NASA/CR–2013-218008

Upgrades to the Probabilistic NAS Platform Air Traffic Simulation Software

George Hunter and Benjamin Boisvert
Saab Sensis Corporation, Campbell, California

June 2013

NASA/CR–2013-218008

Upgrades to the Probabilistic NAS Platform Air Traffic Simulation Software

George Hunter and Benjamin Boisvert
Saab Sensis Corporation, Campbell, California

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NNL10AC94T

June 2013

Available from:

NASA Center for AeroSpace Information
7115 Standard Drive
Hanover, MD 21076-1320
443-757-5802

Table of Contents

Introduction

This document is the final report for NASA BOA: NNL08AA17B, Task Order NNL10AC94T, the Sensis project entitled “Upgrades to the Probabilistic NAS Platform Air Traffic Simulation Software.” This report consists of 17 sections which document the results of the several subtasks of this effort. These 17 sections contain the following documents:

- PNP Separation Assurance Client and Display Enhancements Requirements and Engineering Design
- PNP Separation Assurance Client Software Design Document
- PNP Required Time of Arrival Requirements and Engineering Design
- PNP RTA Conformance Monitor Utility Requirements and Engineering Design
- PNP Required Time of Arrival (RTA) Conformance Monitor Concepts Interface Control Document
- RTA Assignment Client Requirements and Engineering Design
- PNP Weather Avoidance Client and Display Enhancements Requirements and Engineering Design
- PNP Enhanced Terminal Area Modeling Requirements and Engineering Design
- Separation Assurance Software Development Plan
- 2011 Conference paper entitled: “Modeling and Simulation of Interactions Between Traffic Flow Management and Separation Assurance”
- 2012 Conference paper entitled: “NAS-Wide Traffic Flow Management Concept Using Required Time of Arrival, Separation Assurance and Weather Routing”
- PNP developer Guide
- PNP User Guide
- PNP Plan View Display User Guide
- Separation Assurance Developer Guide
- Separation Assurance User Guide
- Required Time of Arrival Client User Guide

These 17 areas of work are documented in Sections A-Q, respectively, in this report.

14809-02

PNP Separation Assurance Client and Display Enhancements Requirements and Engineering Design

Ben Boisvert
George Hunter

Prepared for:

National Aeronautics and Space Administration

Langley Research Center

Hampton, VA 23681

Under:

NASA BOA: NNL08AA17B

December, 2010

Table of Contents

1 Introduction

This document describes software enhancements required to support NASA air traffic control experiments to evaluate advanced separation assurance concepts. Section 2 describes the simulation environment and preliminary work that has been done.

Section 3 describes the requirements for the software to support the upcoming NASA experiments. These are divided into client, server and display categories. Sections 4 and 5 describe design considerations and future enhancements. Section 6 lists the tests to be performed. These sections are also divided into client, server and display categories.

2 Background

This section discusses simulation tool to be used and preliminary tests that have been conducted.

2.1 Probabilistic NAS Platform

In this engineering design we use the Probabilistic NAS Platform (PNP) to implement and test advanced separation assurance (SA) concepts, and the integration of SA and traffic flow management (TFM) concepts.

We developed PNP [1-11], are actively maintaining it, and actively use it for NASA, Joint Planning and Development Office (JPDO), and Federal Aviation Administration (FAA) projects [12-21]. Figure 1 illustrates PNP's client-server architecture.

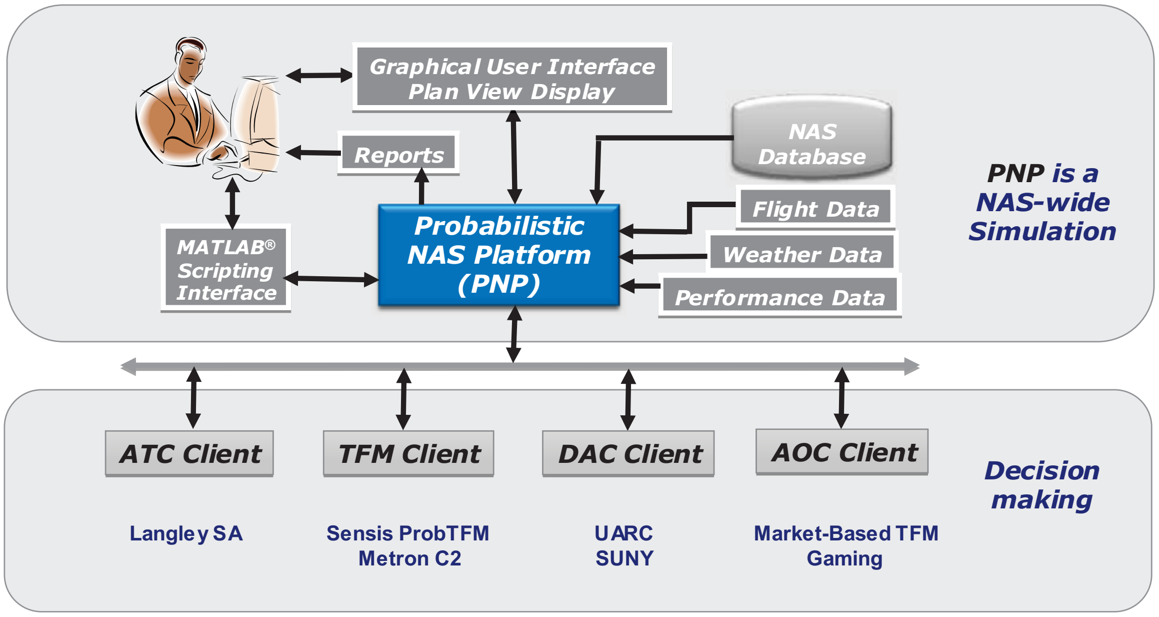
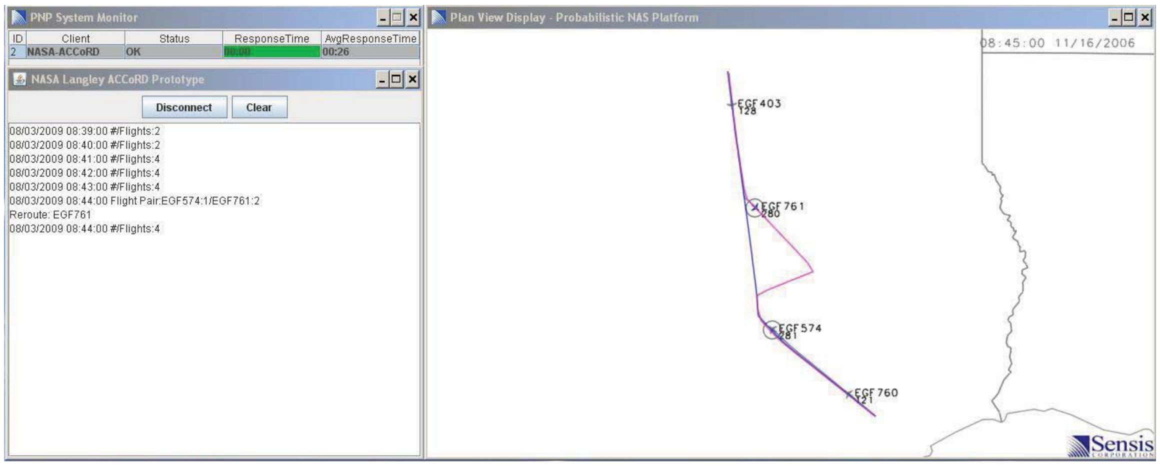


Figure 1 Probabilistic NAS Platform client-server architecture.

2.2 Preliminary tests

We have conducted preliminary tests of separation assurance in PNP. This has included the testing of an ACCoRD prototype developed at the NASA Langley Research Center, as shown in Fig. 2. In these preliminary tests horizontal maneuvers were used to implement 10 mile offset, hypothetical conflict resolutions. These tests used a small set of a dozen flight plans.



ID	Client	Status	ResponseTime	AvgResponseTime
2	NASA-ACCoRD	OK	00:00	00:26

```
08/03/2009 08:39:00 #/Flights:2
08/03/2009 08:40:00 #/Flights:2
08/03/2009 08:41:00 #/Flights:4
08/03/2009 08:42:00 #/Flights:4
08/03/2009 08:43:00 #/Flights:4
08/03/2009 08:44:00 Flight Pair:EGF574:1/EGF761:2
Reroute: EGF761
08/03/2009 08:44:00 #/Flights:4
```

Figure 2 Screen shot of tests of an ACCoRD prototype in the PNP environment.

These tests provided an initial successful integration with PNP. They also revealed that a NAS-wide implementation requires substantial computational resources and that visualization is critical to verification and validation.

3 Requirements

This section lists the software requirements, divided into the PNP separation assurance client, PNP server, and PNP display categories.

3.1 Separation assurance client requirements

1. The SA client shall be a standalone program.
2. The SA client shall support configurable items using INI file format. The INI file format is a de facto standard for configuration files commonly associated with Microsoft Windows.
3. The SA client shall stream output to standard output.
4. The SA client shall stream errors to standard error.
5. The SA client shall use PNP's SimObjects API for communications.
6. The SA client shall send a connect message to the PNP server. The connect message establishes a connection between the PNP server and a client.
7. The SA client shall send a heartbeat response messages to the PNP server in response to a heartbeat request. The heartbeat response message informs the PNP server to continue processing.
8. The SA client shall send a client configuration message to the PNP server. The client configuration message defines the client, and its runtime parameters.
9. The SA client shall request from PNP the flight plans for all flights within a specified distance or time of a specified flight.
10. The SA client shall perform conflict detection of flights for a user-specified conflict detection look ahead time frame.
11. The SA client shall define an application program interface (API) that supports conflict detection algorithms.
12. The SA client shall define the conflict detection look ahead time frame.
13. The SA client shall define the minimum vertical separation distance.
14. The SA client shall define the minimum horizontal separation distance.
15. The conflict detection API (CD-API) shall return loss of separation (LOS) pairs consisting of time, waypoint, and flight IDs.

16. The conflict detection API (CD-API) shall return gain of separation (GOS) pairs consisting of time, waypoint, and flight IDs.
17. The conflict detection API (CD-API) shall return the point of closes approach (PCA) data consisting of time, waypoint, and flight IDs
18. The conflict detection API (CD-API) shall accept 4D trajectory position data.
19. The SA client shall send LOS/GOS flight pairs and PCA data to the PNP server for visualization on the PVD.
20. The SA client shall determine conflict resolution of flights.
21. The SA client shall define an application program interface (API) that supports conflict resolution assurance algorithms.
22. The conflict resolution API (CR-API) shall return resolution maneuvers consisting of modified flight plans (flight plans include the flight ID).
23. The SA client shall perform conflict resolution via turn maneuvers.
24. The SA client shall perform conflict resolution via altitude maneuvers.
25. The SA client shall perform conflict resolution via speed maneuvers.
26. The SA client shall be able to enable and disable turn maneuvers.
27. The SA client shall be able to enable and disable altitude maneuvers.
28. The SA client shall be able to enable and disable speed maneuvers.
29. The conflict resolution API (CR-API) shall accept the LOS, GOS and PCA event data output by the conflict detection API (CD-API).
30. The conflict resolution API (CR-API) shall accept flight plans and their associated trajectory data.
31. The SA client shall send modified flight plans to the PNP server.
32. The SA client shall send a disconnect message to the PNP server when shutting down. The disconnect message allows the PNP server to gracefully shutdown the communications channel.

3.2 PNP server requirements

33. The PNP server shall process and incorporate modified flight plans received from the SA client.
34. The PNP server shall update sector loading based on a modified flight plan.

35. The PNP server shall update airport loading based on a modified flight plan.
36. The PNP server shall update a flight state data based on a modified flight plan.
37. The PNP environment shall include a Trajectory Predictor service that returns trajectory data calculated from an input flight plan.
38. The Trajectory Predictor shall compute the future trajectory data as either *intended* or *projected*, according to an input switch. In the intended mode, the Trajectory Predictor uses the flight plan to predict the future trajectory. In the projected mode, the Trajectory Predictor uses the current velocity vector to predict the future trajectory.
39. The Trajectory Predictor's output trajectory data shall be calculated and output at a user-specifiable data frequency.
40. The Trajectory Predictor's output trajectory data shall consist of latitude, longitude, altitude, time, and velocity for a specified look ahead time (in minutes).
41. The Trajectory Predictor's output trajectory latitude shall be decimal degrees.
42. The Trajectory Predictor's output trajectory longitude shall be decimal degrees.
43. The Trajectory Predictor's output trajectory altitude shall be feet above mean sea level (AMSL).
44. The Trajectory Predictor's output trajectory velocity shall be calculated and output at a user-specifiable (true(wind)/ground airspeed in knots).
45. The Trajectory Predictor's output trajectory time shall be milliseconds since EPOCH.

3.3 PNP display requirements

46. The PVD shall display flight LOS/GOS with a red dog bone.
47. The PVD shall be able to enable the automatic display of the LOS/GOS dog bone.
48. The PVD shall be able to disable the automatic display of the LOS/GOS dog bone.
49. The PVD shall display horizontal profile maneuvers as solid magenta reroutes over solid blue original routes. Note: where routes intersect a magenta line will be displayed.
50. The PVD shall display vertical profile maneuvers as dashed magenta reroutes over solid blue original routes. Note: where routes intersect a magenta line will be dashed over a solid blue line.

51. The PVD shall be able to enable the automatic display of SA maneuvers.

52. The PVD shall be able to disable the automatic display of SA maneuvers.

4 Design considerations

This section describes design considerations for the PNP separation assurance client and PNP server.

4.1 Separation assurance maneuver architecture

There are several solution paths in designing the separation assurance (SA) client architecture. For instance, resolution maneuvers can be constructed as commands (such as vectors in the horizontal plane) which are followed while temporarily suspending the original flight plan. Alternatively, resolution maneuvers can be constructed by amending the flight plan. In the first example, the architecture includes the concept of flying "off-plan," while in the second example the architecture has no such concept. Instead, the flight plan is modified as needed.

We recommend the second example over the first, as it is substantially less complex and is equally capable of supporting resolution maneuvers. We use the concept of a trajectory plan rather than the traditional flight plan simply because when implementing maneuvers more details (e.g., the Mach/CAS profile for a descent) are typically required than are available in a flight plan. Trajectory plans consist of two components, the horizontal route and the vertical profile.

Figure 3 illustrates how four waypoints can be added to a flight plan to fly a resolution maneuver. The resolution maneuver, in this example, consists of four turns which are used to establish a lateral offset and so avoid loss of separation, returning to the original planned track at the completion of the maneuver.

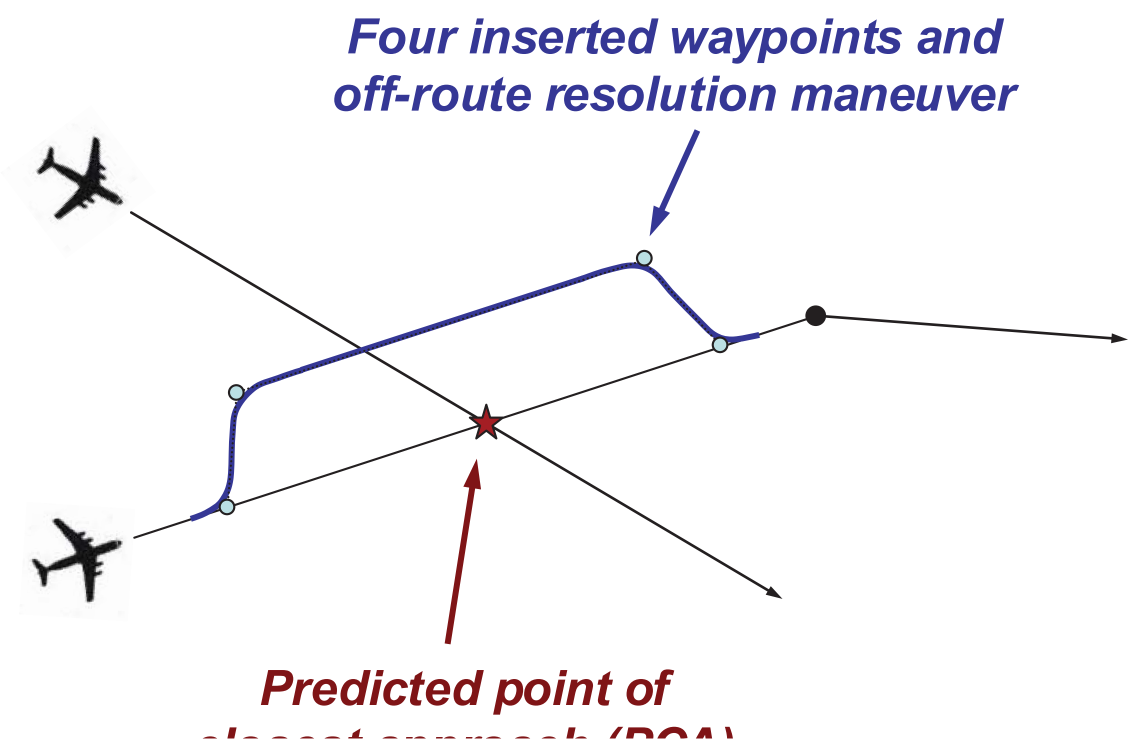


Figure 3 Illustration of a detected conflict which is resolved using turn maneuvers to achieve a lateral offset. In this example method, four waypoints are added to construct the turn maneuvers to achieve an offset and return to the original trajectory.

In our architecture all aircraft always have an associated trajectory plan. The trajectory plan may be retrieved for any flight using a query command, and the current location of the aircraft is identified. To perform a maneuver the trajectory plan is replaced with a new one.

Within the trajectory plan, we separate the horizontal and vertical plane information into two different components. The vertical plane component specifies a series of segments with speed / altitude target states that are to be achieved at the completion of the segment. For instance, consider an aircraft that is currently in a constant speed, level cruise phase segment.

Now consider an altitude maneuver consisting of a descent followed by a level segment, and finally a climb that returns to the original altitude. This would be achieved by inserting three new segments. The three new segments would (i) descend to a target altitude, (ii) remain level at that altitude, and (iii) then climb to the original altitude. The flight would then resume its level cruise phase segment. Speed changes may be implemented in the same way.

These turn, altitude and speed maneuvers can be tested in a trial planning algorithm. That is, rather than implementing a new trajectory plan, one can be hypothesized. We implement this architecture as illustrated in Fig. 4.

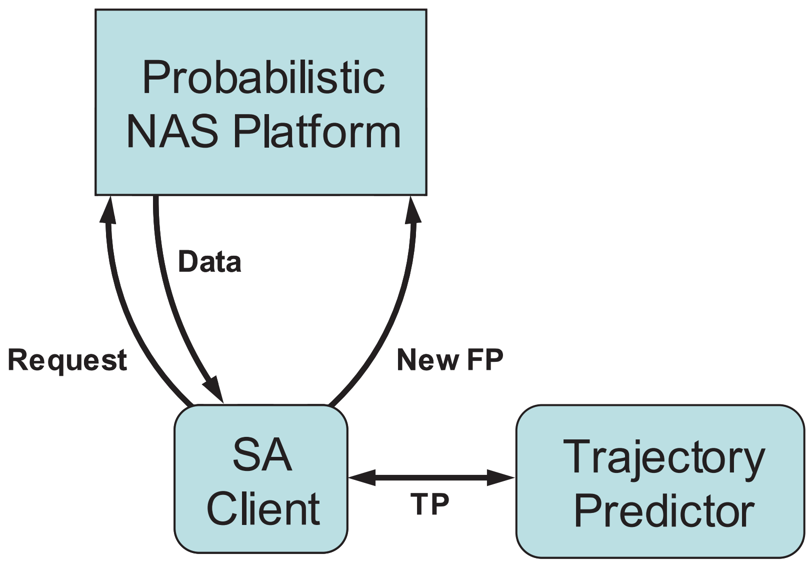


Figure 4 High level separation assurance maneuver architecture.

As Fig. 4 shows, the SA client requests flight plans from PNP. The client then uses the data to derive new flight plans for one or more flights which resolve predicted conflicts. As part of this computation the SA client uses the Trajectory Prediction service to obtain detailed trajectory data based on the flight plans. The trajectory data indicate the predicted trajectory path. Figure 5 shows a more detailed chart of this architecture.

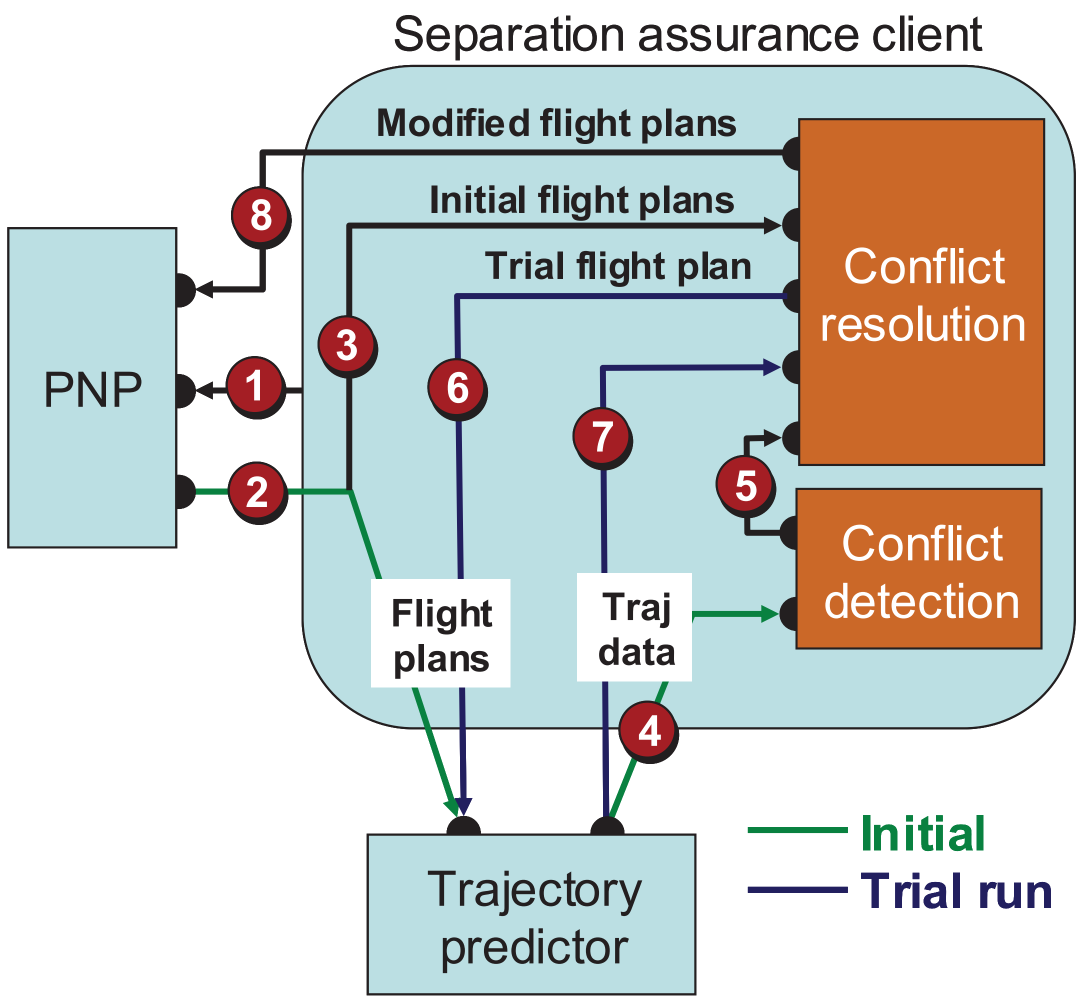


Figure 5 Detailed separation assurance maneuver architecture.

The eight steps identified in the Fig. 5 architecture are described below.

Step 1 The SA client requests the flight plans of all flights within a specified distance or time of the flight of interest.

Step 2 PNP returns the requested flight plans which the SA client sends to the Trajectory Predictor service.

Step 3 The SA client will also send the flight plans to the Conflict Resolution module for its reference.

Step 4 The Trajectory Predictor service returns the trajectory data calculated from the flight plans. These data extend only to the specified look ahead time frame. The SA client sends these data to the Conflict Detection module.

Step 5 The Conflict Detection module returns the point of closest approach (PCA) and loss and gain of separation event data which the SA client sends to the Conflict Resolution module and to

PNP. The SA client also sends the trajectory data (see Step 4) to the Conflict Resolution module for its reference.

Step 6 The Conflict Resolution module computes trial-run resolution maneuvers and their associated flight plan modifications. The SA client sends the new flight plans to the Trajectory Predictor service.

Step 7 The Trajectory Predictor service returns the trajectory data calculated from the flight plans. These data extend only to the specified look ahead time frame. The SA client sends these data to the Conflict Resolution module, as trial plan results.

Step 8 The Conflict Resolution module iterates through Steps 6 and 7 as necessary. Note that as part of its evaluation of the trial plan, the Conflict Resolution module performs a conflict detection check between the modified flights and all other flights. For instance, if two flights are modified, then each one must be checked against all other flights in the set. A possible strategy for performing these checks is to use the Conflict Detection module. After the trial planning process is complete, the Conflict Resolution module returns its final new flight plans. The SA client returns these to PNP for processing and incorporation into the simulation.

Table 1 lists the inputs and outputs of the Trajectory Predictor service.

Table 1 Trajectory Predictor API (TP-API) inputs and outputs.

Inputs	Outputs
A flight plan.	Predicted trajectory data (position and velocity) at the user specified data frequency, based on the input flight plan.

Table 2 lists the inputs and outputs of the Conflict Detection API.

Table 2 Conflict Detection API (CD-API) inputs and outputs.

Inputs	Outputs
Trajectory position data for *N* flights at the user specified data frequency.	Loss of separation event data (time, position and two flight IDs.
Conflict detection look ahead time frame.	Gain of separation event data (time, position and two flight IDs.
Minimum horizontal separation distance.	PCA event data (time, position and two flight IDs.
Minimum vertical separation distance.	
List of *M* flights to check (*M* less than or equal to *N*)	

Table 3 lists the inputs and outputs of the Conflict Resolution API.

Table 3 Conflict Resolution API (CR-API) inputs and outputs.

Inputs	Outputs
Trajectory data (position and velocity) for *N* flights at the user specified data frequency.	Modified flight plans for selected flights.
Flight plans for the *N* flights.	
Loss of separation event data (time and two flight IDs.	
Gain of separation event data (time and two flight IDs.	
PCA event data (time, position and two flight IDs.	
Conflict detection look ahead time frame.	
Minimum horizontal separation distance.	
Minimum vertical separation distance.	
Enabling and disabling switch settings.	

4.2 Conflict detection module

The conflict detection module finds all loss-of-separation events between the current time and the current time plus the conflict detection look ahead time. The loss-of-separation events that the conflict detection module identifies are those between (i) the specified subset of *M* flights and (ii) all the *N* input flights. For instance, if all the flights are specified as flights to evaluate ($M = N$), then the conflict detection module performs an all-against-all check for loss-of-separation events. On the other hand, if only a single flight is specified as the flight to evaluate ($M = 1$), then the conflict detection module performs a one-against-all check for loss-of-separation events. The value of *M* may be anywhere from 1 to *N*.

Loss-of-separation events occur when the horizontal separation between two flights is less than the minimum horizontal separation distance, or the vertical separation between two flights is less than the minimum vertical separation distance. Figure 6 illustrates the geometry of these two minimum separation criteria. Note that the horizontal separation between two aircraft can be computed in a local Cartesian frame of reference, for instance centered at one of the two aircraft.

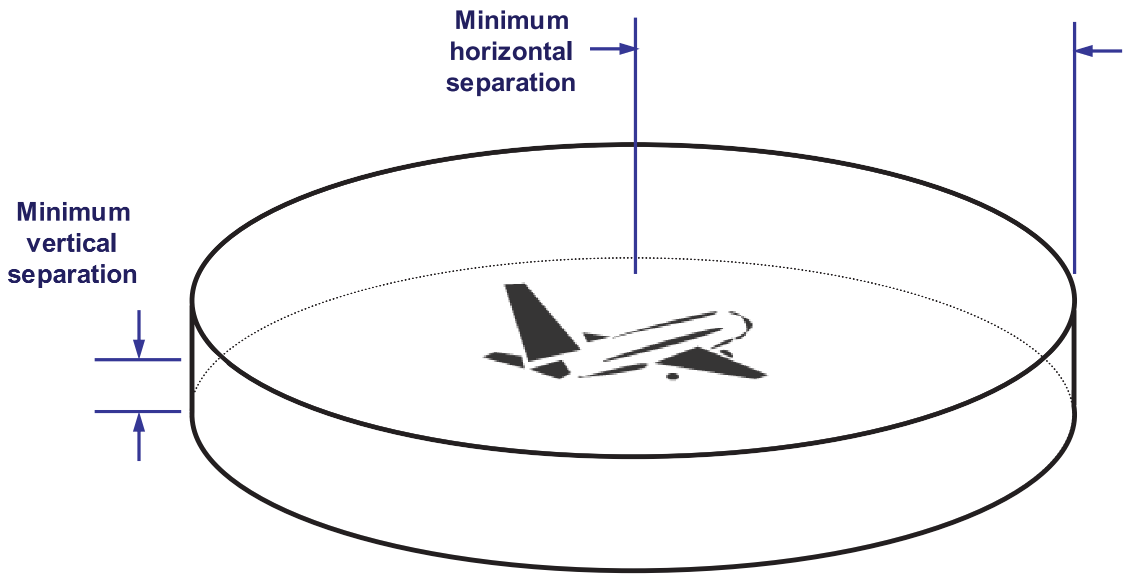


Figure 6 High level separation assurance maneuver architecture.

Loss-of-separation events are not instantaneous, but rather are finite. They include three important sub events: the point at which separation is lost, the point of closest approach, and the point at which separation is gained. Figure 7 illustrates these three sub events.

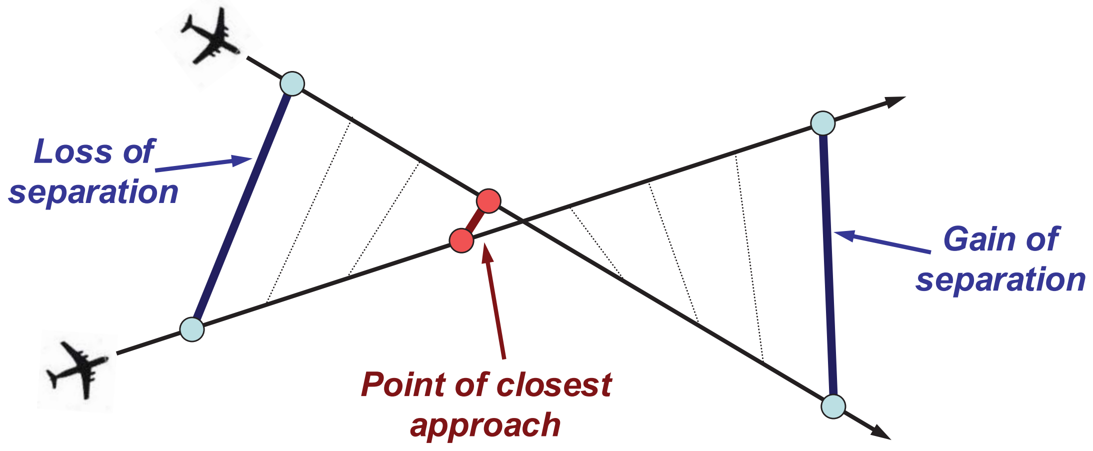


Figure 7 High level separation assurance maneuver architecture.

The input trajectory data are given at discrete time points. We define the loss of separation (LOS) time as the discrete time point at which separation is first lost. We define the gain of separation (GOS) time as the discrete time point at which separation is first gained. And we define the point of closest approach (PCA) as the discrete time point with the least separation of all the time points.

As mentioned above, the conflict detection module returns only those loss-of-separation events that occur in the conflict detection look ahead time frame. But for a given look ahead time frame, a loss-of-separation event may begin and end within the time frame, or it may overlap the time frame, as illustrated in Fig. 8.

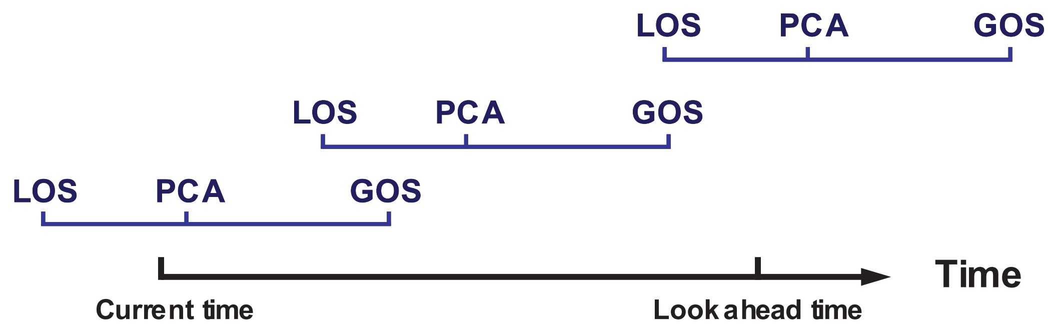


Figure 8 High level separation assurance maneuver architecture.

Figure 8 illustrates three different loss-of-separation events. One precedes the look ahead time frame, one begins and ends within the time frame, and one extends beyond the time frame. Also, note that in the first and third cases, the PCA time point may be in or out of the time frame. In overlap cases, the conflict detection module output reports only the sub events that occur within the conflict detection time frame. For instance, if no LOS time is reported, then this means the aircraft pair are in conflict at the beginning of the time frame (i.e., the current time). On the other hand, if no GOS time is reported, then this means the aircraft pair are in conflict at the end of the time frame.

Also, the predicted runway arrival time, for either flight in a conflict, may occur anywhere within the conflict detection time frame. When this occurs, none of the three conflict events (i.e., LOS, PCA and GOS) are possible after the arrival time.

4.3 Conflict resolution module

As Fig. 5 and Table 3 show, the conflict detection data, as well as the nominal flight plans and associated trajectory data, are input to the Conflict Resolution API (CR-API). The conflict resolution module uses these data to derive maneuvers to resolve the conflicts. Our design segregates the fundamental services (conflict detection and trajectory prediction) used by the conflict resolution module and the conflict resolution algorithm itself.

The objective is to design and implement the fundamental services that are required to support a wide variety of conflict resolution algorithms. The particular conflict resolution algorithm designed and implemented here does not guarantee successful resolution of the detected conflicts. To remain with the project scope, this conflict resolution algorithm is of modest complexity. The objectives of this algorithm are to demonstrate the implementation of a conflict resolution algorithm, exercise the fundamental services, and serve as a nominal conflict resolution algorithm in support of TFM-SA interaction studies.

In the conflict resolution module, each loss of separation event is treated independently. For each loss of separation event, we first ensure that both aircraft are airborne for the duration of the conflict resolution time frame.[1] In other words, the runway departure time must be prior to the current time, and the predicted runway arrival time must be after the GOS event. Also, the PCA must occur within the conflict resolution time frame. If either of these tests fails, then the conflict is ignored. If both pass, then the conflict resolution module evaluates three types of maneuvers: speed change, altitude change and heading change.

4.3.1 Speed change maneuver

The logic first evaluates the speed change maneuver. The speed change maneuver is most desirable because it introduces the least deviation, and therefore introduces the least complexity. The flight does not deviate in its intended altitude profile or its horizontal route. It only changes its crossing times.

Also, the speed change maneuver is usually the most fuel efficient. A speed reduction decreases aerodynamic drag while an altitude maneuver often forces a flight to a less efficient altitude, and a turn maneuver increases the distance flown.

But the speed change maneuver is the most restricted. This is because the speed change maneuver offers the least distance deviation of the three maneuver types. To achieve the necessary separation a longer time span is required. Therefore the speed change maneuver often is insufficient within the conflict resolution time frame.

The speed change maneuver is also geometrically restricted. As Fig. 9 illustrates, the speed change maneuver is ineffective in head-on encounters. In that case, a speed change merely alters the PCA time, but not the PCA itself. And in crossing encounters the effectiveness of the speed change maneuver can be limited. This depends on the precise location of the PCA, and the speed change that is feasible.

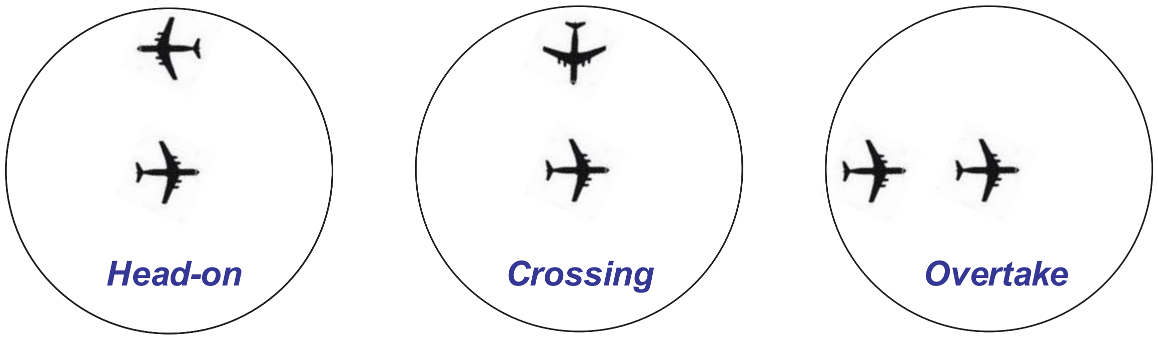


Figure 9 High level separation assurance maneuver architecture.

[1] The conflict resolution time frame is synonymous with the conflict detection time frame.

Commercial transports in en route flight often operate near the limits of their performance envelope. Therefore speed reduction is often more feasible than speed increase. This lack of control symmetry has implications in the crossing encounter geometries. For instance, in certain scenarios a speed *reduction* might aggravate the separation loss. What is needed is a speed increase, but that may not be available.

For these reasons, while the speed control maneuver is often the most desirable, it also has the least availability. Therefore we evaluate the speed control maneuver first, and if it is not feasible then we move to the altitude change maneuver.

To evaluate the speed control maneuver we first check the geometry. We ensure that it is an overtake encounter by ensuring that at the PCA, aircraft A is in the forward hemisphere of aircraft B, and aircraft B is in the [illegible] emisphere of aircraft A. Figure 10 illustrates this check.

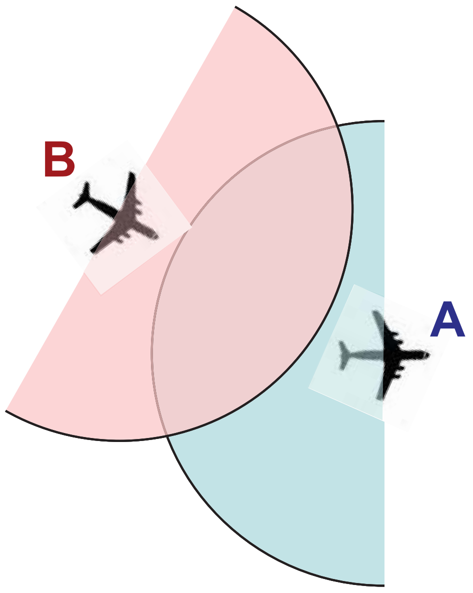


Figure 10 High level separation assurance maneuver architecture.

We use the aircraft velocity vector, and specifically the heading angle, to define the hemisphere, and we require the respective heading angles to be within 45º:

$$\Delta\psi = \left|\psi_A - \psi_B\right| \leq 45^\circ \quad (4.1)$$

where,

ψ_A = heading angle of Aircraft A at PCA (degrees).

We also restrict our speed change maneuver to speed reductions. This means that only the trailing aircraft may be maneuvered in the speed change maneuver. We estimate the required speed reduction (in knots) as:

$$\Delta V = \frac{\Delta P}{\Delta T \cos \Delta\psi} \qquad (4.2)$$

where,

ΔP = minimum horizontal separation – PCA horizontal separation (nmi),
ΔT = PCA time – current time (hours).

We round ΔV up to the next multiple of 5, and we reject the speed change maneuver if it is greater than 20 kts. Note that to pass this test, the PCA must occur within the look ahead time frame (see Fig. 8).

If these geometry and separation tests are passed, then the conflict resolution module enters a trial planning iteration to confirm separation assurance. Specifically, Aircraft B immediately decelerates until its airspeed is reduced by ΔV. This new trajectory is tested against all other trajectories (one against all) using the conflict detection module. The maneuver is accepted if the conflict detection module returns no conflicts. In this case, Aircraft B maintains its new, slower, speed. There is no recovery maneuver.

4.3.2 Altitude change maneuver

If the speed change maneuver is not accepted, then the altitude change maneuver is evaluated. First, the altitude difference between the two aircraft must exceed 50% of the minimum vertical separation. If this is true, then the trial planning step is next. The lower aircraft (Aircraft B) immediately initiates a descent maneuver. The altitude reduction (in feet) is equal to the minimum vertical separation minus the altitude separation at the PCA, plus a buffer of 100 ft:

$$\Delta h = MVS - \left(h_A - h_B\right) + 100 \quad (4.3)$$

where,

MVS = minimum vertical separation (ft),
h_A = altitude of Aircraft A at PCA (ft).

We round Δh up to the next multiple of 100. If the GOS event occurs within the conflict resolution time frame, then Aircraft B performs a recovery maneuver. Otherwise there is no recovery maneuver and the lower altitude is maintained. The recovery maneuver is a climb back to the initial altitude, beginning at the GOS time.

This new trajectory is tested against all other trajectories (one against all) using the conflict detection module. The maneuver is accepted if the conflict detection module returns no conflicts.

4.3.3 Heading change maneuver

If both the speed change and altitude change maneuvers are rejected, then the conflict resolution module attempts to find a heading change maneuver to resolve the conflict. Such maneuvers are only performed for aircraft whose predicted runway arrival time is at least 10 minutes after the GOS time. For aircraft that pass this test, two heading maneuvers are evaluated: a turn maneuver initially to the right, and a turn maneuver initially to the left. Therefore, in this heading change maneuver algorithm, a maximum of four different maneuvers are evaluating using trial planning (i.e., two aircraft and two different trial plan maneuvers per aircraft). The first successful trial plan maneuver that is found is used. If all the maneuvers fail, then no heading change maneuver is implemented. Each trial plan maneuver consists of:

- An approximately 45º turn (to the right or left) commenced immediately,
- A straight segment,
- An approximately 45º turn (to the left or right) to return to the initial heading.

The length of the straight segment is sufficient to generate a separation greater than PCA loss of separation. Its time duration (in hours) is:

$$\Delta T = \frac{\Delta P}{.707 \times \Delta V} \qquad (4.4)$$

where,

ΔP = minimum horizontal separation – PCA horizontal separation (nmi),
ΔV = Aircraft airspeed at current time (kt).

The recovery maneuver is simply the reverse of the conflict avoidance maneuver, commencing at the GOS time plus a time buffer of 30 seconds.

4.4 PNP server design

The engineering design of the PNP server does not change. Modifications that may be required to support the SA client are discussed in the Software Design Document.

5 Testing

This section describes the software testing for the PNP separation assurance client, PNP server, and PNP display enhancements. Table 4lists the PNP SA client tests.

Table 4 PNP SA client tests.

PNP Separation Assurance Client Tests					
#	Requirements	Qualification	Design Traceability	Code Traceability	Build
1	The SA client shall be a standalone program.	Demonstration			1
2	The SA client shall support configurable items using INI file format. The INI file format is a de facto standard for configuration files commonly associated with Microsoft Windows.	Demonstration			1
3	The SA client shall stream output to standard output.	Demonstration			1
4	The SA client shall stream errors to standard error.	Demonstration			1
5	The SA client shall use PNP's SimObjects API for communications.	Inspection			1
6	The SA client shall send a connect message to the PNP server. The connect message establishes a connection between the PNP server and a client.	Analysis			1
7	The SA client shall send a heartbeat response messages to the PNP server in response to a heartbeat request. The heartbeat response message informs	Analysis			1

	the PNP server to continue processing.				
8	The SA client shall send a client configuration message to the PNP server. The client configuration message defines the client, and its runtime parameters.	Analysis			1
9	The SA client shall request from PNP the flight IDs for all flights within a specified distance or time of a specified flight.	Analysis			1
10	The SA client shall perform conflict detection of flights for a user-specified conflict detection look ahead time frame.	Inspection			1
11	The SA client shall define an application program interface (API) that supports conflict detection algorithms.	Inspection			1
12	The SA client shall define the conflict detection look ahead time frame.	Inspection			1
13	The SA client shall define the minimum vertical separation distance.	Inspection			1
14	The SA client shall define the minimum horizontal separation distance.	Inspection			1
15	The conflict detection API (CD-API) shall return loss of separation (LOS) pairs consisting of time, waypoint, and flight IDs.	Inspection			1
16	The conflict detection API	Inspection			1

	(CD-API) shall return gain of separation (GOS) pairs consisting of time, waypoint, and flight IDs.				
17	The conflict detection API (CD-API) shall accept 4D trajectory position data.	Inspection			1
18	The SA client shall send LOS/GOS flight pairs to the PNP server for visualization on the PVD.	Demonstration			1
19	The SA client shall determine conflict resolution of flights.	Analysis			1
20	The SA client shall define an application program interface (API) that supports conflict resolution assurance algorithms.	Inspection			1
21	The conflict resolution API (CR-API) shall return resolution maneuvers consisting of modified flight plans (flight plans include the flight ID).	Analysis			1
22	The SA client shall perform conflict resolution via turn maneuvers.	Analysis			1
23	The SA client shall perform conflict resolution via altitude maneuvers.	Analysis			1
24	The SA client shall perform conflict resolution via speed maneuvers.	Analysis			1
25	The SA client shall be able to enable and disable turn maneuvers.	Demonstration			1

26	The SA client shall be able to enable and disable altitude maneuvers.	Demonstration			1
27	The SA client shall be able to enable and disable speed maneuvers.	Demonstration			1
28	The conflict resolution API (CR-API) shall accept the LOS and GOS event data output by the conflict detection API (CD-API).	Inspection			1
29	The conflict resolution API (CR-API) shall accept flight plans and their associated trajectory data.	Inspection			1
30	The SA client shall send modified flight plans to the PNP server.	Analysis			1
31	The SA client shall send a disconnect message to the PNP server when shutting down. The disconnect message allows the PNP server to gracefully shutdown the communications channel.	Analysis			1

Table 5 lists the PNP server tests.

Table 5 PNP server tests.

PNP Server Tests					
#	Requirements	Qualification	Design Traceability	Code Traceability	Build
32	The PNP server shall be able to process and incorporate modified flight plans received from the SA	Analysis			1

	client.				
33	The PNP server shall update sector loading based on a modified flight plan.	Analysis			1
34	The PNP server shall update airport loading based on a modified flight plan.	Analysis			1
35	The PNP server shall update a flight state data based on a modified flight plan.	Analysis			1
36	The PNP environment shall include a Trajectory Predictor service that returns trajectory data calculated from an input flight plan.	Inspection			1
37	The Trajectory Predictor's output trajectory data shall be calculated and output at a user-specifiable data frequency.	Inspection			1
38	The Trajectory Predictor's output trajectory data shall consist of latitude, longitude, altitude, time, and velocity for a specified look ahead time (in minutes).	Analysis			1
39	The Trajectory Predictor's output trajectory latitude shall be decimal degrees.	Inspection			1
40	The Trajectory Predictor's output trajectory longitude shall be decimal degrees.	Inspection			1
41	The Trajectory Predictor's output trajectory altitude shall be feet above mean sea level (AMSL).	Inspection			1

42	The Trajectory Predictor's output trajectory velocity shall be calculated and output at a user-specifiable (true(wind)/ground airspeed in knots).	Inspection			1
43	The Trajectory Predictor's output trajectory time shall be milliseconds since EPOCH.	Inspection			1

Table 6 lists the PNP display tests.

Table 6 PNP display tests.

PNP Display Tests					
#	Requirements	Qualification	Design Traceability	Code Traceability	Build
44	The PVD shall display flight LOS/GOS with a red dog bone.	Demonstration			3
45	The PVD shall be able to enable the automatic display of the LOS/GOS dog bone.	Demonstration			3
46	The PVD shall be able to disable the automatic display of the LOS/GOS dog bone.	Demonstration			3
47	The PVD shall display horizontal profile maneuvers as solid magenta reroutes over solid blue original routes. Note: where routes intersect a magenta line will be displayed.	Demonstration			3
48	The PVD shall display vertical profile maneuvers as dashed magenta reroutes	Demonstration			3

	over solid blue original routes. Note: where routes intersect a magenta line will be dashed over a solid blue line.				
49	The PVD shall be able to enable the automatic display of SA maneuvers.	Demonstration			3
50	The PVD shall be able to disable the automatic display of SA maneuvers.	Demonstration			3

6 Conclusions

This document describes software enhancements required to support NASA air traffic control experiments to evaluate advanced separation assurance concepts. This includes both requirements and design considerations. The requirements and design considerations fall into client, server and display categories. This document also presents design enhancements for the future, and testing to be conducted.

References

1. George Hunter, "Testing and Validation of NextGen Simulators," AIAA Modeling and Simulation Conference, Chicago, IL, August 2009.

2. George Hunter, Kris Ramamoorthy, "Integration of terminal area probabilistic meteorological forecasts in NAS-wide traffic flow management decision making," 13th Conference on Aviation, Range and Aerospace Meteorology, New Orleans, LA, January, 2008.

3. Kris Ramamoorthy, George Hunter, "The Integration of Meteorological Data in Air Traffic Management: Requirements and Sensitivities," 46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January, 2008.

4. George Hunter, Ben Boisvert, Kris Ramamoorthy, "Advanced Traffic Flow Management Simulation Experiments and Validation," 2007 Winter Simulation Conference, Washington, DC, December, 2007.

5. Kris Ramamoorthy, George Hunter, "Evaluation of National Airspace System Performance Improvement With Four Dimensional Trajectories," AIAA Digital Avionics Systems Conference (DASC), Dallas, TX, October, 2007.

6. George Hunter, Ben Boisvert, Kris Ramamoorthy, "Use of automated aviation weather forecasts in future NAS," The 87th American Meteorological Society Annual Meeting, San Antonio, TX, January, 2007.

7. Kris Ramamoorthy, George Hunter, "Probabilistic Traffic Flow Management in the Presence of Inclement Weather and Other System Uncertainties," INFORMS Annual Meeting, Pittsburgh, PA, November, 2006.

8. Kris Ramamoorthy, Ben Boisvert, George Hunter, "A Real-Time Probabilistic TFM Evaluation Tool," AIAA Digital Avionics Systems Conference (DASC), Portland, OR, October, 2006.

9. Kris Ramamoorthy, George Hunter, "A Trajectory-Based Probabilistic TFM Evaluation Tool and Experiment," Integrated Communications, Navigation and Surveillance Conference (ICNS), Baltimore, MD, May, 2006.

10. Kris Ramamoorthy, George Hunter, "Avionics and National Airspace Architecture Strategies for Future Demand Scenarios in Inclement Weather," AIAA Digital Avionics Systems Conference (DASC), Crystal City, VA, October, 2005.

11. George Hunter, Kris Ramamoorthy, Joe Post, "Evaluation of the Future National Airspace System in Heavy Weather," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Arlington, VA, September 2005.

12. George Hunter, "Probabilistic Forecasting of Airport Capacity," AIAA Digital Avionics Systems Conference (DASC), Salt Lake City, UT, October, 2010.

13. Poornima Balakrishna, George Hunter, "Preliminary NextGen Collaborative Air Traffic Management Analysis," AIAA Digital Avionics Systems Conference (DASC), Salt Lake City, UT, October, 2010.

14. Huina Gao, George Hunter, Frank Berardino, Karla Hoffman, "Development and Evaluation of Market-Based Traffic Flow Management Concepts," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Dallas, TX, September 2010.

15. Huina Gao, George Hunter, "Evaluation of User Gaming Strategies in the Future National Airspace System," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Dallas, TX, September 2010.

16. George Hunter, "Meta Simulation Results for Simultaneous Dynamic Resectorization and Traffic Flow Management," AIAA Digital Avionics Systems Conference (DASC), Orlando, FL, October, 2009.

17. Huina Gao, George Hunter, "Future NAS-Wide User Gaming Preliminary Investigation," AIAA Digital Avionics Systems Conference (DASC), Orlando, FL, October, 2009.

18. George Hunter, "Preliminary Assessment of Interactions Between Traffic Flow Management and Dynamic Airspace Configuration Capabilities," AIAA Digital Avionics Systems Conference (DASC), St. Paul, MN, October, 2008.

19. George Hunter, "Toward an Economic Model to Incentivize Voluntary Optimization of NAS Traffic Flow," AIAA ATIO Conference, Anchorage, AK, September, 2008.

20. George Hunter, "Sensitivity of the National Airspace System Performance to Weather Forecast Accuracy," Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, May, 2008.

21. Kris Ramamoorthy, Ben Boisvert, George Hunter, "Sensitivity of Advanced Traffic Flow Management to Different Weather Scenarios," Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, May, 2007.

14809-02

PNP Separation Assurance Client Software Design Document

Saab Sensis Corporation

Prepared for:

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681

Under:

NASA BOA: NNL08AA17B

August, 2011

Table of Contents

1 Introduction

This document describes software design required to support NASA air traffic control experiments to evaluate advanced separation assurance concepts.

2 Software Architecture

PNP is a client/server architecture. PNP supports multiple clients. Clients may be distributed locally or remotely. PNP's architecture is plug-and-play, so PNP requires no advance knowledge of clients. Clients can dynamically enter and exit the system.

When a client joins the PNP simulation, it registers with PNP and specifies what types of data it will require, and at what intervals.

An airspace client is required to provide the PNP server with the required sectorization information to evaluate flights in the National Airspace System (NAS). The PNP architecture consists of two required Computer Software Configuration Items (CSCI): the PNP server and an airspace client. A Computer Software Configuration Item is the logical grouping of software by functionality to form a single entity. The CSCIs described in this architecture are the Dynamic Airspace Client (DAC), the Probabilistic Traffic Flow Management Client (ProbTFM), and the Separation Assurance Framework Client (SA).

The PNP architecture is scalable and supports an unlimited number of clients. Note that there can be multiple instances of the SA clients dedicated to the separation of a specified aircraft as illustrated below in Figure 1.

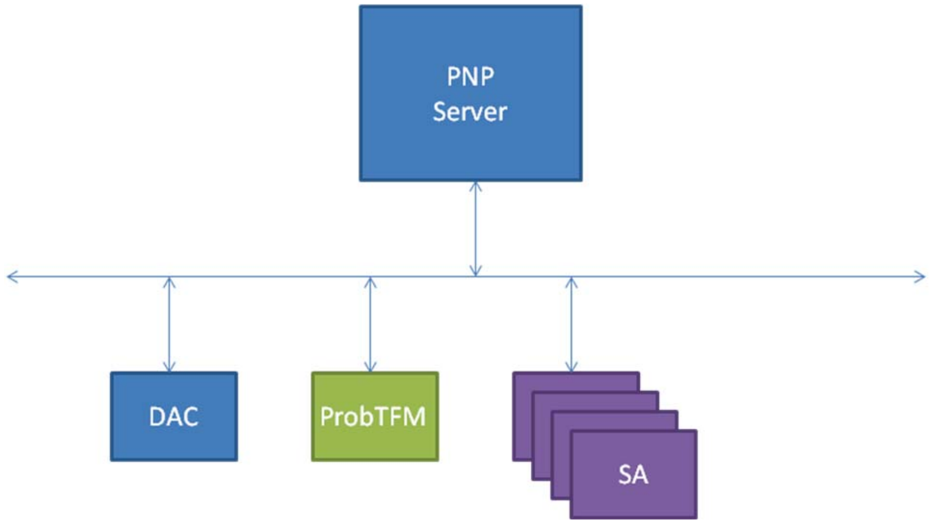


Figure 1 PNP Client/Server Architecture

3 PNP Server

This section describes the PNP server Computer Software Configuration Item.

3.1 Functional Description

The PNP server is responsible for the execution and flow control of the system. The PNP server manages communications, airport capacities/loading, sector definitions/loading (geometries/capacities/monitor alert parameters), display updates, flight data (trajectory models/persistence), and weather models.

PNP server uses meteorological data provided by WSI, including radar reflectivity, winds, and ceiling and visibility. PNP server uses NAS airspace data, airport location and capacity, demand set data containing flight data sets (FDSs), and BADA aircraft performance data (Please refer to the PNP User Agreement for data use requirements).

The PNP server maintains the connections of clients. The PNP server provides client status through the exchange of heartbeat messages. The PNP server sends requested data to clients based on the clients data and interval requests. The PNP server responds interactively with clients based on route requests. The PNP server accepts flight flan modifications, flight delays, and re-sectorization requests [Req#32]. The PNP Server maintains the current state of all flights loaded for a given look ahead period till their arrival. Airport and sector loading is calculated based on the current state of all the flights loaded in the system. Flight state, airport, and sector loading data is re-calculated based on reroutes, delays and re-sectorization [Req#33-35].

Strategic planning is accomplished through predicted trajectory reports, weather and sector modeling. The PNP Server creates minute trajectories. These trajectories provide the predicted position reports used to load the sector tables. The PNP server creates decision support data based on a 15 minute resolution. The decision support data consist of: (1) Sector loading, capacities and capacity reductions based on reflectivity and/or user specified reductions and (2) Airport loading, capacities and capacity reductions based on METARs.

The PNP server uses a Trajectory Predictor (TP) developed by Sensis called PointMass. The TP accepts a user-specifiable resolution in seconds [Req#37]. The TP returns a trajectory based on the following inputs: (1) aircraft type, (2) flight plan, (3) cruise altitude, (4) cruise airspeed, (5) departure time, (6) resolution, (7) initial altitude, (8) initial airspeed, (9) arrival runway altitude, and (10) 4D gridded winds [Req#36]. The TP resultant trajectory consists of latitude, longitude, altitude, time, and velocity (true(wind)/ground airspeed in knots) [Req#38-43].

The PNP server has two runtime modes. In the real-time mode the PNP server connects to WSI's real-time traffic and weather data feed and receives messages WSI's real-time traffic and weather data feed provides NOWRAD, GRIB, METAR, TAF, and SIGMENT data with ASDI data on a 5 minute delay. Time Management is driven by the system clock. In the fast-time mode the PNP server uses archived data sets to read flights and conditions based on an look ahead time. An archive consists flight data sets, sector definitions, airport definitions, NOWRAD images, GRIB data, METAR, TAF, and SIGMENT reports. Time Management is driven by an internal timer.

The PNP server has a Plan View Display (PVD). The PVD allows the user to visualize the NAS. The PVD displays flight data, reflectivity, sector congestion, airport congestion, sector geometries, forecasts, METARs, TAFs and winds. The PVD will display flight LOS/GOS with a red dog bone [Req#44]. The PVD will have a check box button dedicated to the display of the LOS/GOS dog bones [Req#45-46]. The PVD will display horizontal profile maneuvers as solid magenta reroutes over solid blue original routes [Req#47]. The PVD will display vertical profile maneuvers as dashed magenta reroutes over solid blue original routes [Req#48]. The PVD display have a check box dedicated to the display of SA maneuvers [Req#49-50].Toggles have been built in to control the display as the PVD may become cluttered during a session when aircraft counts typically exceed 50,000 as illustrated below in Figure 2.

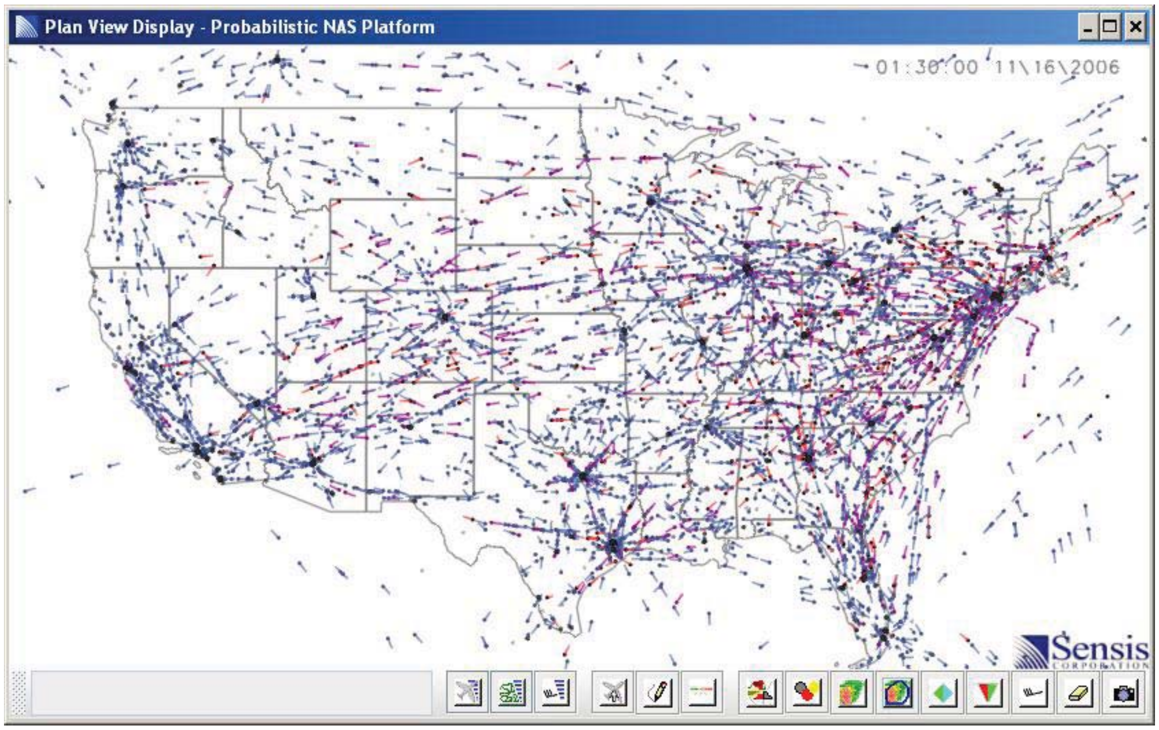


Figure 2 Plan View Display Snapshot

3.1.1 Services

This section describes the PNP server services available to clients:

Sectorization

The PNP server will send out the sectorization at startup and after a resectorization The sectorization data includes: geometries, id/name mapping, subsector mapping, capacities, and min/max altitude definitions.

Airport Capacities

The PNP server will send out the airport capacities. Default airport capacities are based on a 2 point Pareto curve but N point Pareto curve data can also be configured to seed the capacities.

City Pair Routes

The PNP server will send out city pair routes in response to a CityPairRequest message.

Flight Plan Modifications

The PNP server will accept flight plan modifications through BatchResponse message.

Flight Delays

The PNP server will accept flight delays through BatchResponse message.

3.2 Software Design

This section describes the software high level design of the PNP server. The PNP server consist of two threads. One thread dedicated to communications and one thread dedicated to event processing. There are shared objects between the two threads that synchronize information exchange. This is illustrated below in Figure 3.

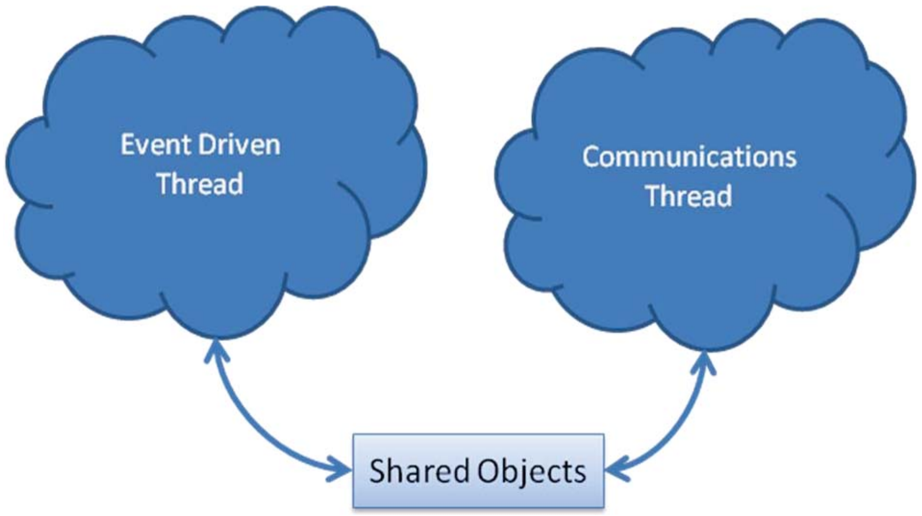


Figure 3 PNP Server Threads

3.2.1 Event Thread

The PNP Server main event thread manages time and timers. The server uses a minute timer to check if any client requests need to be fulfilled and flight state data needs to be updated. The server uses an interval time to run models/algorithms and generate reports and data sets. This is illustrated below in Figure 4.

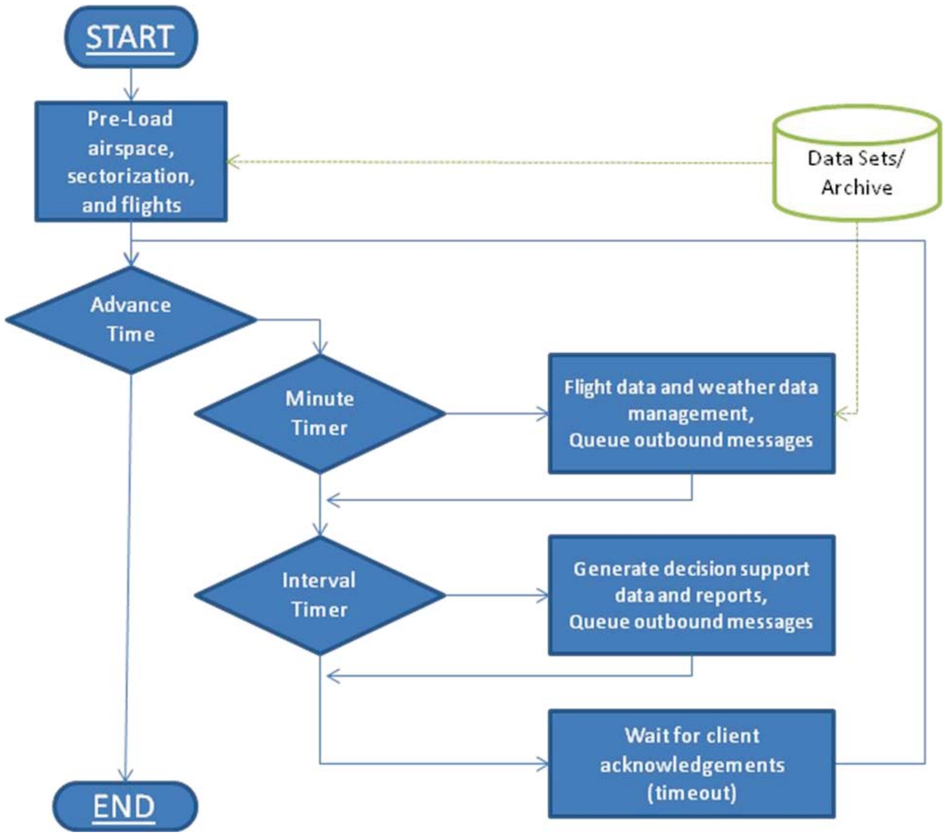

START
Pre-Load airspace, sectorization, and flights
Data Sets/ Archive
Advance Time
Minute Timer
Flight data and weather data management, Queue outbound messages
Interval Timer
Generate decision support data and reports, Queue outbound messages
Wait for client acknowledgements (timeout)
END

Figure 4 PNP Event Thread Design Flow

Initialization

At system startup, the server launches three threads. An initialization thread that loads the airspace and airport capacity datasets then terminates. A communications thread that sets up the server to connection requests and messages. An event thread to control time and scheduled events. The Dynamic Airspace Client (DAC) provides the sectorization and terminates. The event thread loads flights from the adapted flight data set based on the configurable items: start time and look-ahead time. The start time is used to determine what time the simulation is suppose to start. The look-ahead time is used to control how far into the future the server is to load flights. Weather phenomena is loaded and processed based on the start time and configurable parameters.

Steady State

In the main loop of the event thread time is controlled. In the fast-time mode, events are executed as fast as possible. Every iteration of the main loop advances time to next minute. The current simulation time is used to check if time exceeds twice the configurable item: playback time. If any client interval messages needs to be sent, they are queued for sending. If any wind or terminal forecast data exists in the archive for this minute then it is loaded. If any flights depart in this minute then the flight is departed. If any flights arrive in this minute then the flight is unloaded from the system. If any flights depart in this minute plus the look-ahead time then it is loaded into the system. All departed aircraft positions are advanced to this minute's position report. If the current minute also is an interval minute, then if any reflectivity data exists in the archive for this interval it is loaded. The decision support data is created based on the current weather and flight state data. Weather models translate weather data into current/future capacity reductions using persistent forecasts. Capacity reductions are results of percentage of reflectivity sector coverage and METAR/TAF reports. The system reports and chart data are updated based off the latest decision support data: (1) ATGATE and ENROUTE interval loading. (2) airport loading and capacity, (3) METAR and TAF state data, and (4) flight statistics. If any messages were sent to a client, the server waits until it receives an acknowledgement back from the client. If the server does not hear back from a client by the configurable item: timeout then the server reports the error and continues.

Shutdown

At the end of an experiment or a user shutdown request, the server generate reports and terminates each client gracefully.

3.2.2 Communications Thread

The communications thread is responsible for managing the client connections and message exchange. The main loop in the communications thread is infinite. The server opens a connection ready to receive connection requests and messages. The server senses whether a connection is trying to be established, queued messages for sending, and whether a message was received for processing. This is illustrated below in Figure 5.

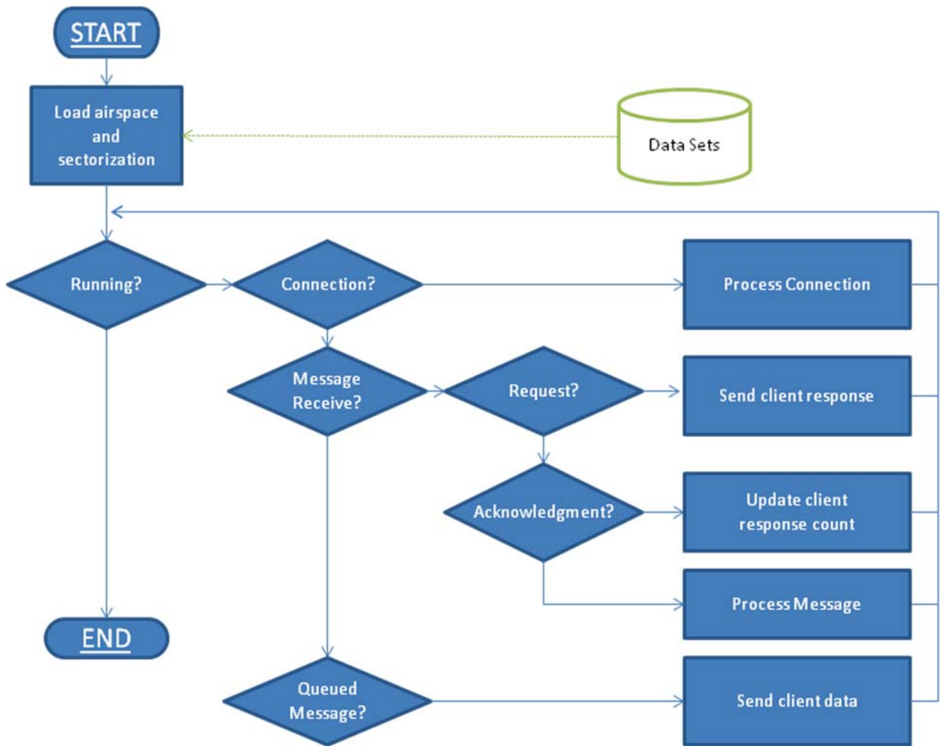


Figure 5 PNP Event Thread Design Flow

3.3 Interfaces

PNP is a client/based architecture. The PNP Server interfaces with clients with TCP/IP sockets. The clients may use a Java API interface library provided by PNP. The PNP API encapsulates all the messages and methods needed to communicate with the PNP Sever. The PNP Server has three main state relationships with clients as depicted in Figure 6.

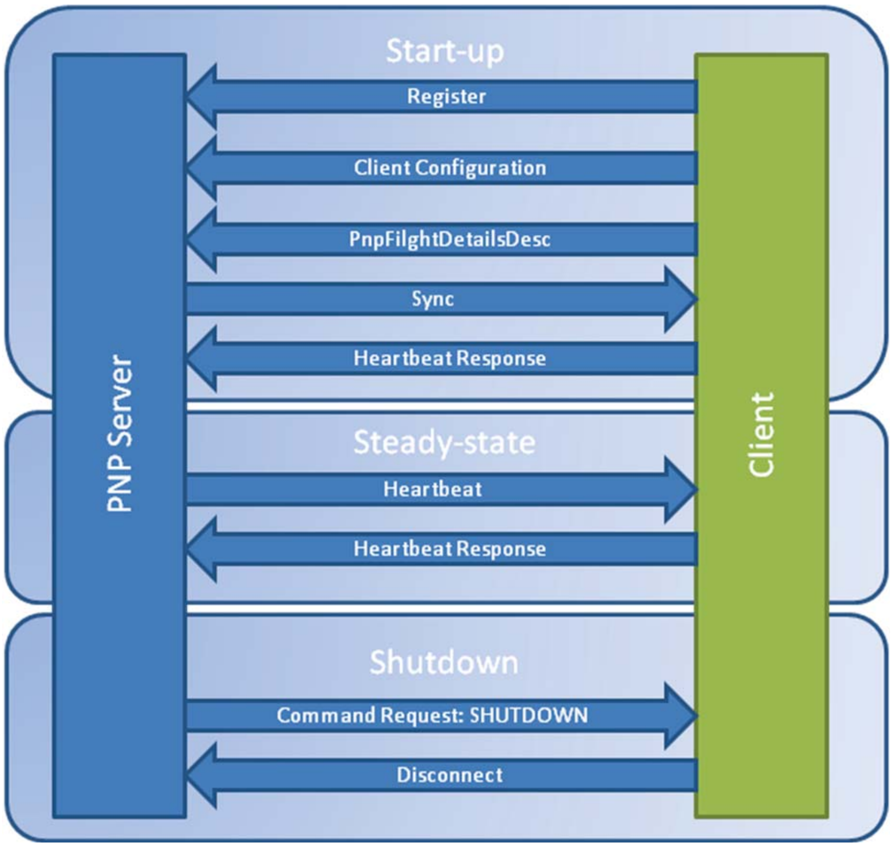


Figure 6 PNP Server Interface Diagram

Start-up

The Start-up state is when a client registers with the server and begins it initialization.

Steady -State

In the Steady-state phase the server communicates with clients in a synchronous cycle. As the server advances time in the simulation, it will periodically reach a time at which messages need to be sent to its clients. At each time point when the server needs to send messages, it will send the appropriate messages, send a heartbeat message to the clients, and wait for the receiving clients to respond. Only after all receiving clients have responded will the PNP server continue to advance time.

Shutdown

The Shutdown phase is either initiated by the server when all flights have landed in an experiment, twice the playback time is surpassed, or via a human in the loop request from the control GUI. A client may leave the simulation by sending a Disconnect message to the server. A client may connect to the simulation again via the Connect message.

4 Dynamic Airspace Client

This section describes the Dynamic Airspace Client (DAC) Computer Software Configuration Item

4.1 Functional Description

The Dynamic Airspace Client is responsible for publishing and maintaining the sector geometries, characteristics, capacities, and references.

4.2 Software Design

The Dynamic Airspace Client creates the sector grid that PNP uses to track flight congestion. The DAC parses sector or subsector geometries to create a sector grid. The sector grid is used to efficiently map positions to sectors. The grid is also used to determine the area reduction based on reflectivity. The DAC maps all the sector information to a single sector id.

The sector information includes:

- Sector name/identifier mapping
- Subsector name/identifier mapping
- Sector/Subsector mapping
- Monitor alert parameter
- Minimum altitude
- Maximum altitude

The DAC design is beyond the scope of this project and left blank intentionally.

4.3 Interfaces

The Dynamic Airspace Client interfaces with the PNP via the Java API library provided by PNP. For every message the client registers for a corresponding ApiListener must be created. The interface messages are illustrated below in Figure 7.

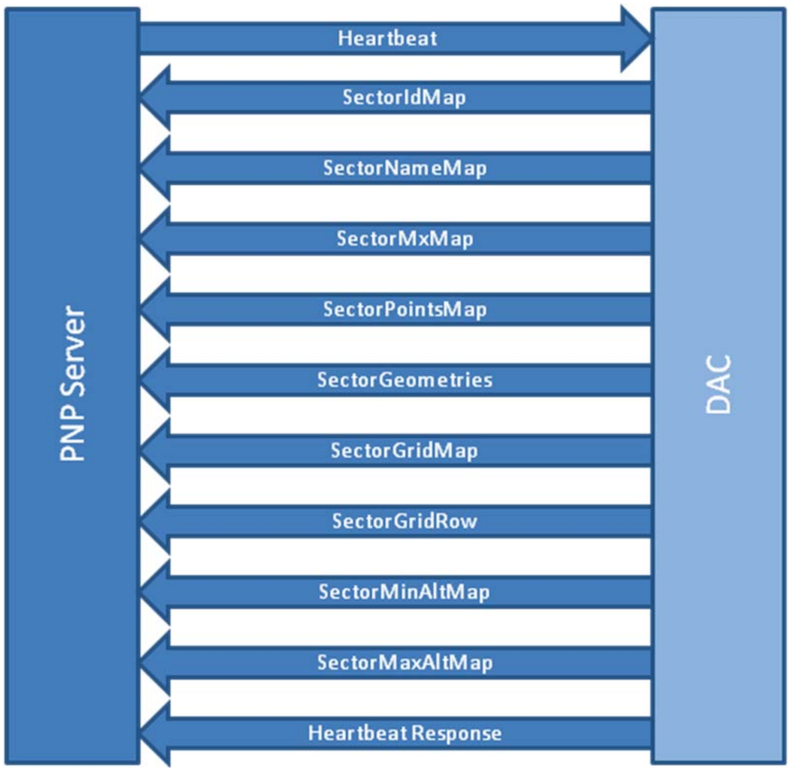


Figure 7 DAC Interface Diagram

5 Probabilistic Traffic Flow Management Client

This section describes the Probabilistic Traffic Flow Management (ProbTFM) Computer Software Configuration Item

5.1 Functional Description

The Probabilistic TFM client reduces airspace congestion by issuing delays and reroutes to the flights likely to contribute most to NAS congestion. By first assessing the likely effect of weather and traffic on NAS capacity, and then enforcing those constraints with delays and reroutes, ProbTFM can distribute expected traffic loads across the NAS, reducing airspace congestion while keeping delays and fuel burn to a minimum. Because its capacity predictions are based on historical data encoded in probability distributions, ProbTFM is able to effectively define constraints where demand is most likely to exceed capacity.

5.2 Software Design

The Probabilistic TFM client design is beyond the scope of this project and left blank intentionally.

5.3 Interfaces

The Probabilistic TFM client interfaces with the PNP via the Java API library provided by PNP. For every message the client registers for a corresponding ApiListener must be created. ProbTFM will create a ClientConfiguration and PnpFlightDetailsDescr message when coding the API interface. ProbTFM will enable predicted positions and the sector schedule in the PnpFlightDetailsDescr object used at instantiation of a client API object. The API automatically sends this message to the server after the client configuration message. The interface messages are illustrated below in Figure 8.

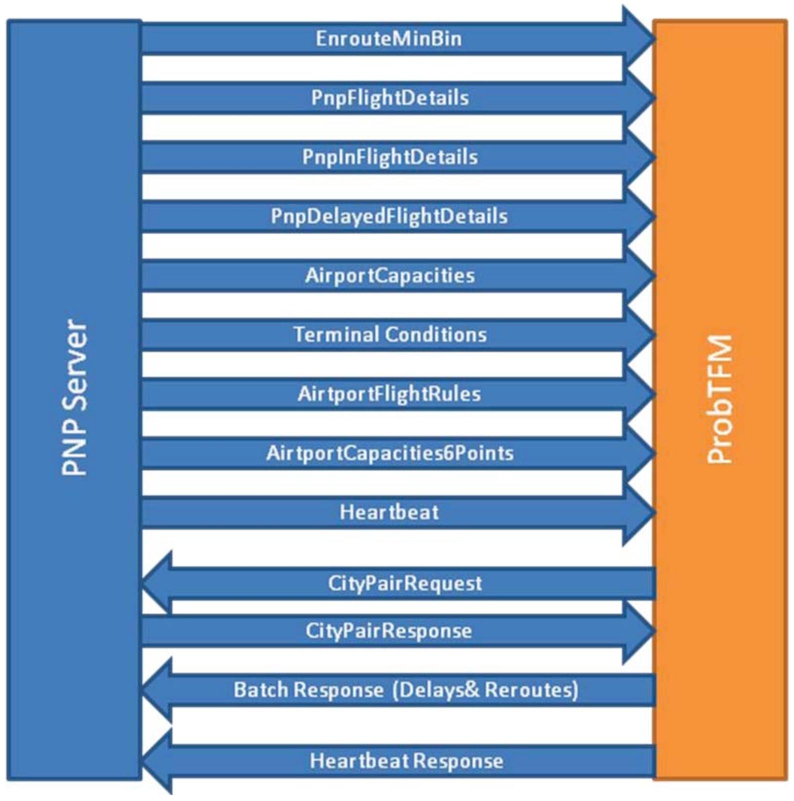


Figure 8 ProbTFM Interface Diagram

6 Separation Assurance Framework Client

This section describes the Separation Assurance Framework (SA) Computer Software Configuration Item

6.1 Functional Description

The Separation Assurance Framework client will provide a framework for developers to select conflict detection and resolution (CDR) algorithms. A Java interface will be created for the conflict detection and resolution implementations. The Java factory creation pattern will be used to support different CDR solutions.

6.2 Software Design

The separation assurance client will be a standalone program [Req#1]. The SA will load user specified inputs using the windows de factor standard for configurations files [Req#2]. The SA will stream output to standard output and errors to standard error [Req#3-4].

The SA will use PNP's SimObjects API for communications [Req#5]. The SimObjects API provides a client with mature methods to connect and communicate with the PNP server. The SA will send a connect message to the PNP server [Req#6]. The connect message establishes a connection between the PNP server and a client. The SA will send a client configuration message to the PNP server [Req#8]. The client configuration message defines the client and its runtime parameters (ownship aircraft id/distance or time). The SA will register for the flight plans for all flights [Req#9]. The SA will establish communications with the server and wait for messages.

When messages are received they are processed until a heartbeat is received. The heartbeat is the trigger that all interval requested messages have been sent from the server. The client will process all the flights into a flight manager. The flight manager create all the predicted trajectories. The flight ids with trajectories will be passed to the conflict detection algorithm. The conflict detection algorithm will process the trajectories based on a user-specified conflict detection look ahead time frame [Req#10] and return conflict data. The SA will send LOS/GOS flight data to the PNP server for visualization on the PVD [Req#18]. The conflict data will be passed to the conflict resolution algorithm. The conflict resolution algorithm will attempt different strategies to resolve conflicts. The conflicts resolution will be passed back to the flight manager. The resolutions will be used to create an in-flight reroute based on the flight modification. This in-flight reroute will be sent back to the server to be implemented [Req#30]. Once conflict resolution has attempted to resolve all the conflicts or a successful flight plan modification has been found, then heartbeat response is sent back to the server [Req#7]. The client waits for the next set of interval requested messages. The SA will send a disconnect message to the PNP server when shutting down before the end of a simulation [Req#31]. This flow is illustrated below in Figure 9.

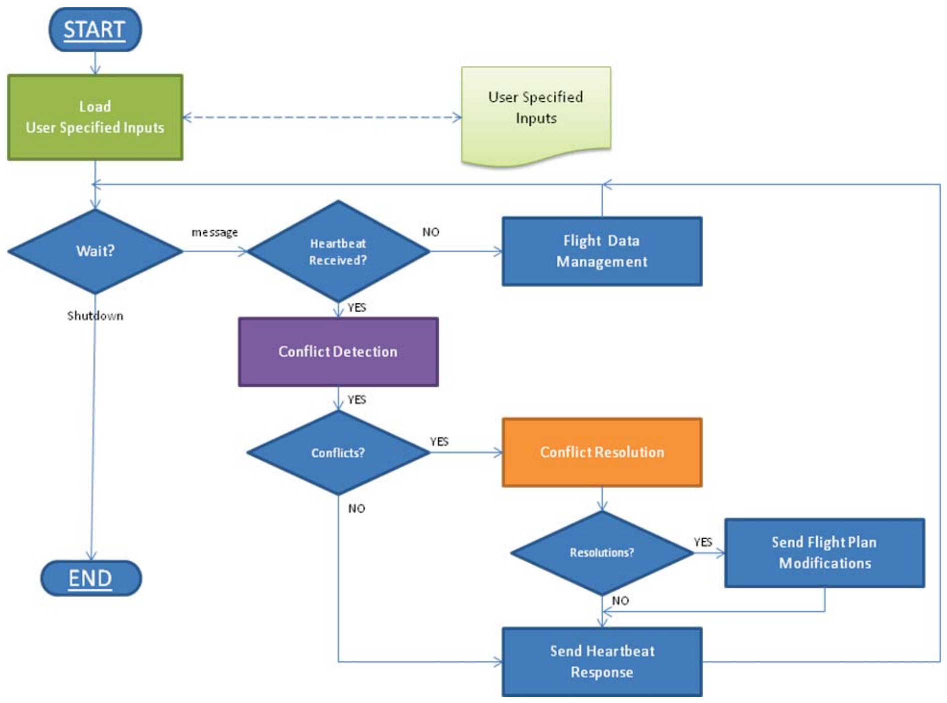


Figure 9 Separation Assurance Design Diagram

6.2.1 Separation Assurance Detailed Design

This section describes the detailed design of the separation assurance client.

Initialization

The SA will load the user specified configuration to establish the (1) trajectory resolution, (2) conflict detection look ahead time, (3) conflict detection minimum horizontal separation distance, (4) conflict detection minimum vertical separation distance, and (5) conflict resolution retry criteria. The SA will establish communications with the server. The SA will register its interval request messages with the server.

Steady-State

The SA will constantly receive and process messages from the server. When flight data is received the flight data manager will check for updates. If the flight plan has not changed then the flight and run the trajectory predictor on new or updated flights store them in the flight data manager. When a heartbeat is received, then the SA will process the trajectories in the flight data manager by performing conflict detection and resolution. The resulting conflict resolutions will be sent to the server as flight modifications. Once all CDR is attempted for all flights then a heartbeat response will be sent back to the server to advance the simulation.

Shutdown

The SA will shutdown communications and reports statistics that have been accumulated throughout the experiment.

6.3 Interfaces

The separation assurance (SA) client interfaces with the PNP via the Java API library provided by PNP. For every message the client registers for a corresponding ApiListener must be created. The SA will create a ClientConfiguration and PnpFlightDetailsDescr message when coding the API interface. The separation assurance client will register for PnpInFlightDetails. The interface messages are illustrated below in Figure 10.

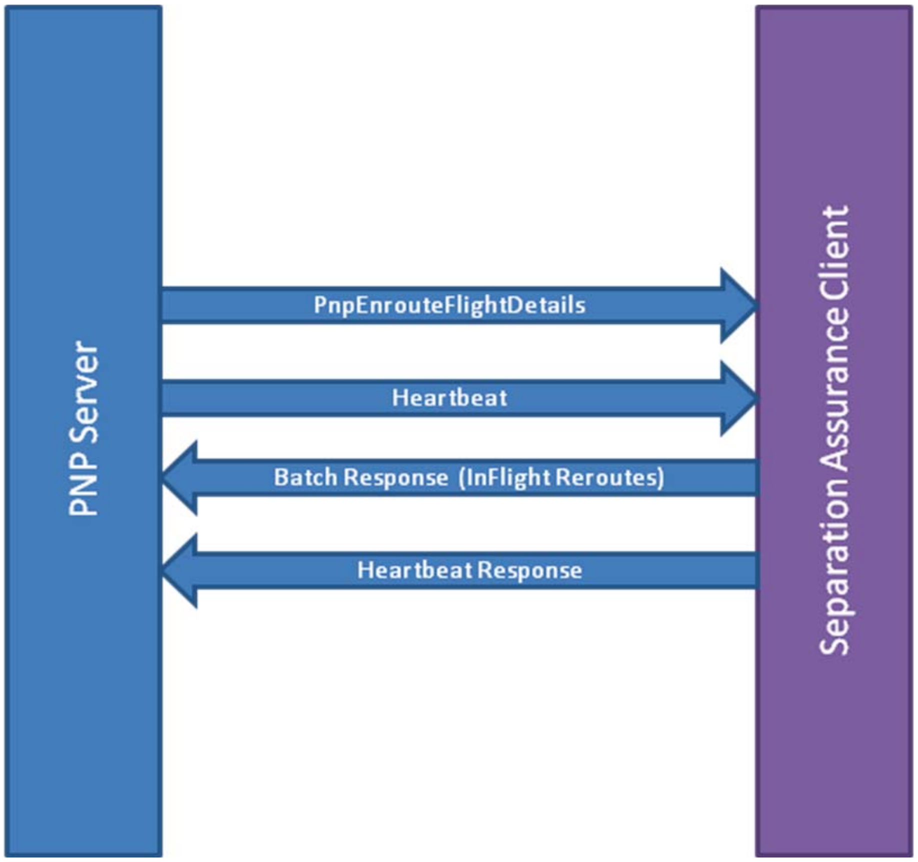

PNP Server
Separation Assurance Client
PnpEnrouteFlightDetails
Heartbeat
Batch Response (InFlight Reroutes)
Heartbeat Response

Figure 10 Separation Assurance Interface Diagram

The ClientConfiguration message can be adapted to request all enroute flights, fleet based enroute flights, ownship enroute flights, and geographical bounded enroute flights. To requests all enroute flights, set the *ownship* flag in the adaptation file to "none". To enable the fleet based enroute flights, set the *ownship* flag in the adaptation file to a fleet identifier i.e. "UPS". To enable the *ownship* enroute flights, set the *ownship* flag in the adaptation file to an aircraft identifier i.e. "AAL1076". To enable the geographical bounded, set the ownship flag in the adaptation file to "polygon" and set the *separationassurancepolygon* flag to a set of decimal latitude/longitudes i.e. "35.1/-101.7,35.1/-91.3,31.1/-91.3,31.1/-101.7"

6.3.1 Pseudo Code

This section provides the separation assurance interface. The separation assurance interface consists of several methods. The *process* method uses the conflict detection and resolution instances to set reroutes, loss of separation list, conflict list and gather statistics. The *clear* method initializes all state data. The *getReroutes*, *getLosList*, and *getConflictList* methods provide assessors to their associated state data.

```
public interface SeparationAssuranceInterface
{
   public void process(ConflictDetectionInterface conflictDetection,
                  ConflictResolutionInterface conflictResolution,
                  long ts);

   public void clear();

   public Map<Integer, ReroutePlan> getReroutes();

   public LosList getLosList();

   public ConflictList getConflictList();

   void createSummaryReport(String experiment);
}
```

Two separation assurance implementations are available in the separation assurance factory. To enable the ownship separation assurance implementation, set the *separationassuranceimplementation* flag in the adaptation file to “ownship”. To enable the surveillance separation assurance implementation, set the *separationassuranceimplementation* flag in the adaptation file to “surveillance”.

```
public class SeparationAssuranceFactory
{
   public static SeparationAssuranceInterface get(String criteria)
   {
      if (criteria.toLowerCase().equals("ownship"))
      {
         return new OwnshipSeparationAssurance();
      }
      else if (criteria.toLowerCase().equals("surveillance"))
      {
         return new SurveillanceSeparationAssurance();
      }
      else
      {
         return null;
```

```
        }
    }
}
```

7 Conflict Detection

This section describes the Conflict Detection Algorithm.

7.1 Functional Description

The Conflict Detection Algorithm will perform conflict detection based on a ownship centric perspective.

7.2 Software Design

The Conflict Detection (CD) will copy all the flight data from the Flight Data Manager. The CD will be ownship centric. If a trial resolution exists, the resolution will be updated for the ownship flight. The CD will load the user specified inputs: (1) conflict detection look ahead time, (2) conflict detection minimum horizontal separation distance, and (3) conflict detection minimum vertical separation distance. The CD will analyze trajectories for all loss-of-separation events based on the current time to the conflict detection look ahead time. The CD will compare the ownship position to every flight in the data set, starting from the current position to last position report defined by the conflict detection look ahead time. If at any point in time an aircraft losses separation with the ownship then the occurrence is saved in the conflict recorder. The CD will post conflict data from the conflict recorder. Once all flight have been analyzed the conflict data will be returned to the calling process.

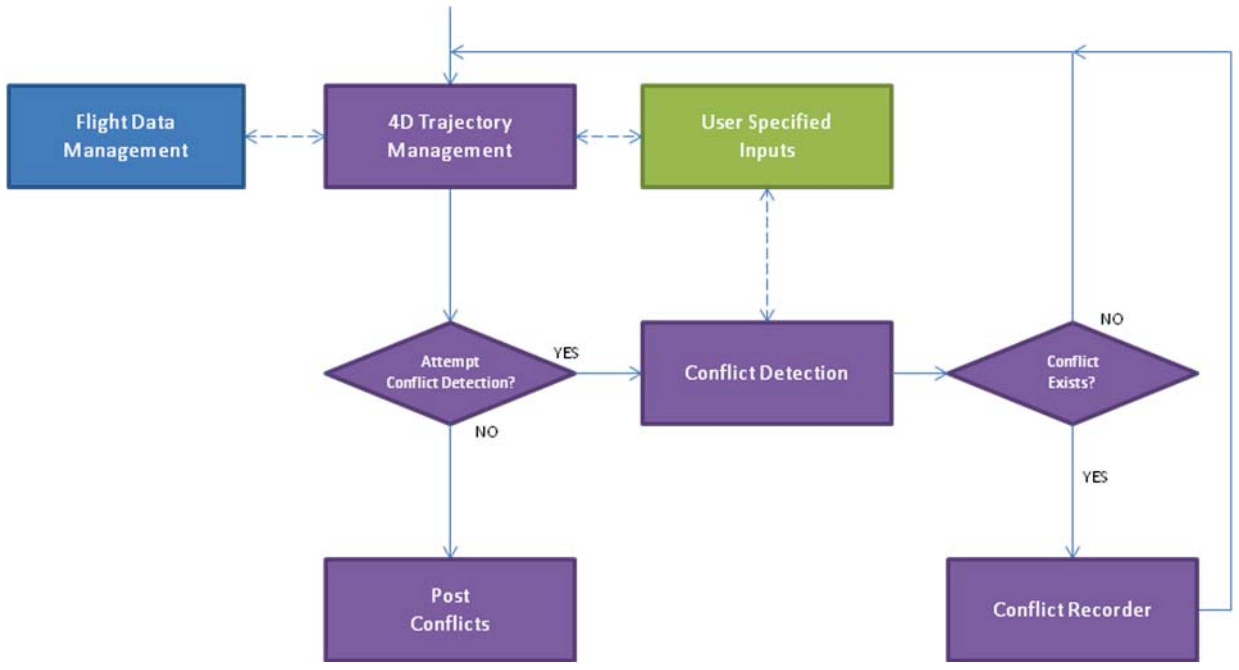


Figure 11 Conflict Detection Design Diagram

7.3 Application Programming Interface

This section describes the Application Programming Interface for conflict detection algorithms [Req#11]. The Conflict Detection Application Programming Interface (CD-API) will provide methods to access and mutate the (1) conflict detection look ahead time frame [Req#12], (2) minimum vertical separation distance [Req#13], and (3) minimum horizontal separation distance [Req#14]. The CD-API will provide a method to access the loss of separation (LOS) pairs consisting of time, waypoint, and flight IDs [Req#15]. The CD-API will provide a method to access the gain of separation (GOS) pairs consisting of time, waypoint, and flight IDs [Req#16]. The CD-API will provide a methods to access and mutate 4D trajectory position data array identified by a flight id [Req#17]. The SA will perform conflict detection of flights based on the user-specified conflict detection look ahead time frame.

7.3.1 Pseudo Code

This section provides the conflict detection interface. The conflict detection interface consists of one method: detectConflicts. A conflict resolution detection would write a detectConflicts method to identify conflict pairs.

```
public interface ConflictDetectionInterface
{

   public List<DetectedConflict> detectConflicts(int ownshipFlightId,
                                    List<FlightDetails> flights,
                                    long ts);

}
```

Two conflict detection implementations are available in the conflict detection factory. To enable the Sensis conflict detection implementation, set the *conflictdetectionimplementation* flag in the adaptation file to “sensis”. To enable the Stratway conflict detection implementation, set the *conflictdetectionimplementation* flag in the adaptation file to “stratway”.

```
public class ConflictDetectionFactory
{
   public static ConflictDetectionInterface get(String criteria)
   {
      if (criteria.toLowerCase().equals("stratway"))
      {
         return new StratwayConflictDetection();
      }
      else
      {
         return new SensisConflictDetection();
      }
```

```
    }
}
```

The DectectedConflict class consists of a conflict pair ids, seconds to loss of separation, seconds to point of closest approach, and seconds to gain of separation.

```
public class DetectedConflict
{
    private int m_MyFlightId;

    private int m_OtherFlightId;

    private int m_SecondsToLossOfSeparation;

    private int m_SecondsToClosestApproach;

    private int m_SecondsToGainOfSeparation;

    public DetectedConflict(int myFlightId,
int otherFlightId,
int secondsToLos,
int secondsToClosestApproach,
int secondsToGainOfSeparation)
    {
        m_MyFlightId = myFlightId;
        m_OtherFlightId = otherFlightId;
        m_SecondsToLossOfSeparation = secondsToLos;
        m_SecondsToClosestApproach = secondsToClosestApproach;
        m_SecondsToGainOfSeparation = secondsToGainOfSeparation;
    }
```

8 Conflict Resolution

This section describes the Conflict Resolution Algorithm.

8.1 Functional Description

The Conflict Resolution will attempt to create flight modifications for a specified aircraft through speed, altitude, or turn maneuvers.

8.2 Software Design

The Conflict Resolution (CR) loads all the conflicts from the CD. all the flight data from the Flight Data Manager. The CR will load the user specified input: conflict resolution parameters. The CR will attempt speed, altitude, and turn maneuvers to reduce the number of conflicts, the maneuver order will be configurable. If an resolution is found that reduces conflicts the flight modifications will be passed to the server as a InFlightReroute in the BatchResponse [Req#19,21].

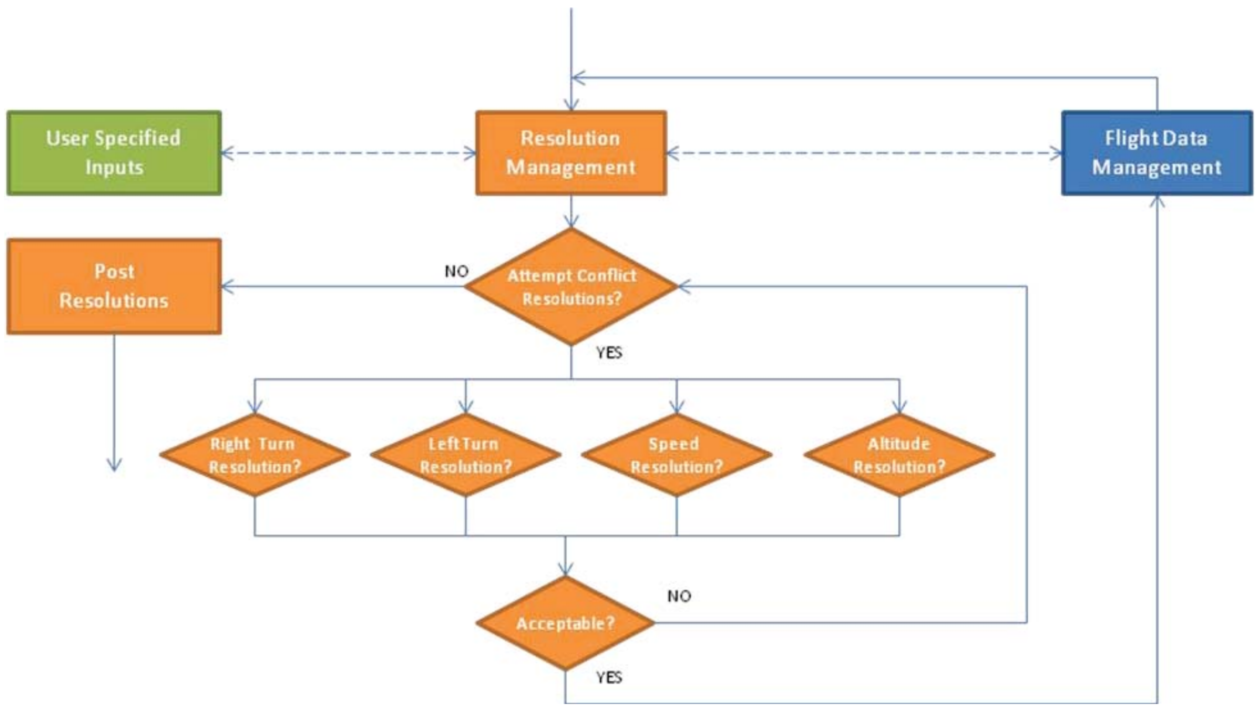


Figure 12 Conflict Resolution Design Diagram

Right Turn Maneuver

The CR will first attempt an right turn maneuver (See EDD 4.3.3). If the conflict is avoided the right turn maneuver is accepted and the flight modification is committed to the flight data manager [Req#22]. If the conflict persists the right turn maneuver is rejected and the flight modification is canceled.

The Turn maneuver resolution is created by first adding all the flown waypoints. Then the current position is added followed by a 90º turn based on the minimum horizontal separation distance. The next position added is based on the distance to the point of closest approach followed by a 90º turn back to the original trajectory.

Left Turn Maneuver

The CR will attempt a left turn maneuver (See EDD 4.3.3) if the conflict still exists. If the conflict is avoided the left turn maneuver is accepted and the flight modification is committed to the flight data manager. If the conflict persists the left turn maneuver is rejected and the flight modification is canceled.

Speed Reduction Maneuver of Trailing Aircraft

The CR will attempt a speed reduction maneuver of the trailing aircraft (See EDD Section 4.3.1) if the conflict still exists. If the conflict is avoided the speed maneuver is accepted and the flight modification is committed to the flight data manager [Req#24]. If the conflict persists the speed maneuver is rejected and the flight modification is canceled.

The speed reduction maneuver is based on the two aircraft with a respective heading angles to be within 45^o at the point of closest approach. This means that only the trailing aircraft may be maneuvered in the speed reduction maneuver. We restrict our speed reduction maneuver to speed reductions during the cruise phase of flight. We restrict stacking maneuvers based on an adaptable maneuver timer. We estimate the required speed reduction (in knots) as: $\Delta V = \dfrac{\Delta P}{\Delta T \cos \Delta\psi}$

where,

ΔP = minimum horizontal separation – PCA horizontal separation (nmi),
ΔT = PCA time – current time (hours).

We round ΔV up to the next multiple of 5, and we reject the speed reduction maneuver if it is greater than 10% of the cruise speed.

Altitude Descent Maneuver

The CR will attempt an altitude maneuver (See EDD 4.3.1) if the conflict still exists. The altitude maneuver will be executed only if the ownship is the lower aircraft in the conflict. If the conflict is avoided the altitude maneuver is accepted and the flight modification is committed to the flight data manager [Req#23]. If the conflict persists the altitude maneuver is rejected and the flight modification is canceled.

The altitude change maneuver is based on the altitude delta of two aircraft. The altitude reduction (in feet) is equal to the minimum vertical separation minus the altitude separation at the PCA, plus a buffer of 100 ft:

$$\Delta h = MVS - \left(h_A - h_B\right) + 100$$

where,

MVS = minimum vertical separation (ft),

h_A = altitude of Aircraft A at PCA (ft).

We round *Δh* up to the next multiple of 100. We restrict our altitude change maneuver to altitude reductions during the cruise phase of flight. We restrict stacking maneuvers based on an adaptable maneuver timer.

8.3 Application Programming Interface

This section describes the Application Programming Interface for conflict resolutions algorithms [Req#20]. The Conflict Resolution Application Programming Interface (CR-API) will provide methods to perform turn, speed, and altitude maneuvers [Req#25-27]. The CR-API will accept the LOS and GOS event data output from the CD-API [Req#28]. The CD-API will provide a methods to access and mutate flight plans and their associated trajectory data [Req#29].

8.3.1 Pseudo Code

This section provides the conflict resolution interface source code. The conflict resolution interface consists of one method: computeConflictResolutionOptions. A conflict resolution implementation would write a computeConflictResolutionOptions method to create resolutions to resolve the conflict between the ownship and nextship.

```
public interface ConflictResolutionInterface
{
   public List<Resolution> computeConflictResolutionOptions(FlightDetails ownship,
                                                            FlightDetails nextship,
                                                            DetectedConflict conflict,
                                                            long ts);
}
```

Two conflict resolution implementations are available in the conflict resolution factory. To enable the Sensis conflict resolution implementation, set the *conflictresolutionimplementation* flag in the adaptation file to “sensis”. To enable the Stratway conflict resolution implementation, set the *conflictresolutionimplementation* flag in the adaptation file to “stratway”.

```
public class ConflictResolutionFactory

{

   public static ConflictResolutionInterface get(String criteria)
   {
        if (criteria.toLowerCase().equals("stratway"))
        {
            return new StratwayConflictResolution();
        }
        else
        {
            return new SensisConflictResolution();
        }
    }
}
```

The resolution class consists of a type, trajectory, and plan. The type describes the resolution (right turn, left turn, speed control, or altitude change). The trajectory is used to test the resolution in the conflict detection evaluation. The plan is sent to the PNP server to implement the resolution in the system.

```
public class Resolution
{
    private RESOLUTION_TYPE m_Type;

    private Traj3D[] m_Trajectory = null;

    private ReroutePlan m_Plan = new ReroutePlan(10);

    public Resolution(RESOLUTION_TYPE type)
    {
        m_Type = type;
    }
}
```

The reroute plan consists of a priority, flight plan, list of flight plan waypoint names, list of flight plan waypoints, or a list of maneuvers. The priority is used to determine if a resolution is implemented in the system if there is a collision with another reroute plan for this flight during this interval. The flight plan, waypoint names, and waypoints describe the reroute to be implemented by the system. The maneuvers describe the maneuver to be implemented by the system.

```
public class ReroutePlan extends Message implements java.io.Serializable
{
    private int m_Priority = 9999;

    private String m_FlightPlan = "";

    private ArrayList<String> m_FlightPlanWaypointNames = new ArrayList<String>();

    private ArrayList<Traj2D> m_FlightPlanWaypoints = new ArrayList<Traj2D>();
    private ArrayList<Maneuver> m_Maneuvers = new ArrayList<Maneuver>();

    public ReroutePlan(int priority)
    {
        m_Priority = priority;
    }

}
```

Attachment 1

Keyword	Description	Default Values
serverrequestinterval	The time interval in which PNP server will send messages to the SA client	1
minimumhorizontalseparation	The minimum horizontal distance in nautical miles used to determine loss of separation	10
minimumverticalseparation	The minimum horizontal distance in feet used to determine loss of separation	1000
conflictdetectionimplementation	The conflict detection implementation used to determine loss of separation	sensis | stratway
conflictresolutionimplementation	The conflict resolution implementation used to determine resolutions	sensis | stratway
separationassuranceimplementation	The separation assurance implementation used to separate air traffic	sensis | stratway
separationassurancedistance	The distance in which two aircraft are in proximity	10
separationassurancetime	The look ahead time in which two aircraft are in proximity	12
minimumaltitudebound	The minimum altitude in which an aircraft is considered for separation	18000
trajectorytimestepinseconds	The time step in which trajectories are created	10
maneuvertimer	The time in minutes in which an aircraft cannot perform another maneuver	10
outputpath	The directories in which to output reports	data\\reports

14809-04

PNP Required Time of Arrival Requirements and Engineering Design

George Hunter

Prepared for:

National Aeronautics and Space Administration

Langley Research Center

Hampton, VA 23681

Under:

NASA BOA: NNL08AA17B

July, 2011

Table of Contents

1 Introduction

This document describes software enhancements required to support NASA air traffic control experiments to evaluate advanced national airspace system (NAS) concepts. Section 2 describes the simulation environment and provides an overview of the required time of arrival (RTA) concept.

Section 3 describes the requirements for the software. These are divided into PNP architecture, client, and utility categories. Sections 4 and 5 describe design considerations and future enhancements. Section 6 lists the tests to be performed.

2 Background

This section discusses simulation tool to be used and provides an overview of the required time of arrival (RTA) concept.

2.1 Probabilistic NAS Platform

In this engineering design we use the Probabilistic NAS Platform (PNP) to implement and traffic flow management (TFM), separation assurance (SA), traffic spacing and regional rerouting to avoid heavy weather and congestion concepts.

We developed PNP [1-11], are actively maintaining it, and actively use it for NASA, Joint Planning and Development Office (JPDO), and Federal Aviation Administration (FAA) projects [12-21]. Figure 1 illustrates PNP's client-server architecture.

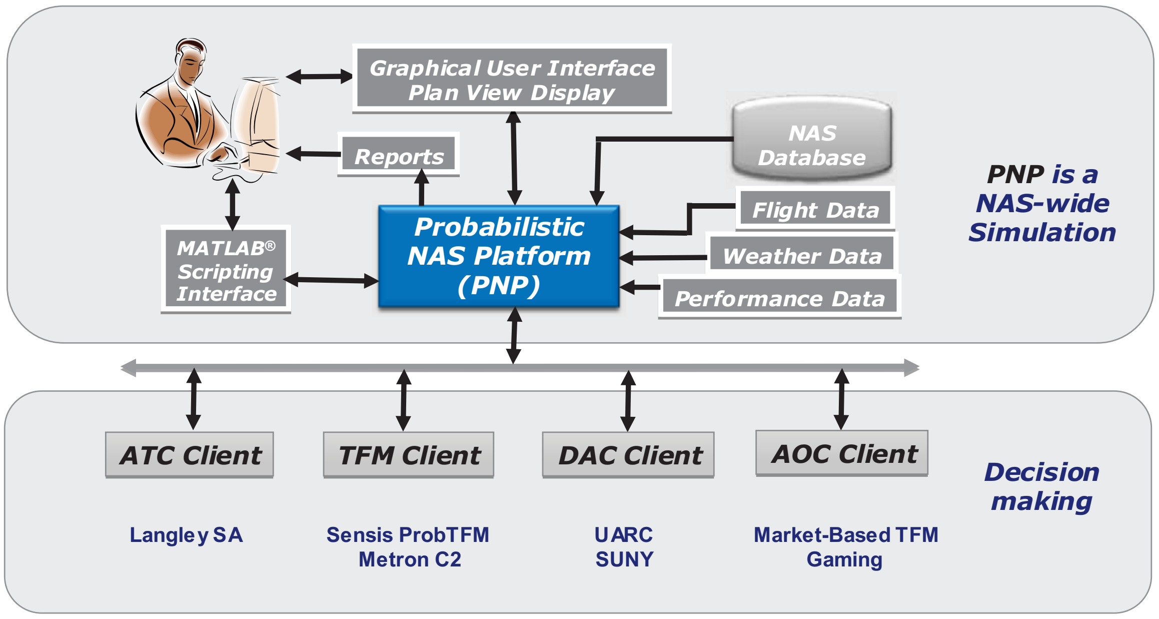
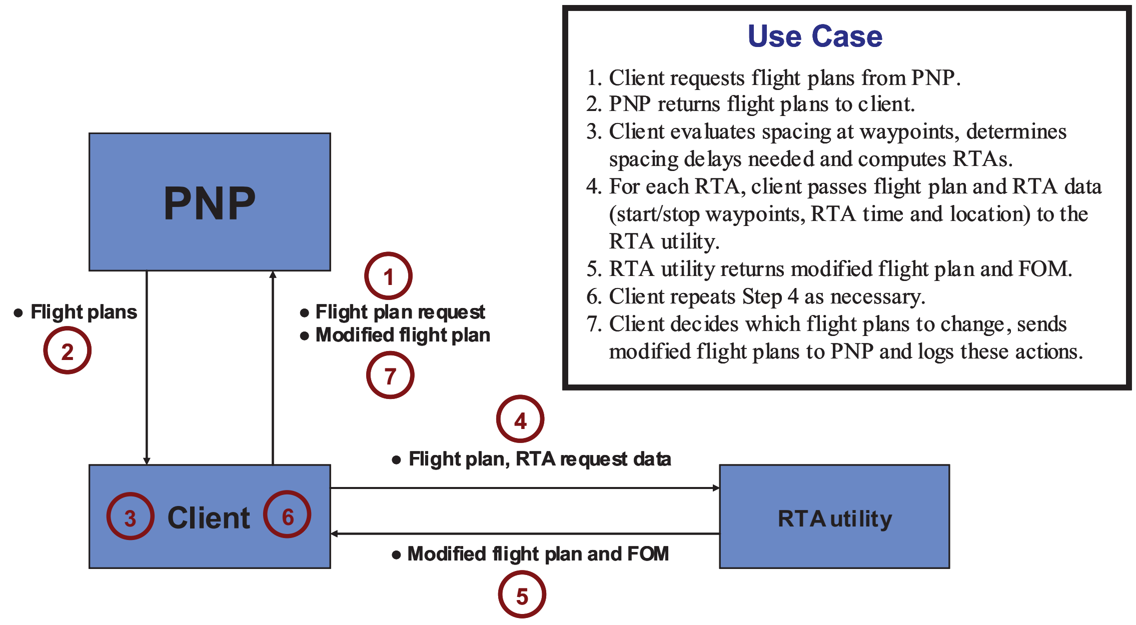


Figure 1 Probabilistic NAS Platform client-server architecture.

2.2 Overview of the required time of arrival concept

A required time of arrival is a desired crossing time for a flight which is attached to a geographically-fixed position along the flight plan route. An RTA is implemented by first determining the RTA value, and then modifying the flight plan to meet the RTA value. The flight plan can be modified in several different ways. Our first priority is to use modifications to the horizontal route to meet the RTA. Second, as a stretch goal we use modifications to the descent segment, including both speed control and the top of descent (TOD) control. Third, a future enhancement (or stretch goal) is to use speed control.

Within the PNP architecture we implement an RTA utility that clients can use. Clients make an RTA request and the RTA utility returns the modified flight plan along with a figure of merit indicating the quality of the RTA solution. Figure 2 shows the high level architecture and use case.



Use Case

1. Client requests flight plans from PNP.
2. PNP returns flight plans to client.
3. Client evaluates spacing at waypoints, determines spacing delays needed and computes RTAs.
4. For each RTA, client passes flight plan and RTA data (start/stop waypoints, RTA time and location) to the RTA utility.
5. RTA utility returns modified flight plan and FOM.
6. Client repeats Step 4 as necessary.
7. Client decides which flight plans to change, sends modified flight plans to PNP and logs these actions.

Figure 2 The PNP RTA high level architecture and use case.

Any client can use the RTA utility to derive flight plan modifications for any flight. Furthermore, there is no limit to the number of RTAs that can be implemented per flight. But the RTA utility handles only one request per call, so clients that use the RTA utility to implement multiple RTAs for a flight must make multiple calls to the RTA utility. In other words, multiple RTAs are implemented serially, one at a time.

Finally, there is no restriction on the RTA value itself. Crossing times may imply positive or negative delays for the flight. Of course negative delays, which require the flight to reach the waypoint earlier than planned, may be more difficult to achieve.

3 Requirements

This section lists the software requirements, divided into the PNP architecture, client, and utility categories.

3.1 PNP architecture

1. RTAs shall be specifiable for any and all flights in the simulation.
2. Each RTA shall be assigned to a flight plan waypoint or a range value.
3. The client shall have available to it several flight plan and trajectory functions for convenience. These include insertion of a waypoint into a flight plan, calculation of the range value of a waypoint, and conversion between range to the destination airport and range from the origin airport.
4. Stretch goal: The PNP flight plan architecture shall include a change history which indicates all changes that have been made to the flight plan during the simulation run, the simulation time of the change, the type of change and the reason for the change.

3.2 PNP client

5. All RTA actions shall be logged.

3.3 PNP RTA utility

6. The RTA utility shall accept as input a flight plan, the RTA crossing time, and the RTA geographical location. The RTA utility shall also accept as input an indication of which control strategy to use and associated input data. For turn control strategy, for instance, the input data may include two geographical locations indicating the earliest and latest execution of the RTA maneuver (i.e., the “maneuver boundaries”).
7. The RTA utility shall accept the geographical locations (for the maneuver boundaries and the RTA) in terms of (i) a flight plan waypoint, (ii) a range value from the origin airport, or (iii) a range value to the destination airport. The range values are specified in terms of distance along the flight’s ground track.
8. The RTA utility shall return as output a modified flight plan that attempts to meet the RTA.
9. The RTA utility shall return a figure of merit (FOM) of its RTA solution, indicating how accurately the RTA will be met, in seconds. An early arrival (i.e., the modified flight plan arrives at the specified geographical location before the RTA) shall be indicated using a negative FOM. A late arrival shall be indicated using a positive FOM. In other words, the FOM shall be computed as the planned crossing time – RTA.

10. The RTA utility shall implement multiple control strategies to meet the RTA. Only one control strategy may be used by the client per call.
11. The RTA utility shall implement a general turn strategy that modifies the flight plan horizontal route to meet the RTA.
12. Stretch goal: The RTA utility shall implement a descent segment strategy that modifies the descent speed profile to meet the RTA.
13. Stretch goal: The RTA utility shall implement a descent segment strategy that modifies the top of descent to meet the RTA.

4 Design considerations

There are no design considerations at this time.

5 Future enhancements

This section describes enhancements that are outside the scope of this effort but may be implemented in the future.

5.1 PNP architecture

1. The PNP flight plan architecture shall include a change history which indicates all changes that have been made to the flight plan during the simulation run, the simulation time of the change, the type of change and the reason for the change.

5.2 PNP client

There are no client enhancements at this time.

5.3 PNP RTA utility

2. The RTA utility shall implement a general speed strategy that modifies the airspeed to meet the RTA. This strategy shall include both airspeed increase and decrease strategies.
3. The RTA utility shall support the use of forecasted trajectories that do not match the actual trajectory that PNP will simulate. For instance, the forecasted trajectory may be based on a forecasted wind field rather than the actual wind field.

6 Testing

Each of the requirements in Section 3 will be verified either by demonstration, analysis or inspection.

7 Conclusions

This document describes software enhancements required to support NASA air traffic control experiments to evaluate advanced RTA concepts.

References

1. George Hunter, "Testing and Validation of NextGen Simulators," AIAA Modeling and Simulation Conference, Chicago, IL, August 2009.

2. George Hunter, Kris Ramamoorthy, "Integration of terminal area probabilistic meteorological forecasts in NAS-wide traffic flow management decision making," 13th Conference on Aviation, Range and Aerospace Meteorology, New Orleans, LA, January, 2008.

3. Kris Ramamoorthy, George Hunter, "The Integration of Meteorological Data in Air Traffic Management: Requirements and Sensitivities," 46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January, 2008.

4. George Hunter, Ben Boisvert, Kris Ramamoorthy, "Advanced Traffic Flow Management Simulation Experiments and Validation," 2007 Winter Simulation Conference, Washington, DC, December, 2007.

5. Kris Ramamoorthy, George Hunter, "Evaluation of National Airspace System Performance Improvement With Four Dimensional Trajectories," AIAA Digital Avionics Systems Conference (DASC), Dallas, TX, October, 2007.

6. George Hunter, Ben Boisvert, Kris Ramamoorthy, "Use of automated aviation weather forecasts in future NAS," The 87th American Meteorological Society Annual Meeting, San Antonio, TX, January, 2007.

7. Kris Ramamoorthy, George Hunter, "Probabilistic Traffic Flow Management in the Presence of Inclement Weather and Other System Uncertainties," INFORMS Annual Meeting, Pittsburgh, PA, November, 2006.

8. Kris Ramamoorthy, Ben Boisvert, George Hunter, "A Real-Time Probabilistic TFM Evaluation Tool," AIAA Digital Avionics Systems Conference (DASC), Portland, OR, October, 2006.

9. Kris Ramamoorthy, George Hunter, "A Trajectory-Based Probabilistic TFM Evaluation Tool and Experiment," Integrated Communications, Navigation and Surveillance Conference (ICNS), Baltimore, MD, May, 2006.

10. Kris Ramamoorthy, George Hunter, "Avionics and National Airspace Architecture Strategies for Future Demand Scenarios in Inclement Weather," AIAA Digital Avionics Systems Conference (DASC), Crystal City, VA, October, 2005.

11. George Hunter, Kris Ramamoorthy, Joe Post, "Evaluation of the Future National Airspace System in Heavy Weather," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Arlington, VA, September 2005.

12. George Hunter, "Probabilistic Forecasting of Airport Capacity," AIAA Digital Avionics Systems Conference (DASC), Salt Lake City, UT, October, 2010.

13. Poornima Balakrishna, George Hunter, "Preliminary NextGen Collaborative Air Traffic Management Analysis," AIAA Digital Avionics Systems Conference (DASC), Salt Lake City, UT, October, 2010.

14. Huina Gao, George Hunter, Frank Berardino, Karla Hoffman, "Development and Evaluation of Market-Based Traffic Flow Management Concepts," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Dallas, TX, September 2010.

15. Huina Gao, George Hunter, "Evaluation of User Gaming Strategies in the Future National Airspace System," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Dallas, TX, September 2010.

16. George Hunter, "Meta Simulation Results for Simultaneous Dynamic Resectorization and Traffic Flow Management," AIAA Digital Avionics Systems Conference (DASC), Orlando, FL, October, 2009.

17. Huina Gao, George Hunter, "Future NAS-Wide User Gaming Preliminary Investigation," AIAA Digital Avionics Systems Conference (DASC), Orlando, FL, October, 2009.

18. George Hunter, "Preliminary Assessment of Interactions Between Traffic Flow Management and Dynamic Airspace Configuration Capabilities," AIAA Digital Avionics Systems Conference (DASC), St. Paul, MN, October, 2008.

19. George Hunter, "Toward an Economic Model to Incentivize Voluntary Optimization of NAS Traffic Flow," AIAA ATIO Conference, Anchorage, AK, September, 2008.

20. George Hunter, "Sensitivity of the National Airspace System Performance to Weather Forecast Accuracy," Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, May, 2008.

21. Kris Ramamoorthy, Ben Boisvert, George Hunter, "Sensitivity of Advanced Traffic Flow Management to Different Weather Scenarios," Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, May, 2007.

14809-06

PNP RTA Conformance Monitor Utility Requirements and Engineering Design

George Hunter

Prepared for:

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681

Under:

NASA BOA: NNL08AA17B

October, 2011

Table of Contents

1 Introduction

This document describes software enhancements required to support NASA air traffic control experiments to evaluate advanced national airspace system (NAS) concepts. Section 2 describes the simulation environment and provides an overview of the required time of arrival (RTA) conformance monitor concept.

Section 3 describes the requirements for the software. These are divided into PNP architecture, utility and client categories. Sections 4 and 5 describe design considerations and future enhancements. Section 6 lists the tests to be performed.

RTA conformance monitoring is part of a larger context, including these steps:

1. Specify an RTA for a flight
2. Monitor the flight's RTA conformance
3. Attempt to use maneuvers to reestablish RTA when out of conformance
4. Reformulate RTA when attempts to reestablish RTA fail

We define Step 1 as a TFM function. The TFM client analyzes multiple flights in a traffic flow, and determines how delay should be distributed among the flights. RTA assignments are a result of this analysis.

We assign Steps 2 and 3 to the RTA Conformance Monitor utility. And we define Step 4 as another TFM function. By assigning Steps 1 and 4 to the TFM client and Steps 2 and 3 to the Conformance Monitor utility, we focus the former on problems involving multiple flights (and traffic flow considerations), and the latter on problems involving single flights.

2 Background

This section discusses simulation tool to be used and provides an overview of the required time of arrival (RTA) conformance monitor concept.

2.1 Probabilistic NAS Platform

In this engineering design we use the Probabilistic NAS Platform (PNP) to implement RTA conformance monitoring. We developed PNP [1-11], are actively maintaining it, and actively use it for NASA, Joint Planning and Development Office (JPDO), and Federal Aviation Administration (FAA) projects [12-21]. Figure 1 illustrates PNP's client-server architecture.

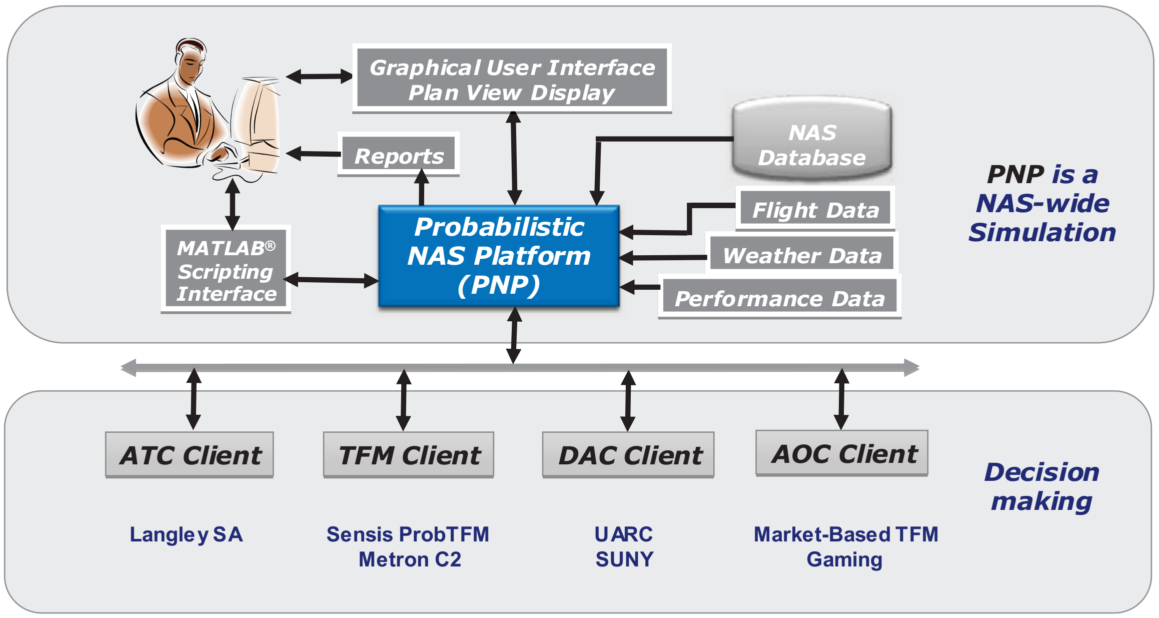


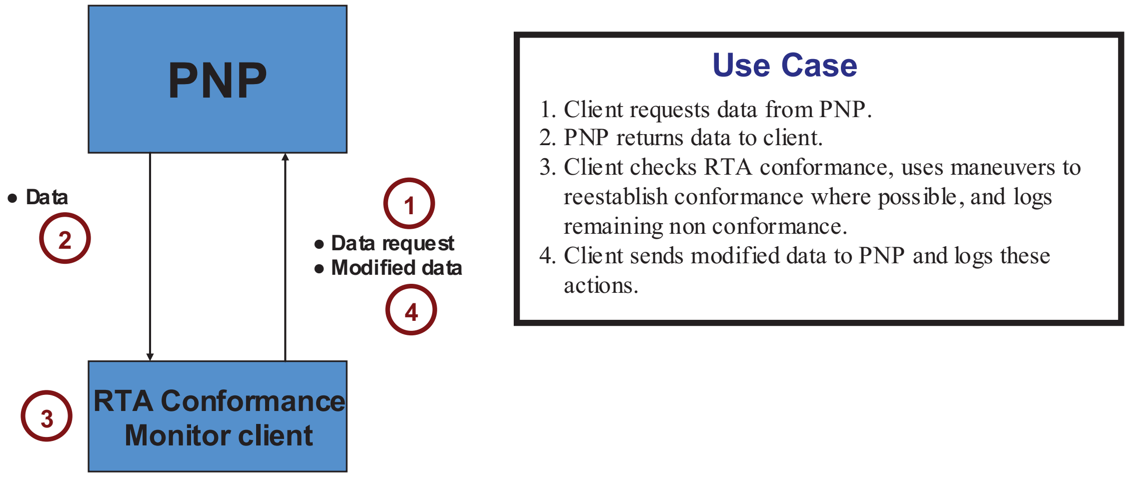
Figure 1 Probabilistic NAS Platform client-server architecture.

2.2 Overview of the RTA conformance monitor concept

An RTA is a desired crossing time for a flight which is attached to a geographically-fixed position along the flight plan route. An RTA is implemented by first determining the RTA value, and then modifying the flight plan to meet the RTA value, if necessary. At this stage of development we use the flight's destination airport as the geographic location associated with the RTA. In later stages this may be generalized to allow the RTA to be anywhere along the flight plan's trajectory.

PNP clients, such as the TFM client, will need to use the RTA conformance monitoring functionality. At this time there is no requirement for PNP to use conformance monitoring the as a client. Therefore we architect the conformance monitoring as a utility, for use by PNP clients. We also consider the possible future enhancement of using the RTA Conformance Monitor as a PNP client. We ensure that our architecture supports such a future enhancement.

Therefore we consider two cases: the RTA conformance monitor as a PNP client and as a utility for use by other PNP clients. Figure 2 shows the high level architecture and use case for the RTA conformance monitoring client.

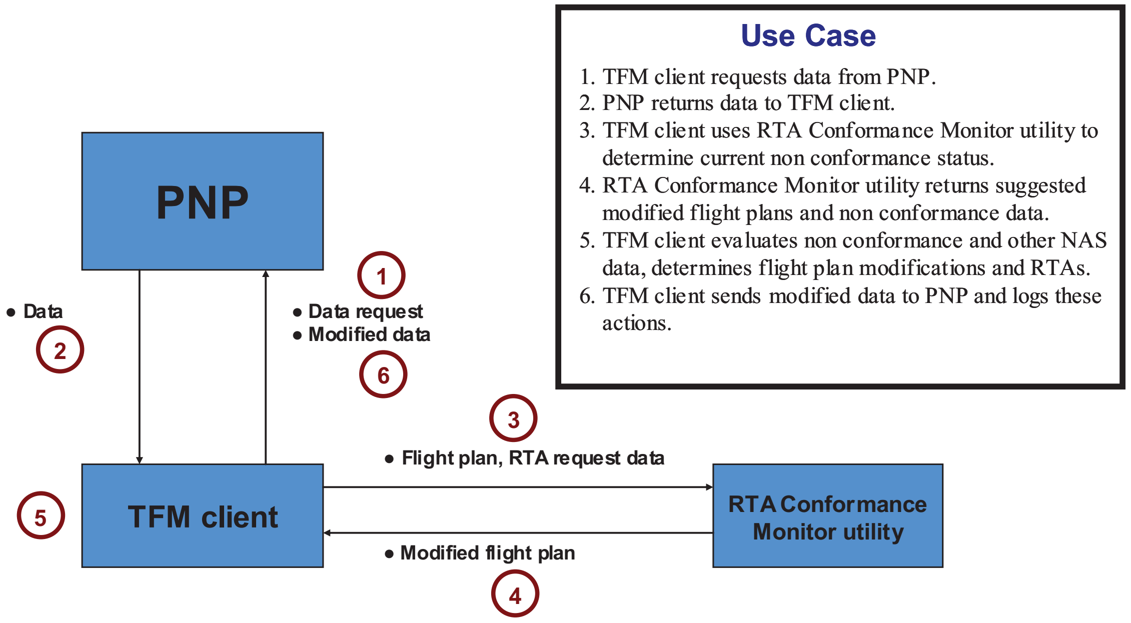


Use Case

1. Client requests data from PNP.
2. PNP returns data to client.
3. Client checks RTA conformance, uses maneuvers to reestablish conformance where possible, and logs remaining non conformance.
4. Client sends modified data to PNP and logs these actions.

Figure 2 The high level architecture and use case for the RTA conformance monitoring client.

Figure 2 shows the high level architecture and use case for the RTA conformance monitoring utility.



Use Case

1. TFM client requests data from PNP.
2. PNP returns data to TFM client.
3. TFM client uses RTA Conformance Monitor utility to determine current non conformance status.
4. RTA Conformance Monitor utility returns suggested modified flight plans and non conformance data.
5. TFM client evaluates non conformance and other NAS data, determines flight plan modifications and RTAs.
6. TFM client sends modified data to PNP and logs these actions.

Figure 3 The high level architecture and use case for the RTA conformance monitoring utility.

3 Requirements

This section lists the software requirements, divided into the PNP architecture, Conformance Monitor client / utility, and other client categories.

3.1 PNP architecture

1. RTA fields shall be added to the PNP flight plan format. These RTA fields shall include the predicted arrival time, if not already available.
2. A change history shall be added to the PNP flight plan format. This change history shall indicate which client made the change, the simulation time of day of the change, the type of change, and the reason for the change. Types of changes shall include: gate delay, and in-flight maneuver. Reasons for changes shall include: weather avoidance, separation assurance, meet initial RTA, RTA conformance, congestion management.
3. PNP shall include a list of all flights known to be out of RTA conformance. This list serves as a quick reference for clients dealing with such flights (so the client need not search through all flights). This is the “RTA conformance list.”

3.2 PNP RTA Conformance Monitor utility

4. The RTA conformance monitor shall be callable by a PNP client as a utility.
5. The utility shall accept as input a time window.
6. The utility shall check for non conformance within the input time window.
7. The utility shall use an RTA time tolerance value.
8. The utility shall perform its non conformance check only for flights that are on the RTA conformance list.
9. The utility shall perform its non conformance check only for flights that have an RTA value within the input time window.
10. The utility shall perform its non conformance check using the RTA value stored in the flight plan.
11. The utility shall check for non conformance by determining if a flight’s predicted arrival time at the RTA point differs from the RTA by more than the RTA time tolerance.
12. The utility shall output suggested flight plan modifications to meet the RTAs. The utility shall not modify any flight plans.
13. The utility shall output a list of flights that cannot be brought into conformance (if any) using any of the available maneuvers, and output files for post analysis.

3.3 Other PNP clients

14. Any client that modifies a flight plan shall add that flight to the RTA conformance list, indicating that that flight may be out of conformance.

4 Design considerations

PNP flight plans include the predicted arrival times for all waypoints as well as the destination airport. Often these predicted arrival times have no error. Errors arise when the forecasted trajectory is different from the actual trajectory that PNP simulates. Such differences can arise when a client uses a trajectory prediction model that is different from the PNP trajectory model. Such differences can also arise when inputs to the trajectory prediction model differ between the client and PNP. For instance, this would occur if the client does not have accurate aircraft weight data, wind forecasts, and so forth.

For PNP simulation runs that do not have these error sources, predicted arrival times have no error. While this is not a general case, it is typical. For many air traffic control research problems, the lack of winds and otherwise trajectory prediction errors is of little or no consequence. The majority of our research PNP simulation runs do not have these error sources.

These cases present a significant opportunity for the RTA Conformance Monitor client and utility. Specifically, because there is no arrival time prediction error, all RTA violations (i.e., cases of non conformance) can be known in advance. Such violations only occur when a flight is delayed or maneuvered. This means that the RTA Conformance Monitor need not search through all aircraft, but could simply find the cases of non conformance on a list.

Such a list (the "RTA conformance list") could be maintained by PNP. A flight would be added to the list when the flight plan is modified by a client. This includes gate delays, reroutes, in flight maneuvers, and so forth.

Therefore there are two cases to consider: the perfect arrival time prediction case and the more general case in which arrival times contain prediction errors.

In the latter case, the RTA Conformance Monitor must first determine all flights that have an RTA that falls within the input time window. The RTA Conformance Monitor then ensures that each of these flights has a sufficiently recent trajectory prediction, such that the predicted arrival times in the flight plan are reasonably up to date. And finally, the RTA Conformance Monitor must then check for RTA conformance for all these flights.

Clearly there is substantially more algorithm complexity and computational resource requirements in the more complex, general case. At this time we do not include this more complex and resource intensive case, in our architecture. But we consider this case and ensure that our architecture is easily scalable to handle the general case.

5 Future enhancements

As discussed in Section 2.2, we use the flight's destination airport as the geographic location associated with the RTA. In later stages this may be generalized to allow the RTA to be anywhere along the flight plan's trajectory.

As discussed in Section 4, a future enhancement is to handle the more general case in which the predicted arrival times may have various error sources.

As discussed in Section 2.2, the conformance monitor utility could be modified so that it can also be called directly by PNP as a client.

6 Testing

Each of the requirements in Section 3 will be verified either by demonstration, analysis or inspection.

7 Conclusions

This document describes software enhancements required to support NASA air traffic control experiments to evaluate advanced TFM concepts. In particular, these concepts specify RTAs for flights, and require RTA conformance monitoring functionality to maintain those RTAs or alert the TFM algorithm of RTAs which cannot be maintained.

References

1. George Hunter, "Testing and Validation of NextGen Simulators," AIAA Modeling and Simulation Conference, Chicago, IL, August 2009.

2. George Hunter, Kris Ramamoorthy, "Integration of terminal area probabilistic meteorological forecasts in NAS-wide traffic flow management decision making," 13th Conference on Aviation, Range and Aerospace Meteorology, New Orleans, LA, January, 2008.

3. Kris Ramamoorthy, George Hunter, "The Integration of Meteorological Data in Air Traffic Management: Requirements and Sensitivities," 46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January, 2008.

4. George Hunter, Ben Boisvert, Kris Ramamoorthy, "Advanced Traffic Flow Management Simulation Experiments and Validation," 2007 Winter Simulation Conference, Washington, DC, December, 2007.

5. Kris Ramamoorthy, George Hunter, "Evaluation of National Airspace System Performance Improvement With Four Dimensional Trajectories," AIAA Digital Avionics Systems Conference (DASC), Dallas, TX, October, 2007.

6. George Hunter, Ben Boisvert, Kris Ramamoorthy, "Use of automated aviation weather forecasts in future NAS," The 87th American Meteorological Society Annual Meeting, San Antonio, TX, January, 2007.

7. Kris Ramamoorthy, George Hunter, "Probabilistic Traffic Flow Management in the Presence of Inclement Weather and Other System Uncertainties," INFORMS Annual Meeting, Pittsburgh, PA, November, 2006.

8. Kris Ramamoorthy, Ben Boisvert, George Hunter, "A Real-Time Probabilistic TFM Evaluation Tool," AIAA Digital Avionics Systems Conference (DASC), Portland, OR, October, 2006.

9. Kris Ramamoorthy, George Hunter, "A Trajectory-Based Probabilistic TFM Evaluation Tool and Experiment," Integrated Communications, Navigation and Surveillance Conference (ICNS), Baltimore, MD, May, 2006.

10. Kris Ramamoorthy, George Hunter, "Avionics and National Airspace Architecture Strategies for Future Demand Scenarios in Inclement Weather," AIAA Digital Avionics Systems Conference (DASC), Crystal City, VA, October, 2005.

11. George Hunter, Kris Ramamoorthy, Joe Post, "Evaluation of the Future National Airspace System in Heavy Weather," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Arlington, VA, September 2005.

12. George Hunter, "Probabilistic Forecasting of Airport Capacity," AIAA Digital Avionics Systems Conference (DASC), Salt Lake City, UT, October, 2010.

13. Poornima Balakrishna, George Hunter, "Preliminary NextGen Collaborative Air Traffic Management Analysis," AIAA Digital Avionics Systems Conference (DASC), Salt Lake City, UT, October, 2010.

14. Huina Gao, George Hunter, Frank Berardino, Karla Hoffman, "Development and Evaluation of Market-Based Traffic Flow Management Concepts," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Dallas, TX, September 2010.

15. Huina Gao, George Hunter, "Evaluation of User Gaming Strategies in the Future National Airspace System," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Dallas, TX, September 2010.

16. George Hunter, "Meta Simulation Results for Simultaneous Dynamic Resectorization and Traffic Flow Management," AIAA Digital Avionics Systems Conference (DASC), Orlando, FL, October, 2009.

17. Huina Gao, George Hunter, "Future NAS-Wide User Gaming Preliminary Investigation," AIAA Digital Avionics Systems Conference (DASC), Orlando, FL, October, 2009.

18. George Hunter, "Preliminary Assessment of Interactions Between Traffic Flow Management and Dynamic Airspace Configuration Capabilities," AIAA Digital Avionics Systems Conference (DASC), St. Paul, MN, October, 2008.

19. George Hunter, "Toward an Economic Model to Incentivize Voluntary Optimization of NAS Traffic Flow," AIAA ATIO Conference, Anchorage, AK, September, 2008.

20. George Hunter, "Sensitivity of the National Airspace System Performance to Weather Forecast Accuracy," Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, May, 2008.

21. Kris Ramamoorthy, Ben Boisvert, George Hunter, "Sensitivity of Advanced Traffic Flow Management to Different Weather Scenarios," Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, May, 2007.

14809-09

PNP Required Time of Arrival (RTA) Conformance Monitor Concepts Interface Control Document

Ben Boisvert

Prepared for:

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681

Under:

NASA BOA: NNL08AA17B

November, 2011

Table of Contents

1 Introduction

This document describes the interface control document required to support NASA air traffic control experiments to evaluate the required time of arrival (RTA) conformance monitor concepts.

2 Software Architecture

The RTA Conformance Monitor Utility is a set of methods in a library that can be utilized by a client or the server. The PNP server will provide a list of all flights known to be out of RTA conformance.

3 Interfaces

3.1 Flight Change Record

The Flight Change Record class is used to track flight plan changes over time. The creation of a delay or reroute requires the creation of a Flight Change Record such that all flight change transaction can be traced throughout the system.

FlightChangeRecord		
Field	Type	Description
m_Client	String	Indicates the client that submitted the flight change record.
m_Timestamp	long	The time that the flight change record was assigned.
m_Rta	long	The required time of arrival associated with the flight change record.
m_Status	FlightChangeRecord.STATUS	The status of the flight change record. Values: SUBMITTED, IMPLEMENTED, PRIORITY_CONFLIGHT_IMPLEMENTED, PRIORITY_CONFLIGHT_IGNORED, OVERCOME_BY_EVENTS
m_Reason	FlightChangeRecord.RESAON	The reason the flight change record was created. Values: WEATHER_AVIODANCE, SEPARATION_ASSURANCE, INITIAL_RTA, RTA_CONFORMANCE, CONGESTION_MANAGEMENT
m_Type	FlightChangeRecord.TYPE	The type of change in the flight change

		record. Values: GATE_DELAY, PREDEPARTURE_REROUTE, INFLIGHT_MANEUVER, INFLIGHT_REROUTE

3.2 Delay Plan

The Delay Plan class is used to request that the PNP server institute a delay for a flight.

DelayPlan		
Field	Type	Description
m_DelayInSeconds	int	Indicates the delay in seconds for a flight.
m_FlightChangeRecord	FlightChangeRecord	Indicates who, when, and why flight is being delayed.

3.3 Reroute Plan

The Reroute Plan class is used to request that the PNP server institute a reroute for a flight.

ReroutePlan		
Field	Type	Description
m_FlightPlan	String	Indicates the ASDI flight plan.
m_FlightPlanWaypointNames	ArrayList<String>	List of flight plan waypoint names.
m_FlightPlanWaypoints	ArrayList<Traj2D>	List of 2D trajectory flight plan waypoints.
m_Maneuvers	ArrayList<Maneuver>	List of maneuvers
m_ FlightChangeRecord	FlightChangeRecord	Indicates who, when, and why flight is being delayed.

3.4 Flight Details

The Flight Details class has been updated to contain the following fields to support the RTA enhancements.

FlightDetails		
Field	Type	Description
m_FlightClassification	FlightClassification.CLASSIFICATION	Indicates the classification of a flight. Values: VFR, IFR, AFR, UAS

m_Rta	long	The required time of arrival.
m_FlightChangeHistory	Map<Long, FlightChangeRecord>	List of flight change records by time.

3.5 RTA Non-Conformance List

The RTA Non-Conformance List class is a list of all flights known to be out of RTA conformance

PnpRtaNonConformanceList		
Field	Type	Description
m_FlightDetails	List<FlightDetails>	List of RTA non-conformant flights.

3.6 RTA Spacing Maneuver

The RTA Spacing Maneuver class contains the maneuver need to achieve a RTA.

RtaSpacingManeuver		
Field	Type	Description
m_ImplementManeuverTime	long	Indicates the time in which a RTA maneuver is implemented.
m_RequiredArrivalTime	long	The required time of arrival.
m_SpeedRtaEnabled	RtaSpacingManeuver.TYPE	Type of RTA maneuver Values: CRUISE_CAS, DESCENT_PROFILE, DOGLEG

4 Conclusion

This document describes software interfaces required to support NASA air traffic control experiments to evaluate advanced TFM concepts. In particular, these interfaces specify RTAs for flights.

1214809-03

RTA Assignment Client Requirements and Engineering Design

Ben Boisvert
George Hunter

Prepared for:

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681

Under:

NASA BOA: NNL08AA17B/Task Order NNL10AC94T

March, 2012

Table of Contents

1 Introduction

This document describes software enhancements required to support NASA air traffic control experiments to evaluate advanced, cockpit-based, traffic flow management concepts including separation assurance. Section 2 describes the simulation environment and preliminary work that has been done.

Section 3 describes the requirements for the software to support upcoming NASA experiments. Sections 4 describes design considerations and future enhancements. Section 5 lists the tests to be performed and Section 6 gives the conclusions.

2 Background

This section discusses simulation tool to be used and preliminary tests that have been conducted.

2.1 *Probabilistic NAS Platform*

In this engineering design we use the Probabilistic NAS Platform (PNP) to implement and test advanced traffic flow management (TFM) concepts including separation assurance (SA).

We developed PNP [1-11], are actively maintaining it, and actively use it for NASA, Joint Planning and Development Office (JPDO), and Federal Aviation Administration (FAA) projects [12-21]. Figure 1 illustrates PNP's client-server architecture.

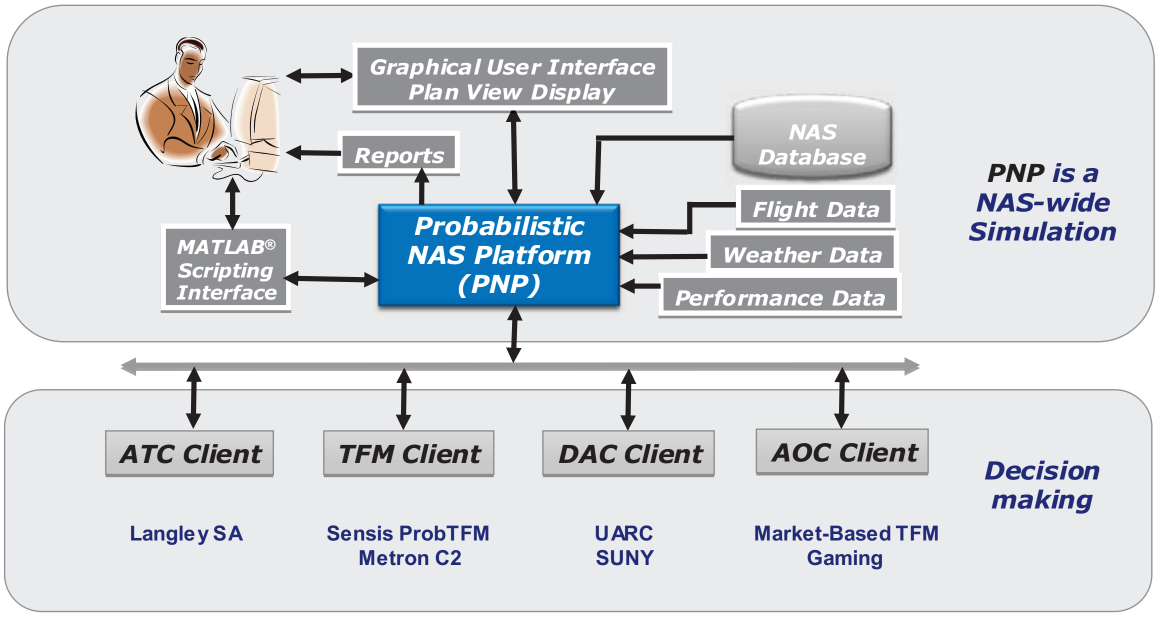


Figure 1 Probabilistic NAS Platform client-server architecture.

In this project we add Weather Avoidance and RTA Assignment clients to PNP. As Fig. 2 shows, these clients will be used with the SA and ProbTFM clients.

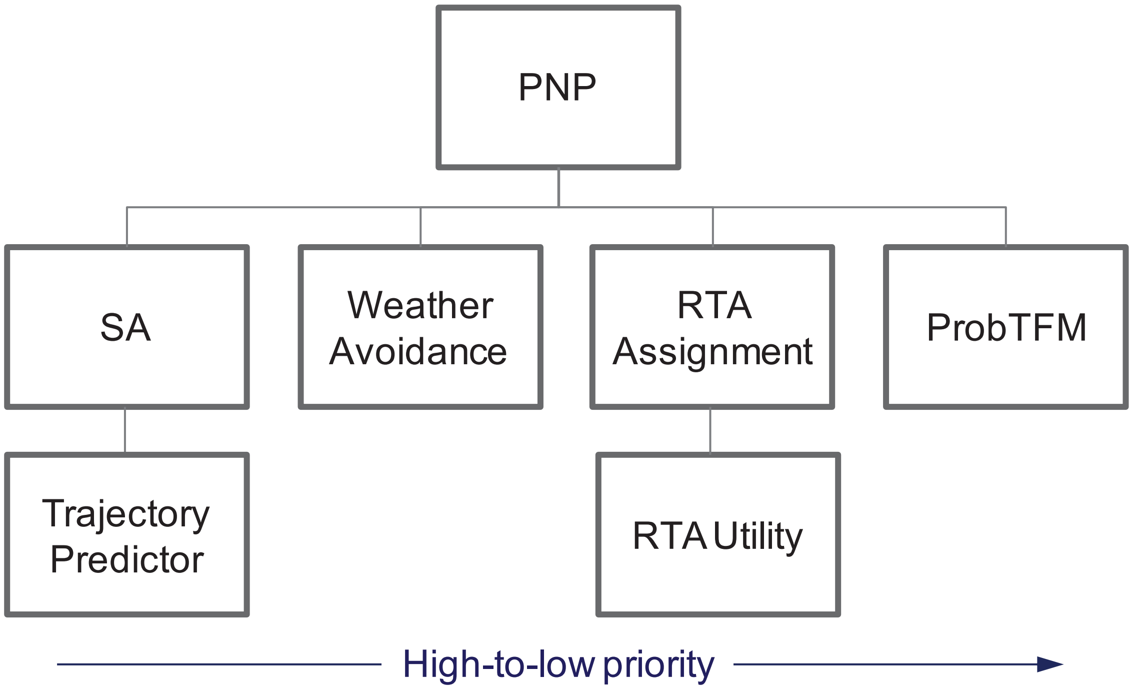


Figure 2 High level experiment architecture indicating maneuver priority used to resolve conflicting client requests.

2.2 *Preliminary tests*

In previous efforts we have constructed and tested the SA and ProbTFM clients. We also constructed and tested the RTA utility which evaluates flight times for different maneuvers and designs maneuvers to meet a given RTA.

3 Requirements

This section lists the software requirements for the RTA Assignment client.

1. The RTA Assignment client shall schedule an initial RTA for all feasible aircraft at the arrival airport.
2. The RTA Assignment client shall monitor RTA conformance for all feasible flights.
3. The RTA Assignment client shall attempt to assign a new RTA in the event a flight cannot meet the assigned RTA.
4. If the RTA Assignment client fails to successfully assign an RTA to a flight, that failure shall be logged to the system.
5. The RTA Assignment client shall not interfere with the operation of the other PNP clients.

4 Design considerations

This section describes design considerations for the RTA Assignment client and PNP server.

4.1 Use case and assumptions

This section describes the nominal use case for the 2012 experiment which will use the RTA Assignment client, along with the SA, Weather Avoidance and ProbTFM clients. PNP is run in a scenario with heavy traffic and significant en route convective weather but light airport surface weather. Therefore the capacities of the en route airspace sectors are impacted by the presence of convective weather and consequently incur time-varying capacity reductions, but the airports have constant, clear weather, capacity.

In every 15 minute time bin PNP calls the four clients: SA, Weather Avoidance, RTA Assignment and ProbTFM. The SA client is run with standard en route separation assurance settings.

The Weather Vectoring client has two separate components, a strategic component for pre departure flights and a tactical component for en route flights. The strategic component runs first. It uses a database of fixed routes to search for alternate routes for flights which are forecasted to penetrate too much convective weather. An alternate route is selected for such flights if its associated weather penetration is less than the tolerance.

Next the tactical component runs. It computes reroutes around local weather for flights that are forecasted to penetrate too much weather. For both the strategic and tactical components, for flights that are rerouted a flag is set in the flight plan if the flight becomes out of conformance of a previously assigned RTA.

The RTA Assignment client assigns RTAs for the first time to flights in their en route phase. The RTA Assignment client also monitors RTA conformance. For flights that are out of conformance, the client attempts to reestablish conformance.

For the ProbTFM client rerouting is turned off as well as en route sector congestion. Only gate delays are used to resolve airport congestion. Sector congestion is turned off by scaling the sector capacities globally to infinity.

4.2 RTA Assignment client architecture

Figure 3 shows that the RTA Assignment client consists of two main parts corresponding to its two main tasks. First, the client assigns the initial RTAs for all flights, and second, the client monitors RTA conformance and attempts to reassign RTAs for flights out of conformance.

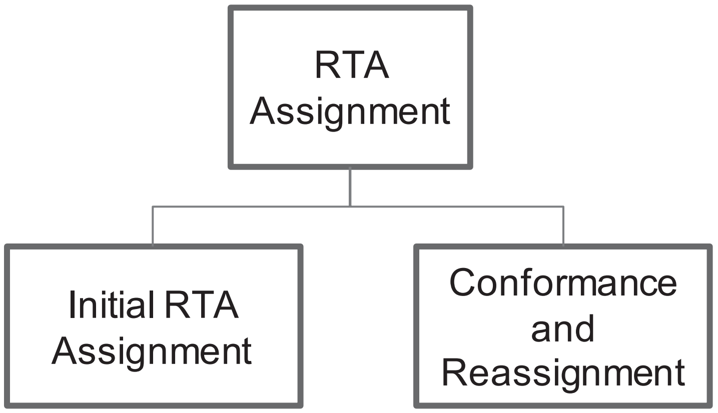


Figure 3 RTA Assignment client high level architecture.

4.3 Initial RTA assignment

Flights have their initial RTA assigned when they enter an adaptable distance or time from the destination airport. Of course, once a flight has an RTA assigned, then it no longer will be considered in the initial RTA assignment part of the RTA Assignment client.

The initial RTA may not be possible to assign for some flights, depending on parameter settings. For example, very short flights may land before an RTA can be assigned. Therefore, it may not be feasible to assign an RTA to some flights. It is important that this and other clients, as well as PNP, be robust to this and not assume that all flights have RTAs. Flights without RTAs simply fly normally. Their arrival time is dictated by the flight plan and environmental variables.

The RTA assignment logic is shown in Fig. 4. For eligible flights (i.e., those flights that have not had an initial RTA assignment and have entered the RTA assignment distance or time window), the RTA utility is used to determine a reasonable arrival window, defined by the earliest and latest arrival times. The earliest arrival time is computed using only speed control (no altitude or path control). An adaptable parameter (default is 20 minutes) is added to this to obtain the latest ETA.

The algorithm next steps from this earliest to latest time, in increments of the airport minimum time spacing. The airport minimum time spacing is a constant parameter, specific for each airport, equal to the airports 15-minute time interval capacity divided by 900 seconds, to obtain the minimum time interval, in seconds, between operations.

When used at airports, this spacing parameter is not intended to represent the literal spacing between operations. Clearly airports can operate with simultaneous operations on different runways, for instance. Instead, this spacing parameter is merely a modeling device to ensure that the airport is not overloaded in finite time windows, such as 15-minute time intervals. Ultimately, this spacing parameter is intended for use with waypoints, where it can represent the literal traffic spacing.

As the algorithm steps from earliest to latest time, it searches for a time point which is conflict free. That is, a time point at which there is no other already scheduled operation (departure or arrival) that is within the airport's minimum time spacing of that time point.

When an open slot is found, it is assigned to the flight by modifying the flight plan. Also the airport's schedule list is updated. If no such open slot is found in the search, then the RTA assignment failure is logged.

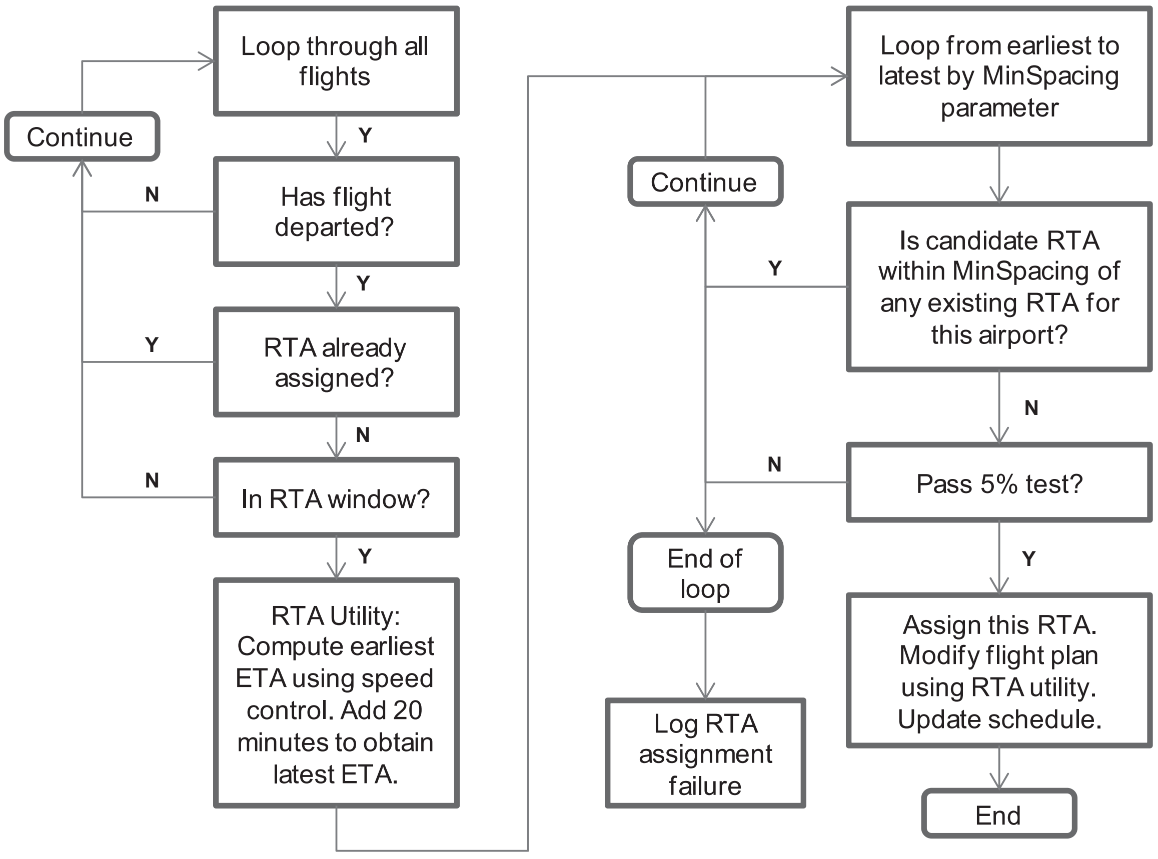

Loop through all flights
Continue
Y
N
Has flight departed?
Y
Y
RTA already assigned?
N
N
In RTA window?
Y
RTA Utility: Compute earliest ETA using speed control. Add 20 minutes to obtain latest ETA.
Loop from earliest to latest by MinSpacing parameter
Continue
Y
Is candidate RTA within MinSpacing of any existing RTA for this airport?
N
N
Pass 5% test?
End of loop
Y
Log RTA assignment failure
Assign this RTA. Modify flight plan using RTA utility. Update schedule.
End

Figure 4 Initial RTA assignment logic.

Note that in the search for an open slot, flights with an already assigned RTA are not modified. The only degree of freedom in this search is the own-ship RTA.

The initial RTA assignment logic also has a "5% test." The purpose of this test is to build into the schedule occasional open slots so that subsequent non conforming flights can be handled more easily. In other words, a schedule that is not 100% fully packed with flights can support subsequent delays with less disruption.

The 5% test is used for initial RTA assignments. It is not used for RTA reassignment, when a flight is delayed due to RTA non conformance. In this test, a random number generator (RNG) is used to occasionally skip an open slot, thus leaving it open. An adaptable input threshold is used to specify the chances of passing the test. The test is passed if the RNG produces a value that exceeds the input threshold. For instance, if the RNG produces uniformly distributed random numbers in a range from 0.0 to1.0, then a threshold value of 0.05 means that the test will be passed 95% of the time, and likewise it will fail 5% of the time.

4.4 RTA conformance monitor

The RTA Assignment client also monitors the conformance of flights that already have an RTA assignment. The conformance monitor has two components: the monitor component and the reassignment component.

In this experiment, a flight may become out of conformance because it was maneuvered to maintain separation (by the SA client) or to avoid weather (by the Weather Avoidance client). In future experiments there may be other causal factors. For instance, forecast errors in the en route winds may cause a sufficient error in the estimated time of arrival to create an RTA loss of conformance. Note that in that hypothetical case, there was no client action that caused the loss of conformance, and so it could not be anticipated. The only way to check for such loss of conformance is to update the ETAs for all flights that have an RTA assignment.

In this experiment, because loss of conformance can only be caused by client actions (SA and Weather Avoidance clients), it is possible to know which flights are out of conformance without computing their ETA. For instance, when clients cause a maneuver that forces loss of conformance, the client could add the flight to a list of non conforming flights. The RTA Assignment client could then simply consult that list. Whether or not such a strategy should be used is a software engineering issue.

Conformance is judged to be lost when the ETA deviates from the RTA by a value greater than an adaptable threshold parameter.

The RTA reassignment component uses logic similar to the initial RTA assignment logic discussed above (see Section 4.3).

4.5 Schedule creation

In the RTA assignment logic above, it is implicit that there is a schedule for each airport, consisting of a list of all the departure and arrival RTAs, which are currently assigned. The RTA assignment logic consults the schedule in order to find an open slot. And once an open slot is assigned, the schedule must be updated to include the new RTA that has been assigned. So an RTA is not only stored in the flight plan of the flight, it also is stored in the airport's schedule. Therefore, in addition to the initial RTA assignment and the RTA conformance, the RTA Assignment client also must use and maintain the schedule for each airport.

The airport schedule may also be used and maintained by other clients and the PNP servers. Therefore, This is a cross-cutting design issue. The details of how and where the airport schedules are initially created, and subsequently managed, is a software engineering design issue. Here we make several comments regarding the schedule.

For departures, the schedule departure times (SDTs) are immediately available from the initial demand set. Therefore the airport schedule could be populated with all the departures for the entire day at the beginning of the simulation. Or this process could be done at regular intervals during the simulation, such as every two hours. At the extreme, this could done in every 15 minute time interval, populating the schedule with only the departures for the current time interval.

In general, as Fig. 5 illustrates, the closer to departure time and so the more frequent this process is, the more it favors arrivals. On the other hand, departures will tend to have more priority, over the arrivals, if this process is less frequent, and done over a longer time window. We suggest a happy medium, that reflects operational procedures, where this process is executed approximately one-two hours prior to departure. But a shorter time may be desired to give arrivals higher priority.

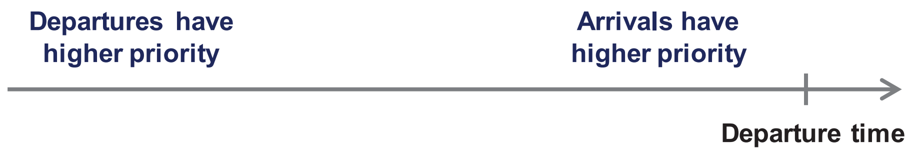

Departures have higher priority
Arrivals have higher priority
Departure time

Figure 5 Insertion time in airport schedule for departures influences relative priority.

Note that when an SDT is inserted into the schedule, assignment logic similar to that discussed above in Section 4.1 is used. This is because the new SDT being inserted may conflict with operations (departures or arrivals) already inserted into the schedule.

For arrivals, the schedule arrival times (SATs) are also available from the initial demand set. But given the schedule uncertainties, the SAT probably should not be inserted into the airport schedule. Instead, we wait until the RTA has been formally assigned, as described above in Section 4.1. At that point the RTA is inserted into the airport schedule, as indicated in Fig. 4.

4.6 Schedule maintenance

In addition to the initial insertion of the departure and arrival times (SDTs and RTAs, respectively) into the airport schedule, the schedule also must be maintained as the day evolves. In particular, anytime a departure or arrival time is modified, it must be done in accordance with the airport schedule, ensuring the move is to an open slot. For arrivals, this was discussed in Section 4.1. Departures may also have their (departure) time modified. In this case it is not referred to as an RTA, but nonetheless it does occupy a slot in the airport schedule, and so changes to the departure time requires schedule maintenance.

In the current experiment, departure times are modified only by the ProbTFM client, in the form of gate delays. In the future, other clients such as the airline operations center (AOC) client could change departure times or even cancel flights.

Where and how the schedule maintenance is performed is a software engineering issue.

4.7 Discrete time versus continuous time

The airport schedule may be formatted in discrete or continuous time. In the discrete time format, fixed time slots are computed and enforced using the airport minimum spacing parameter. The continuous time format, on the other hand, allows for continuously varying, arbitrary departure and arrival times. The only constraint is that the airport minimum spacing is not violated. Figure 6 illustrates these two formats.

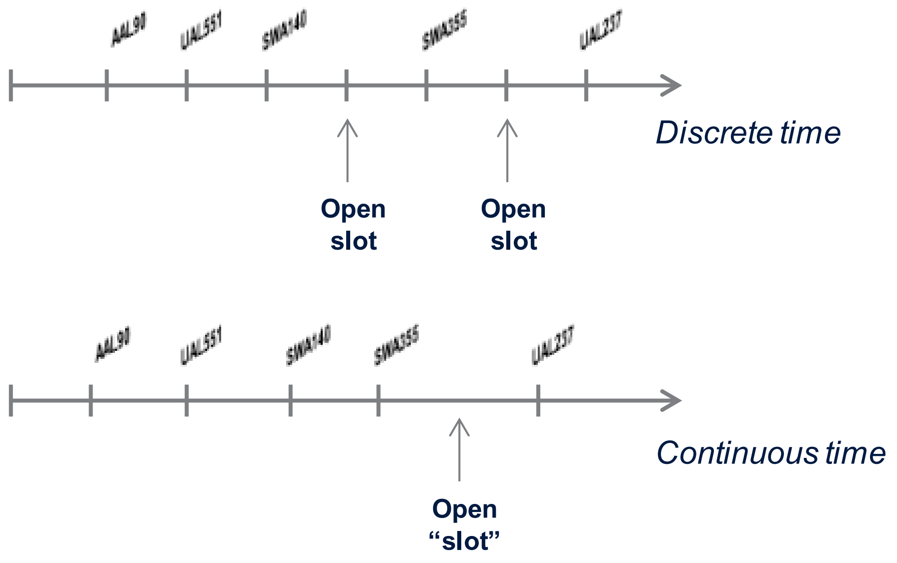


Figure 6 Discrete time versus continuous time schedule.

As Fig. 6 illustrates, in the discrete time format it is easier to maximize the use of airport capacity. If all the slots are used, then the throughput equals the capacity. In the continuous time format, on the other hand, there are packing issues. Departure and arrival times that are not tightly packed create extra, unused, time in the schedule. This results in fewer open slots, and lower throughput.

The continuous time format, however, supports variable airport minimum spacing that can result from using a Pareto curve description of the capacity, or from environmental impacts on the airport capacity. Such variations in airport capacity are difficult to support with the discrete time format.

The choice of timing format is a software engineering issue.

4.8 Future enhancements

There are several enhancements that could be implemented in the future. Here are some candidate enhancements:

- Move the RTA point from the airport to the terminal area metering fixes.
- Add en route winds and the effect of forecast error of these winds.

- Add time-varying airport capacity due to Pareto curve modeling or weather effects.
- Improve the SA client algorithm.

5 Testing

This section describes the software testing for this implementation. Table 1 lists the tests.

Table 1 PNP SA client tests.

PNP Separation Assurance Client Tests					
#	Requirements	Qualification	Design Traceability	Code Traceability	Build
1	The RTA Assignment client shall schedule an initial RTA for all feasible aircraft at the arrival airport.	Demonstration			1
2	The RTA Assignment client shall monitor RTA conformance for all feasible flights.	Demonstration			1
3	The RTA Assignment client shall attempt to assign a new RTA in the event a flight cannot meet the assigned RTA.	Demonstration			1
4	If the RTA Assignment client fails to successfully assign an RTA to a flight, that failure shall be logged to the system.	Demonstration			1
5	The RTA Assignment client shall not interfere with the operation of the other PNP clients.	Inspection			1

6 Conclusions

This document describes software enhancements required to support NASA air traffic control experiments to evaluate advanced, cockpit-based, TFM concepts with separation assurance. This includes both requirements and design considerations. This document also presents design enhancements for the future, and testing to be conducted.

References

1. George Hunter, "Testing and Validation of NextGen Simulators," AIAA Modeling and Simulation Conference, Chicago, IL, August 2009.

2. George Hunter, Kris Ramamoorthy, "Integration of terminal area probabilistic meteorological forecasts in NAS-wide traffic flow management decision making," 13th Conference on Aviation, Range and Aerospace Meteorology, New Orleans, LA, January, 2008.

3. Kris Ramamoorthy, George Hunter, "The Integration of Meteorological Data in Air Traffic Management: Requirements and Sensitivities," 46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January, 2008.

4. George Hunter, Ben Boisvert, Kris Ramamoorthy, "Advanced Traffic Flow Management Simulation Experiments and Validation," 2007 Winter Simulation Conference, Washington, DC, December, 2007.

5. Kris Ramamoorthy, George Hunter, "Evaluation of National Airspace System Performance Improvement With Four Dimensional Trajectories," AIAA Digital Avionics Systems Conference (DASC), Dallas, TX, October, 2007.

6. George Hunter, Ben Boisvert, Kris Ramamoorthy, "Use of automated aviation weather forecasts in future NAS," The 87th American Meteorological Society Annual Meeting, San Antonio, TX, January, 2007.

7. Kris Ramamoorthy, George Hunter, "Probabilistic Traffic Flow Management in the Presence of Inclement Weather and Other System Uncertainties," INFORMS Annual Meeting, Pittsburgh, PA, November, 2006.

8. Kris Ramamoorthy, Ben Boisvert, George Hunter, "A Real-Time Probabilistic TFM Evaluation Tool," AIAA Digital Avionics Systems Conference (DASC), Portland, OR, October, 2006.

9. Kris Ramamoorthy, George Hunter, "A Trajectory-Based Probabilistic TFM Evaluation Tool and Experiment," Integrated Communications, Navigation and Surveillance Conference (ICNS), Baltimore, MD, May, 2006.

10. Kris Ramamoorthy, George Hunter, "Avionics and National Airspace Architecture Strategies for Future Demand Scenarios in Inclement Weather," AIAA Digital Avionics Systems Conference (DASC), Crystal City, VA, October, 2005.

11. George Hunter, Kris Ramamoorthy, Joe Post, "Evaluation of the Future National Airspace System in Heavy Weather," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Arlington, VA, September 2005.

12. George Hunter, "Probabilistic Forecasting of Airport Capacity," AIAA Digital Avionics Systems Conference (DASC), Salt Lake City, UT, October, 2010.

13. Poornima Balakrishna, George Hunter, "Preliminary NextGen Collaborative Air Traffic Management Analysis," AIAA Digital Avionics Systems Conference (DASC), Salt Lake City, UT, October, 2010.

14. Huina Gao, George Hunter, Frank Berardino, Karla Hoffman, "Development and Evaluation of Market-Based Traffic Flow Management Concepts," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Dallas, TX, September 2010.

15. Huina Gao, George Hunter, "Evaluation of User Gaming Strategies in the Future National Airspace System," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Dallas, TX, September 2010.

16. George Hunter, "Meta Simulation Results for Simultaneous Dynamic Resectorization and Traffic Flow Management," AIAA Digital Avionics Systems Conference (DASC), Orlando, FL, October, 2009.

17. Huina Gao, George Hunter, "Future NAS-Wide User Gaming Preliminary Investigation," AIAA Digital Avionics Systems Conference (DASC), Orlando, FL, October, 2009.

18. George Hunter, "Preliminary Assessment of Interactions Between Traffic Flow Management and Dynamic Airspace Configuration Capabilities," AIAA Digital Avionics Systems Conference (DASC), St. Paul, MN, October, 2008.

19. George Hunter, "Toward an Economic Model to Incentivize Voluntary Optimization of NAS Traffic Flow," AIAA ATIO Conference, Anchorage, AK, September, 2008.

20. George Hunter, "Sensitivity of the National Airspace System Performance to Weather Forecast Accuracy," Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, May, 2008.

21. Kris Ramamoorthy, Ben Boisvert, George Hunter, "Sensitivity of Advanced Traffic Flow Management to Different Weather Scenarios," Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, May, 2007.

1214809-02

PNP Weather Avoidance Client and Display Enhancements Requirements and Engineering Design

Ben Boisvert
George Hunter

Prepared for:

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681

Under:

NASA BOA: NNL08AA17B/Task Order NNL10AC94T

March, 2012

Table of Contents

1 Introduction

This document describes software enhancements required to support NASA air traffic control experiments to evaluate advanced Weather Avoidance concepts in the context of flight-deck-based traffic flow management. Section 2 describes the simulation environment and preliminary work that has been done. Section 3 describes the requirements for the software to support the upcoming NASA experiments. Section 4 describes design considerations and future enhancements. Section 5 lists the tests to be performed. The document is closed with the conclusions in Section 6, followed by a list of references.

2 Background

This section discusses the simulation tool to be used and preliminary tests that have been conducted.

2.1 Probabilistic NAS Platform

In this engineering design, we use the Probabilistic NAS Platform (PNP) to implement and test advanced Weather Avoidance concepts.

PNP has been developed as a platform to evaluate advanced NAS concepts.[1-21] PNP is a NAS-Wide simulation that can be tailored to simulate current day or future scenarios. PNP is scalable such that researchers can investigate various aspects of the NAS through multiple clients, as depicted in Figure 1.

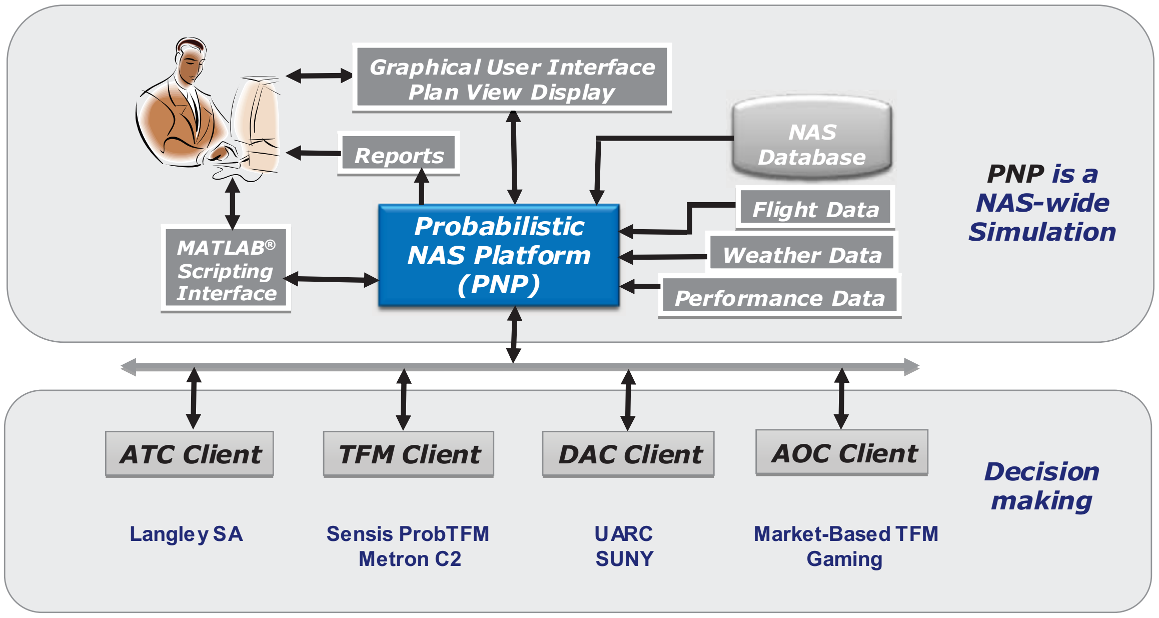


Figure 1 Probabilistic NAS Platform client-server architecture.

2.2 Preliminary Tests

In previous efforts, we have conducted preliminary tests of weather avoidance, rerouting, and gridding in PNP. Those efforts included basic weather cell identification and simple avoidance routes, as shown in Figure 2.

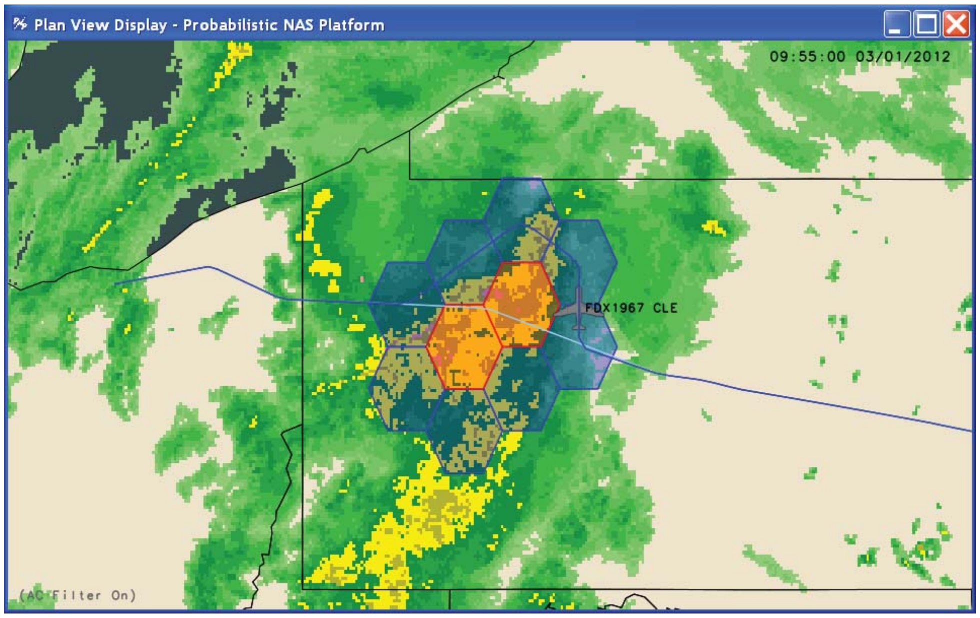


Figure 2 Screen shot of a weather avoidance reroute in the PNP environment.

These tests provided an initial successful integration with PNP. They also revealed that a NAS-wide implementation requires substantial computational resources and that visualization is critical to verification and validation.

3 Requirements

This section lists the software requirements, divided into the PNP Weather Avoidance client, PNP server, and PNP display categories.

3.1 Weather Avoidance Client Requirements

1. The client shall run periodically during the PNP simulation.
2. The client shall search all flights that have departed or are within a user-specifiable time of departing to determine if the flight is projected to fly nearby heavy convective weather.
3. The client shall determine how to modify the flight plan to avoid the weather.
4. The client shall evaluate several candidate gate delays or pre-departure reroutes in a “what if” trial-planning approach.
5. The client shall assign gate delays.
6. The client shall evaluate several candidate in-flight reroutes in a “what if” trial-planning approach.
7. The client shall assign reroutes.
8. The client shall create a report of metrics evaluating the success of the weather avoidance algorithm.
9. The client shall log failed attempts to reroute flights around weather.

3.2 PNP Server Requirements

10. The PNP server shall report rerouting statistics based on the Required Time of Arrival, Separation Assurance, and Weather Avoidance activities.

3.3 PNP Display Requirements

11. The PNP display shall incorporate Required Time of Arrival field into the Flight Display Window.
12. The PNP display shall incorporate reroute type information (Separation Assurance, Congestion, and Weather Avoidance) into the Flight Display Window.

4 Design Considerations

This section describes design considerations for the PNP Weather Avoidance client and PNP server.

4.1 Weather Avoidance Maneuver Architecture

The Weather Avoidance client will be a client in the PNP client-server architecture as illustrated in Figure 3. The Weather Avoidance client will communicate with the PNP server using the SimObjects API.

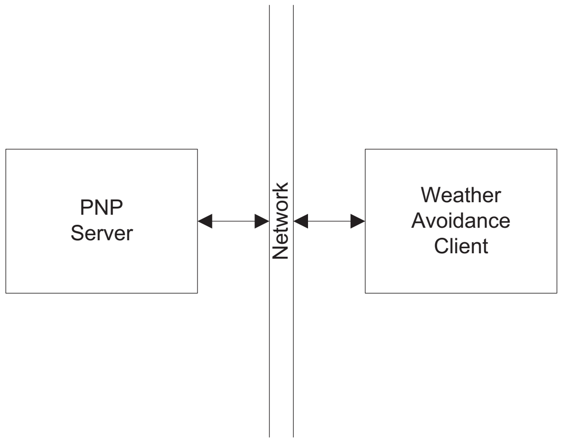


Figure 3 High-level Weather Avoidance architecture.

The Weather Avoidance client will request flight and reflectivity updates (NOWRAD) from the server, based on the interval parameter. NOWRAD data are sent every 5 minutes, so an interval less than 5 minutes is not recommended. The Weather Avoidance client will process the reflectivity image into a graphical cell representation that will be used to reroute flights. Reroutes will be generated by the Weather Avoidance client and sent to the PNP server for implementation, as illustrated in Fig. 4.

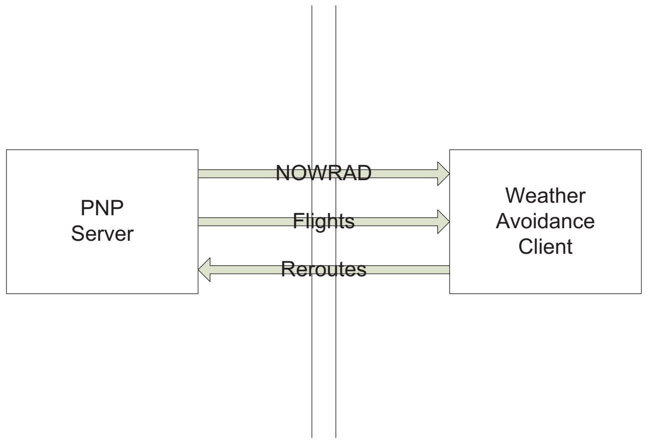


Figure 4 Weather Avoidance message flow.

The PNP server will accept reroutes based on the reroute order of precedence listed in Table 1.

Table 1 Reroute Precedence.

Type	Order
Separation Assurance	Highest
Weather Avoidance	High
RTA conformance	Medium
Congestion Avoidance	Lowest

4.1.2 Weather Forecasting

The Weather Avoidance client will use a nowcast and persistence forecast to predict weather. The nowcast uses the current reflectivity image to represent the convective weather for the current time period. The nowcast will be used to report the number of bad weather flight incursions. The persistence forecast uses the current reflectivity image as the forecast for the future time periods. Each forecast will have an adaptable parameter to determine the coverage threshold. The PNP universal grid (4.1 nmi) will be used to determine weather cell population coverage. The weather cell will be a hexagon creating a honeycomb grid, as illustrated in Fig. 5.

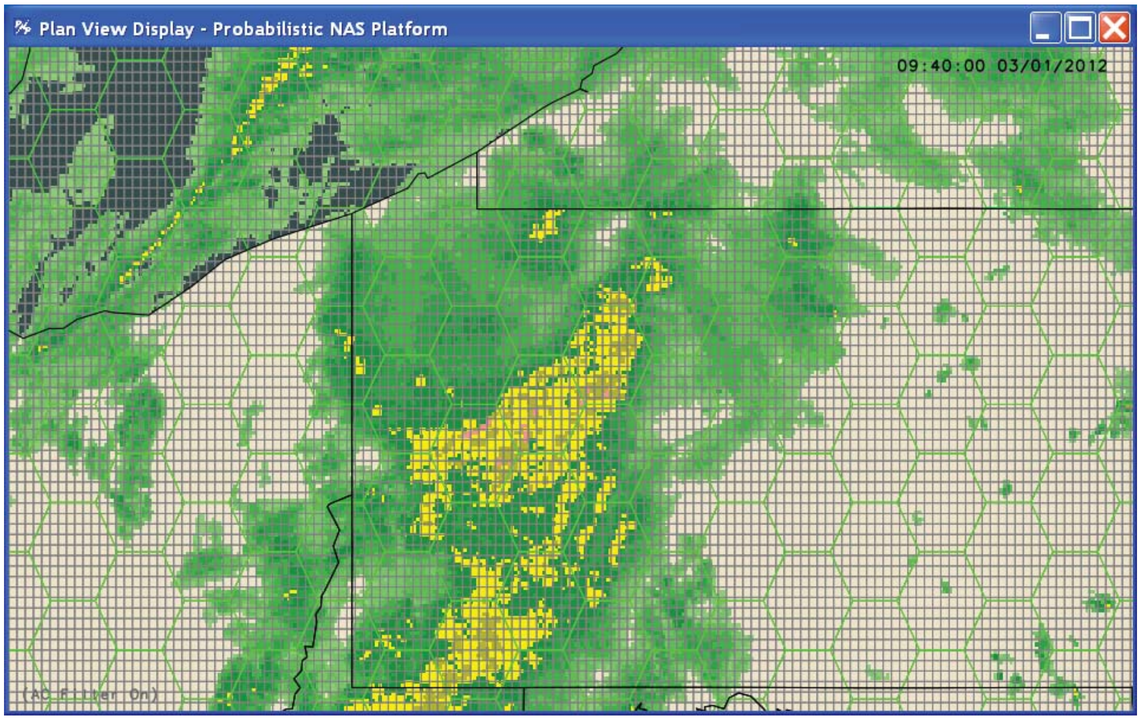


Figure 5 PNP Universal Grid and Weather Cells.

4.1.3 Weather Cell Management

The bad weather cells will be identified using the persistence forecast. A bad weather cell is identified when the percentage of universal grid elements exceeds the coverage threshold, as illustrated in Fig. 6. The weather cell hexagon has a radius of approximately 17.5 nmi. There are approximately 78 grid elements per weather cell. The universal grid element is considered populated with bad weather if the universal grid element dbz value is greater than 40 dbz.

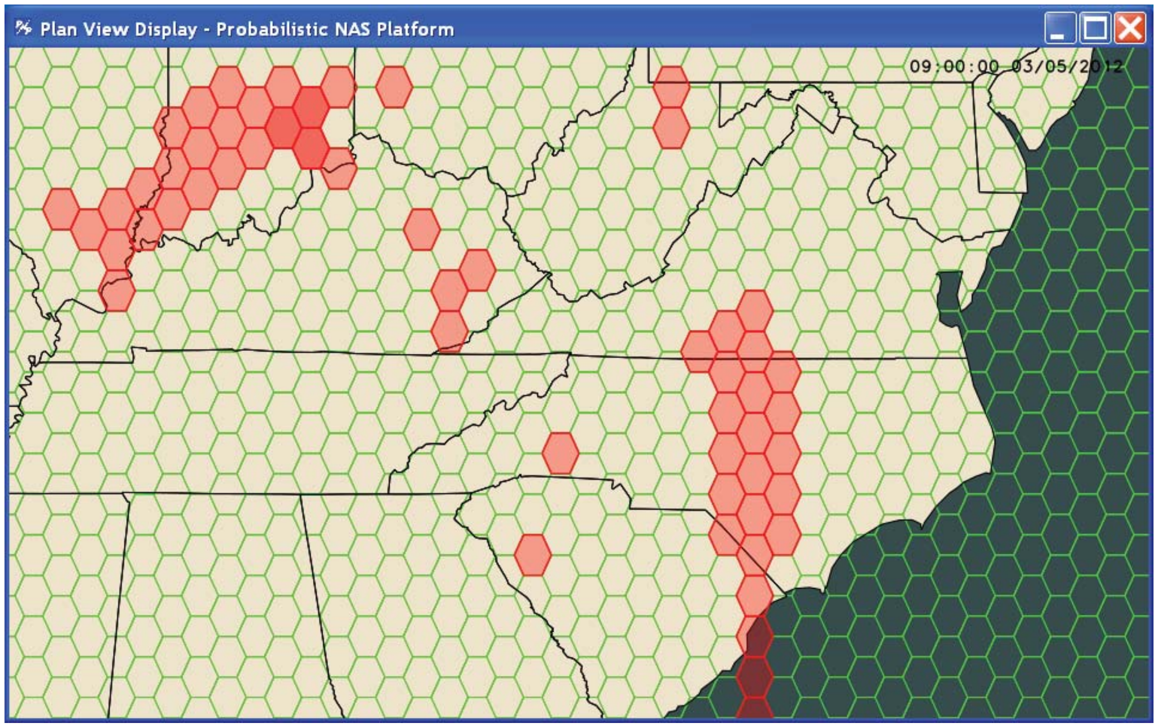


Figure 6 Red Bad Weather Cells.

4.1.4 Weather Clustering

The bad weather cells will be correlated to form weather clusters, as illustrated in Fig. 7. As neighboring bad weather cells are linked, they form the weather cluster. The clustering of bad weather cells forms the representation of a storm, which then can be avoided.

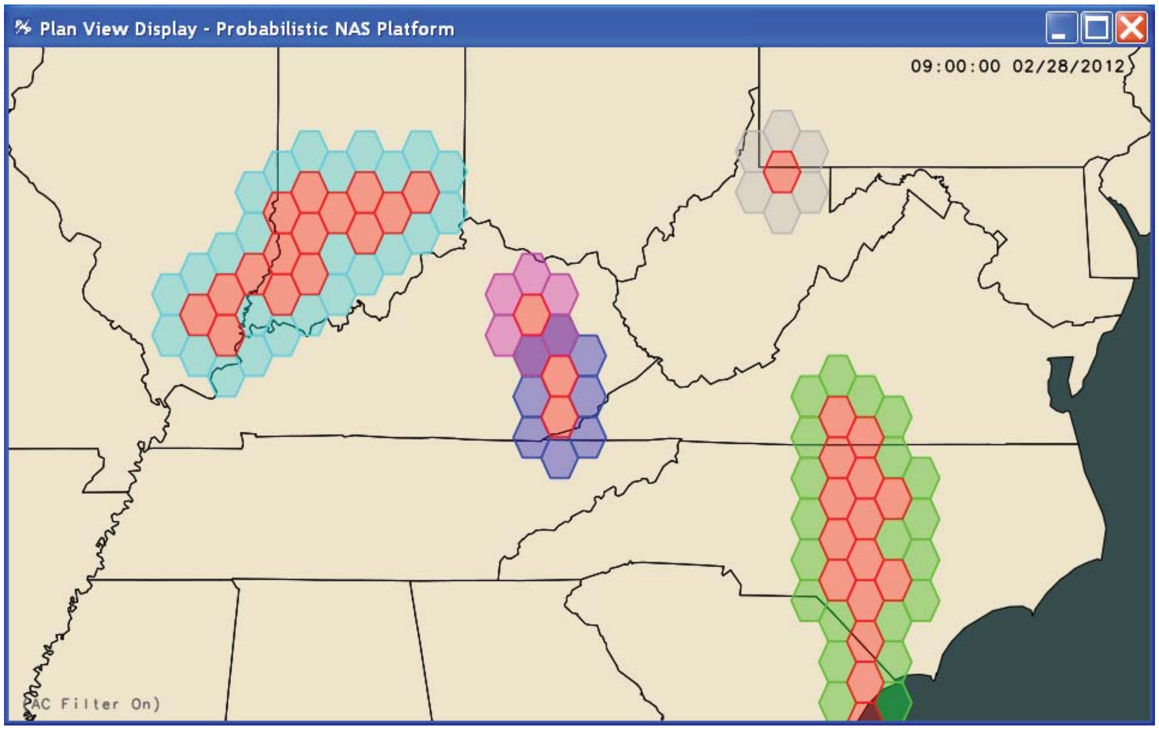


Figure 7 Weather Clusters.

4.2 Routing

The Weather Avoidance client will use both strategic and tactical routing using the persistence forecast. The tactical routing occurs in the en route phase while the strategic routing occurs prior to departure.

4.2.1 Tactical Routing

A flight will be considered for tactical routing if its tracks in the planning horizon traverse bad weather cells. The planning horizon will consist of two adaptable relative time parameters (start and end of the planning horizon). If a flight is predicted to traverse a bad weather cell in the planning horizon, then the first bad weather cell encountered is used to determine the weather cluster. The neighboring good weather cells of the weather cluster will be used as candidate reroute cells. These candidate cells will be passed into a Dijkstra algorithm to determine the shortest path. This shortest path is then spliced into the current flight plan to create a new reroute for the flight, as illustrated in Fig. 8. Route optimization will be investigated to smooth reroutes, reducing path length and complexity.

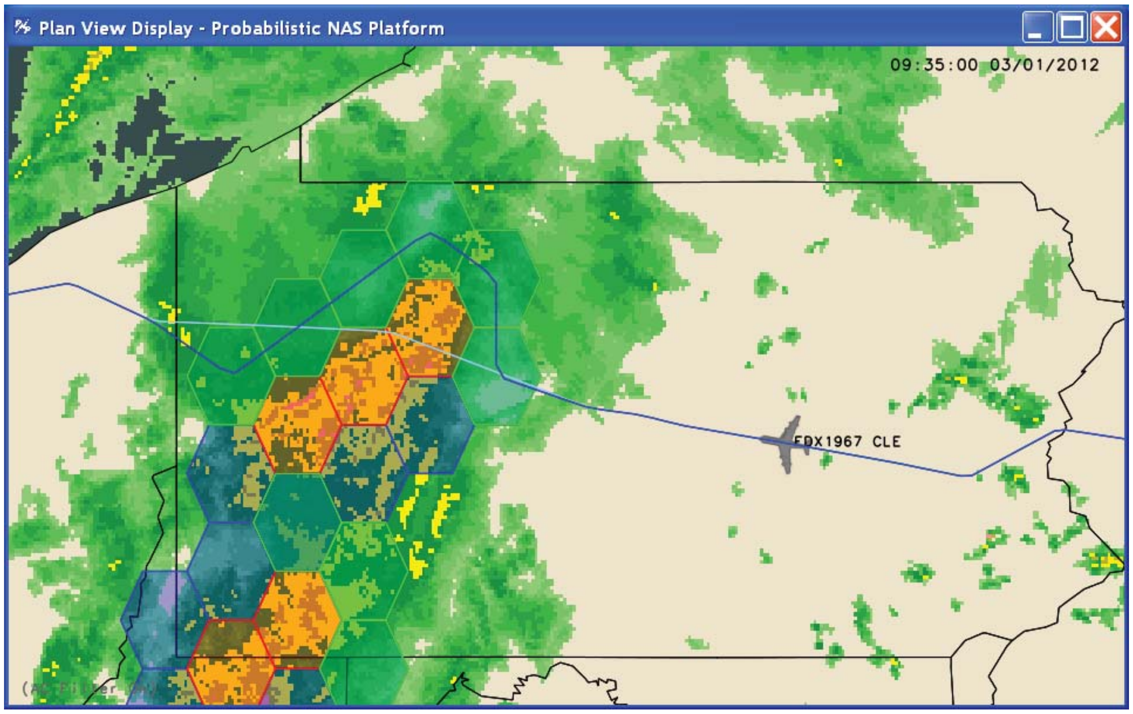


Figure 8 Tactical Routing Around Bad Weather Cluster.

4.2.2 Strategic Routing

Pre-departure flights will be considered for strategic routing if their tracks are predicted to traverse bad weather cells. The strategic routing algorithm will calculate how many minutes a flight will encounter bad weather based on the persistence forecast. Then, the strategic routing algorithm will calculate how many minutes a flight will encounter bad weather using different routes. If an alternate route is available which encounters less bad weather, then it is selected as a strategic reroute. If multiple city-pair routes are available which encounter less bad weather, then the shortest among them will be chosen, as illustrated in Fig. 9. The selected city-pair routes will be saved for an adaptable time period. As routes are saved and reused over an adaptable time period, they will serve as the "game-plan" for that time period.

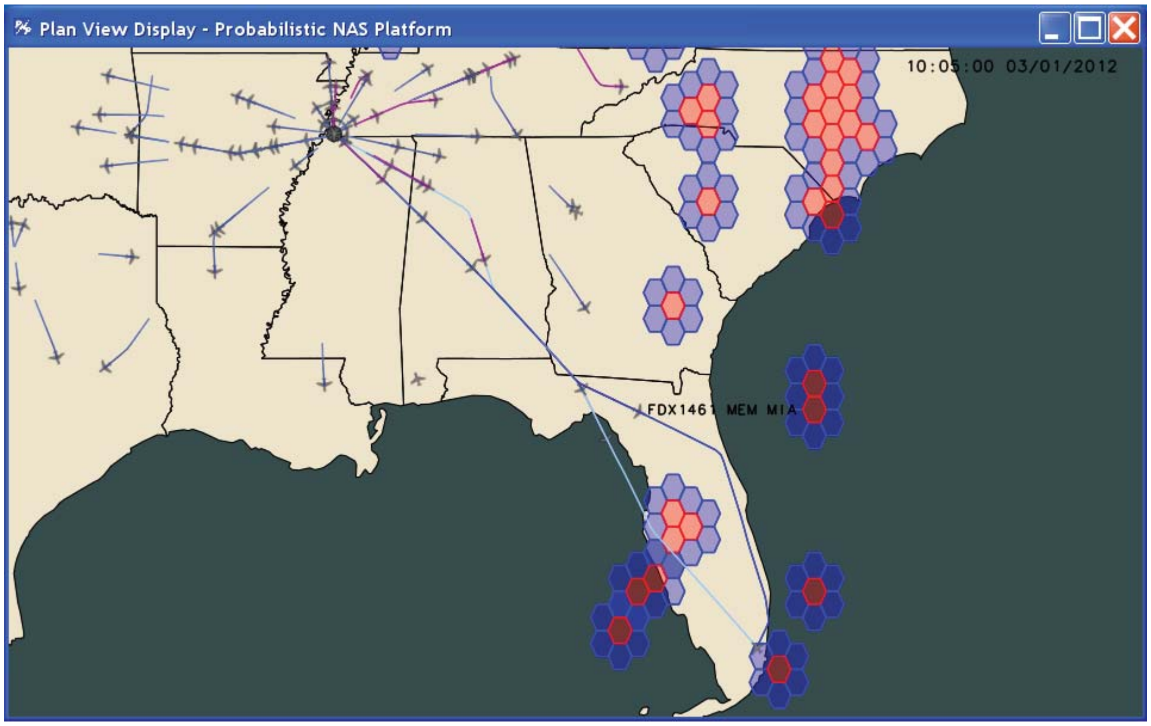


Figure 9 Strategic Routing Around Bad Weather Cluster.

4.3 Detail Design

This section describes the detailed design of the weather avoidance client**.**

4.3.1 Main Executive Thread

The Weather Avoidance client consists of four main modules, as shown in Figure 10. The Weather Avoidance client's main executive will extend a Java thread. The thread will initialize adaptation and start communications with the server. The main thread will use the SimObjects API creating a client configuration object and registering for the *Heartbeat*, *Command*, *NOWRAD*, *pre-departure* and *en-route flight* messages. The main thread will register callback methods for every registered message. The heartbeat callback method will processes the NOWRAD image file and create a forecast based on the nowcast and persistence forecast coverage thresholds. If strategic routing is enabled, the flight manager will process all pre-departure flights for strategic reroutes. If tactical routing is enabled, the flight manager will process all en-route flights for tactical reroutes. The *NOWRAD* callback method will store the current NOWRAD images for processing. The *pre-departure* and *en-route flight* callback methods will store the flights in the flight manager. The *Command* callback method will process all system commands. If reroutes are created by the routing managers, the main thread will create a reroute plan and send it to the PNP server for processing.

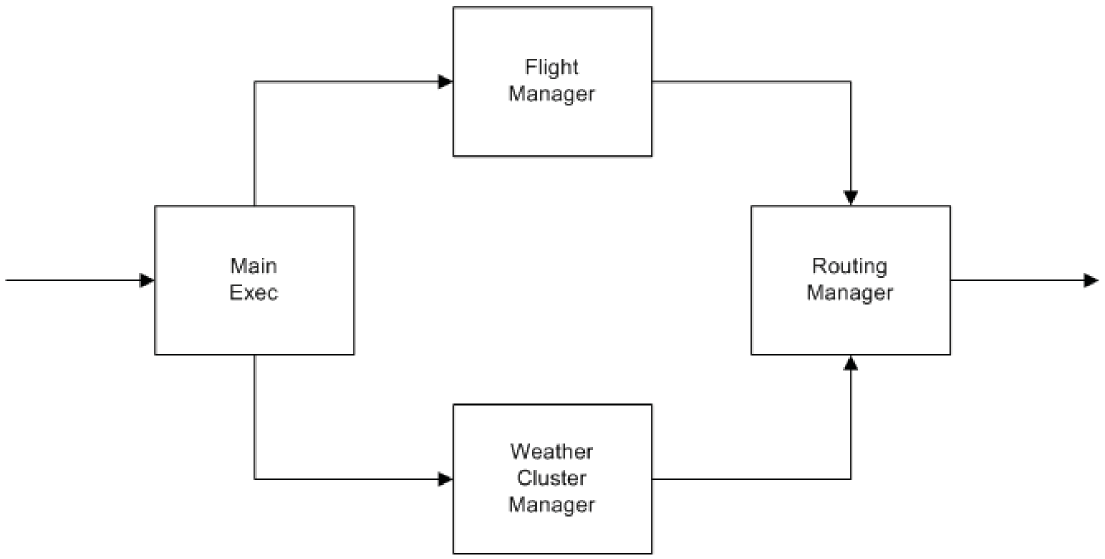


Figure 10 Weather Avoidance Detail Design.

4.3.2 Strategic Routing Manager

The Strategic Routing Manager creates strategic reroutes, avoiding bad weather for pre-departure flights, as shown in Figure 11. The manager calculates the flight minutes in bad weather based on the forecast for a flight. Then, the manager evaluates alternative city-pair routes for that flight. If the alternative routes have less flight minutes in bad weather, a reroute is recommended. If multiple routes exist with less flight minutes in bad weather, the shortest reroute will be selected. The selected route for a given city pair will be saved in the "game-plan." The game-plan will maintain these selected routes for an adaptable direction. Every pre-departure flight is evaluated by the Strategic Routing Manager.

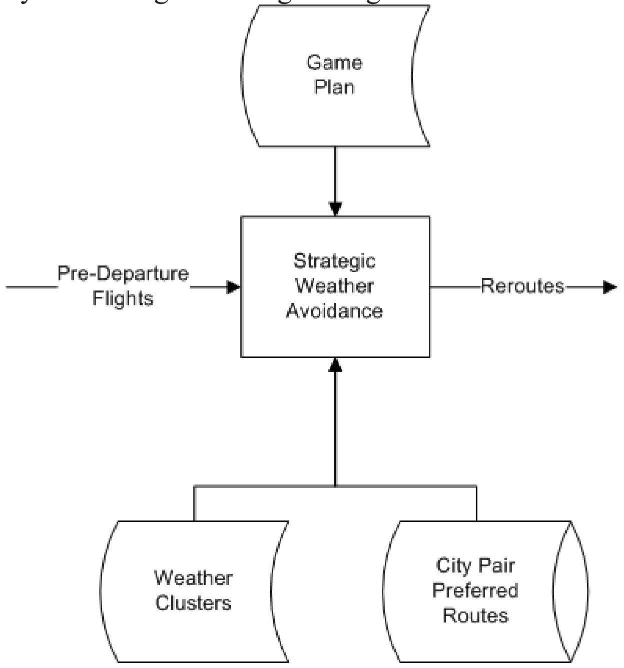


Figure 11 Strategic Routing.

4.3.3 Tactical Routing Manager

The Tactical Routing Manager creates tactical reroutes, avoiding bad weather for en-route flights, as shown in Figure 12. The Tactical Routing Manager determines the flight minutes in bad weather based on the forecast for a flight. Then, the manager evaluates rerouting based on buffer cells using the Dijkstra algorithm. The first weather cluster that a flight encounters in the planning horizon is identified. The buffer cells associated with the first weather cluster encountered are considered candidate cells for the Dijkstra Algorithm. The first and last buffer cell and all their associated neighbor cells are linked together to create a network for the Dijkstra algorithm. The Dijkstra algorithm is executed, and the resultant path is used as a reroute. The Dijkstra path is stitched into the current flight plan. Every en-route flight is evaluated by the Tactical Routing Manager.

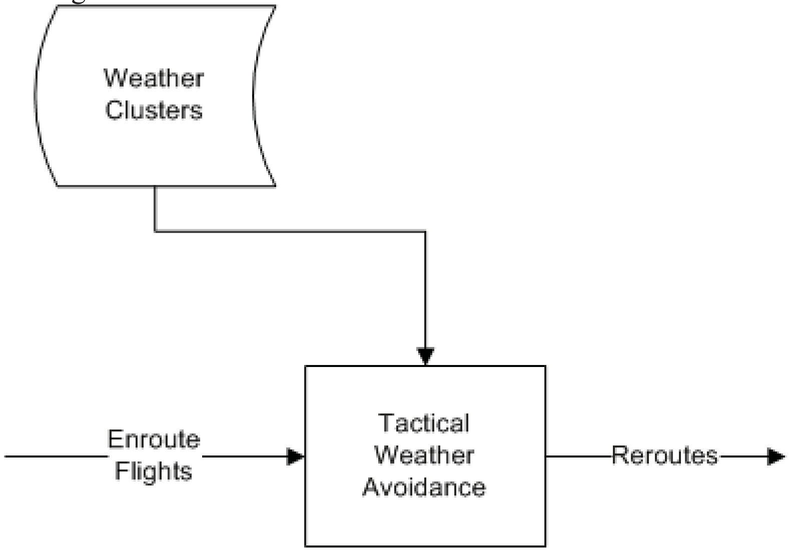


Figure 12 Tactical Routing.

4.3.3.1 Route Optimization

The reroutes created by the Dijkstra algorithm may not be optimal. Smoothing may be achieved by evaluating the route using the neighbor buffer cell of the first and last node of the reroute. The smoothed reroute would then be verified that it does not encounter bad weather. If a smoothed route encounters a bad weather cell, then it is rejected and the optimization process terminates.

4.4 Future Enhancements

This section describes enhancements that are outside the scope of this effort but may be implemented in the future.

4.4.1 Perfect Time Varying Forecast

TBD

4.4.2 Strategic Reroutes Post Departure

TBD

5 Testing

This section describes the software testing for the PNP Weather Avoidance client, PNP server, and PNP display enhancements. Table 2 lists the PNP SA client tests.

Table 2 PNP SA Client Tests.

PNP Weather Avoidance Client Tests					
#	Requirements	Qualification	Design Traceability	Code Traceability	Build
1	The client will run periodically during the PNP simulation.	Demonstration	Inspection	Inspection	1
2	The client searches all flights that have departed or are within a user specifiable time of departing to determine if the flight is projected to fly nearby heavy convective weather.	Demonstration	Inspection	Inspection	1
3	The client determines how to modify the flight plan to avoid the weather.	Demonstration	Inspection	Inspection	1
4	The client will evaluate several candidate gate delays in a "what if" trial-planning approach.	Inspection	Inspection	Inspection	1
5	The client will assign gate delays.	Analysis	Inspection	Inspection	1
6	The client will evaluate several candidate in-flight reroutes in a "what if" trial-planning approach.	Inspection	Inspection	Inspection	1
7	The client will assign reroutes.	Analysis	Inspection	Inspection	1
8	The client will create a report of metrics evaluating the success of the weather	Analysis	Inspection	Inspection	1

	avoidance algorithm.				
9	The client shall log failed attempts to reroute flights around weather.	Demonstration	Inspection	Inspection	1
10	The PNP server shall report rerouting statistics based on the Required Time of Arrival, Separation Assurance, and Weather Avoidance activities.	Demonstration	Inspection	Inspection	1
11	The PNP display shall incorporate Required Time of Arrival field into the Flight Display Window.	Demonstration	Inspection	Inspection	1
12	The PNP display shall incorporate reroute type information (Separation Assurance, Congestion, and Weather Avoidance) into the Flight Display Window.	Demonstration	Inspection	Inspection	1

6 Conclusions

This document describes software enhancements required to support NASA air traffic control experiments to evaluate advanced Weather Avoidance concepts. This includes both requirements and design considerations. This document also presents design enhancements for the future, and testing to be conducted.

References

1. George Hunter, "Testing and Validation of NextGen Simulators," AIAA Modeling and Simulation Conference, Chicago, IL, August, 2009.

2. George Hunter, Kris Ramamoorthy, "Integration of terminal area probabilistic meteorological forecasts in NAS-wide traffic flow management decision making," 13th Conference on Aviation, Range and Aerospace Meteorology, New Orleans, LA, January, 2008.

3. Kris Ramamoorthy, George Hunter, "The Integration of Meteorological Data in Air Traffic Management: Requirements and Sensitivities," 46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January, 2008.

4. George Hunter, Ben Boisvert, Kris Ramamoorthy, "Advanced Traffic Flow Management Simulation Experiments and Validation," 2007 Winter Simulation Conference, Washington, DC, December, 2007.

5. Kris Ramamoorthy, George Hunter, "Evaluation of National Airspace System Performance Improvement With Four Dimensional Trajectories," AIAA Digital Avionics Systems Conference (DASC), Dallas, TX, October, 2007.

6. George Hunter, Ben Boisvert, Kris Ramamoorthy, "Use of automated aviation weather forecasts in future NAS," The 87th American Meteorological Society Annual Meeting, San Antonio, TX, January, 2007.

7. Kris Ramamoorthy, George Hunter, "Probabilistic Traffic Flow Management in the Presence of Inclement Weather and Other System Uncertainties," INFORMS Annual Meeting, Pittsburgh, PA, November, 2006.

8. Kris Ramamoorthy, Ben Boisvert, George Hunter, "A Real-Time Probabilistic TFM Evaluation Tool," AIAA Digital Avionics Systems Conference (DASC), Portland, OR, October, 2006.

9. Kris Ramamoorthy, George Hunter, "A Trajectory-Based Probabilistic TFM Evaluation Tool and Experiment," Integrated Communications, Navigation and Surveillance Conference (ICNS), Baltimore, MD, May, 2006.

10. Kris Ramamoorthy, George Hunter, "Avionics and National Airspace Architecture Strategies for Future Demand Scenarios in Inclement Weather," AIAA Digital Avionics Systems Conference (DASC), Crystal City, VA, October, 2005.

11. George Hunter, Kris Ramamoorthy, Joe Post, "Evaluation of the Future National Airspace System in Heavy Weather," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Arlington, VA, September, 2005.

12. George Hunter, "Probabilistic Forecasting of Airport Capacity," AIAA Digital Avionics Systems Conference (DASC), Salt Lake City, UT, October, 2010.

13. Poornima Balakrishna, George Hunter, "Preliminary NextGen Collaborative Air Traffic Management Analysis," AIAA Digital Avionics Systems Conference (DASC), Salt Lake City, UT, October, 2010.

14. Huina Gao, George Hunter, Frank Berardino, Karla Hoffman, "Development and Evaluation of Market-Based Traffic Flow Management Concepts," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Dallas, TX, September, 2010.

15. Huina Gao, George Hunter, "Evaluation of User Gaming Strategies in the Future National Airspace System," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Dallas, TX, September, 2010.

16. George Hunter, "Meta Simulation Results for Simultaneous Dynamic Resectorization and Traffic Flow Management," AIAA Digital Avionics Systems Conference (DASC), Orlando, FL, October, 2009.

17. Huina Gao, George Hunter, "Future NAS-Wide User Gaming Preliminary Investigation," AIAA Digital Avionics Systems Conference (DASC), Orlando, FL, October, 2009.

18. George Hunter, "Preliminary Assessment of Interactions between Traffic Flow Management and Dynamic Airspace Configuration Capabilities," AIAA Digital Avionics Systems Conference (DASC), St. Paul, MN, October, 2008.

19. George Hunter, "Toward an Economic Model to Incentivize Voluntary Optimization of NAS Traffic Flow," AIAA ATIO Conference, Anchorage, AK, September, 2008.

20. George Hunter, "Sensitivity of the National Airspace System Performance to Weather Forecast Accuracy," Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, May, 2008.

21. Kris Ramamoorthy, Ben Boisvert, George Hunter, "Sensitivity of Advanced Traffic Flow Management to Different Weather Scenarios," Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, May, 2007.

14809-22

PNP Enhanced Terminal Area Modeling Requirements and Engineering Design

George Hunter

Prepared for:

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681

Under:

NASA BOA: NNL08AA17B

November 30, 2012

Table of Contents

1 Introduction

This document describes software and data enhancements required to support NASA air traffic control experiments to evaluate advanced national airspace system (NAS) concepts. In particular, this document describes the enhancements for improved terminal area modeling in the Probabilistic NAS Platform (PNP).

Section 2 describes the simulation environment and provides an overview of the required time of arrival (RTA) concept. Section 3 describes the requirements for the software. Sections 4 and 5 describe design considerations and future enhancements, respectively. Section 6 describes the tests to be performed.

2 Background

This section provides an overview of the enhanced terminal modeling to be implemented and the PNP simulation tool.

2.1 Probabilistic NAS Platform

In this engineering design we use the Probabilistic NAS Platform (PNP) to implement enhanced terminal modeling. We developed PNP [1-11], are actively maintaining it, and actively use it for NASA, Joint Planning and Development Office (JPDO), and Federal Aviation Administration (FAA) projects [12-23]. Figure 1 illustrates PNP's client-server architecture.

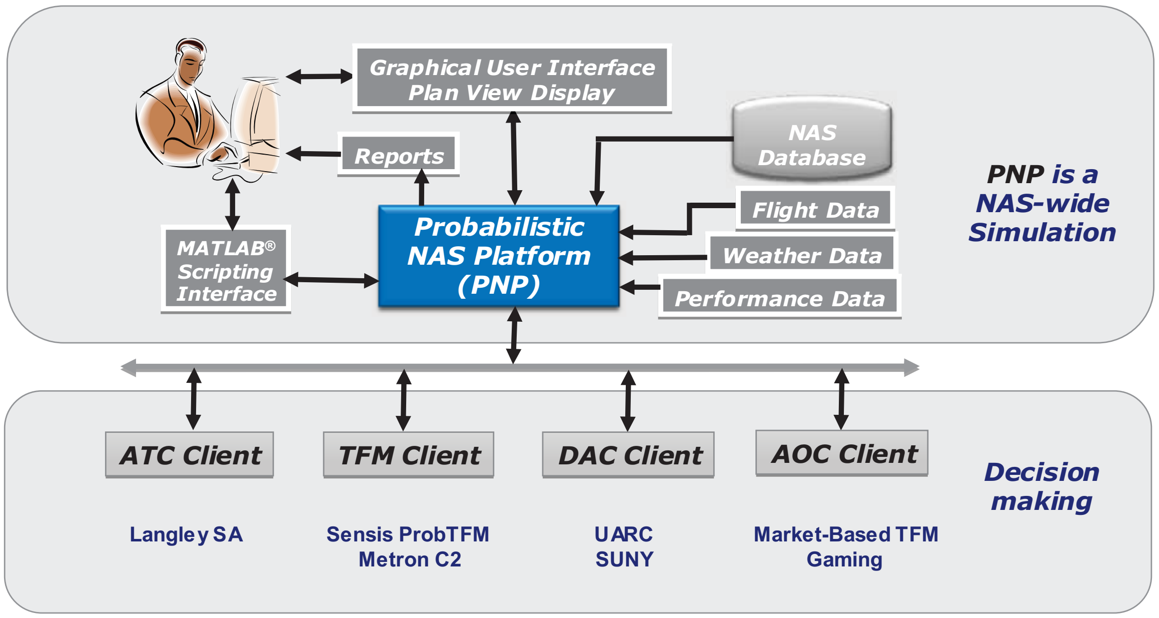
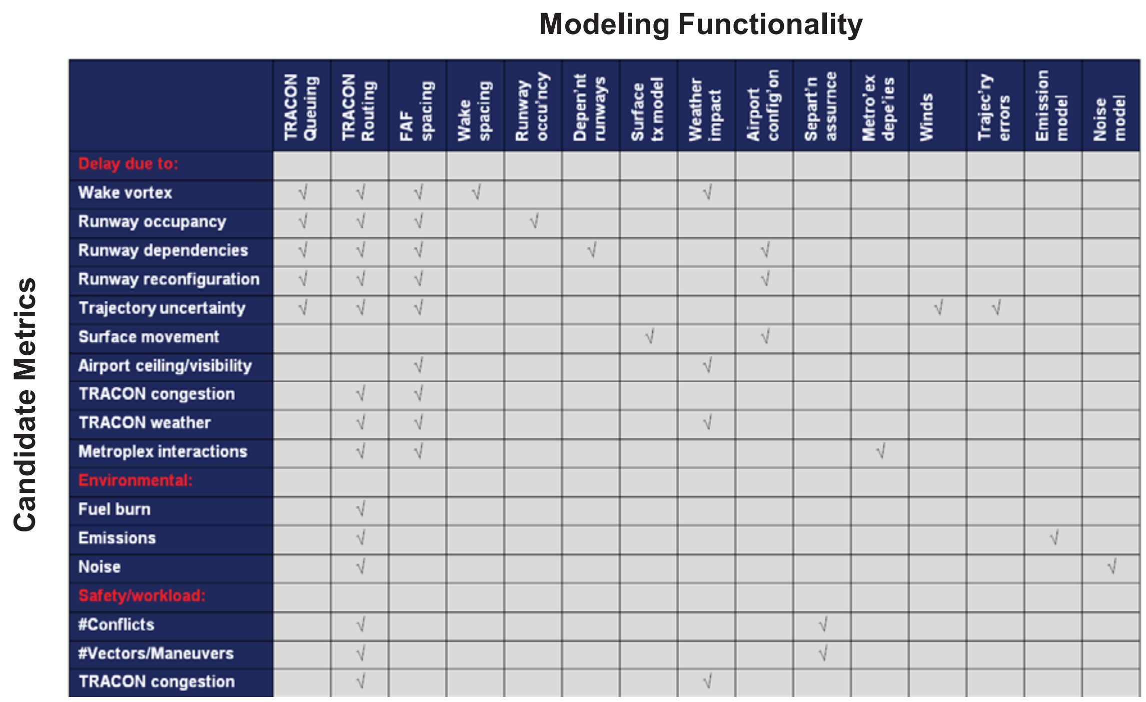


Figure 1 Probabilistic NAS Platform client-server architecture.

2.2 Overview of terminal area modeling

Terminal area modeling is a large problem. It may include, for instance, TRACON (Terminal Radar Control) routing, final approach fix and wake vortex spacing, runway occupancy, surface traffic, various weather impacts, airport configuration, separation assurance, metroplex dependencies, wind effects on trajectories, various trajectory prediction errors, and different environmental impacts. Figure 2 summarizes these models and some respective metrics that could be used.


Modeling Functionality

Candidate Metrics

	TRACON Queuing	TRACON Routing	FAF spacing	Wake spacing	Runway occu'ncy	Depen'nt runways	Surface tx model	Weather impact	Airport config'on	Separt'n assurnce	Metro'ex depe'ies	Winds	Trajec'ry errors	Emission model	Noise model
Delay due to:															
Wake vortex	√	√	√	√				√							
Runway occupancy	√	√	√		√										
Runway dependencies	√	√	√			√			√						
Runway reconfiguration	√	√	√						√						
Trajectory uncertainty	√	√	√									√	√		
Surface movement							√		√						
Airport ceiling/visibility			√					√							
TRACON congestion		√	√												
TRACON weather		√	√					√							
Metroplex interactions		√	√								√				
Environmental:															
Fuel burn		√													
Emissions		√												√	
Noise		√													√
Safety/workload:															
#Conflicts		√								√					
#Vectors/Maneuvers		√								√					
TRACON congestion		√						√							

Figure 2 Overview of terminal area modeling and metrics.

Figure 3 shows an example architecture of how these various models could interact.

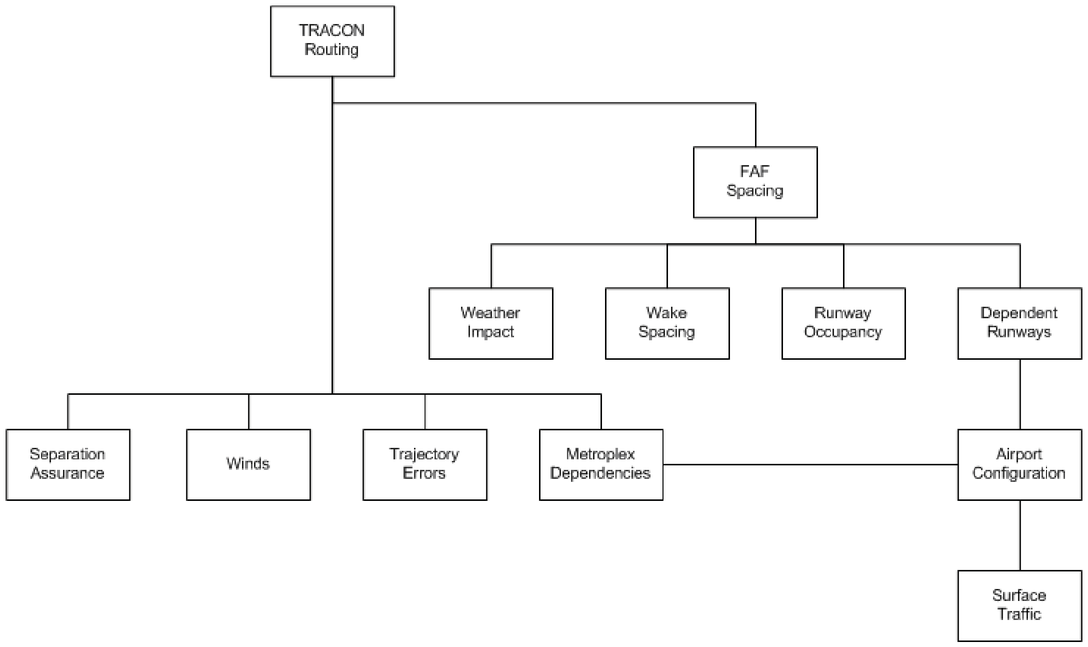


Figure 3 Hypochart summarizing relationships between terminal area models.

In keeping with our "build-a-little, test-a-little" development approach, this contract does not develop a complete terminal area for PNP. Instead, here we first select foundational models and data that together provide a working, albeit limited, terminal capability. This first set of models includes algorithms for realistic TRACON routings which are smoothly integrated into existing flight plans, runway assignments and wake vortex spacing. This work also includes the collection and use of NAS operational data at major terminal areas to populate the models and obtain realistic results.

This design document establishes the requirements and specific design approach for each of these models and data collection. Therefore, the focus and emphasis of our design document is on these foundational models and data. However, we also consider the greater context of terminal modeling, since it is crucial to design these models so that they lay the groundwork for the other models.

We also ensure that our design approach is not tied to any particular algorithm or strategy we implement. For instance, we will implement a runway assignment logic and algorithm in Task 3, but alternative solutions can be used instead.

2.3 Limitations

As Section 2.2 discusses, terminal area modeling is extensive. In this effort we begin with the important, foundational terminal area modeling components. We do not model, however, several elements of the TRACON. These elements include:

- No modeling of the impact of convective weather on terminal area routes in the TRACON and the extended terminal area.
- Limited modeling of site-dependent trends, such as how the runways at major airports are divided between arrivals and departures.
- No airport reconfiguration model indicating when and how to reconfigure.
- No modeling of runway dependencies.
- No modeling of terminal area fuel burn.
- No modeling of terminal area emissions and noise.
- No modeling of terminal area safety considerations.
- No modeling of runway occupancy.
- No modeling of the surface movement phase.
-

Also, there are certain limitation within the modeling that is implemented in this work. These limitations include:

- At this time metroplex interactions are not modeled.
- At this time runways may not serve both arrivals and departures within given 15 minute time slot.
- At this time airport reconfiguration events must occur on the 15 minute time bin boundaries.

- At this time airport reconfiguration events must be declared at least 45 minutes prior to occurring.
- At this time we do not adjust flights once they are within 30 minutes of landing (30 minute freeze horizon).

3 Requirements

This section lists the software and data requirements, divided into the major task categories.

3.1 User-specifiable terminal area routes

1. PNP shall be augmented with a new airport model that models the individual runways and supports terminal area modeling including: terminal area routes, separation assurance in the terminal area, runway balancing and wake vortex final approach spacing.
2. Realistic arrival terminal routes (ATRs) and departure terminal routes (DTRs) shall be derived for a common configuration at least three major airports.

3.2 Merge en route and terminal routes

3. Flights shall be assigned an initial ATR and DTR, which augment their flight plan.
4. PNP clients shall be able to change (reassign) ATRs and DTRs to flights, dynamically during the simulation run.
5. ATRs and DTRs shall be mergeable with the flight plan so that the new flight plan includes an ATR and a DTR in addition to the original flight plan, and the result is a realistic flight trajectory

3.3 Runway assignments

6. A new PNP client shall be created to assign runways dynamically to arrivals and departures, accounting for the relative runway use.

3.4 Enforce wake spacing

7. A new PNP client shall be created that enforces a minimum spacing between the consecutive arrivals and consecutive departures, for each runway.
8. A wake vortex minimum spacing utility shall be used to determine realistic wake vortex separation minimums.

3.5 Operational terminal area data

9. Airport site specific data shall be collected, processed and analyzed to support the creation of realistic terminal area modeling.

4 Design considerations

This section describes the design considerations for the major tasks.

4.1 User-specifiable terminal area routes

In this task, we modify PNP to allow the user to specify detailed arrival and departure routes and their associated runways for any airport in the simulation. This tasks includes the following subtasks:

- Define a new PNP airport model to support terminal area routes and wake vortex spacing.
- Populate the new PNP airport models, for particular airports, with the necessary data.

4.1.1 Airport runway model

In PNP, airports have been defined as a node, where the airport capacity is modeled as a Pareto curve indicating how the arrival and departure capacities are related, in different conditions. We now introduce an enhancement to the airport model which models the specific runways at an airport. We refer to the original model as the *node model*, and the new model as the *runway model*.

All airports will continue to be modeled using the node model, as before. But now, an airport's model may be augmented with the runway model. Therefore, the runway model is an optional fidelity enhancement.

<u>Why is the runway model optional?</u> Ultimately we intend to create runway models for many of the airports in PNP. We believe this runway model will be important for the larger airports, but likely will not be important for the smaller airports. Also, initially we likely will create runway models for a limited number of airports. Therefore, we make the runway model optional because (i) we do not believe it is required for the smaller airports and (ii) we want to test and evaluate this feature with our initial airport runway models.

<u>Why does the runway model augment the node model rather than replace it?</u> Even if a runway model exists for an airport, there may be clients that perform traffic planning using a node model. This is because the node model tends to be strategic and the runway model tends to be tactical. The node model is important for long term TFM planning, say several hours into the future. In this timeframe, overall airport loading is managed, even though specific runway assignments are unknown. In fact making specific runway assignments in the strategic time frame could be a meaningless exercise. Therefore the node model is available to strategic planners, such as the ProbTFM client, and the runway model is available to tactical planners, such as the RunwayAssignmentAndSpacing client (Section 4.4). Note that the node model can be adjusted if it does not accurately match the runway model. Also, strategic planners, such as ProbTFM, can follow a more loose TFM policy. In this case the node model would have less influence.

Details of the airport runway model. In the runway model, an airport has one or more configurations. In each configuration, a set of runways is defined and for each and every runway, arrival terminal routes (ATRs) and departure terminal routes (DTRs) are defined. During the PNP simulation, runways are dynamically assigned to serve as either an arrival or a departure runway. A runway may only serve as an arrival or a departure runway in a given time bin (a future enhancement is to allow for mixed usage runways). Obviously when a runway serves as an arrival runway, then the ATRs that lead to it are used, and the DTRs that come from it are not used. Likewise when a runway serves as a departure runway, then the ATRs that lead to it are not used, and the DTRs that come from it are used.

Arrival runways are defined by two waypoints: the outer marker and the runway threshold. The outer marker is 5 nmi upstream from the runway threshold. All ATRs must terminate with the outer marker and runway threshold waypoints, associated with the runway the flight is using. The use of these two waypoints, rather than merely a single waypoint at the runway, ensures that flights are properly lined up on the final approach.

Likewise, departure runways are defined by two waypoints as well: a point on the runway, such as the runway threshold, and a departure fix located approximately 3 nmi from the runway, in the direction that the flights depart. Table 1 summarizes the parameters in the runway model.

Table 1 Airport runway model parameters.

Runway model parameters
Number of configurations defined.
For each configuration, number of runways.
For each runway, the number of ATRs and DTRs.
For each ATR, the strategic RAI (Section 4.3.1).
For each ATR, the tactical RAI (Section 4.3.1).
For each route, the number of waypoints.
For each route, the index or identifier of the runway it uses.
For each waypoint, the latitude, longitude and optionally the altitude values.

Note that a given metering fix may have multiple ATRs which appear identical, except that toward the end they fly to a different outer marker/runway combination. Also, a given metering fix may have multiple ATRs which fly to the same outer marker/runway, but use different routes to get there.

For graphical display purposes, the runway model may also contain geometrical data of the airport and terminal area, such as the runway boundaries and TRACON boundary.

4.1.2 Candidate route construction methods and method selection

This section discusses how we construct terminal area routes. We describe three candidate methods for constructing terminal area routes and explain how we use a combination of them.

Published routes. Published terminal area routing data, such as the standard departure and arrival routes (i.e., SIDs and STARs) are a valuable source of information on how flights approach and depart major airports. These routes, however, often do not provide the complete trajectory from the airport to the en route airspace. For example, Fig. 4 shows the SIDs and STARs for the Dallas-Ft. Worth (DFW) airport.

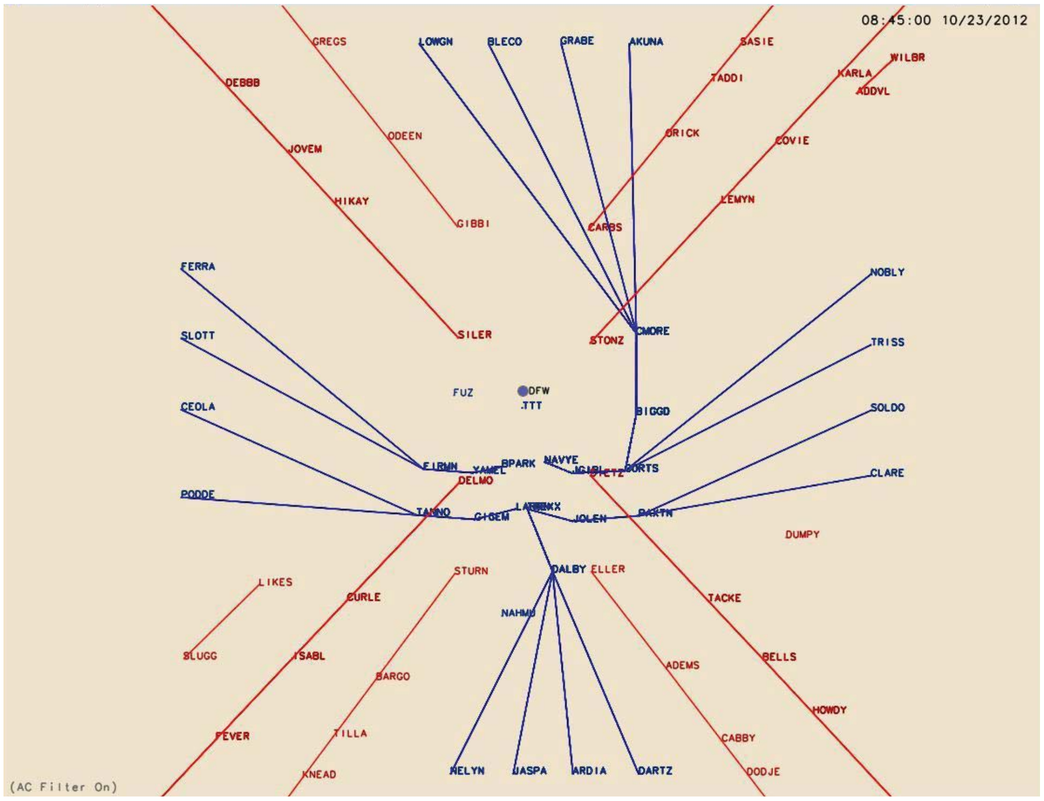


Figure 4 DFW STARs (red) and SIDs (blue).

Figure 4 shows that the SIDs and STARs, and in particular the STARs, do not provide the detailed TRACON routings leading to the DFW airport.

Previous studies. Another source of terminal area routing data are the previous studies that have constructed such routes for various purposes. In fact we at Saab-Sensis have been heavily involved in airport modeling, including the terminal area routes. Figure 5 shows an example of south flow routes for both DFW arrivals and departures we have constructed.

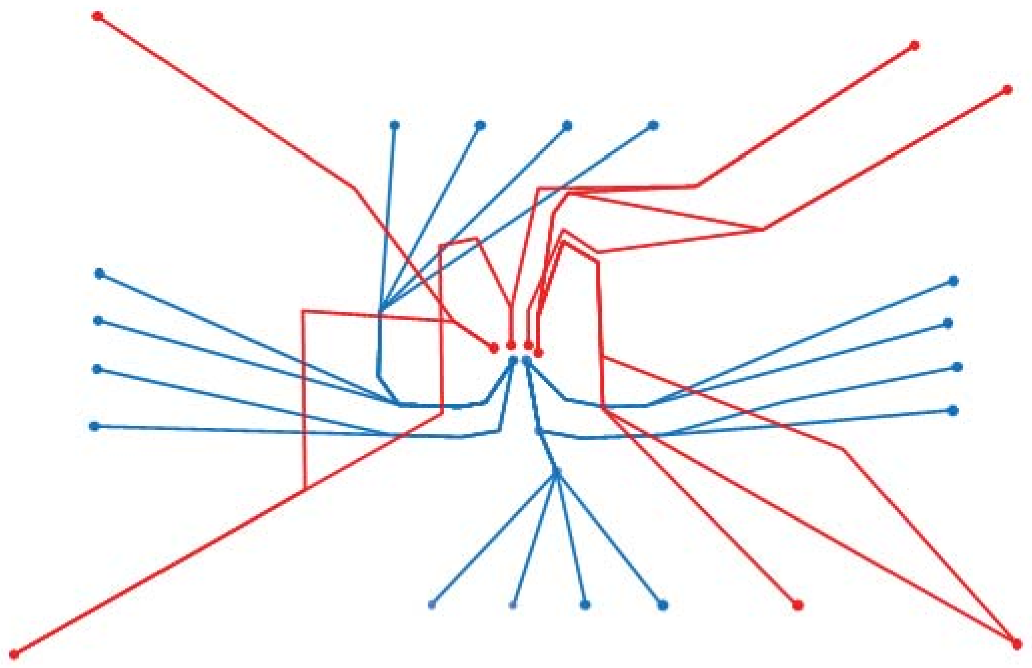


Figure 5 Example of previously derived DFW arrival (red) and departure (blue) routes for the south flow configuration. [24]

While the Fig. 5 routing data are of the appropriate level of detail and accuracy for our requirements, this is unfortunately the exception rather than the rule. In our literature search of both published and unpublished terminal routing data, we found data with a wide range of completeness and level of detail. What we did not find is a database of terminal area routes for most or all of the major airports, with the Fig. 5 level of detail, and for the major configurations used. While previous work can assist us in particular cases, such as in this DFW example, it is a limited resource.

Automated analysis of radar data. We can design terminal area routes empirically using an automated analysis of radar tracking data. There are two major challenges with this approach: reconstructing the trajectories and identifying nominal routes.

Our radar tracking data sources are ASDI and ASDE-X, neither of which provide a complete radar track from airport runway to the TRACON boundary. ASDI tracking data sometimes cuts off in the TRACON. In such cases it does not include the entire trajectory, all the way to, or from, the runway. This occurs, for instance, when an arrival descends and goes over the horizon, when observed from an en route tracking radar. ASDE-X tracking data, on the other hand, includes the segment of the flight near the airport but may cut off in the middle of the TRACON.

Therefore, to reconstruct a complete radar track from airport runway to the TRACON boundary, we could identify flights in the ASDE-X and ASDI tracking data, associate the same flight in those two different data sources, and combine those two radar tracks to form a single track. Of course, in general, the two different radar tracks overlap, or there is a gap between them. So the process of combining the two radar tracks is complicated.

The second challenge is to identify nominal routes. That is, given a large number of radar tracks for arrivals and departures from several weeks or months of ASDI and ASDE-X tracking data, we must identify a small set of standard arrival and departure routes. And for each route we must identify the runway used.

One method to identify these nominal routes is to cluster the radar tracks, and select the centroid track within each cluster. Such clustering requires that we define a difference metric, that quantifies the distance between routes. One method that has been used with success is to draw straight lines between (i) the two initial waypoints and (ii) the two final waypoints, of the two respective tracks. This produces a polygon and the difference metric is computed as the area of the polygon.

Once the tracks have been clustered, the centroid track of each cluster is found as the track that has the minimum sum of root mean square distances to all the other tracks in the cluster.

There are several complications to such clustering methods. First, the all-against-all distances must be computed between all the radar tracks. Next the number of expected clusters must be guessed. And the clusters need to be evaluated manually before used. Small clusters may be rejected and clusters may inadvertently include radar tracks that clearly are not using the same route. This can occur, for example, if the number of specified clusters is too small. So several cluster runs are typically needed to ensure reasonable results. Also, the final centroid track selected may need to be smoothed.

<u>Our route construction method</u>. The three different methods discussed above can all assist us in creating terminal area routes, but they all have important drawbacks as well. The published routes often do not provide complete TRACON routes. Previous studies are of inconsistent detail, fidelity and coverage. And the automated analysis of radar data presents several complexities.

After examining these three possible methods, we concluded that the best approach is to construct the ATRs and DTRs manually using an all-source approach. Previous studies can provide valuable input data for certain airports, but in most cases our baseline approach is to synthesize the published SIDs and STARs with our radar data in a manual analysis.

In this approach we examine (i) the published route data and (ii) archived radar data. By looking at both sources we use engineering judgment to derive a set of arrival and departure routes representative of the typical routings used in practice. In this approach it is important that the archived radar data are selectively graphed so that the graph shows typical routes. In particular, we narrow the time window of the data such that it represents only a single airport configuration. Also, we graph only arrivals or only departures, on one particular graph. Figure 6 shows an example 5-hour time window of DFW south flow radar tracks for arrivals.

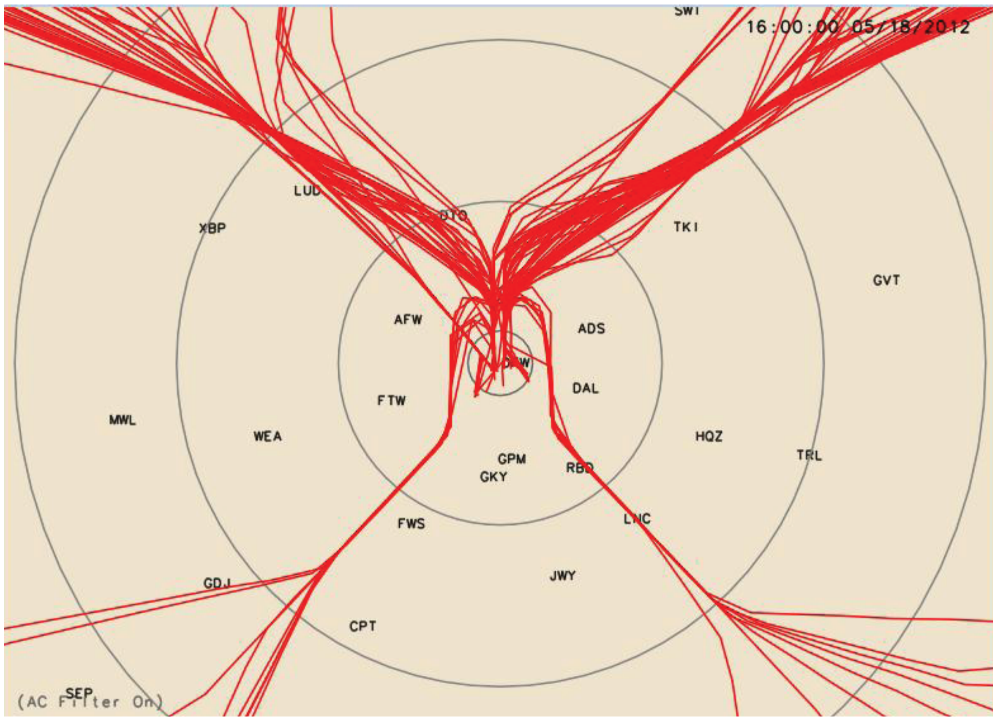


Figure 6 DFW sample arrival radar tracks from ASDI, 16:00-21:00, 5/18/2012.

Radar data from a long time window is often confusing. Such graphs include both arrivals and departures from multiple configurations, and the many anomalous routings that are used, for example when weather impacts the terminal area. Therefore it is important to use graphs of selected radar data. We use these graphs, in conjunction with the published routes to synthesize a set of representative, realistic, ATRs and DTRs in different configurations. We also use our geographic airport runway data to construct the runway and final/departure approach fix waypoints.

In this approach we use what we refer to as a *fixed route* method. A *variable route* contains segments that may vary. For instance, the downwind leg of an ATR may vary in length, which in turn causes the final approach leg to vary. This can be used to control the outer marker arrival time and the final approach spacing. Also, turns can vary which alters the subsequent segments as well. Variable route schemes add complexity both to the route construction process and to the algorithms using the routes. For example, geometrical calculations and logic are required to compute the route given the settings for the variable segments.

Fixed routes, on the other hand, have no variable segments. The geometry of fixed routes is constant. To model routing variations, such as the final approach extensions just described, one must use multiple fixed routes.

Therefore, in our set of ATRs for a given airport configuration, we include both (i) the different ATRs that come from the different metering fixes and (ii) the minor differences that a particular route can have. For these minor differences, we construct different routes. These differences are typically toward the end of the route, as the flight approaches the airport. For example, a particular ATR may use a final approach extension, as discussed, or it may feed multiple runways. In either case, we create multiple ATRs to represent these differences. As we construct these routes, we populate their RAI mappings (Section 4.3.1).

Once constructed, these routes may be used in PNP by inserting the list of waypoints into the flight plan for each flight. Obviously the DTR would be inserted at the beginning of the flight plan route, and the ATR at the end. Or, instead of inserting the terminal routes, pointers may be inserted into the flight plan indicating which routes are used. In addition to the waypoints, the particular runway that is used is also stored, as part of the DTRs and ATRs. This provides the runway identity to clients that perform terminal area rerouting, runway balancing and spacing.

4.2 Merge en route and terminal routes

In this task, we will develop a method of merging en route trajectories with the ATRs and DTRs resulting in seamless trajectories from en route to the ground.

A typical demand set contains a wide variety of en route trajectories, some of which may not be well aligned with the ATR or DTR to which the flight is assigned. In such instances, merger logic is needed to modify the initial or final en route segment so that it does smoothly proceed from and proceed to the DTR and ATR, respectively. Simply put, some adjustment may be required to ensure a smooth transition between the en route trajectory and the TRACON trajectory. We offer two solutions to this merger problem.

4.2.1 Great circle routes

A straightforward merging strategy is simply not to use the filed flight plan route. Instead, the aircraft flies a great circle route, connecting the DTR and ATR.

4.2.2 Filed flight plan routes

Great circle routes are a straightforward solution to the merger problem, but in some studies the filed flight plan route will be preferred. In that case, we merge that route with the DTR and ATR by trimming each end of the filed flight plan route, if necessary, so that it does not overlap with the two terminal routes. We do this by ensuring that the distance, from the airport, to the first waypoint in the filed flight plan is at least 1.5 times the distance, from the airport, to the final waypoint in the DTR.

Similarly, at the other end we ensure that the distance, from the airport, to the final waypoint in the filed flight plan is at least 1.5 times the distance, from the airport, of the first waypoint in the ATR[2]. If either of these conditions are not met, then the filed flight plan route is trimmed until the condition is met.

Note that we do not allow round-robin flights, which otherwise could pose difficulties for this logic. Also, flights of very short distance could pose difficulties as their departure and arrival routes could overlap. In this case all the flight plan waypoints should be trimmed. In any case, when there are no waypoints in the final flight plan route, then the logic reverts to the 4.2.1 great circle strategy.

4.3 Runway assignments

In this task, we create the terminal area route architecture that will be used in the RunwayAssignmentAndSpacing client (Section 4.4).

4.3.1 Arrivals

For each airport configuration, each ATR has an identifier, such as an index. For example, for *N* ATRs the index could have values 1 through *N*, so that the ATRs are numbered 1 through *N*. All flights in PNP always have an assigned ATR. A flight's initial ATR is assigned off-line, prior to the PNP run, and stored in the input demand set. Any logic may be used to determine the best ATR for the flight. For instance, the ATR that is closest to the flight's great circle route may be selected.

During the PNP simulation, clients may change a flight's ATR assignment. To assist clients with ATR selection, we introduce the concept of the route availability index (RAI). The RAI is an index that points to a list of preferred ATRs for that flight. At any given simulation time, during the PNP simulation, clients may use a flight's RAI to fetch the list of preferred ATRs.

<u>Where does a flight's RAI come from?</u> The RAI value comes from the ATR that the flight is assigned to. Every ATR has two RAIs associated with it: the *strategic RAI* and the *tactical RAI*. Clients may use both these values, as described below.

<u>Why do ATRs have two RAI values?</u> The purpose of the strategic RAI is both to save compute time and add realism. For instance, an en route flight from the northeast may need to be reassigned to a different ATR. Such a route reassignment may be needed for runway balancing or separation. But when evaluating the alternatives, ATRs from the southwest (on the opposite corner of the TRACON) should not be considered. In fact, we would like to narrow the list of eligible routes to only the other northeast ATRs. This saves compute time and adds realism. Therefore, the strategic RAI for a northeast arrival

[2] The value of 1.5 is tentative.

route, for example, points to a list which contains all the ATRs from the northeast, but no other ATRs.

But when the flight is closer to the destination airport (such as when the flight is inside or near the destination TRACON), then reassignments should be narrowed to an even shorter list of ATRs that are very similar to the assigned ATR. For instance, such a tactical reassignment could be limited only to ATRs that are identical to the assigned route, except for minor variations at the end of the route, such as final approach extensions or a switch to a neighboring runway. The tactical RAI points to a list of these candidate ATRs. Therefore, whereas the purpose of the strategic RAI is both to save compute time and add realism, the tactical RAI has the additional purpose of facilitating last-minute delays or runway changes.

To summarize, every arrival route has two RAI values, the strategic RAI and the tactical RAI. The strategic RAI points to a list of ATRs that are similar to the route at the strategic level. The tactical RAI points to a list of ATRs that are similar to the route at the tactical level.

<u>What are the ground rules for using the RAIs?</u> In general, clients are responsible for ensuring realism and feasibility when reassigning ATRs to flights. For instance, if a en route flight's current great circle distance to the destination airport is shorter than that of the initial waypoint of an ATR (i.e., the flight is currently "inside" of the ATR), then the flight probably should not be switched to that ATR.

To assist clients, the strategic and tactical RAIs adhere to a simple ground rule. For the strategic RAI, it points to ATRs that are feasible when the flight's great circle distance to the destination airport is greater than 50 nmi. For the tactical RAI, it points to ATRs that are feasible when the flight's great circle distance to the destination airport is greater than 20 nmi[3].

In this current version of PNP we implement both the strategic and tactical RAIs (Section 4.1), but in this version we do not use the tactical RAIs in the client decision making. The tactical RAIs are available for future enhancements in the client decision making.

4.3.2 Departures

For each airport configuration, as with ATRs, each DTR has an identifier such as an index. For *N* DTRs, the index could have values 1 through *N*. All flights in PNP always have an assigned DTR. A flight's initial DTR is assigned off-line, prior to the PNP run, and stored in the input demand set. Any logic may be used to determine the best DTR for the flight. For instance, the DTR that is closest to the flight's great circle route may be selected. During the PNP simulation, clients may change a flight's DTR assignment.

[3] The value of 20 and 50 nmi are tentative.

4.4 Enforce wake spacing

In this task, we develop a client that performs both runway assignments and spacing for both arrivals and departures at an airport. The client's name is RunwayAssignmentAndSpacing and it evaluates upcoming arrivals and departures at an airport. For arrivals the client adjusts the ATRs and outer marker arrival times. For departures the client adjusts the departure times and runway assignments. The departure runway assignments are changed by assigning a different DTR. These adjustments are made to balance the runway use and maintain proper final approach spacing for arrivals and runway spacing for departures.

The RunwayAssignmentAndSpacing client function logic is as follows.

The client operates on one airport at a time. It operates on a future time window defined by user specifiable start and stop times, relative to the current time. Default values are 30 and 45 minutes. Therefore the default time window consists of the third time bin into the future (or equivalently, the second time bin following the current time bin).[4]

The client fetches the list of all arrival and all departure flights scheduled to arrive and depart, respectively, at the airport, in the time window. `NArrivals` is set to the number of arrivals and `NDepartures` is set to the number of departures.

The client fetches the airport configuration defined for this time window. Because reconfiguration events must occur at the boundaries of time bins, the client is guaranteed that the configuration will be constant for the current run. `NRunways` is set to the number of runways.

The client assigns each runway to be either an arrival or departure runway. First, it computes the number of desired arrival and departure runways as follows:

```
NArrivalRunways = ( float(NArrivals)/float(NDepartures+NArrivals) )*NRunways;
if(NArrivalRunways >= NRunways) NArrivalRunways = NRunways - 1;
NDepartureRunways = NRunways - NarrivalRunways;
```

Next the client assigns each runway to be either an arrival or departure runway, such that the total number of arrival runways equals `NArrivalRunways` and the total number of departure runways equals `NDepartureRunways`. Ideally site adaptation information would be used to follow runway assignment trends at each airport. For now, we make the specific runway assignments at random.

[4] Note that it is possible that very short flights may depart inside this 30 minute time window.

Now the client is ready for runway balancing and spacing. The client first handles arrivals and then departures (although this ordering is arbitrary).

The client loops through all arrival flights, in order of their scheduled arrival time at the outer marker.[5] For each flight, if its ATR leads to a runway which currently is assigned as a departure runway, then the route's strategic RAI is used to find a different ATR which leads to an arrival runway. The flight is assigned this ATR. At this time, when multiple ATRs are available, any of the ATRs may be selected.

For that assigned runway, the time spacing between the flight and the preceding flight (on the same runway) is computed, at the outer marker.

For the given aircraft types, the minimum required wake vortex spacing is computed.

If there is no spacing violation (i.e., if the computed spacing is greater than the minimum required spacing), then execution goes back to the top of the loop, and the next flight is considered.

If there is a spacing violation, then the route's strategic RAI is used to find an alternate ATR with no spacing violation. Of the available alternatives, the ATR is selected which gives the flight the earliest arrival time at the outer marker.[6]

If no such alternative ATR is available (i.e., all the alternate ATRs also have spacing violations), then the ATR with the smallest spacing violation (i.e., the least difference between the computed spacing and the minimum required spacing), is selected. Then the RTA utility is used to delay the flight such that the outer marker spacing will meet the minimum required spacing.

If the RTA fails, then we simply use the ATR with the smallest spacing violation.

Execution goes back to the top of the loop, and the next arrival flight is considered.

Next the client loops through all departure flights, in order of their scheduled departure times at the runway. For each flight, if its DTR uses a runway which currently is assigned as an arrival runway, then the flight is assigned to a departure runway. At this time, when multiple runways are available, any of the runways may be selected. And the DTR, from the selected runway, is selected using the same logic as in the initial route assignment (e.g., select the DTR that is closest to the flight's great circle trajectory).

For the assigned runway, the time spacing between the flight and the preceding flight (on the same runway) is computed, at the runway.

[5] This scheduled arrival time could be in reference to the runway, rather than the outer marker, if necessary.
[6] This estimated arrival time could be in reference to the runway, rather than the outer marker, if necessary.

For the given aircraft types, the minimum required wake vortex spacing for departures is computed. For now, we can use the same spacing computation used for arrivals.

If there is no spacing violation (i.e., if the computed spacing is greater than the minimum required spacing), then execution goes back to the top of the loop, and the next departure flight is considered.

If there is a spacing violation, then the other departure runways are considered, using the scheduled departure time. Any one of the available runways where there is no spacing violation (i.e., the computed spacing is greater than the minimum required spacing) is used. The selection can be random. A new DTR is assigned, corresponding to the new runway assignment.

If no such alternative runway is available (i.e., all the alternate runways also have spacing violations), then the runway with the smallest spacing violation (i.e., the least difference between the computed spacing and the minimum required spacing), is selected. Then the flight's departure time is delayed such that the runway spacing will meet the minimum required spacing. A new DTR is assigned, corresponding to the new runway assignment.

Execution goes back to the top of the loop, and the next departure flight is considered.

Note that for both the arrivals and departures, the first (earliest) arrival and departure, at each runway, will require the corresponding last flight from the previous time bin, in order to compute the spacing. Note also that if the runway has switched from a departure to an arrival runway, or vice versa, due to a reconfiguration event or merely due to a runway assignment change, then there is no preceding flight.

Note that, as mentioned in the above algorithm, reconfiguration events are restricted to occur at the boundaries between time bins, so there is no need to consider the possibility of a reconfiguration event in this client. Also reconfiguration events must be declared at least 45 minutes in advance, so the client has the correct configuration to work with.

Finally, note that a wake vortex minimum separation requirement utility is required to compute the minimum inter arrival spacing for arrivals and departures. As an intermediate solution, we can use the same wake vortex minimum separation requirements for departures as for arrivals.

4.5 Operational terminal area data

In this task, we collect and process the various airport and terminal area data. The design considerations have already been discussed in the previous sections.

5 Future enhancements

This section lists enhancements that are outside the scope of this effort but may be implemented in the future. There are many such enhancements. Here we list the major items which these enhancements would support:

- Allow for mixed use runways (i.e., both arrivals and departures in the same time bin).
- Adjust TRACON trajectories (e.g., adjust routes or apply speed or altitude control) for flights that are currently within the TRACON to support final approach spacing management and/or for separation assurance within the TRACON.
- Model the impact of convective weather on terminal area routes in the TRACON and the extended terminal area.
- Incorporate site-dependent trends, such as runway and metering fix use.
- Create an airport reconfiguration model based on airport-specific trends.
- Add trajectory uncertainty.
- Model runway dependencies.
- Model terminal area fuel burn.
- Model terminal area emissions and noise.
- Model terminal area safety considerations.
- Model metroplex interactions.
- Model runway occupancy.
- Include surface movement phase.

6 Testing

Each of the requirements in Section 3 will be verified either by demonstration, analysis or inspection.

7 Conclusions

This document describes software enhancements required to support NASA air traffic control experiments to evaluate advanced terminal area concepts in a NAS-wide context.

References

1. George Hunter, "Testing and Validation of NextGen Simulators," AIAA Modeling and Simulation Conference, Chicago, IL, August 2009.

2. George Hunter, Kris Ramamoorthy, "Integration of terminal area probabilistic meteorological forecasts in NAS-wide traffic flow management decision making," 13th Conference on Aviation, Range and Aerospace Meteorology, New Orleans, LA, January, 2008.

3. Kris Ramamoorthy, George Hunter, "The Integration of Meteorological Data in Air Traffic Management: Requirements and Sensitivities," 46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January, 2008.

4. George Hunter, Ben Boisvert, Kris Ramamoorthy, "Advanced Traffic Flow Management Simulation Experiments and Validation," 2007 Winter Simulation Conference, Washington, DC, December, 2007.

5. Kris Ramamoorthy, George Hunter, "Evaluation of National Airspace System Performance Improvement With Four Dimensional Trajectories," AIAA Digital Avionics Systems Conference (DASC), Dallas, TX, October, 2007.

6. George Hunter, Ben Boisvert, Kris Ramamoorthy, "Use of automated aviation weather forecasts in future NAS," The 87th American Meteorological Society Annual Meeting, San Antonio, TX, January, 2007.

7. Kris Ramamoorthy, George Hunter, "Probabilistic Traffic Flow Management in the Presence of Inclement Weather and Other System Uncertainties," INFORMS Annual Meeting, Pittsburgh, PA, November, 2006.

8. Kris Ramamoorthy, Ben Boisvert, George Hunter, "A Real-Time Probabilistic TFM Evaluation Tool," AIAA Digital Avionics Systems Conference (DASC), Portland, OR, October, 2006.

9. Kris Ramamoorthy, George Hunter, "A Trajectory-Based Probabilistic TFM Evaluation Tool and Experiment," Integrated Communications, Navigation and Surveillance Conference (ICNS), Baltimore, MD, May, 2006.

10. Kris Ramamoorthy, George Hunter, "Avionics and National Airspace Architecture Strategies for Future Demand Scenarios in Inclement Weather," AIAA Digital Avionics Systems Conference (DASC), Crystal City, VA, October, 2005.

11. George Hunter, Kris Ramamoorthy, Joe Post, "Evaluation of the Future National Airspace System in Heavy Weather," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Arlington, VA, September 2005.

12. George Hunter, "Probabilistic Forecasting of Airport Capacity," AIAA Digital Avionics Systems Conference (DASC), Salt Lake City, UT, October, 2010.

13. Poornima Balakrishna, George Hunter, "Preliminary NextGen Collaborative Air Traffic Management Analysis," AIAA Digital Avionics Systems Conference (DASC), Salt Lake City, UT, October, 2010.

14. Huina Gao, George Hunter, Frank Berardino, Karla Hoffman, "Development and Evaluation of Market-Based Traffic Flow Management Concepts," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Dallas, TX, September 2010.

15. Huina Gao, George Hunter, "Evaluation of User Gaming Strategies in the Future National Airspace System," AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Dallas, TX, September 2010.

16. George Hunter, "Meta Simulation Results for Simultaneous Dynamic Resectorization and Traffic Flow Management," AIAA Digital Avionics Systems Conference (DASC), Orlando, FL, October, 2009.

17. Huina Gao, George Hunter, "Future NAS-Wide User Gaming Preliminary Investigation," AIAA Digital Avionics Systems Conference (DASC), Orlando, FL, October, 2009.

18. George Hunter, "Preliminary Assessment of Interactions Between Traffic Flow Management and Dynamic Airspace Configuration Capabilities," AIAA Digital Avionics Systems Conference (DASC), St. Paul, MN, October, 2008.

19. George Hunter, "Toward an Economic Model to Incentivize Voluntary Optimization of NAS Traffic Flow," AIAA ATIO Conference, Anchorage, AK, September, 2008.

20. George Hunter, "Sensitivity of the National Airspace System Performance to Weather Forecast Accuracy," Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, May, 2008.

21. Kris Ramamoorthy, Ben Boisvert, George Hunter, "Sensitivity of Advanced Traffic Flow Management to Different Weather Scenarios," Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, May, 2007.

22. George Hunter, Ben Boisvert, "Modeling and Simulation of Interactions Between Traffic Flow Management and Separation Assurance," *AIAA Modeling and Simulation Conference*, Portland, OR, August 2011.

23. George Hunter, Ben Boisvert, Ty Marion, Jerry Smith, (in press) "Traffic Flow Management and Separation Assurance Simulation Experiment," *AIAA Digital Avionics Systems Conference (DASC)*, Williamsburg, VA, October 2012.

24. Sebastian Timar, Nadia Bess, “ACES-TME KDFW Terminal Airspace and Surface Design,” Saab-Sensis Corporation, draft.

14809-01

Upgrades to the Probabilistic NAS Platform Air Traffic Simulation Software

Software Development Plan

Ben Boisvert
Sensis Corporation

Prepared for:

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681

Under:

NASA BOA: NNL08AA17B

November, 2010

TABLE OF CONTENTS

1 Scope

This document addresses the requirements set forth in the NASA Procedural Requirements (NPR) Document 7150.2A. This NPR imposes requirements on procedures, design considerations, activities, and tasks used to acquire, develop, assure, and maintain software created and acquired by or for NASA programs.

1.1 Software Development Plan

This Software Development Plan (SDP) establishes the plan for software implementation, test, and qualification for the Upgrades to the Probabilistic National Airspace System(NAS) Platform (PNP) Air Traffic Simulation Software under the NASA BOA: NNL08AA17B.

1.2 Project Overview

The objectives of this effort are to upgrade and enhance the functionality of PNP to support the research requirements of the Systems and Portfolio Analysis (SPA) Research Focus Area (RFA) of NASA's Airspace Program Systems Analysis Integration and Evaluation (SAIE) Project. The milestones supported by this work are:

1. SAIE.SPA.4.01	Portfolio Analysis
2. SAIE.SPA.3.03	System Level Benefits Assessments of Combined Concepts
3. SAIE.SPA.4.02	Design Study 2
4. SAIE.SPA.2.05	System Constraints, Demand/ Capacity Analysis

1.3 System Overview

The Probabilistic NAS Platform (PNP) is both a research tool and a decision support platform to analyze air traffic and traffic flow management (TFM) issues. Real-time PNP provides meteorological data and traffic advisories in which an operator or Air Traffic Manager can make more informed and accurate decisions. Playback PNP evaluates current or future TFM concepts using projected or actual traffic demands sets with recorded historical meteorological data. The PNP system is to be upgraded to support Separation Assurance procedures. A Separation Assurance client is to be created to test and demonstrate separation assurance functionality in PNP.

1.4 Document Overview

This SDP defines the processes to be followed for all software activities necessary to accomplish the development.

1.5 Software Management/Organization

Sensis management/organization is a matrix of many professionals from different organizations. System Engineering will be responsible for the algorithmic and system design of this project. Network & Hardware Support will be responsible for the setting up of the repository and any hardware acquisitions necessary to complete this project. Integration and Test will be responsible for the project integration and test into the baseline PNP system. Software will be responsible for the development, documentation, test, quality, and delivery of the products on this project.

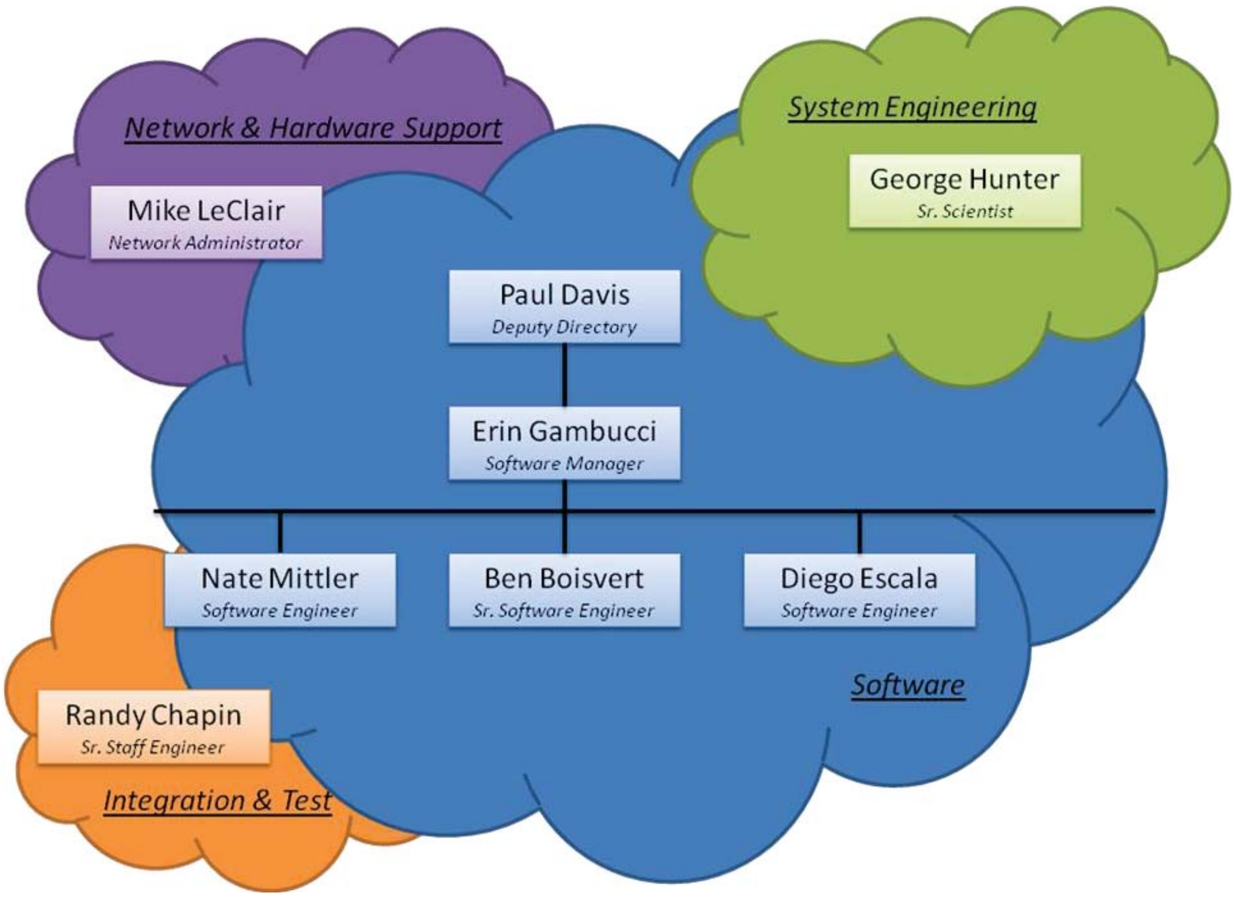


Figure 1: Sensis Matrix Organization for the Upgrades to the Probabilistic NAS Platform Air Traffic Simulation Software

1.6 Software Processes

Sensis will follow the following processes in line with the SDP described in NPR-7150.2A.

1.6.1 Requirements Phase

- System Engineering will develop an Engineering Design Document (EDD). The EDD conforms to an internal Sensis template which has been used for several years.

- Software Engineering will develop a Software Requirements Document (SRD) and Software Requirements Specification (SRS) according to the guidance provided by NPR-7150.2A.
- SRD and SRS will be peer reviewed and delivered to NASA for approval

1.6.2 Design Phase

Software Engineering will develop a Software Design Document (SDD) including a Interface Design Description (IDD) according to the guidance provided by NPR-7150.2A

- Software Engineering will review SDD for EDD compliance
- SDD will be peer reviewed and delivered to NASA for approval

1.6.3 Coding Phase

- Software Engineering will develop code according to the SDD
- Software Engineering will generate the software reports and update the software users manual as needed

1.6.4 Testing Phase

System/Software Engineering will develop a Software Test Plan with Software Test Procedures

1.6.5 Delivery Phase

- Software Engineering will package the software, manuals, and reports into a deliverable package
- Software Engineering will deliver deliverable to NASA for deliverable acceptance.

1.7 Software Procedures

Sensis will follow the following procedures in line with the established Sensis processes.

1.7.1 Requirements Phase

- All requirements documents will be identified in the SDP
- All documents will be developed in MS-Word adhering to any required contractual format requirements.
- All documents must be peer reviewed

1.7.2 Design Phase

- All design documents will be identified in the SDP

- All documents will be developed in MS-Word adhering to any required contractual format requirements.
- All documents must be peer reviewed

1.7.3 Coding Phase

- Software code will be developed in the IntelliJ IDE using the Sensis template.
- Software code will be controlled by subversion.
- Software unit tests will be written in JUnit.
- Software issues will be tracked using JIRA.
- Software performance issues will be analyzed with the YourKit profiler.
- Software quality will be monitored with Sonar.
- All software documents/reports will be identified in the SDP
- All software documents/reports will be developed in MS-Word adhering to any required contractual format requirements.

1.7.4 Testing Phase

- All testing documents/reports will be identified in the SDP
- All testing documents/reports will be developed in MS-Word adhering to any required contractual format requirements.

1.7.5 Delivery Phase

- Software Engineering will package the software, manuals, and reports into a deliverable package
- Software Engineering will deliver deliverable to NASA for deliverable acceptance.

2 Software Development (Life-Cycle Model)

Sensis will use an incremental (Multi-build) development life-cycle. The incremental development life-cycle is a variation of the traditional waterfall development life-cycle. Incremental development determines user needs and defines system requirements up front, then performs the rest of development in a sequence of builds. The first build incorporates part of the planned capabilities, the next build adds more capabilities, and so on, until the system is complete. The approach is best suited to projects in which the requirements are well defined and understood and the design and technology are proven and mature.

Table1: Build Schedule

Build	Description	Date
1	SA Client	Mar 2011
2	Stratway integration	Jun 2011
3	Final Delivery	Sep 2011

The advantages of the incremental development life cycle include:

- Reduced risk of schedule slips, requirements changes, and user acceptance problems

- Increased manageability
- Feedback from early builds can be incorporated into later builds
- End user / requirements changes can be incorporated before final version is delivered
- User validation of the product as it is being developed
- Deferred implementation of unstable requirements until after issues have been resolved
- Early operational training on interim versions of the product
- Early validation of operational procedures
- Well-defined checkpoints with customer and end users

2.1 Requirements Development

Sensis will create a Software Requirements Document (SRD). Sensis will identify, develop, document, approve, and maintain software requirements based on analysis of customer requirements, customer feedback, and the operational concepts. Sensis will perform software requirements analysis based on the requirements. The software requirements analysis determines the requirement's correctness, consistency, completeness, feasibility, and verifiability. The software requirements analysis activities include the allocation of functional, non-functional, and performance requirements to functions in a requirements matrix for tractability.

2.1.2 Software Requirements Specification

The SDD will include a Software Requirements Specification (SRS) Section. The SRS contains:

- CSCI requirements
- Qualification provisions (e.g., demonstration, test, analysis, inspection).
- Bidirectional requirements traceability.
- Requirements partitioning for phased delivery.

2.1.3 Requirements Management

Sensis will manage changes to the software requirements via versions of the Software Requirements Document. Cost, technical, and schedule impacts will be analyzed and reflected in the software schedule as needed. Sensis will identify and initiate corrective actions, and track requirements until inconsistencies are resolved. Sensis will perform requirements validation to ensure that the software will perform as intended. The acceptance test plan will be the vehicle to perform requirements validation.

2.2 Software Design

Sensis will create a Software Design Document (SDD) that defines the software architecture, components, modules, interfaces, and data for the software to satisfy the requirements. The Software Design Document will consist of software architectural and detailed design. Computer software configuration items (CSCIs) will be identified and requirements will be mapped to these CSCIs. Sensis will perform and maintain bidirectional traceability between the software requirements and the software design.

2.2.1 Software Design Document

The Software Design Document describes the design of a CSCI. It describes the CSCI-wide design decisions, the CSCI architectural design, and the detailed design required to implement the software.

The SDD will include a Software Design Description Section. The Software Design Description contains:

- CSCI architectural design.
- CSCI decomposition and interrelationship between components.

2.2.2 Interface Design Description

The SDD will include an interface design description section. The Interface Design Description describes the interface characteristics of CSCI's developed on this project. The Interface Design Description contains:

- Type of interface to be implemented.
- Specification of individual data elements.
- Specification of communication methods that the interfacing entities will use for the interface.

2.2.3 Software Implementation

Sensis will implement the software design into software code. Sensis will ensure that Sensis software coding methods, standards, and/or criteria are adhered to and verified. Sensis will unit test the software that is developed for this project. Sensis will perform and maintain bidirectional traceability from the software design and the software code. Sensis will perform software peer review/inspection on developed code and unit tests. Sensis will create a user manual for the separation client. Sensis will create Software Version Document (SVD) for each software release. Sensis will release software reports for every software release.

Existing tools that Sensis will utilize in its development:

- IntelliJ IDEA IDE – IntelliJ automates coding standardization and syntax checking for the Java programming language.
- JUnit – A unit test framework for the Java programming language.
- YourKit Profiler - A profiling framework to analyze CPU and memory usage.
- JIRA – An issue and project tracking framework for software development teams to improve code quality and the speed of development.
- Sonar - An open source quality management platform, dedicated to continuously analyze and measure technical quality, from the project portfolio to the class method.

2.2.4 Software Peer Review/Inspection Report

Sensis will perform and create software peer review/inspection reports.

The Software Peer Review/Inspection Report contains:

- Identification and review/inspection time and date.
- Summary on total time expended on review/inspection.
- Participant information.
- Total number of defects found.
- Peer review/inspection results summary (i.e., pass, re-inspection required).
- Listing of all review/inspection defects.

2.2.5 Software User Manual

Sensis will update the software user manual.

The software user manual contains:

- Software summary.
- Access to the software: first-time user of the software, initiating a session, and stopping and suspending work.
- Assumptions and limitations.

2.2.6 Software Version Description

The software version description identifies and describes a software version. The description is used to release, track, and control a software version.

The Software Version Description contains:

- Full identification of the system and software.
- Executable software.
- Instructions for building the executable software.
- Software product files
- Change requests and/or problem reports implemented in the current software version since the last Software Version Description was published.

2.2.7 Software Reports

Sensis will provide software reports at the end of each software release. The software metrics report provides data to the project for the assessment of software cost, technical, and schedule progress.

The software metrics report contains:

- Number of requirements included in a completed build/release (planned vs. actual).
- Number of software Problem Reports/Change Requests (new, open, closed, severity).
- Number of requirements verified.

- Results from static code analysis tools.

2.2.8 Software Quality

Sensis will provide software quality reports at the end of each release. The software quality report provides data to the project for the assessment of quality.

The software quality report contains:
- Lines of code/Classes
- Rule of compliance/Violations
- Comments/Duplications
- Complexity
- Package tangle index
- Code coverage

2.2.9 Software Safety

This program is classified as CLASS D and not Safety Critical therefore no safety plan will be developed.

2.2.10 Software Security and Privacy

Sensis maintains an Export Control System and the Stratway software is being controlled under that system.

2.3 Software Testing

Sensis will perform testing to verify the software functionality and remove defects. Sensis will perform software testing as defined in the Software Test Plan. The Software Test Plan will consist of Software Test Procedures that verify requirements. Sensis will evaluate test results and document the evaluation. Sensis will document defects identified during testing and track to closure. Sensis will update Software Test Plan(s) and Software Test Procedure(s) to be consistent with software requirements. Sensis will perform and maintain bidirectional traceability from the Software Test Procedures to the software requirements. Sensis will ensure that the software system is validated on the targeted platform.

2.3.1 Software Test Procedures

The Software Test Procedures describe the test preparations, test cases, and test procedures to be
used to perform qualification testing of a CSCI or a software system or subsystem.

The Software Test Procedures contain:
- Test preparations, including hardware and software.
- Test descriptions

- Requirements traceability.
- Identification of test configuration.

2.3.2 Software Test Report

The Software Test Report is a record of the qualification testing performed on a CSCI, a software system or subsystem, or other software-related item.

The Software Test Report contains:
- Overview of the test results.
- Detailed test results.
- Test log.

2.3.3 Verification and Validation

Sensis will perform verification and validation at the unit level. Unit tests will be automated through the use of JUnit. JUnit forces a complete test to be identified automate the verification and validation of the outputs.

2.4 Software Operations, Maintenance, and Retirement

Sensis will complete and deliver the software product to the customer with appropriate documentation to support the operations and maintenance phase of the software's life cycle.

3 Supporting Software Life-Cycle

This section describes the processes and procedures in place to support the software life-cyle.

3.1 Software Configuration Management

Sensis will develop a Software Configuration Management Plan that describes the functions, responsibilities, and authority for the implementation of software configuration management for the project. Sensis will track and evaluate changes to the software. Sensis will use version control to manage all configuration items developed for this project. Sensis will establish and implement procedures controlling each type of configuration item developed for this project. Sensis will prepare and maintain records of the configuration status of configuration items per software release. Sensis will establish and implement procedures for the storage, handling, delivery, release, and maintenance of deliverable software products. Sensis will create Software Change Request/Problem Reports.

3.1.1 Software Change Request/Problem Report

The Software Change Request/Problem Report shall contain:

- Identification of the software item.
- Description of the problem or change to enable problem resolution or justification for and the nature of the change, including: assumptions/ constraints and change to correct software error.
- Originator of Software Change Request/Problem Report and originator's assessment of priority/severity.
- Description of the corrective action taken to resolve the reported problem or analysis and evaluation of the change or problem, changed software configuration item, schedules, cost, products, or test.
- Life-cycle phase in which problem was discovered or in which change was requested.
- Approval or disapproval of Software Change Request/Problem Report.
- Date problem discovered.
- Status of problem.
- Configuration of system and software when problem is identified (e.g., system/software configuration identifier or list of components and their versions).
- Any workaround to the problem that can be used while a change is being developed or tested.

3.2 Risk Management

Sensis will identify, analyze, plan, track, control, communicate, and document software risks that put contract milestones at risk.

3.2.1 Software Peer Reviews/Inspections

Sensis will perform and report on software peer reviews/inspections for:
- Software Requirements Document.
- Software Design Document.
- Software Test Plan.
- Software code as developed on this project.

4 Conclusions

This document serves as the software development plan for the implementation, test, and qualification for the upgrades to the PNP Air Traffic Simulation Software under the NASA BOA: NNL08AA17B.

Modeling and Simulation of Interactions Between Traffic Flow Management and Separation Assurance

George Hunter[7] and Ben Boisvert[8]
Sensis Corporation, Campbell, CA, 95682

The integration of strategic and tactical actions in simulations of the national airspace presents several challenges. Yet such an integration is important for design evaluation and assessment of interactions between strategic and tactical decision making. Here we describe our modeling and integration of traffic flow management and separation assurance decision making in a national airspace simulation. We also present preliminary simulation results regarding the interaction between these two decision makers.

Nomenclature

ATC	=	Air traffic control
ATM	=	Air traffic management
CPMF	=	Capacity probability mass function
DAC	=	Dynamic airspace configuration
LPMF	=	Loading probability mass function
NAS	=	National air space
NextGen	=	Next generation NAS
PMF	=	Probability mass function
PNP	=	Probabilistic NAS Platform
SA	=	Separation assurance
TFM	=	Traffic flow management
TP	=	Trial plan

I. Introduction

Air traffic management services are often confronted with high-demand scenarios. Because excessive delays are costly to users, there is a substantial on-going effort to use intelligent strategies that minimize system delays even when demand is high. These intelligent strategies use models of system dynamics, forecasts of the system demand and capacity, and automated decision making tools. There is a spectrum of such decision making tools, ranging from the tactical to the strategic level. At the tactical end of the spectrum, a key function is to help

[7] Senior Scientist, 1700 Dell Avenue, Campbell, CA, member, AIAA.
[8] Senior Engineer, 2017 Cunningham Drive., #407, Hampton, VA 23666.

maintain separation between aircraft. At the strategic end of the spectrum, a key function is to help manage demand with strategic rerouting or delays, or manage capacity with airspace redesigns.

While these disparate and wide ranging decision support tools can provide substantial efficiency gains, they also can present complex challenges and opportunities for ATM designers. For instance, different tools, and their underlying strategies, may interact in unforeseen ways, causing unintended consequences. A TFM tool may resolve demand overloads assuming a particular airspace design while a completely separate airspace design tool is implementing a new airspace geometry. Because both the TFM and DAC decision support tools are strategic and so share a common time frame, they are particularly amenable to unintended feedback influences. [1]

But interactions between different decision making functions may also present opportunities. For instance, in recent years substantial progress has been made in the modeling and design of advanced, NextGen separation assurance decision support tools and methods. [2-12] Such improved tactical decision making may assist the strategic TFM decision making because demand need not be so tightly managed. The purpose of TFM is to manage demand so system loading is manageable at the tactical level. TFM creates the strategic plans so that ATC is not overwhelmed. But if tactical ATC functions are more robust, then a looser TFM policy may be possible. So there is a relationship with potential opportunities between the strategic and tactical decision making. Although they operate on very different time scales, they are not independent of each other.

Although it is usually not quantified, TFM decision making necessarily defines a given policy which determines how strictly demand is managed. A loose TFM policy reduces strategic delay at the cost of increasing the expected demand-to-capacity ratio in the NAS. A tight TFM policy, on the other hand, increases strategic delay in order to decrease the expected demand-to-capacity ratio in the NAS.

Although this concept of a TFM policy often goes unspoken, it is quite real and it defines how the strategic planning deals with uncertainties. There is substantial uncertainty in demand-to-capacity projections over the strategic TFM planning time frame (e.g., six hours into the future). These uncertainties arise from both demand and capacity unknowns. Capacity uncertainty is due primarily to meteorological phenomena which are difficult to predict accurately. These meteorological phenomena can degrade both airport and airspace capacities, and they can also lead to demand uncertainties. For instance, flight delays from earlier in the day can cascade and cause delays or cancellations later in the day. And once airborne, en route winds may influence flight times differently than forecasted. Aside from meteorological effects, demand uncertainties also arise from a variety of factors in the user operations, such as mechanical problems or crew scheduling disruptions.

Given these substantial uncertainties in the projected loading and demand, there can be significant uncertainty in the forecasted demand-to-capacity ratio, or *congestion*. Therefore a zero tolerance TFM policy (i.e., the tightest TFM policy), in which strategic TFM initiatives are used to ensure zero probability of future congestion, would be intolerably expensive in terms of delay. This is because infeasible levels of delay would be required to ensure that even the worst-case demand and capacity scenarios cannot result in any congestion.

Therefore a zero tolerance TFM policy in today's system is not practical. Either the forecasting uncertainties need to be reduced or a looser TFM policy is required. In the former, reduction of capacity uncertainty is not likely to be practical, and reduction in loading uncertainty reduces efficiency. This is because last-minute decision making is compromised. Such loading uncertainty allows for efficient decision making, for instance, by not forcing users to commit to a particular plan.

On the other hand, today's NAS allows for over-scheduling practices and demand levels are sufficiently high that overloading of surface and airspace resources would be inevitable if the traffic was not managed. Therefore a zero management (i.e., *open-loop*) TFM policy in today's system is also not practical.

Given these realities, the NAS implicitly uses a medium TFM policy. The NAS TFM policy is not overly tight as in the zero tolerance policy. Nor is the NAS TFM policy overly loose as in the zero management policy, allowing traffic to freely flow with no restrictions on demand. The NAS's use of a medium TFM policy means there is a finite probability of local overloading that must be handled by local TFM initiatives or ATC SA functions.

A key question is: What are the tradeoffs between TFM policy and the ATC solution? Advanced ATC SA capabilities are on the horizon, but what opportunities will they present at the strategic level? Specifically, will such advanced NextGen SA capabilities accommodate a looser TFM policy? If so, what reduction in the strategic TFM delay can be expected? And how does that TFM delay reduction compare with delays caused by the advanced ATC SA capabilities?

Here we assemble the modeling and simulation platform required to address these questions. To investigate these issues we need explicit representations of both the tactical and strategic decision making, in a full, NAS-wide, simulation. These simulations are required to evaluate and verify the interactions in detail, and this raises the problem of combining fast and slow phenomena in the same simulation. Combining such widely disparate time scales, in a large-scale (NAS-wide) simulation can substantially increase the computer resources required.

In this paper we present our design and implementation of a NAS-wide simulation that provides sufficient fidelity to model both the tactical and strategic decision makers and their intersections, but avoids unnecessary details that would intolerably escalate the compute requirements. Our simulation has both real-time and fast-time modes. Finally, we present preliminary experimental results and conclusions obtained from using our simulation with a mature probabilistic traffic flow management decision support tool and a simple separation assurance tool.

II. Probabilistic NAS Platform Simulation

For our simulation we use the Probabilistic NAS Platform environment. [13-31] Figure 1 shows the high level PNP architecture. The PNP architecture is convenient for this effort because, as Fig. 1 shows, all decision making is segregated into clients that attach to the PNP server. PNP itself models the NAS and its dynamics, but makes no decisions. It provides NAS data to the decision-making clients, and accepts decisions from those clients.

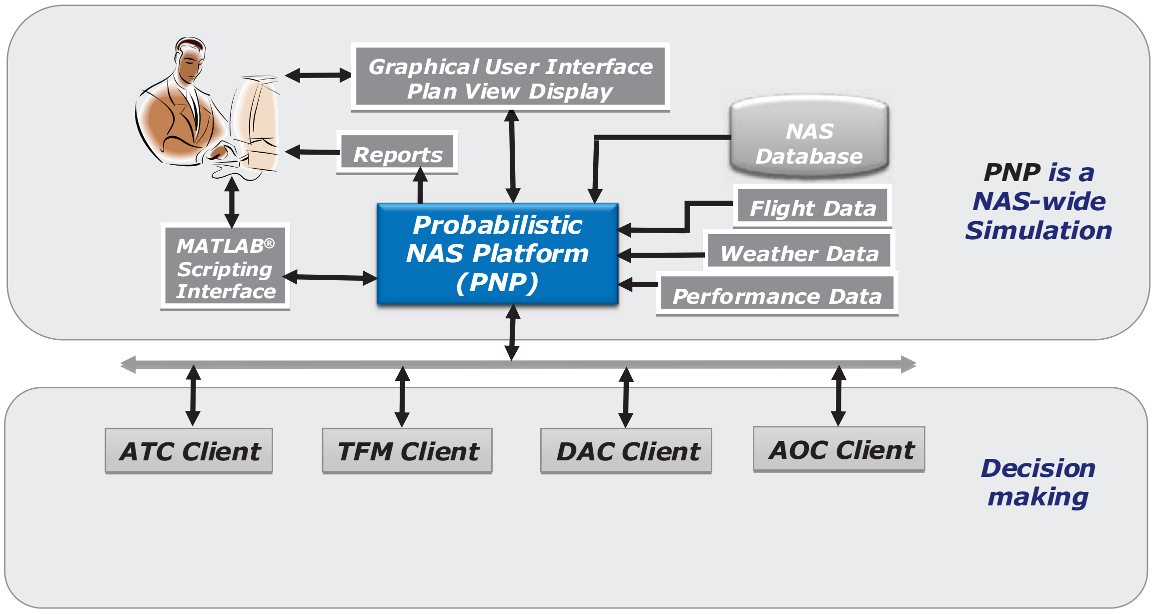


Figure 1. High level PNP architecture including typical decision making clients.

PNP can be run in a real-time mode or a playback mode. In the real-time mode, meteorological and traffic data feeds supply current information that PNP uses to model the state of the NAS. To propagate aircraft trajectories, PNP uses PointMass, [32] a fast and accurate trajectory predictor.

In playback mode ProbTFM runs in fast-time using archived weather and traffic data. The weather data are selected from a list of historical days from which the meteorological data have been archived. The traffic data, on the other hand, may either represent an historical day, or a future demand day. For example, a day in the year 2014 can be run, with a hypothetical fleet mix such as an increase in the very light jets.

The input data are read in by the PNP data processor. The data processor reads, parses, and processes these data. Downstream of the data processor, the PNP modules are mode insensitive. Those processes operate in the same way, regardless of whether the real-time or the fast-time mode is run, and regardless of data source.

The PNP real-time mode can be used to generate real-time, automated TFM advisories. Both the real-time and the fast-time modes can be used as a development environment for ATM researchers to experiment and evaluate candidate decision support tools or other technology or infrastructure enhancements.

Another key input to PNP is the airspace definitions data. We use current NAS airspace definitions; however, user-defined airspace definitions can instead be input to PNP. Therefore, ATM researchers can experiment with new airspace designs, and even dynamically redesigned airspace geometries.

III. Traffic Flow Management Client

For the TFM decision support tool we use ProbTFM which stochastically models both system capacity and loading forecasts, as functions of both weather and traffic uncertainties. These capacity and loading forecasts are modeled as probability mass functions (CPMFs and LPMFs, respectively). ProbTFM computes CPMFs and LPMFs for all NAS capacity elements, in 15 minute time intervals, for the future hours of the day. The CPMF and LPMF, for a particular capacity element and time window, are compared to derive a non dimensional congestion cost.

The total congestion cost for a flight is then tallied according to the NAS capacity elements it transits.

This congestion computation uses our congestion cost function. Equation (1) shows that this sector congestion cost (SCC) function is a modified convolution of the CPMF and LPMF.

$$SCC = \sum_{i=L\min} \sum_{j=C\min} (i-j) \times LPMF(i) \times CPMF(j) \quad (1)$$

For example, Fig. 2 illustrates a notional LPMF which slightly overlaps its corresponding CPMF. If there was no overlap then the SCC computed in Eq. (1) would be zero. The slight overlap, however, introduces a small SCC.

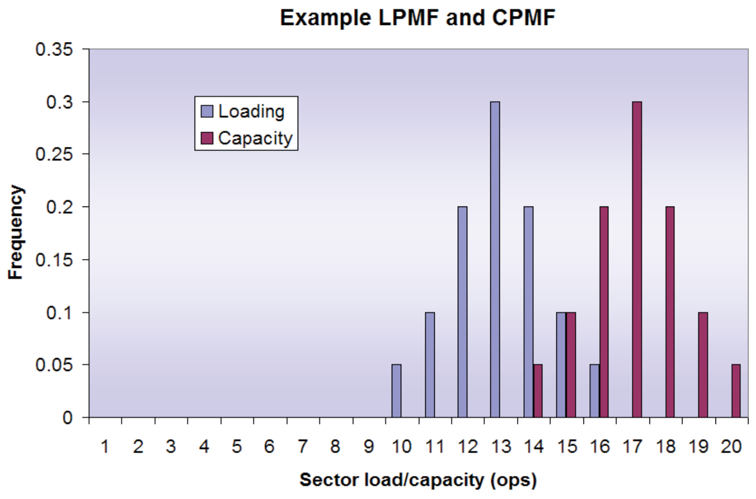


Figure 2. Notional LPMF and CPMF with slight overlap.

In Fig. 2 the distance between the LPMF mean value and CPMF mean value is four operations. As Fig. 3 shows, the greater the overlap the greater the congestion cost. The cost grows as the LPMF slides to greater values or, equivalently, the CPMF slides to lower values.

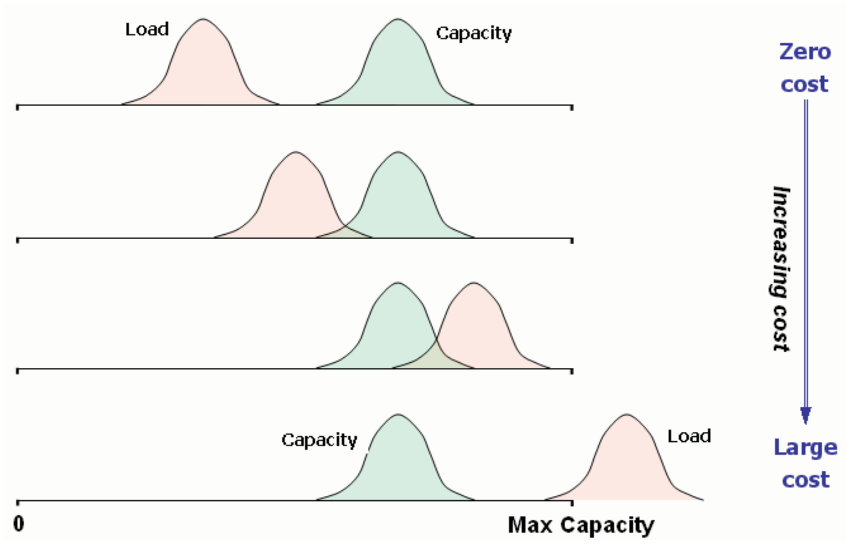


Figure 3. Congestion cost model from Eq. 1.

Inherent in our congestion cost model is the forecast accuracy, for both the CPMF and LPMF. The congestion cost model is based on PMFs, and the accuracy of the forecast is imbedded in the PMFs. Therefore the congestion cost accounts for forecast accuracy. The model can be used for any look ahead time (LAT) or for current time (where the PMF becomes a spike at the known value).

With our models of the weather impact on NAS capacity, we use the Eq. (1) cost function to evaluate the congestion cost at airports and in the NAS airspace, in future time intervals. We use these costs, in turn, to evaluate the congestion cost of each departing flight. High congestion cost flights typically are involved in more than one area of congestion. Thus, the NAS congestion is reduced more efficiently by resolving the high cost flights first. Figure 4 illustrates the high-level ProbTFM logic.

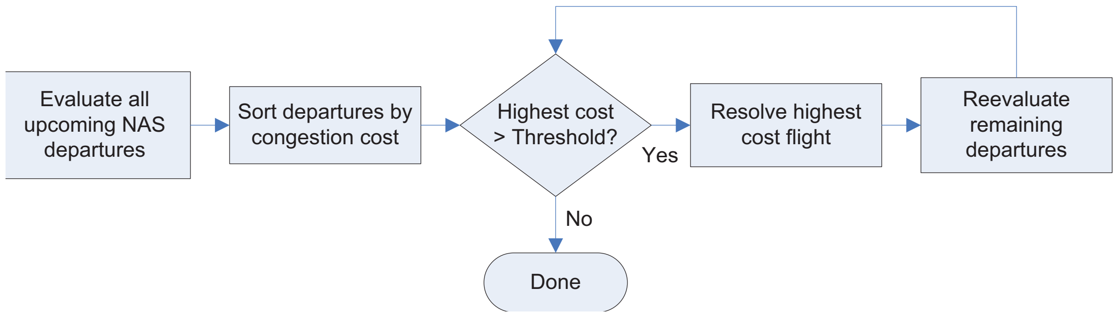

Evaluate all upcoming NAS departures
Sort departures by congestion cost
Highest cost > Threshold?
Yes
Resolve highest cost flight
Reevaluate remaining departures
No
Done

Figure 4. ProbTFM high-level logic for managing demand in congestion scenarios.

As Fig. 4 shows, inherent in the ProbTFM logic is a TFM policy setting. Flight congestion costs, computed from Eq. (1), are compared with a threshold value. This threshold value is a user-specified input the defines the TFM policy. A low threshold results in a tight TFM policy, as even minor levels of forecasted congestion will trigger strategic delays. A high threshold, on the other hand, results in a loose TFM policy, as significant levels of forecasted congestion are required to trigger strategic delays.

IV. Separation Assurance Client

In our PNP separation assurance client all aircraft always have an associated trajectory plan. The trajectory plan may be retrieved for any flight using a query command, and the current location of the aircraft is identified. To perform a maneuver the trajectory plan is replaced with a new one.

Within the trajectory plan, we separate the horizontal and vertical plane information into two different components. The vertical plane component specifies a series of segments with speed / altitude target states that are to be achieved at the completion of the segment. For instance, consider an aircraft that is currently in a constant speed, level cruise phase segment.

Now consider an altitude maneuver consisting of a descent followed by a level segment, and finally a climb that returns to the original altitude. This would be achieved by inserting three new segments. The three new segments would (i) descend to a target altitude, (ii) remain level at that altitude, and (iii) then climb to the original altitude. The flight would then resume its level cruise phase segment. Speed changes may be implemented in the same way.

These turn, altitude and speed maneuvers can be tested in a trial planning algorithm. That is, rather than implementing a new trajectory plan, one can be hypothesized. We implement this architecture as illustrated in Fig. 5.

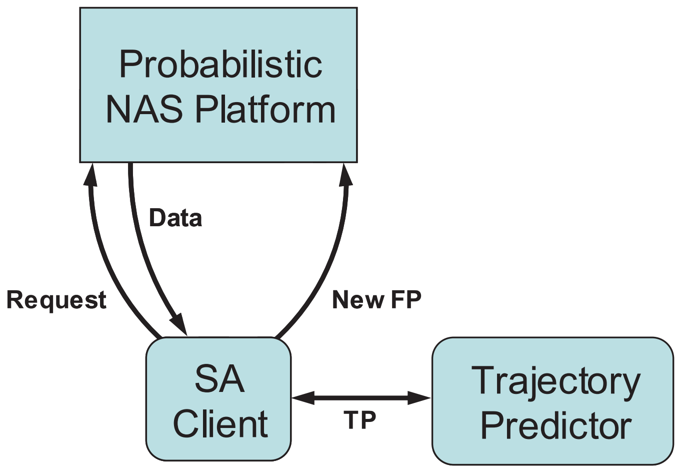


Figure 5. **High level separation assurance maneuver architecture with trial plan capability.**

As Fig. 5 shows, the SA client requests flight plans from PNP. The client then uses the data to derive new flight plans for one or more flights which resolve predicted conflicts. As part of this computation the SA client uses the Trajectory Prediction service to obtain detailed trajectory data based on the flight plans. The trajectory data indicate the predicted trajectory path. Figure 6 shows a more detailed chart of this architecture.

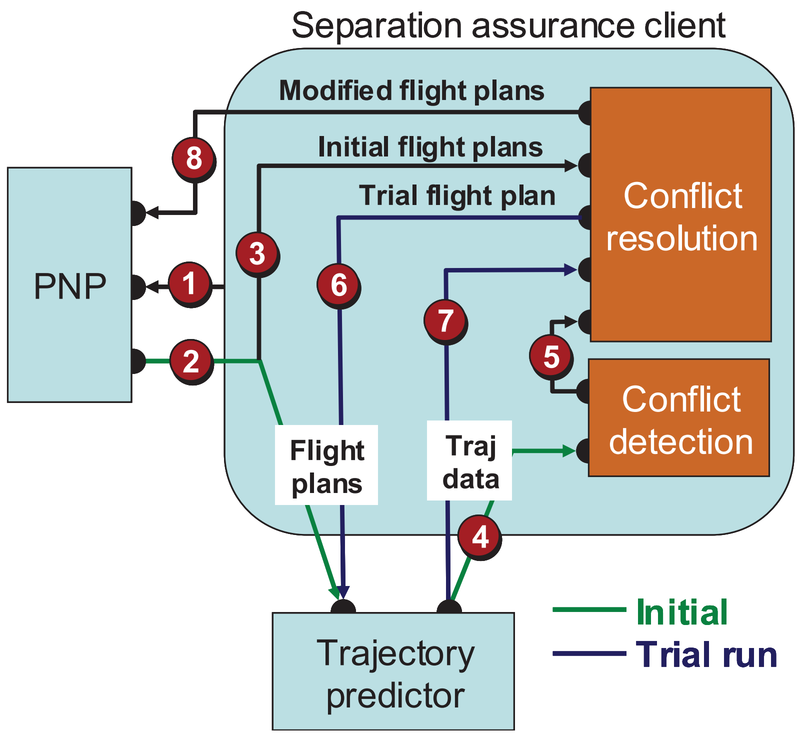


Figure 6. **Detailed separation assurance maneuver architecture.**

The eight steps identified in the Fig. 6 architecture are described in Table 1.

Table 1. Major steps in PNP separation assurance client.

Step 1	The SA client requests the flight plans of all flights within a specified distance or time of the flight of interest.

Step 2	PNP returns the requested flight plans which the SA client sends to the Trajectory Predictor service.
Step 3	The SA client will also send the flight plans to the Conflict Resolution module for its reference.
Step 4	The Trajectory Predictor service returns the trajectory data calculated from the flight plans. These data extend only to the specified look ahead time frame. The SA client sends these data to the Conflict Detection module.
Step 5	The Conflict Detection module returns the point of closest approach (PCA) and loss and gain of separation event data which the SA client sends to the Conflict Resolution module and to PNP. The SA client also sends the trajectory data (see Step 4) to the Conflict Resolution module for its reference.
Step 6	The Conflict Resolution module computes trial-run resolution maneuvers and their associated flight plan modifications. The SA client sends the new flight plans to the Trajectory Predictor service.
Step 7	The Trajectory Predictor service returns the trajectory data calculated from the flight plans. These data extend only to the specified look ahead time frame. The SA client sends these data to the Conflict Resolution module, as trial plan results.
Step 8	The Conflict Resolution module iterates through Steps 6 and 7 as necessary. Note that as part of its evaluation of the trial plan, the Conflict Resolution module performs a conflict detection check between the modified flights and all other flights. For instance, if two flights are modified, then each one must be checked against all other flights in the set. A possible strategy for performing these checks is to use the Conflict Detection module. After the trial planning process is complete, the Conflict Resolution module returns its final new flight plans. The SA client returns these to PNP for processing and incorporation into the simulation.

Within this SA client we implemented a simple conflict resolution algorithm for purposes of investigating the TFM-SA interactions described in Section I. Our conflict resolution algorithm uses flight intent data in both the detection and resolution functions. Also, conflicts in terminal areas are ignored as we are not using a detailed terminal area model. The resolver uses single aircraft maneuvers rather than coordinating maneuvers of both aircraft.

When conflicts are detected the conflict resolution algorithm categorizes them as either a merge, cross or head-on geometry, as illustrated in Fig. 7.

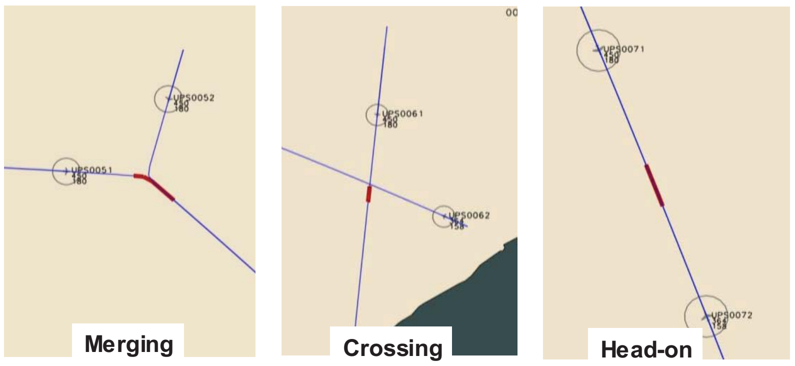


Figure 7. Conflict resolver encounter geometry categories.

The conflict resolver then uses a trial planning procedure to evaluate the candidate maneuvers. A speed reduction is the first preference, followed by an altitude reduction, followed by right and left turns.

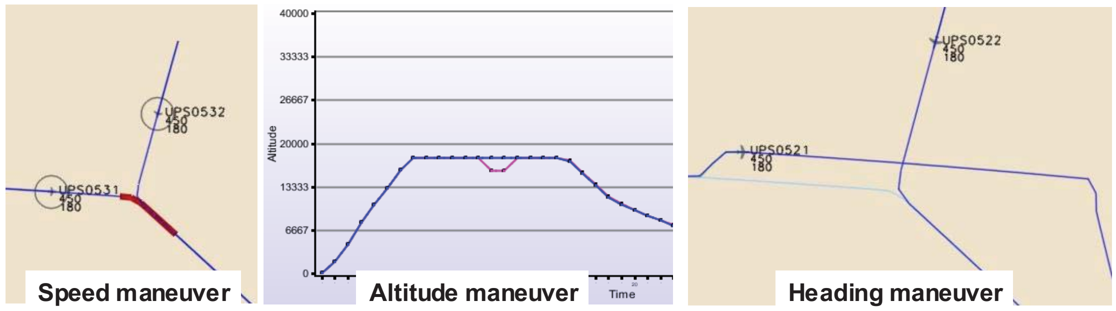
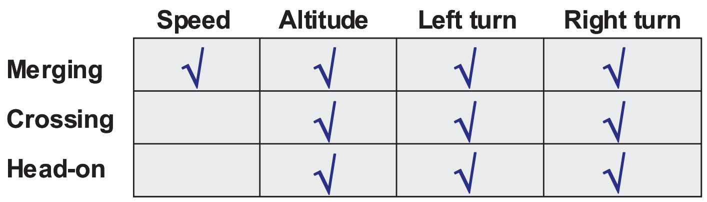


Figure 8. Candidate maneuvers in order of preference, left to right.

The speed maneuver is used only in a merging geometry. Otherwise, all the candidate maneuvers are possible for the different geometries, as illustrated in Fig. 9.



	Speed	Altitude	Left turn	Right turn
Merging	√	√	√	√
Crossing		√	√	√
Head-on		√	√	√

Figure 9. Encounter geometries and permissable candidate maneuvers.

V. Experiment Results

We used the SA and TFM clients described above in the PNP NAS-wide simulation to make a preliminary investigation of the interactions between these tactical and strategic decision making functions. For this experiment we used November 16, 2006, a Thursday, as our scenario day. The traffic demand is heavy in this scenario (60,404 flights) with intense convective weather activity in the eastern NAS, as shown in Fig. 10.

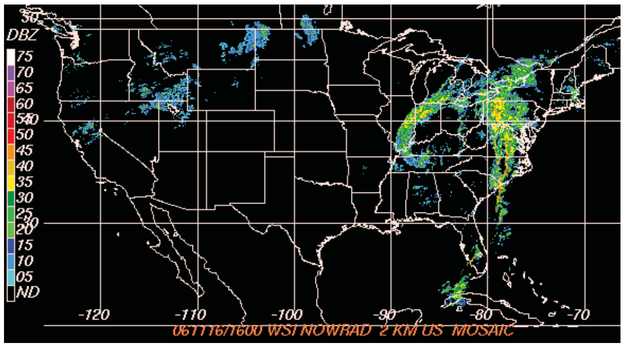


Figure 10. November 16, 2006: NexRad Reflectivity, 1600Z.

In this experiment we set our adaptation parameters to run the conflict resolver every five minutes in the simulation time and to detect and resolve conflicts over a 15 minute time horizon. We set the minimum horizontal separation to 5 nmi and the minimum vertical separation to 1000 ft.

As Fig. 11 shows, slightly less than 10% of the resolutions were speed maneuvers and the remainder were roughly equally divided between altitude and turn maneuvers.

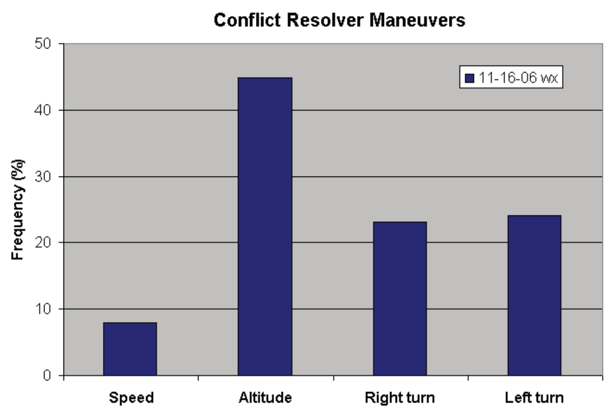


Figure 11. Frequency of the four different SA maneuver types.

As discussed in Section I, the TFM policy determines the demand-to-capacity levels which the tactical ATC functions must accommodate. If these levels are relatively high then the tactical ATC must exert more control to maintain separation. Figure 12 shows how the tactical loading varies with TFM Policy in this experimental scenario.

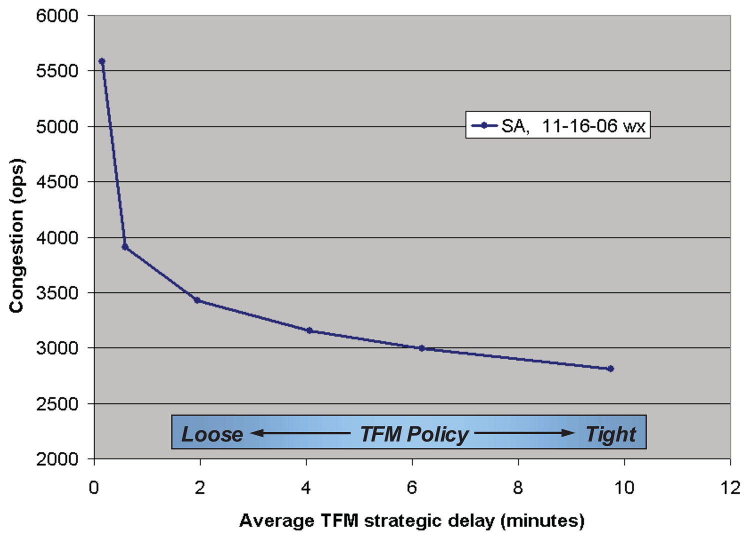


Figure 12. NAS congestion, accounting for weather impacts on capacity, versus TFM policy.

Figure 12 shows that the tactical loading drops off rapidly as the TFM moves from an open-loop policy to even a very loose policy. The use of small amounts of strategic delay is quite efficient in managing the forecasted demand-to-capacity ratios. In a heavy weather and traffic scenario such as this, however, additional tightening is likely to be required in order for congestion to reach manageable levels.

Figure 13 shows how the SA conflict resolution count varies with the TFM policy. The figure shows the expected trend that as TFM policy is loosened the increased loading shown in Fig 12 translates into increased conflicts and therefore increased resolutions.

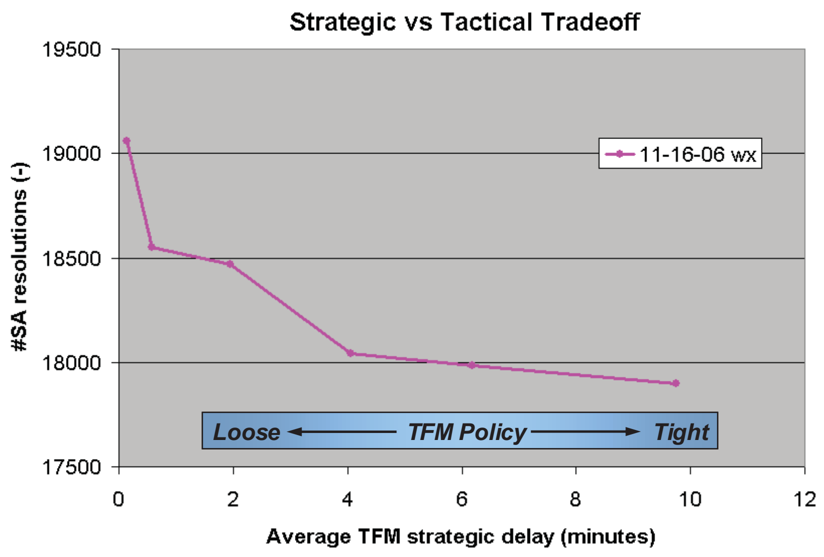


Figure 13. SA conflict resolution count versus TFM policy.

Although the number of resolutions required increases as the TFM policy is loosened, Fig. 13 shows that the rise is not dramatic. Over the span of 10 minutes of strategic TFM delay, the resolution count varies by about 5%. This resolution count of about 20k translates into additional delay minutes. So in addition to the strategic delay that results from the TFM policy, there is also a tactical delay component. But whereas the strategic TFM delay decreases as the TFM policy is loosened, the tactical SA delay increases.

This suggests a tradeoff between the strategic TFM delay and the tactical SA delay. Our results suggest this tradeoff is dominated by the reduction in the strategic TFM delay for two reasons. The first reason is that the tactical SA delay is about an order of magnitude less than the strategic TFM delay, as Fig. 14 shows.

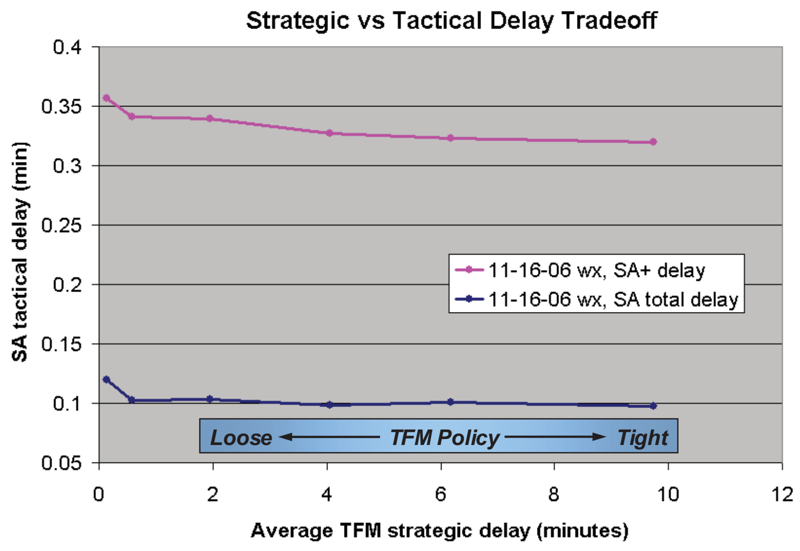


Figure 14. SA tactical delay for positive and total delay cases, versus TFM strategic delay.

We divide the tactical SA delay into two categories: resolutions with positive delay (SA+) and resolutions with negative delay. Resolutions with positive delay add about a minute of flight time, on average. This translates into about 0.3 minutes of delay per total number of flights (60,404 in our scenario), as shown in Fig. 14. This level of delay is minor compared to the strategic TFM delay.

The second reason why the tradeoff between strategic TFM delay and tactical SA delay is insignificant is that SA resolutions with negative delay further reduce the tactical delay substantially. As Fig. 14 shows, when these negative delay, flight time reductions, are accounted for, the total tactical delay reduces from about 0.3 to about 0.1 minutes, a reduction of a factor of three.

VI. Conclusion

In this paper we discuss fundamental TFM concepts which relate the TFM policy to demand-to-capacity levels in the NAS and in turn the tactical ATC workload. We also discuss how the TFM policy relates to the interaction between strategic TFM and tactical SA decision making in the NAS. To investigate these TFM-SA interactions, we implement a simulation environment to run TFM-SA integration experiments, using a simple SA conflict resolution algorithm.

We use this simulation tool to perform a preliminary investigation of the fundamental TFM-SA interaction issue of delay tradeoff. Specifically, if advanced NextGen SA capabilities accommodate a looser TFM policy, then what reduction in the strategic TFM delay can be expected? And how does that TFM delay reduction compare with delays caused by the tactical SA capabilities? Finally, what is the overall system delay reduction offered by NextGen SA tools?

Our results suggest that the strategic TFM delay reductions offered by a looser TFM policy are substantially greater than the tactical SA delay increases incurred. For instance, an aggressive loosening of TFM policy would reduce the TFM delay by five minutes per flight or more. Such a strategic delay reduction would induce only about 0.3 minutes of tactical delay per flight, as a result of conflict resolutions that add positive delay. And when negative delay resolutions are included, the tactical delay per flight reduces to about 0.1 minutes per flight. Therefore the tactical delay increase can be an order of magnitude less than the strategic delay decrease.

Therefore, from a system delay reduction perspective, local ATC SA functionality that allows for a looser TFM policy could provide significant delay reduction savings. Obviously there are many other factors to consider in implementing such a NextGen functionality such as operational, procedural and safety. But given that those issues are adequately addressed, ATC SA functionality has substantial delay savings potential.

Note that this conclusion does not hinge on substantially higher traffic levels in the future. With modest growth or even in today's system, demand/capacity imbalances, requiring TFM delay, are common. Such TFM delay has the potential for reduction, by TFM policy loosening made possible by ATC SA enhancements.

Further investigation is required to determine the specific levels of TFM loosening that can be achieved with the ATC SA enhancements that are on the horizon. To investigate this and other more detailed questions we need a higher resolution conflict resolution algorithm or settings. For instance, we would need to increase the SA client run frequency from once very 5 minutes to a faster rate, so conflicts are not missed.

Acknowledgments

The authors would like to thank Jeremy Smith, Ty Marien and Patricia Glaab for their guidance, suggestions and oversight of this project. The authors would also like to thank WSI, a meteorological data supplier to the government and commercial markets, for its data

contributions. The authors would also like to thank Mitre for its contributions to the ProbTFM LPMF model.

References

1. George Hunter, "Meta Simulation Results for Simultaneous Dynamic Resectorization and Traffic Flow Management," *AIAA Digital Avionics Systems Conference*, Orlando, FL, October, 2009.
2. Robert Vivona, David Karr, David Roscoe, "Pattern-Based Genetic Algorithm for Airborne Conflict Resolution," *AIAA Guidance, Navigation and Control Conference*, Keystone, CO, August, 2006.
3. Cesar Munoz, Ricky Butler, Anthony Narkawicz, Jeffrey Maddalon, George Hagen, "A Criteria Standard for Conflict Resolution: A Vision for Guaranteeing the Safety of Self-Separation in NextGen," NASA TM-2010-216862, October, 2010.
4. David Wing, Jennifer Murdoch, James Chamberlain, Maria Consiglio, Sherwood Hoadley, Clay Hubbs, Michael Palmer, "Function Allocation with Airborne Self-Separation Evaluated in a Piloted Simulation," *International Congress of the Aeronautical Sciences*, 2010.
5. Huabin Tang, John Robinson, Dallas Denery, "Tactical Conflict Detection in Terminal Airspace," *Journal of Guidance, Control, and Dynamics*, 34:2, pp. 403-413, March-April, 2011.
6. Todd Lauderdale, Andrew Cone, Aisha Bowe, "Relative Significance of Trajectory Prediction Errors on an Automated Separation Assurance Algorithm," *Ninth USA/Europe Air Traffic Management Research and Development Seminar*, 2011.
7. David Blum, David Thipphavong, Tamika Rentas, Ye He, Xi Wang, Elisabeth Pate-Cornell, "Safety Analysis of the Advanced Airspace Concept using Monte Carlo Simulation," *AIAA Guidance, Navigation, and Control Conference and Modeling and Simulation Technologies Conference*, Toronto, Canada, August, 2010.
8. David Thipphavong, "Accelerated Monte Carlo Simulation for Safety Analysis of the Advanced Airspace Concept," *10th AIAA Aviation Technology, Integration, and Operations Conference*, Fort Worth, TX, September, 2010.
9. Russell Paielli, "Tactical Conflict Resolution using Vertical Maneuvers in Enroute Airspace," *AIAA Journal of Aircraft*, 45:6, November-December, 2008.
10. Heinz Erzberger, "The Automated Airspace Concept," Fourth USA/Europe Air Traffic Management Research and Development Seminar, 2001.
11. Andrew Lacher, David Maroney, Andrew Zeitlin, Unmanned Aircraft Collision Avoidance – Technology Assessment and Evaluation Methods," Mitre, July 2008.
12. Andrew Zeitlin, Michael McLaughlin, "Safety of Cooperative Collision Avoidance for Unmanned Aircraft," *25th Digital Avionics Systems Conference*, Portland OR, October 2006.
13. George Hunter, "Probabilistic Forecasting of Airport Capacity," *AIAA Digital Avionics Systems Conference*, Salt Lake City, UT, October, 2010.
14. Poornima Balakrishna, George Hunter, "Preliminary NextGen Collaborative Air Traffic Management Analysis," *AIAA Digital Avionics Systems Conference*, Salt Lake City, UT, October, 2010.
15. Huina Gao, George Hunter, Frank Berardino, Karla Hoffman, "Development and Evaluation of Market-Based Traffic Flow Management Concepts," *AIAA Aviation Technology, Integration and Operations Forum*, Dallas, TX, September 2010.
16. Huina Gao, George Hunter, "Evaluation of User Gaming Strategies in the Future National Airspace System," *AIAA Aviation Technology, Integration and Operations Forum*, Dallas, TX, September 2010.
17. Huina Gao, George Hunter, "Future NAS-Wide User Gaming Preliminary Investigation," *AIAA Digital Avionics Systems Conference*, Orlando, FL, October, 2009.
18. George Hunter, "Preliminary Assessment of Interactions Between Traffic Flow Management and Dynamic Airspace Configuration Capabilities," *AIAA Digital Avionics Systems Conference*, St. Paul, MN, October, 2008.
19. George Hunter, "Toward an Economic Model to Incentivize Voluntary Optimization of NAS Traffic Flow," *AIAA Aviation Technology, Integration and Operations Forum*, Anchorage, AK, September, 2008.
20. George Hunter, "Sensitivity of the National Airspace System Performance to Weather Forecast Accuracy," *Integrated Communications, Navigation and Surveillance Conference*, Herndon, VA, May, 2008.
21. Kris Ramamoorthy, Ben Boisvert, George Hunter, "Sensitivity of Advanced Traffic Flow Management to Different Weather Scenarios," *Integrated Communications, Navigation and Surveillance Conference*, Herndon, VA, May, 2007.
22. George Hunter, "Testing and Validation of NextGen Simulators," *AIAA Modeling and Simulation Conference*, Chicago, IL, August 2009.
23. George Hunter, Kris Ramamoorthy, "Integration of terminal area probabilistic meteorological forecasts in NAS-wide traffic flow management decision making," *13th Conference on Aviation, Range and Aerospace Meteorology*, New Orleans, LA, January, 2008.

24. Kris Ramamoorthy, George Hunter, "The Integration of Meteorological Data in Air Traffic Management: Requirements and Sensitivities," *46th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, January, 2008.

25. George Hunter, Ben Boisvert, Kris Ramamoorthy, "Advanced Traffic Flow Management Simulation Experiments and Validation," *2007 Winter Simulation Conference*, Washington, DC, December, 2007.

26. Kris Ramamoorthy, George Hunter, "Evaluation of National Airspace System Performance Improvement With Four Dimensional Trajectories," *AIAA Digital Avionics Systems Conference*, Dallas, TX, October, 2007.

27. George Hunter, Ben Boisvert, Kris Ramamoorthy, "Use of automated aviation weather forecasts in future NAS," *The 87th American Meteorological Society Annual Meeting*, San Antonio, TX, January, 2007.

28. Kris Ramamoorthy, George Hunter, "Probabilistic Traffic Flow Management in the Presence of Inclement Weather and Other System Uncertainties," *INFORMS Annual Meeting*, Pittsburgh, PA, November, 2006.

29. Kris Ramamoorthy, Ben Boisvert, George Hunter, "A Real-Time Probabilistic TFM Evaluation Tool," *AIAA Digital Avionics Systems Conference*, Portland, OR, October, 2006.

30. Kris Ramamoorthy, George Hunter, "A Trajectory-Based Probabilistic TFM Evaluation Tool and Experiment," *Integrated Communications, Navigation and Surveillance Conference*, Baltimore, MD, May, 2006.

31. Kris Ramamoorthy, George Hunter, "Avionics and National Airspace Architecture Strategies for Future Demand Scenarios in Inclement Weather," *AIAA Digital Avionics Systems Conference*, Crystal City, VA, October, 2005.

32. James Phillips, "An Accurate and Flexible Trajectory Analysis," *World Aviation Congress*, Anaheim, CA, October, 1997.

NAS-WIDE TRAFFIC FLOW MANAGEMENT CONCEPT USING REQUIRED TIME OF ARRIVAL, SEPARATION ASSURANCE AND WEATHER ROUTING

George Hunter and Benjamin Boisvert Sensis Corporation, Campbell, VA
Jeremy Smith and Ty Marien, NASA Langley Research Center, Hampton, VA

Abstract

Substantial research efforts in recent years have investigated advanced planning and decision support tools for the National Airspace System. These tools range from the tactical to the strategic level and include conflict detection and resolution, weather vectoring, traffic spacing, flight conformance monitoring, dynamic airspace configuration, and traffic flow management. Research efforts have also investigated how these different capabilities may work together. Here we construct a system-wide simulation with several prototype decision support tools, of varying levels of fidelity, to investigate their interactions and the resulting system-wide performance. Our initial experimental architecture consists of separation assurance, traffic spacing using required time of arrival, tactical weather vectoring, conformance monitoring, strategic weather routing and traffic flow management using gate delay only.

Introduction

Service providers in the national airspace (NAS) are often confronted with high-congestion scenarios caused by high demand, capacity disruptions, or a combination of the two. Such high-congestion scenarios often result in costly delays. Therefore users and NAS policy makers are interested in several different NAS enhancements to help expedite traffic flow in such scenarios. These enhancements impact the flight deck avionics as well as the ground-based airline operations centers and the air traffic management service providers. For instance, these enhancements include increased information flow and collaboration between these entities, improved avionics and flight-deck decision support tools, improved navigation capabilities, advanced traffic flow management (TFM) [1], dynamic airspace configuration [2], flight-specific weather routing, flight plan conformance monitoring, inter-aircraft spacing management and advanced separation assurance technologies [3-5].

These advanced technologies and strategies use models of system dynamics, forecasts of the system demand and capacity, and automated decision making tools. There is a spectrum of such decision making tools, ranging from the tactical to the strategic level. At the tactical end of the spectrum, decision making tools may help maintain separation between aircraft. At the strategic end of the spectrum, decision making tools may help manage demand with strategic rerouting or delays, or manage capacity with airspace redesigns.

While these wide ranging enhancements can provide substantial efficiency gains, they also present challenges for ATM designers. First, these different tools, and their underlying strategies, may interact in unforeseen ways, causing unintended consequences. For instance, a traffic flow management tool may resolve demand overloads assuming a particular airspace design while a completely separate airspace design tool is implementing a new airspace geometry. Because both the TFM and DAC decision support tools are strategic and so share a common time frame, they are particularly amenable to unintended feedback influences. This TFM-DAC interaction is the subject of on-going research to understand its effects and to design mitigation strategies [6-7].

Interactions between tactical tools, such as separation assurance, and the strategic tools are more straightforward since their time constants are so well separated. Indeed, designers of these tools often can initially ignore the other, or treat it as a simple first-order effect. Ultimately, however, more detailed treatment is required, for interactions between the tactical and strategic levels can be significant. For instance, at the strategic level, a TFM decision support tool requires forecasts of system capacity. But airspace capacity can depend on how traffic patterns are controlled at the tactical level, by separation assurance tools, for instance. This dependency on the tactical solution can be even more pronounced in the presence of heavy weather.

Recently studies have begun to examine interactions between such temporally segregated tools. We found, for instance, that a successful implementation of a tactical separation assurance capability substantially off-loads the strategic TFM initiatives. In one experiment we found two orders of magnitude leverage. That is, TFM delay could be reduced by about 100 times the delay introduced by the tactical separation assurance tool [8].

Here we describe new enhancements to our simulation which help to explore and evaluate these interactions: a traffic spacing tool which uses required time of arrival (RTA) and flight-specific strategic and tactical weather rerouting. These capabilities can be used to model flight-deck based decision making which may interact with ground-based systems. In this paper we present our models of these capabilities. Our next step is to exercise our simulation to evaluate the performance of these capabilities combined with other decision support tools. For our initial experiments we use the Probabilistic NAS Platform (PNP) simulation. Our initial experimental architecture consists of (i) strategic probabilistic TFM in the pre departure phase to reduce forecasted airport overloads, (ii) strategic weather rerouting in the pre departure phase to avoid major weather systems, (iii) tactical weather rerouting to take advantage of available airspace in the vicinity of convective weather, (iv) en route separation assurance and (v) traffic spacing at the departure and arrival metering fixes using RTA. Figure 1 illustrates our simulation system. Our simulation has both real-time and fast-time modes.

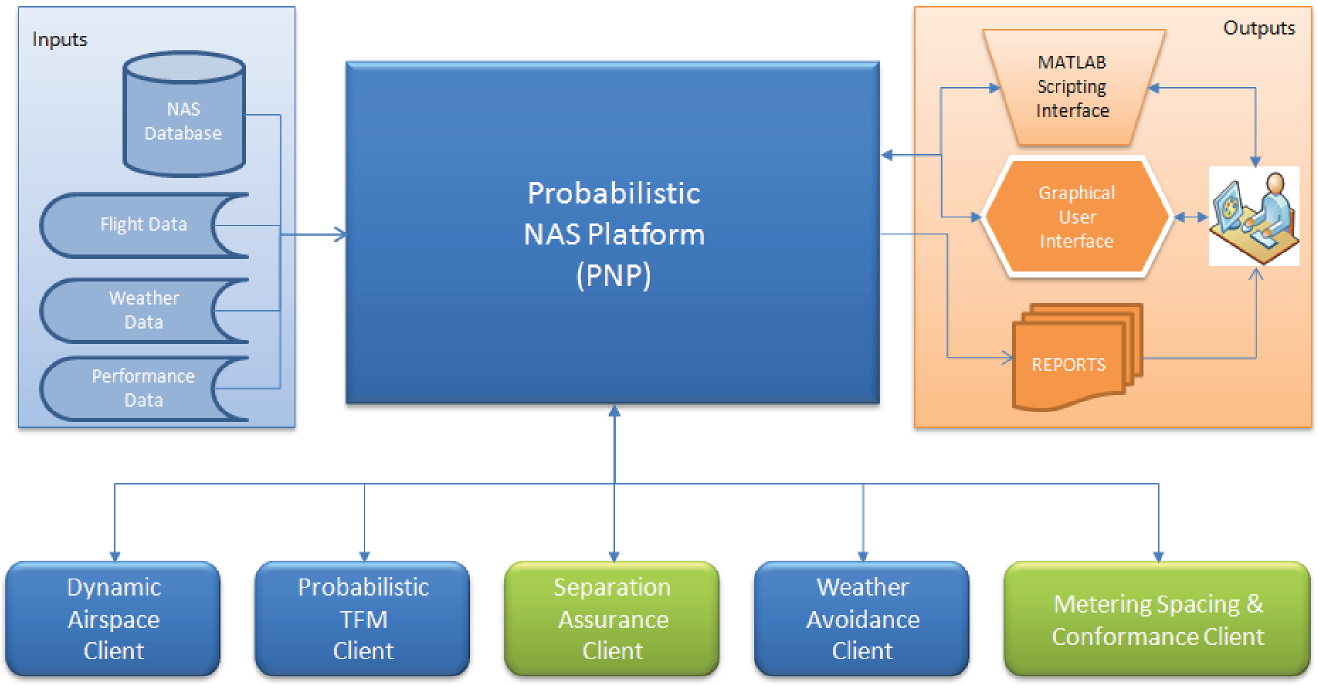


Figure 1. PNP Architecture

This architecture attempts to achieve safe weather routes and separation standards with a minimum of delay and congestion. The metering spacing is enforced on all departure and arrivals. Departure and arrival metering fixes are used to determine the spacing. Traffic separation is accomplished through conflict detection and resolution. Safe weather routes avoid heavy weather. Traffic Flow management is regulated by using forecasted probabilistic airport capacities. And conformance monitoring corrects for forecasted loss of spacing due to weather rerouting or separation assurance maneuvers.

RTA-based Traffic Spacing Concept

The RTA-based traffic spacing concept performs departure and arrival spacing at the metering fix. Departure spacing is assigned pre-departure, while arrival spacing is assigned en-route.

Metering Spacing

The RTA-based traffic spacing client requires all flights to be assigned a departure and arrival metering waypoint.

Departure Spacing

This concept manages departures by slotting flights by the departure metering fix crossing time. A flight is slotted when an open slot exists for its departure metering fix crossing time. If the slot is not available for that metering fix and crossing time, the next available slot for that metering fix is used. A pre-departure delay is created to adjust the metering crossing time. Figure 2 illustrates how the departure spacing concept handles a test case of ten duplicate flights, which originally had identical metering fix scheduled crossing times, from BWI to ATL.

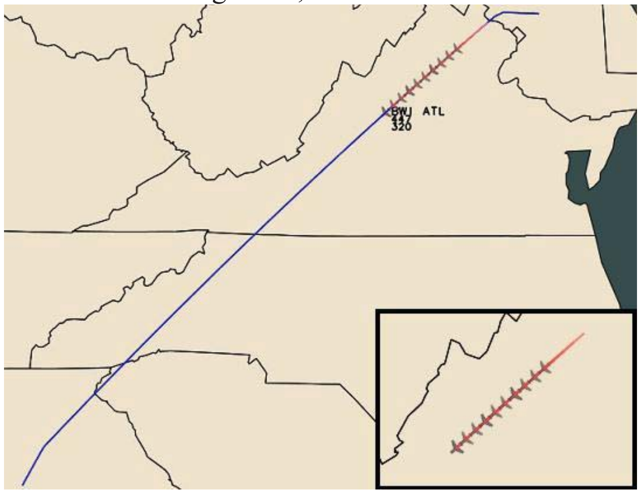


Figure 2. Departure Spacing

Arrival Spacing

The RTA-based traffic spacing concept manages arrivals by slotting flights by the arrival metering fix crossing time. This is done when a flight enters the freeze horizon zone. The freeze horizon zone is a configurable distance from the arrival airport. A flight is slotted when an open slot exists for its current arrival metering fix crossing time. If the slot is not available for that metering fix and crossing time, maneuvers are attempted to find an available slot. Three maneuvers are executed in the following order: descent profile modification, speed control, and path stretch.

The descent profile maneuver attempts to adjust the speed profile to obtain a flight window in which the arrival metering fix can be crossed. Likewise the speed control maneuver adjusts the cruise speed and the path stretch maneuver uses dog-leg maneuvers. Figure 3 shows a path stretching test case of ten duplicate flights from BWI to ATL which required delay to meet their metering fix RTA assignments.

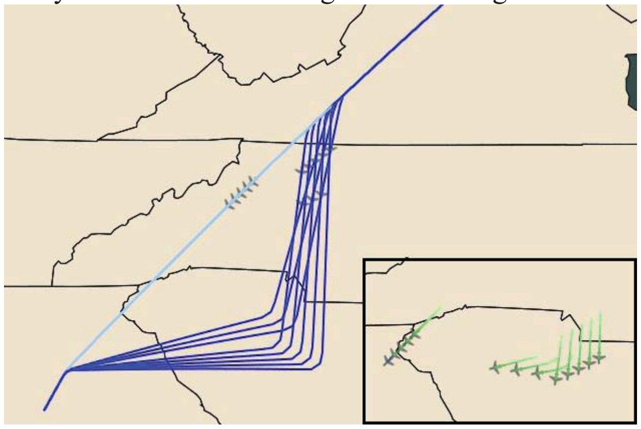


Figure 3. Path Stretch Maneuvers

The flight window obtained from a maneuver is used to find an arrival slot. If an arrival slot is not available the failure to obtain slot is logged and the flight continues as scheduled. Note that flights are slotted based on their performance in their flight envelope not the next available arrival slot.

Conformance Spacin

The RTA-based traffic spacing concept performs conformance monitoring, based on the estimated arrival metering fix crossing time. If a flight's estimated crossing time loses conformance then the flight is re-sequenced based on its flight window and the next available arrival time slot.

Separation Assurance Concept

The Separation Assurance Concept [8] is a first-order tool that attempts to separate aircraft based on separation standards. The separation standards consist of the following configurable items: conflict detection look-ahead time, minimum horizontal separation distance, minimum vertical separation distance, and conflict resolution criteria. Conflict detection of aircraft is accomplished by an *N*-squared search. The closest conflict to each aircraft is addressed, unless it has been previously solved by the other conflict aircraft, then the next closest conflict is addressed. Note the optimal solution is not calculated. The conflict calculates the loss of separation (LOS), point of closest approach (PCA) and gain of separation (GOS) time and distance based on the aircraft's current position and predicted trajectory, as illustrated in Fig. 4.

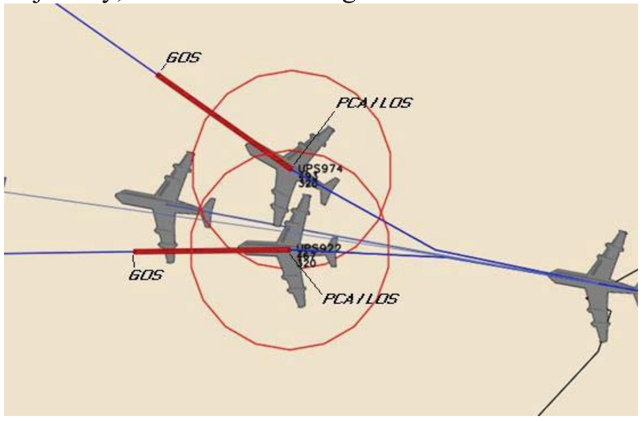


Figure 4. LOS, PCA, and GOS

A trial planner is used to evaluate four maneuvers in the following priority: speed control, altitude change, right turn, and left turn. The speed control maneuver reduces the current speed to avoid the conflict. Similarly the altitude change maneuver changes the current altitude and the left and right turns change the horizontal path to avoid the conflict, as illustrated in Fig. 5. The resulting flight plan change is evaluated to confirm that the conflict is averted and no new conflicts arise. If a resolution is not found, then the failure is logged and the flight continues on its current trajectory.

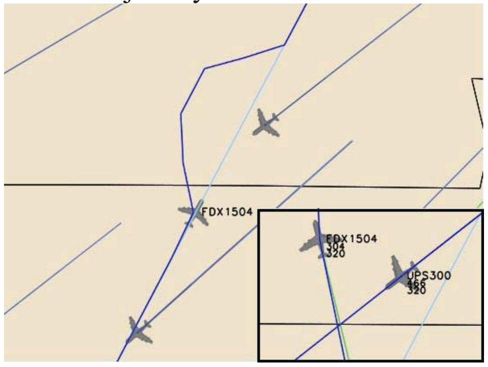


Figure 5. Left Turn Conflict Resolution

Probabilistic TFM Concept

The Probabilistic TFM concept [9-14] is a traffic flow management tool capable of tracking, forecasting and resolving NAS congestion. Its algorithms determine which flights are critical to the congestion problem, and the best TFM strategy for resolving NAS congestion.

The ProbTFM algorithms are stochastic in that they account for the inherent uncertainties in the NAS demand and capacity, rather than using deterministic approximations. This is important as both demand and capacity forecasts can have substantial uncertainty. ProbTFM models and accounts for these uncertainties in its solution. The Probabilistic TFM concept is used to delay departures based on airport congestion, as illustrated in Fig. 6.

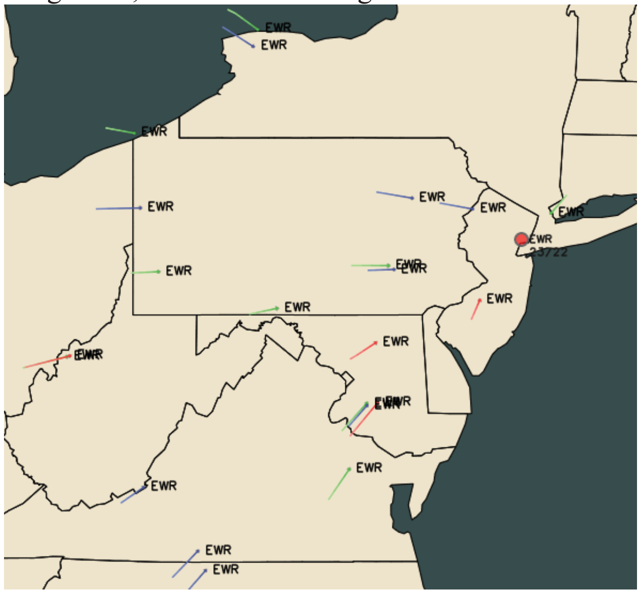


Figure 6. Newark (EWR) Airport Congestion

ProbTFM models both system capacity and loading forecasts as functions of both weather and traffic uncertainties. These capacity and loading forecasts are modeled as probability mass functions (CPMFs and LPMFs, respectively). ProbTFM computes CPMFs and LPMFs for all NAS capacity elements, in 15 minute time intervals, for the future hours of the day. The CPMF and LPMF, for a particular capacity element and time window, are compared to derive a non dimensional congestion cost.

This congestion computation uses a modified convolution of the CPMFs and LPMFs, as shown in Eq. (1). The total congestion cost for a flight is then tallied according to the NAS capacity elements it transits.

$$SCC = \sum_{i=L\min} \sum_{j=C\min} (i-j) \times LPMF(i) \times CPMF(j) \quad (1)$$

For example, Fig. 7 illustrates a notional LPMF which slightly overlaps its corresponding CPMF. If there was no overlap then the SCC computed in Eq. (1) would be zero. The slight overlap, however, introduces a small SCC.

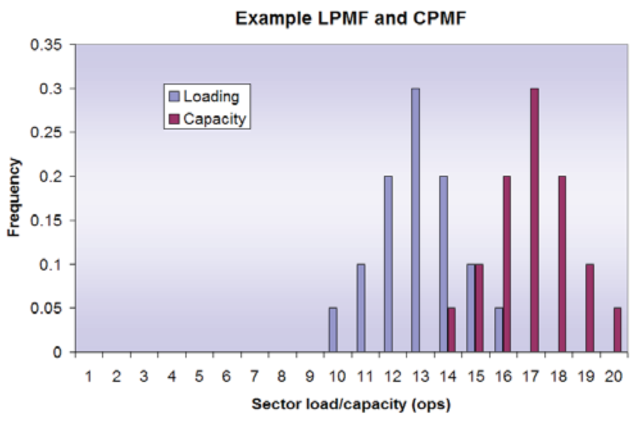


Figure 7. Notional LPMF and CPMF with slight overlap.

As Fig. 8 shows, the greater the overlap the greater the congestion cost. The cost grows as the LPMF slides to greater values or, equivalently, the CPMF slides to lower values.

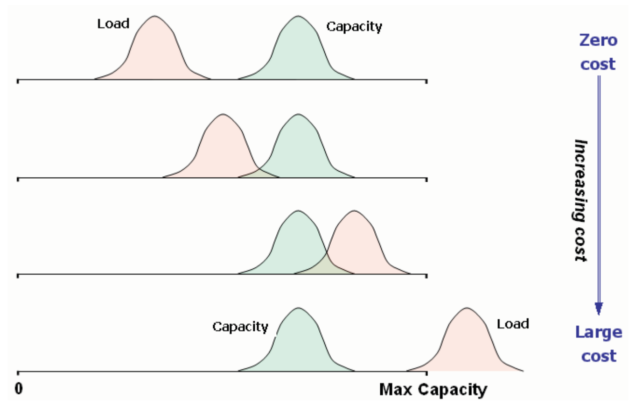


Figure 8. Congestion cost model from Eq. 1.

Inherent in our congestion cost model is the forecast accuracy, for both the CPMF and LPMF. The congestion cost model is based on PMFs, and the accuracy of the forecast is imbedded in the PMFs. Therefore the congestion cost accounts for forecast accuracy.

With our models of the weather impact on NAS capacity, we use Eq. (1) to evaluate the congestion cost at airports and in the NAS airspace, in future time intervals. We use these costs, in turn, to evaluate the congestion cost of each departing flight. High congestion cost flights typically are involved in more than one area of congestion. Thus, the NAS congestion is reduced more efficiently by resolving the high cost flights first. Figure 9 illustrates the high-level ProbTFM logic.

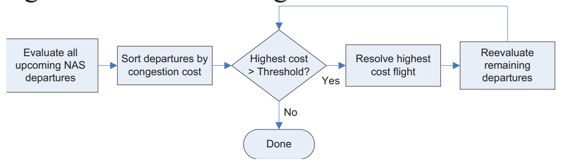

Evaluate all upcoming NAS departures
Sort departures by congestion cost
Highest cost > Threshold?
Yes
Resolve highest cost flight
Reevaluate remaining departures
No
Done

Figure 9. ProbTFM high-level logic for managing demand in congestion scenarios.

As Fig. 9 shows, inherent in the ProbTFM logic is a cost threshold. Flight congestion costs, computed from Eq. (1), are compared with the threshold value. This threshold value is a user-specified input the defines the TFM policy. A low threshold results in a tight TFM policy, as even minor levels of forecasted congestion will trigger strategic delays. A high threshold, on the other hand, results in a loose TFM policy, as significant levels of forecasted congestion are required to trigger strategic delays.

Weather Rerouting Concept

The weather rerouting concept attempts to avoid airspace that contains heavy convection. The concept uses variable-sized polygons to model the airspace. The current reflectivity image, or *nowcast*, represents the convective weather for the current time period. The nowcast is used to report the number of hazardous-weather flight incursions. The concept also uses a weather forecast for the future time periods. Each forecast has an adaptable parameter to determine the coverage threshold. The PNP universal grid (4.1 nmi) is used to determine weather cell population coverage. The weather cell is a hexagon creating a honeycomb grid, as illustrated in Fig. 10.

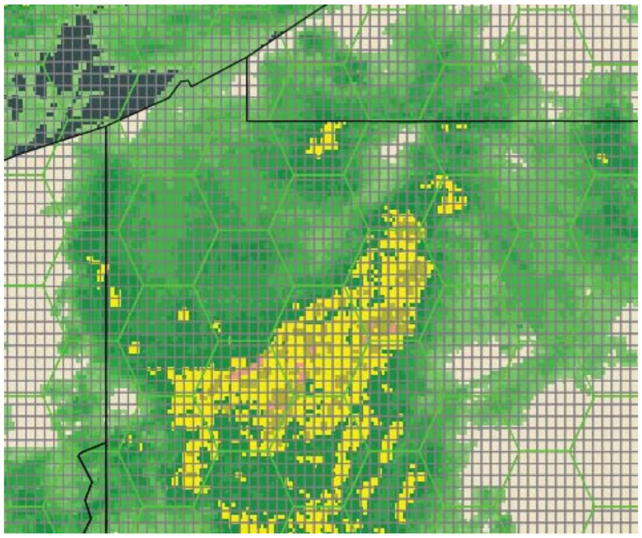


Figure 10. PNP Universal Grid and Weather Cells

The hazardous weather cells are identified using the persistence forecast. A hazardous weather cell is identified when the percentage of universal grid elements exceeds the coverage threshold, as illustrated in Fig. 11 as red hexagons. The weather cell hexagon size is configurable. The universal grid element is considered populated with hazardous weather if the universal grid element reflectivity is greater than 40 dbz. The hazardous weather cells are correlated to form weather clusters, as illustrated in Fig. 11. As neighboring hazardous weather cells are linked, they form the weather cluster. The clustering of hazardous weather cells forms the representation of a storm, which then can be avoided. The neighbor cells are identified to seed a Dijkstra algorithm to calculate the shortest path around a weather cluster, as illustrated in Fig. 11 as blue hexagons.

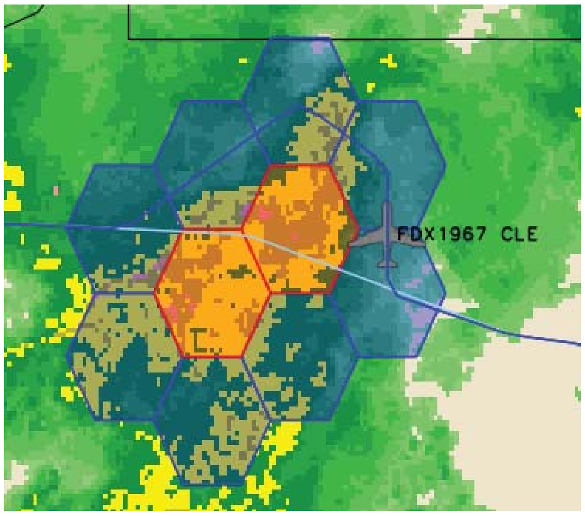


Figure 11. Weather Cluster

Routing

The weather rerouting concept uses the weather forecast to compute both strategic and tactical routing. The tactical routing occurs in the en route phase while the strategic routing occurs prior to departure.

Tactical Routing

A flight is considered for tactical routing if its tracks in the planning horizon traverse hazardous weather cells. The planning horizon consists of two adaptable relative time parameters (start and end of the planning horizon). If a flight is predicted to traverse a hazardous weather cell in the planning horizon, then the first hazardous weather cell encountered is used to determine the weather cluster. The neighboring clear weather cells of the weather cluster are used as candidate reroute cells. These candidate cells are passed into a Dijkstra algorithm to determine the shortest path. This shortest path is then spliced into the current flight plan to create a new reroute for the flight, as illustrated in Fig. 12. Route optimization is used to smooth the initial vectoring around a weather cluster. The number of acceptable route waypoints is configurable. This parameter controls path length and complexity.

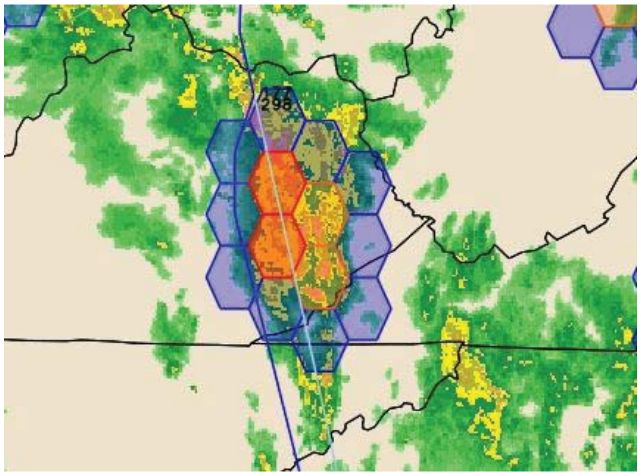


Figure 12. Tactical Rerouting

Strategic Routing

Pre-departure flights are considered for strategic rerouting if their tracks are predicted to traverse hazardous weather cells. The strategic routing algorithm calculates how many minutes a flight will encounter hazardous weather based on the forecast. Then the strategic routing algorithm calculates how many minutes a flight will encounter hazardous weather using different routes, as illustrated in Fig. 13. If an alternate route is available which encounters less hazardous weather, then it is selected as a strategic reroute. If multiple city-pair routes are available which encounter less hazardous weather, then the shortest among them is chosen, as illustrated in Fig. 14. The selected city-pair routes are saved for an adaptable time period. As routes are saved and reused over a configurable time period, they serve as the "game-plan" for that time period.

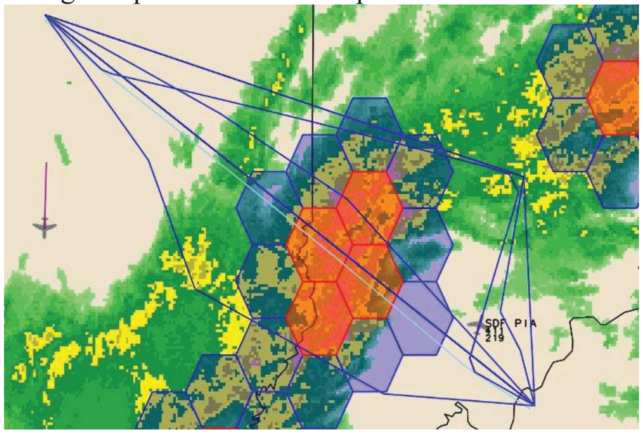


Figure 13. Strategic Rerouting

The city-pair routes were created based on ASDI routes collected over several days. Thousands of routes were then analyzed and paired down by their arc deltas.

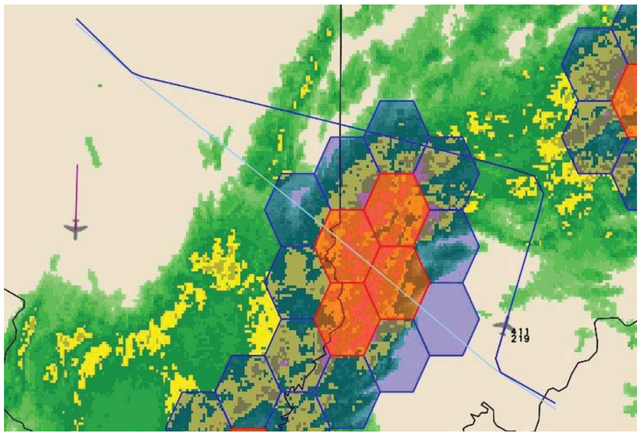


Figure 14. Strategic Reroute Selection

Probabilistic NAS Platform

For our simulation we use the Probabilistic NAS Platform environment [15-17]. The PNP architecture is convenient for this effort because, as Fig. 1 shows, all decision making is segregated into clients that attach to the PNP server. PNP itself models the NAS and its dynamics. It provides NAS data to the decision-making clients, and accepts decisions from those clients.

PNP can be run in a real-time mode or a playback mode. In the real-time mode, meteorological and traffic data feeds supply current information that PNP uses to model the state of the NAS. To propagate aircraft trajectories, PNP uses PointMass, a fast and accurate trajectory predictor [16,18].

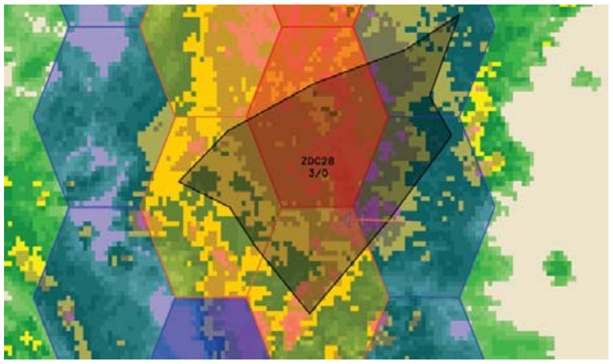
In playback mode ProbTFM runs in fast-time using archived weather and traffic data. The weather data are selected from a list of historical days from which the meteorological data have been archived. The traffic data, on the other hand, may either represent an historical day, or a future demand day which has been modeled. For example, a day in the year 2014 can be run, with a hypothetical fleet mix such as an increase in the very light jets.

The input data are read in by the PNP data processor. The data processor reads, parses, and processes these data. Downstream of the data processor, the PNP modules are mode insensitive. Those processes operate in the same way, regardless of whether the

real-time or the fast-time mode is run, and regardless of data source.
The PNP real-time mode can be used to generate real-time, automated TFM advisories. Both the real-time and the fast-time modes can be used as a development environment for ATM researchers to experiment and evaluate candidate decision support tools or other technology or infrastructure enhancements.
Another key input to PNP is the airspace definitions data. We use current NAS airspace definitions; however, user-defined airspace definitions can instead be input to PNP. Therefore, ATM researchers can experiment with new airspace designs, and dynamically redesigned airspace geometries as well.

The Probabilistic NAS Platform is responsible for sector and airport congestion accounting. Sectors can have a reduced capacity due to weather coverage as illustrated in Fig 15.

The Probabilistic NAS Platform receives all flight updates from all the concepts and determines the hierarchy of which flight updates is executed. The flight update priority is separation assurance, weather avoidance, spacing, and congestion management.



Conclusions

This paper describes an RTA-based traffic spacing and weather rerouting concepts and how they could fit within a candidate system architecture. The other system components include conflict detection and resolution, flight conformance monitoring, and traffic flow management using gate delay only.

Figure 15. Sector CongestionAcknowledgements

The authors would also like to thank WSI, a meteorological data supplier to the government and commercial markets, for its data contributions. The authors would also like to thank Mitre for its contributions to the ProbTFM LPMF model.

References

[1] Federal Aviation Administration, Air Traffic Organization, “Traffic Flow Management in the National Airspace System,” October 2009.

[2] P. Kopardekar, K. Bilimoria, B. Sridhar, “Initial Concepts for Dynamic Airspace Configuration,” *7th AIAA Aviation Technology, Integration and Operations Conference*, Belfast, Northern Ireland, September 2007.

[3] David Wing, Jennifer Murdoch, James Chamberlain, Maria Consiglio, Sherwood Hoadley, Clay Hubbs, Michael Palmer, “Function Allocation with Airborne Self-Separation Evaluated in a Piloted Simulation,” *International Congress of the Aeronautical Sciences*, 2010.

[4] Huabin Tang, John Robinson, Dallas Denery, “Tactical Conflict Detection in Terminal Airspace,” *Journal of Guidance, Control, and Dynamics*, 34:2, pp. 403-413, March-April, 2011.

[5] Todd Lauderdale, Andrew Cone, Aisha Bowe, “Relative Significance of Trajectory Prediction Errors on an Automated Separation Assurance Algorithm,” *Ninth USA/Europe Air Traffic Management Research and Development Seminar*, 2011.

[6] George Hunter, “Preliminary Assessment of Interactions Between Traffic Flow Management and Dynamic Airspace Configuration Capabilities,” *AIAA Digital Avionics Systems Conference (DASC)*, St. Paul, MN, October, 2008.

[7] George Hunter, “Meta Simulation Results for Simultaneous Dynamic Resectorization and Traffic Flow Management,” *AIAA Digital Avionics Systems Conference (DASC)*, Orlando, FL, October, 2009.

[8] George Hunter, Ben Boisvert, “Modeling and Simulation of Interactions Between Traffic Flow Management and Separation Assurance,” AIAA Modeling and Simulation Conference, Portland, OR, August, 2011.

[9] Kris Ramamoorthy, Ben Boisvert, George Hunter, “A Real-Time Probabilistic TFM Evaluation Tool,”

AIAA Digital Avionics Systems Conference (DASC), Portland, OR, October, 2006.

[10] George Hunter, Ben Boisvert, Kris Ramamoorthy, “Use of automated aviation weather forecasts in future NAS,” The 87th American Meteorological Society Annual Meeting, San Antonio, TX, January, 2007.

[11] George Hunter, Kris Ramamoorthy, “Integration of terminal area probabilistic meteorological forecasts in NAS-wide traffic flow management decision making,” 13th Conference on Aviation, Range and Aerospace Meteorology, New Orleans, LA, January, 2008.

[12] Huina Gao, George Hunter, Frank Berardino, Karla Hoffman, “Development and Evaluation of Market-Based Traffic Flow Management Concepts,” AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Dallas, TX, September 2010.

[13] George Hunter, “Probabilistic Forecasting of Airport Capacity,” AIAA Digital Avionics Systems Conference (DASC), Salt Lake City, UT, October, 2010.

[14] Kris Ramamoorthy, George Hunter, “Toward an Empirical Estimation of Weather Impact on Airport Capacity,” AIAA Aviation Technology, Integration and Operations (ATIO) Forum, Indianapolis, IN, September 2012.

[15] Kris Ramamoorthy, George Hunter, “Evaluation of National Airspace System Aggregate Performance Sensitivity,” AIAA Digital Avionics Systems Conference (DASC), Dallas, TX, October, 2007.

[16] George Hunter, Ben Boisvert, Kris Ramamoorthy, “Advanced Traffic Flow Management Simulation Experiments and Validation,” 2007 Winter Simulation Conference, Washington, DC, December, 2007.

[17] George Hunter, “Testing and Validation of NextGen Simulators,” AIAA Modeling and Simulation Conference, Chicago, IL, August 2009.

[18] James Phillips, “An Accurate and Flexible Trajectory Analysis,” *World Aviation Congress*, Anaheim, CA, October, 1997.

31st Digital Avionics Systems Conference

October 14-18, 2012

PNP Developer's Guide

Creating and Using Clients with the SimObjects API

8/17/2011
Sensis Corporation

Table of Contents

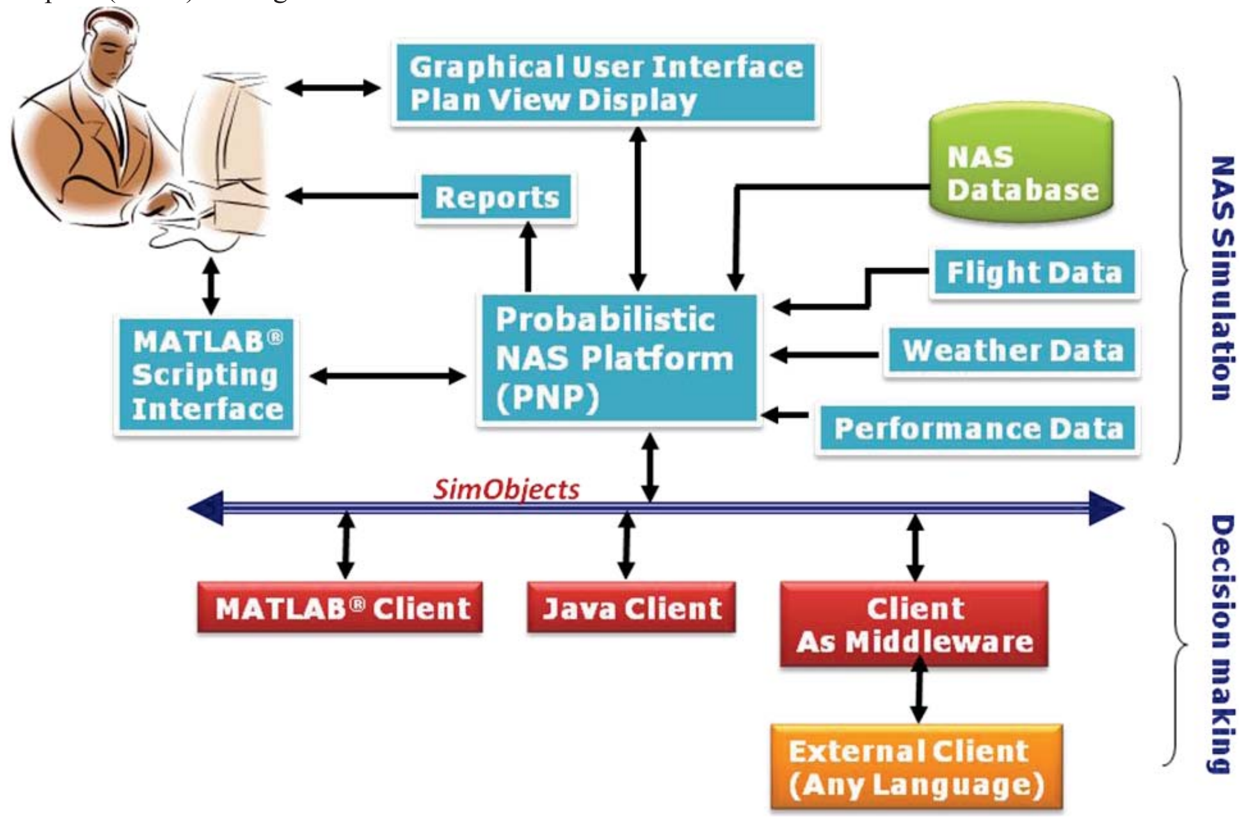
Introduction

This PNP Developer's Guide focuses mainly on the development of PNP clients. The PNP clients model NAS decision making such as in traffic flow management, airspace reconfiguration and user operations. The PNP clients may be created in various computer languages, and during a simulation run they may dynamically enter and exit the run.

PNP Overview

SW Architecture

PNP is a server, and clients can dynamically enter and exit during the run, using the PNP interface. Clients can request a variety of data messages, and can return to PNP modifications, including flight plan modifications and airspace (sector) redesigns.



PNP Server

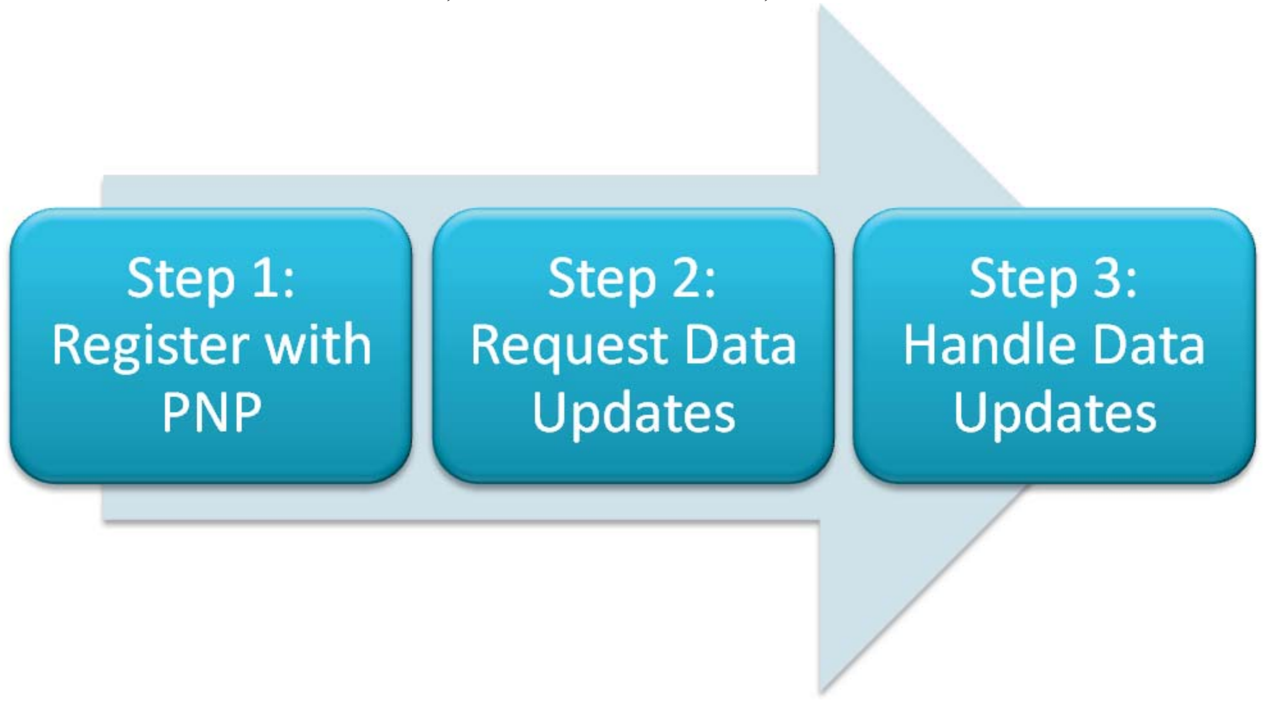
The PNP server provides data to clients using the SimObjects API. The following sections describe methods of requesting data from PNP and the types of data that may be requested by clients.

Data

PNP uses several types of data. These include meteorological data provided by WSI, including radar reflectivity, winds, and ceiling and visibility. PNP also uses NAS airspace data, airport location and capacity, demand set data containing flight data sets (FDSs), and BADA aircraft performance data. Please refer to the PNP User Agreement for data use requirements.

PNP Clients

PNP clients model NAS components. By connecting to the server and registering with PNP, a client can request the data it needs to model NAS components and decision making.
Clients may be written in either Java or MATLAB, and may be standalone or middleware. Standalone clients perform all processing themselves, whereas middleware clients gather data from PNP, run an external program to process the data, and send the results back to PNP.
Whether a client is Java or MATLAB, standalone or middleware, it will need to follow these three basic steps:


Step 1: Register with PNP
Step 2: Request Data Updates
Step 3: Handle Data Updates

The PNP Communications Cycle

The PNP server communicates with clients in a synchronous cycle. As PNP advances time in the simulation, it will periodically reach a time at which messages need to be sent to its clients. At each time point when the PNP server needs to send messages, it will send the appropriate messages, send a heartbeat message to the clients, and wait for the receiving clients to respond. Only after all receiving clients have responded will the PNP server continue to advance time. This asynchronous communications cycle ensures repeatability of the simulation, and is illustrated in Figure 2.

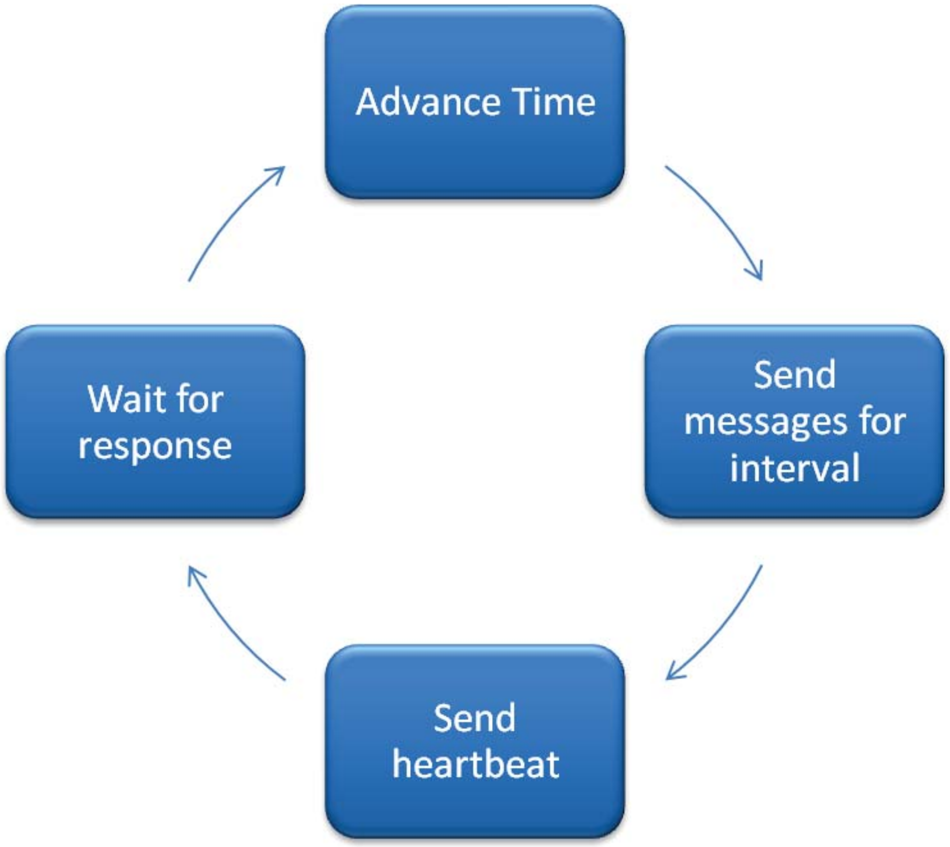


Figure 2. The PNP Communications Cycle

The SimObjects API

SimObjects is the PNP API. It uses TCP/IP to transfer messages between PNP and its clients.

Messaging

The SimObjects API consists of over forty messages, which are used to transfer data between PNP and its clients.
Messages in PNP exist as ***`Serializable`*** Java classes named after the data they contain. `PnpFlightDetails`, for example, contains a list of flights and information pertaining to those flights. All message classes exist in the package ***pnp.share.network.messages***.
Please refer to the SimObjects API documentation for details on API messages.

Dynamic Clients

Clients in PNP are referred to as dynamic because they can enter the simulation at any time. When connecting to PNP, each client specifies its configuration parameters, along with its data requirements. This provides for a flexible client/server architecture, where the server does not require advance knowledge of its clients, but can provide targeted data to each client based on its requirements.

Creating a PNP Java Client

Required Libraries

The following libraries need to be included by a Java client.

Library	Use
pnp_common.jar	Contains shared API messages
commons-logging-api.jar	Used for error logging
commons-logging-av.jar	Used for error logging
log4j-1.2.9.jar	Used for logging
seagull-common.jar	Contains many utility functions utilized by several API messages
seagull-position.jar	Contains classes for defining position data
seagull-units.jar	Used by position classes for specifying position units

Connecting to PNP from a Java client

Connecting a client to PNP is easy and requires only three steps.

Step 1: Register for Messages

In order to connect and register with PNP, a client must create and send a list of message requests to PNP. These requests specify the messages PNP should send to the client, as well as how often each message should be sent. To create a message request, simply register for messages using the *MessageRegistryManager* singleton. Three parameters are required to register a message:

- The class of the API message object
- The interval, in minutes of simulation time, at which PNP is to send the message to the client
- An object implementing the ApiListener interface to invoke when the requested message type is received

The ApiListener interface contains only one method, which will be called when the requested message is received. The only parameter of the *actionPerformed* method is the message object. The signature of the method is:

```
    void actionPerformed(Object o);
```

Example

The following example requests a list of all enroute flights every 15 minutes:

```
    /**
     * Register for messages desired to be received from PNP
     */
    private void registerForMessages()
    {
        // We'd like to receive all messages at 15-minute intervals
        int interval = 15;

        MessageRegistryManager registry = MessageRegistryManager.instance();

        // Register for messages
        registry.register(PnpEnrouteFlightDetails.class, interval, m_MessageHandler);
    }
```

In this example, *m_MessageHandler* is an *ApiListener* object. In your client, you may use the same *ApiListener* to handle many messages, or you may use different *ApiListener*s for different messages or sets of messages. It is up to you.

Step 2: Create a Client Configuration Object

Now, you need to create a *ClientConfiguration* object to tell PNP a little bit about your client. Some rudimentary information is required, such as the name of your client, so that PNP knows how to list your client in the System Monitor. Other settings in the configuration object are optional, and you may use these as required by your client.

A full description of the methods in *ClientConfiguration* is available in the Javadoc for the API, available in the ***doc/*** directory of your PNP release.

To create a basic *ClientConfiguration* object, simply call the constructor of *ClientConfiguration* with the name of your client as the only parameter.

```
        // Create a configuration object for this client
        ClientConfiguration config = new ClientConfiguration("My First Client");
```

Step 3: Initialize the Connection

Once the client configuration object has been created, the connection with the server can be initiated. To start the connection, simply create a *PnpFlightDetailsDescr* object and use it to call the *init* method to start the connection.
The *PnpFlightDetailsDescr* object simply gives the PNP server some guidance as to what information to include in flight data sent from the server to the client. This allows the server to optimize network usage by not sending data to the client that the client doesn't need.

Example

The following example creates a *PnpFlightDetailsDescr* that requires that only the sector schedule data be filled out for flight objects sent by the server.

```
        // Set up the flight details descriptor
        PnpFlightDetailsDescr flightDetailsDescr = new PnpFlightDetailsDescr();
        flightDetailsDescr.setSectorScheduleFlag(true);
```

At this point, we are ready to connect to the PNP server. To do this, simply call *init*:

```
        // Connect to PNP
        init(pnp_server_name, config, flightDetailsDescr);
```

Here, *pnp_server_name* is the network name of the host computer on which PNP is running, and *config* is the *ClientConfiguration* object.

Real-world Example

The following is a real-world example of code used to connect a client to the PNP server, and shows how all the pieces fit together to establish the client-server connection.

```java
public LpThread(String serverName)
{
    m_ServerName = serverName;

    registerForMessages();
    initPnpConnection();
}

/**
 * Connect to the PNP server
 */
private void initPnpConnection()
{
    // Obtain the name of the PNP server to connect to
    String pnp_server_name = m_ServerName;

    // Create a configuration object for this client
    ClientConfiguration config = new ClientConfiguration("LP Solver Gateway");

    config.setDataLookaheadTimeBinOffset((int) m_LookaheadTime.as(Units.Minutes) / 15);

    // Set up the flight details descriptor
    PnpFlightDetailsDescr flightDetailsDescr = new PnpFlightDetailsDescr();
    flightDetailsDescr.setSectorScheduleFlag(true);

    // Connect to PNP
    init(pnp_server_name, config, flightDetailsDescr);
}

/**
 * Register for messages desired to be received from PNP
 */
private void registerForMessages()
{
    // We'd like to receive all messages at 15-minute intervals
    int interval = 15;

    MessageRegistryManager registry = MessageRegistryManager.instance();

    // Register for messages
    registry.register(PnpAtGateFlightDetails.class, interval, m_MessageHandler);
    registry.register(PnpEnrouteFlightDetails.class, interval, m_MessageHandler);

    registry.register(Heartbeat.class, interval, m_MessageHandler);
    registry.register(Sync.class, interval, m_MessageHandler);

    registry.register(SectorIdMap.class, interval, m_MessageHandler);
    registry.register(SectorNameMap.class, interval, m_MessageHandler);
    registry.register(SectorMxMap.class, interval, m_MessageHandler);
    registry.register(SectorCount.class, interval, m_MessageHandler);
    registry.register(SectorWxCoverage.class, interval, m_MessageHandler);
    registry.register(SectorRxReduction.class, interval, m_MessageHandler);
```

```
        registry.register(AirportCapacities.class, interval, m_MessageHandler);
        registry.register(AirportFlightRules.class, interval, m_MessageHandler);

        registry.register(TerminalConditions.class, interval, m_MessageHandler);
    }
```

Sending data from your client

Sending data from a client to the PNP server is easy. The OCMM contains a *send* method which will send any message to PNP. The message will be a *Serializable* object from the ***pnp.share.network.messages*** package of PNP.
Using the class from the previous section, the following method could be used to send messages back to PNP:

```
        private void send(Object message)
        {
                m_Client.send(message);
        }
```

Responding to PNP heartbeat messages

Once the PNP server has sent all requested messages to your client for the given interval, it will send a `Heartbeat` message to your client, and will await a response before moving the simulation forward. The purpose of this is to keep the simulation synchronous and repeatable. The client should only respond after it has processed all of the data it has received for that interval.
To respond to the heartbeat message, the client need only send a `HeartbeatResponse` message to the server:

```
        send(new HeartbeatResponse())
```

Interacting with PNP

A client can interact with PNP by modifying flights. PNP clients can delay or reroute flights, and the following sections explain how to do just that.

The BatchResponse Message

All delays and reroutes are performed using the BatchResponse message. The BatchResponse message allows a client to specify which (if any) flights to delay or reroute. This message should be sent to PNP once the client has received a Heartbeat message and has done all of its processing.
The BatchResponse message contains the following constructor:

```
    public BatchResponse(int dstId,
                         Map<Integer, Short> delayMap,
                         Map<Integer, ReroutePlan> rerouteMap,
                         Map<Integer, ReroutePlan> inflightRerouteMap)
    {
        m_DstId = dstId;
        m_DelayMap = delayMap;
        m_RerouteMap = rerouteMap;
        m_InflightRerouteMap = inflightRerouteMap;
    }
```

Delaying Flights

The delay map in the BatchResponse is a map that is ordered by flight ID. Each flight ID maps to a *short* value specifying the number of minutes by which to delay the flight. PNP will apply the specified delay to each flight in the *delayMap* provided.

Rerouting Flights

Flight reroutes fall into two categories:

- Pre-departure reroutes, applied to flights that have not yet departed
- Inflight reroutes, applied to flights that are already enroute

As with delays, the maps specified for reroutes are keyed by flight ID. The ReroutePlan provided simply contains the new list of 3-dimensional positions, listed at one-minute intervals, for each flight. The *rerouteMap* specifies any pre-departure reroutes requested by the client. The *inflightRerouteMap* specifies any inflight reroutes requested by the client.
All reroutes requested by the client are implemented by PNP.

Best Practices for developing PNP clients

In order to make the best use possible of PNP and SimObjects, the following have been identified as best practices to follow when creating clients for PNP:

- **Create separate threads for processing vs. connection management**

  A PNP client performs whatever simulation tasks are appropriate for it. While it is performing this processing, it may need to send or receive messages from PNP. Keeping the communications handling code in a separate thread from the processing code allows messages to be sent and received while the processing code is running. This added flexibility significantly enhances the capability of any client developed in this way.

- **Send all processing results to PNP before sending a `HeartbeatResponse`**

  A `HeartbeatResponse` tells PNP that your client has finished its processing. Once PNP has received this message, it continues advancing time, and will send a new set of messages to your client for processing. If the client has not finished sending its results to PNP before it receives the new data, the repeatability of the simulation may be affected, as the order in which messages are sent and received may change if the same simulation is run again. Additionally, if the client's internal data are not cleared before the new data are received, the client may end up with incorrect data. To ensure against this, the client should wait until after sending all its results before sending the `HeartbeatResponse` message.

Glossary

Term	Definition
API	Application Programming Interface
NAS	National Airspace System
OCMM	Object Client Message Manager. Utility for managing a client’s data connection with PNP.
PNP	Probabilistic NAS Platform
SimObjects	The PNP API

Appendix A: Conflict Detection and Resolution

Some client applications detect airspace conflicts, or imminent losses of separation between aircraft. The PNP Plan View Display (PVD) is capable of depicting conflicts, as well as losses of separation, when data on such events are supplied by a client.

Conflict-related API Messages

The PNP API contains two messages that may be used by clients for pointing out airspace conflicts and losses of separation. These are:

- ***ConflictList***
  The *ConflictList* message contains a list of *Conflict* objects, each representing an imminent loss of separation between two aircraft.

- ***LosList***
  The *LosList* message contains a list of *LossOfSeparation* objects, each representing a current loss of separation between two aircraft.

Both of these messages are documented in the API JavaDoc documentation contained in the ***doc/*** folder of your PNP release.

Display of CD&R Information

When a conflict or loss of separation is sent from the client to the server, the server will display the conflict on the PVD for the duration of the time bin. This information is cleared by the server at the end of the time bin, at which it is allowed to expire if not re-sent by the client. This behavior exists because conflicts or losses of separation may be resolved during the time bin. Therefore, a client that detects conflicts and loss of separation (LOS) events must re-send all conflict and LOS information at every time bin during which the conditions are active.

Probabilistic NAS Platform

v1.5

User Guide

8/16/2011
Sensis Corporation

TABLE OF CONTENTS

INTRODUCTION

The Probabilistic NAS Platform (PNP) is both a research tool and a decision support platform to analyze air traffic and traffic flow management (TFM) issues. Real-time PNP provides real time meteorological data and traffic advisories in which an operator or Air Traffic Manager can make more informed and accurate decisions. Playback PNP evaluates current or future TFM concepts using projected or actual traffic demands sets with recorded historical meteorological data.

SCOPE

This document guides the PNP user in the design and execution of PNP simulations.
Topics covered include:

- Software Overview
- Configuring PNP
- Running PNP

SOFTWARE

ARCHITECTURE

The PNP software has a client/server architecture. PNP supports multiple clients. Clients may be distributed locally or remotely. As **Figure 3. PNP Architecture** illustrates, these processes communicate with one another via the SimObjects API.

SimObjects coordinates the exchange of data between the PNP server and its clients in the form of serialized Java objects. Data integrity is assured through the use of TCP/IP communications on dedicated socket channels. PNP clients may dynamically join a simulation, and use a subscribe mechanism to request periodic, customized data updates from PNP.

Clients may be developed in Java or MATLAB. Additionally, external, third-party software may be integrated into a PNP simulation through the use of a dedicated middleware client, which communicates both with the external software using any desired mechanism, and with PNP server using the SimObjects API.

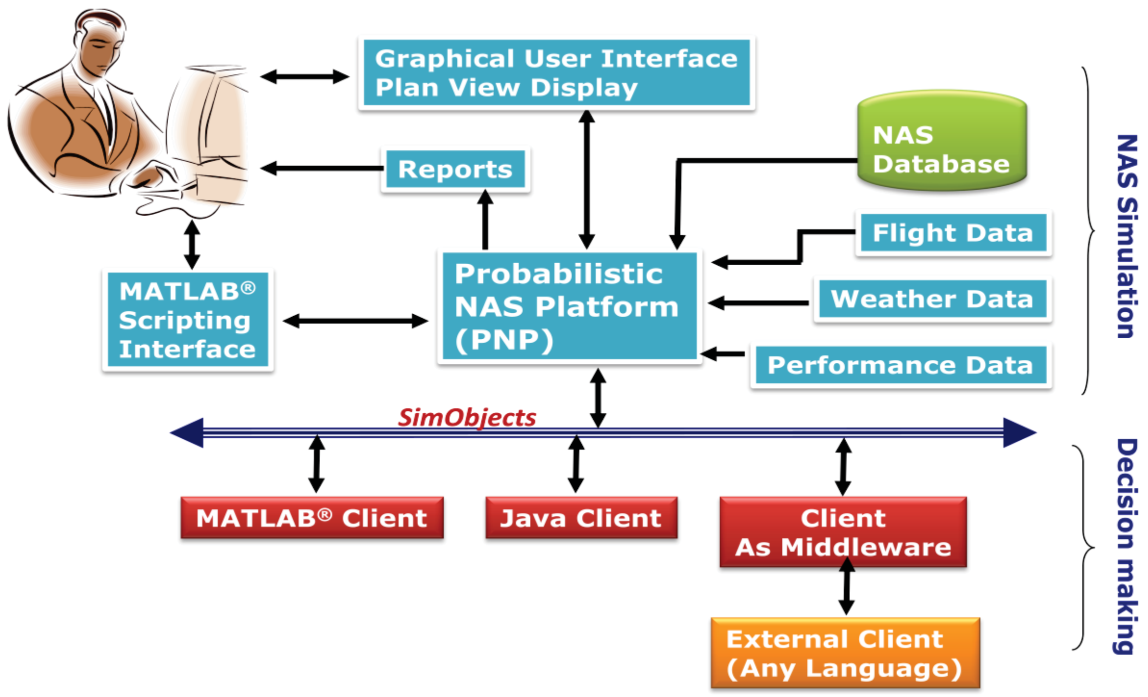


Figure 3. PNP Architecture

DESIGN

SERVER

The PNP executive is the server. It is responsible for the execution and flow control of the system. The PNP executive manages communications, airport capacities, sector definitions (geometries/capacities/monitor alert parameters), display updates, flights (trajectory models/persistence), and weather models.

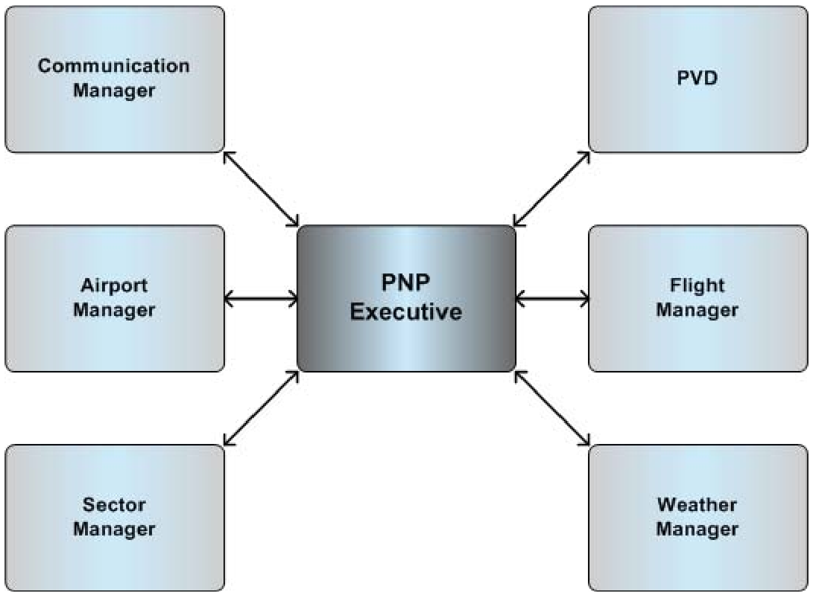


Figure 4. PNP Server Design

CLIENTS

The PNP architecture is scalable and supports an unlimited number of clients. Clients may be created using Java or MATLAB, and communicate with PNP using the SimObjects API. PNP's client-server architecture is plug-and-play, so PNP requires no advance knowledge of clients. Clients are added to a PNP simulation by specifying the configuration file for each client to be run. Each client's configuration file specifies how to run the client, so once the configuration file is specified, PNP requires no knowledge about the client.

When a client joins the PNP simulation, it registers with PNP and specifies what types of data it will require, and at what intervals. Clients may request many types of airspace data, including traffic and weather data. *See the SimObjects API documentation for details*.

Clients may process data received from PNP and take action that directly affects the simulation. Clients may delay or reroute flights, or modify the airspace. *See the SimObjects API documentation for details*.

PROBABILISTIC TFM (PROBTFM) CLIENT

The Probabilistic TFM Client determines delays and reroutes to reduce congestion by creating a probabilistic distribution based on sector and airport congestion.

WEATHER AVOIDANCE (WAC) CLIENT

The Weather Avoidance Client determines delays and reroutes to avoid congestion and weather based on sector loading and reflectivity data***.*** ***(Work in progress)***

AIRLINE OPERATIONS (AOC) CLIENT

The Airline Operations Client determines delays and reroutes based on airline schedules and operational mode.

GROUND DELAY PROGRAM (GDP) CLIENT

The Ground Delay Program Client determines delays based on airport congestion.

SEPARATION ASSURANCE (SA) CLIENT

The Separation Assurance Client determines reroutes based on loss of separation events.

REQUIRED TIME FO ARRIVAL (RTA) CLIENT

The Required Time of Arrival Client determines reroutes based on meeting scheduled RTAs. ***(Work in progress)***

DYNAMIC AIRSPACE (DAC) CLIENT

The Dynamic Airspace Client determines sectors (subsectors) definitions and capacities.

WSI CLIENT

The WSI Client sends live weather products to the PNP Server.

MESSAGES

The messages in the system are serialized objects. The following section identifies the current messages in the system.

REQUEST MESSAGES

Request Messages	
Message Type	**Source**
PNP Flight Details	**Server**
Sector Loading	**Server**
Airport Loading	**Server**
Airport Conditions	**Server**
Airport Capacities	**Server**
Reroute Request	**Client**
Sector Geometries	**Server**

Heartbeat	**Server**

RESPONSE MESSAGES

Response Messages	
Message Type	**Source**
Delays	**Client**
Reroutes	**Client**
Reroute Response	**Server**
Sector Data	**Client**
See SimObjects API documentation for complete details	

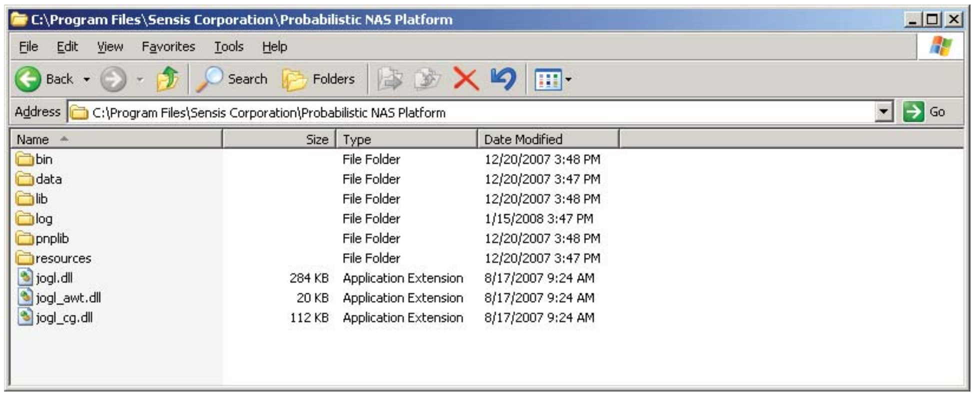
INPUTS

DATA SETS

PNP requires the following data sets to define its airspace and aircraft.

DIRECTORIES

The PNP directory structure contains six sub-directories, as shown in Fig. 4. The ***bin*** directory contains the command files necessary to run the software. The ***log*** directory contains the runtime output and error logs. If a user wants to report a problem with the system, it is necessary that all the ***log*** directory files be forwarded to the PNP maintenance staff. The ***lib***, ***pnplib***, and ***resource*** directories contain the files necessary to run the PNP application.


C:\Program Files\Sensis Corporation\Probabilistic NAS Platform
File Edit View Favorites Tools Help
Back Search Folders
Address C:\Program Files\Sensis Corporation\Probabilistic NAS Platform
Go

```
Name                Size  Type                   Date Modified
bin                       File Folder            12/20/2007 3:48 PM
data                      File Folder            12/20/2007 3:47 PM
lib                       File Folder            12/20/2007 3:48 PM
log                       File Folder            1/15/2008 3:47 PM
pnplib                    File Folder            12/20/2007 3:48 PM
resources                 File Folder            12/20/2007 3:47 PM
jogl.dll          284 KB  Application Extension  8/17/2007 9:24 AM
jogl_awt.dll       20 KB  Application Extension  8/17/2007 9:24 AM
jogl_cg.dll       112 KB  Application Extension  8/17/2007 9:24 AM
```

DATA

The ***data*** directory contains information necessary for PNP to run, as well as data required to seed the PNP application.

BADA

The ***bada*** subdirectory contains all the files necessary for the trajectory engine to fly aircraft. The trajectory engine uses these BADA performance data to calculate the exact position of each aircraft at every minute of the simulation, giving a high-fidelity representation of all the traffic in the NAS.

CONFIG

The ***config*** subdirectory contains information necessary to configure PNP. This information includes specifications on available clients, as well as other data required to run PNP. When adding a client to PNP, the clients.cfg file must be updated to reflect changes to the list of clients.

DISPLAY

The ***display*** subdirectory contains initialization files used to configure the PNP displays.

INI

The ***ini*** subdirectory contains initialization files used to configure the PNP server as well as its clients. Each client has a unique set of configuration commands. Please refer to the documentation for each client for instructions on how to properly configure it.

PVD

The ***pvd*** subdirectory contains all the files necessary to display objects on the PVD.

REPORTS

The ***reports*** subdirectory holds all the runtime reports that are generated. The runtime reports use the comma-separated values file format (.CSV) that stores tabular data. This format is commonly acceptable by any spreadsheet application, word processor, and MATLAB.

SECTORMAPPING

The ***sectormapping*** subdirectory contains all the files necessary to define the sectors of the system. PNP is installed with sector data from January, 2007. However, the user may provide a different set of sector data, and configure PNP to use those data. Please refer to the Airspace Parameters section of this document for details on how to configure PNP sector data parameters.

FILES

FLIGHT DATA SETS

The FDS Analysis is used to analyze future and current air traffic modeling. AvDemand is used to generate flight data sets for PNP experiments. The flight data set format is tabular data used to capture a flight with the following fields:

FDS Fields		
Field Name	**Used by PNP?**	**Example Value**
Flight ID		**34**
Airline	√	**AAL234/009**
Aircraft Type	√	**B752**

Departure Airport	√	**LAX**
Arrival Airport	√	**MIA**
Original Fixes	√	**2037/7104 2018/6970 2010/6840 1950/6600 1909/6377 1857/6179 1836/5989 1751/5406 1569/4907 1548/4817**
Modified Fixes		
Cruise Altitude	√	**350**
Cruise Speed	√	**452**
Scheduled Gate Departure Time		**10**
Gate Departure Trigger Time		**-1**
Gate Ground Delay Departure Time		**-1**
Updated Gate Departure Time	√	**810**
Flight Plan Trigger Time		**-1**
Flight Cancelled Flag		**55**

SECTOR GEOMETRIES AND MONITOR ALERT PARAMETERS

These data sets create the sectorization for PNP.

Airspace Data Sets

Data Sets	Directory
Sector Mapping	C:\ARCHIVE\AIRSPACE\sectormapping

AIRSPACE DATA SETS

The airspace data sets are used to create the airspace for PNP. The NFDC data sets are provided by the FAA every 56 days. The City Pairs data sets were data mined from historical ASDI routes. The AvDafif data sets were obtained from the National Geospatial-Intelligence Agency. The Airport Mapping datasets were created by Sensis Systems Engineers. The custom data sets have been created to fill gaps in the airspace definition. These data sets are used to define PNP's airspace.

Airspace Data Sets	
Data Sets	**Directory**
NFDC	C:\ARCHIVE\AIRSPACE\nfdc
Fallingrain	C:\ARCHIVE\AIRSPACE\fallingrain
City Pairs	C:\ARCHIVE\AIRSPACE\citypairs
AvDafif	C:\ARCHIVE\AIRSPACE\avdafif
Airport Mapping	C:\ARCHIVE\AIRSPACE\airportmapping
Custom	C:\ARCHIVE\AIRSPACE\custom

BASE OF AIRCRAFT DATA (BADA)

The BADA data is used by the Sensis Trajectory model to simulate aircraft performance.

PNP SERVER PARAMETERS

The PNP Server parameters control the runtime attributes of the experiment. The parameters control what FDS, weather, sectors, and airspace data an experiment utilizes. The parameters control the speed, display, and models utilized in an experiment.

The various PNP Server parameters are specified in the PNP Server initialization (.ini) file and are broken up into parameter sets. The following sections describe the PNP Server parameter sets found in the PNP initialization file.

CONTROL PARAMETERS

Control parameters provide basic connection and initialization information for PNP Server. These parameters do not typically change between experiments, and are meant to be used to standardize a set of runs and ensure that some basic parameters are used across several sets of runs.

Control Parameters

Field Name	Field Description	Default Value
starttime	The starttime parameter specifies the start time used to begin processing the FDS air traffic dataset. This is the start time of the simulation.	07:45
playbackruntime	The playbackruntime parameter specifies the number of hours of flights from the FDS to process. A value of 24, for example, will simulate 24 hours of departures. The simulation runs until all aircraft have landed.	24
fdsoffset	The fdsoffset parameter specifies the number of hours to offset the processing to the FDS. This is used to compensate for time discrepancies in the FDS such as time zones. For example, if an FDS was created with departure times local to the GMT-8 (Pacific) timezone, the departure times in the flight need to be offset by +8 hours to convert them to GMT, in which PNP operates.	8

fasttimeflag	The fasttimeflag parameter determines the speed of the PNP clock during playback. If this flag is set to *true,* PNP will use 100% of the CPU and run as fast as possible. Note: This will make the PVD interaction slower and less responsive. Setting the flag to *false* allows the system to attempt to run in uniform time.	true

KNOB PARAMETERS

Knobs represent a series of parameters that can be adjusted to tweak the behavior of the simulation.

Knob Parameters

Field Name	**Field Description**	**Default Value**
lookaheadtime	Specifies the number of hours of departures to pre-load in advance of the current simulation time. This allows predictive algorithms (such as the at-gate loading counter) to have visibility into future NAS loading	2
airportcapacitymultiplier	The factor by which to multiply all airport capacities	1.0
sectorcapacitymultiplier	The factor by which to multiply all sector capacities	1.0
greatcircleroute	Specifies whether to disregard filed flight plan routes and use great circle routes instead	false
pointmasstrajectory model	Specifies whether PNP should use the Point Mass trajectory model for aircraft simulation ▪ When set to *true*, PNP will use the point mass model, which is fast and has high accuracy. ▪ When set to *false*, PNP will use the kinematics model, which is ultra fast and has slightly lower accuracy	false

WEATHER PARAMETERS

The weather parameters control how PNP processes weather data.

Weather Parameters

Field Name	Field Description	Default Value
Weather Timestamp Offset	The Weather Timestamp Offset parameter contains the number of hours to offset the processing on the WSI archive data. This is used to compensate for time discrepancies in the WSI recorded archive.	0
reflectivityflag	The Reflectivity parameter instructs the PNP to enable processing of reflectivity data	true
Propagate Intervals	The Propagate Intervals parameter instructs the PNP to use a reflectivity persistence forecast for X time intervals, when forecasting convection. The persistence forecast is sometimes more desirable than forecast polygons. *The recommended use of persistence is at least one hour (an entry of 4).*	4
forecast	The forecast parameter instructs the PNP to process the WSI forecast convection polygons and make the forecast available to clients. The forecast convection polygons obtained from WSI are applied to same grid as the sectors. The percentage grid coverage is computed per sector. This convection forecast coverage is prorogated forward based on the convection time window provided by WSI.	false
windsflag	The Winds parameter instructs the PNP to process the WSI wind data. The winds data are used by the trajectory	false

	models.	
metarreportsflag	The METAR parameter instructs the PNP to process the Aviation Routine Weather Reports.	true
tafreportsflag	The TAF parameter instructs the PNP to process the Terminal Area Forecasts.	true

DISPLAY PARAMETERS

The display parameters affect the amount of information that is shown on PNP's graphical display during a simulation.

Display Parameters

Field Name	Field Description	Default Value
pvdlevel	The PVD Level parameter specifies the amount of information to show on the display. Acceptable values range from 0 to 3, where a value of 0 disables the PVD entirely, and a value of 3 enables the display of all available information.	0

Level	Performance Description
0	No PVD
1	PVD with airport & sector congestion only.
2	PVD with congestion and aircraft only.
3	Interactive mode. PVD with congestion, aircraft, and weather. **Note: The system runs slower as the*

		processor is freed to update the PVD requests.

chart	The Chart parameter specifies whether a statistics chart should be shown during the simulation. The chart contains real-time statistics on delays, reroutes, and number of flights processed.	true
datestring	The Date String parameter specifies the date string to be shown at the top-right corner of the PVD (if the PVD is shown). This date should match the date of the simulation data being processed.	

DATASET PARAMETERS

The Dataset parameters control which files PNP uses as its source for weather and traffic data. Note that all directory definitions use the double-backslash (\\) to separate directories. Double backslashes are necessary for the proper operation of PNP.

Dataset Parameters

Field Name	Field Description	Default Value
wsiarchive	The WSI Archive parameter is the directory of the WSI archive to be processed. The WSI Archive contains all the weather data. If a user wants to run with weather, this field needs to contain the WSI Archive and the appropriate Weather Parameters must be selected.	
avdemandinputfile	The AvDemand Input File parameter is the directory and filename of the Flight Data Set (FDS) file containing all the planned flights for the simulation.	
airportcapacityfile	The Airport Capacity File parameter is the directory and filename of the airport capacity definitions.	\\ARCHIVE\\AIRSPACE\\airportmapping\\ 2007_baseline_capacities.csv
tafairportfile	The TAF Airport File parameter is the directory and filename of the terminal airports of interest.	\\ARCHIVE\\AIRSPACE\\airportmapping\\ 2006_taf_airports.csv

AIRSPACE PARAMETERS

The airspace parameters control which files PNP uses as its source for airspace data. Note that all directory definitions use the double-backslash (\\) to separate directories. Double backslashes are necessary for the proper operation of PNP.

Airspace Parameters

Field Name	Field Description	Default Value
sectoraltitudelmits	The Sector Altitude Limits parameter is the directory and filename of the sector altitude limit definitions.	\\ARCHIVE\\AIRSPACE\\sectormapping\\ 2007_sector_altitude_limits.csv
supersectorfile	The Super Sector File parameter is the directory and filename of the super sector geometry definitions.	\\ARCHIVE\\AIRSPACE\\sectormapping\\ 2007_01_18_sectors_super.dat
highsectorfile	The High Sector File parameter is the directory and filename of the high sector geometry definitions.	\\ARCHIVE\\AIRSPACE\\sectormapping\\ 2007_01_18_sectors_high.dat
lowsectorfile	The Low Sector File parameter is the directory and filename of the low sector geometry definitions.	\\ARCHIVE\\AIRSPACE\\sectormapping\\ 2007_01_18_sectors_low.dat
traconfile	The TRACON Geometries File parameter is the directory and filename of the TRACON geometry definitions.	\\ARCHIVE\\AIRSPACE\\sectormapping\\ 2007_01_18_tracon.dat
sectorcapacityfile	The Sector Capacity File parameter is the directory and filename of the sector capacity definitions.	\\ARCHIVE\\AIRSPACE\\sectormapping\\ 2007_sector_capacities.dat

OUTPUT PARAMETERS

The Output parameters specify what data should be output by PNP.

Output Parameters

Field Name	Field Description	Default Value
description	A description of the experiment	
experiment	The name of the experiment	
flightdataoutput	The Flight Data parameter instructs PNP whether to generate flight details data or not.	true
flightplanreroutesoutput	The Flight Plan Reroutes Output parameter instructs PNP whether to generate flight plan reroutes or not.	true
sectorloadingoutput	The Sector Loading Output parameter instructs PNP whether to generate sector loading data or not.	true
airportloadingoutput	The Airport Loading Output parameter instructs PNP whether to generate airport loading data or not.	true
sectorcapacityoutput	The Sector Capacity Output parameter instructs PNP whether to generate sector capacity data or not.	true

airportcapacityoutput	The Airport Capacity Output parameter instructs PNP whether to generate airport capacity data or not.	true

OUTPUTS

The PNP generates several reports that can be used for analysis and post processing by other applications. These reports are described in the following sections.

REPORTS

SUMMARY REPORT

The Summary Report is generated by the Flight Data Set (FDS) Analysis. This report contains the following fields:

Field	Description
Experiment	The name of the expirement
TotalDepartures	Total number of departures
TotalMxCongestion	Number of airspace congestion events based on sector MAP value
AvgMxCongestion	Average airspace congestion per congestion event based on sector MAP value
TotalWxCongestion	Number of airspace congestion events based on weather reduced MAP value
AvgWxCongestion	Average airspace congestion per congestion event based on weather reduced MAP value
AirportMxCongestion	Number of airport congestion events based on airport capacity
AirportAvgMxCongestion	Average airport congestion per congestion event based on airport capacity
AirportWxCongestion	Number of airport congestion events based on weather reduced airport capacity

AirportAvgWxCongestion	Average airport congestion per congestion event based on weather reduced airport capacity
DepartureDelayCount	The number of pre-departure delays
PreDepartureDelay	The total minutes of pre-departure delays
AvgPreDepartureDelayCount	Average delay per pre-departure delay
PreDepartureRerouteCount	The number of pre-departure reroutes
PreDepartureRerouteDelay	The total minutes of flight time added for pre-departure reroutes that increase flight time
PreDepartureRerouteNegativeDelay	The total minutes of flight time savings for pre-departure reroutes that reduce flight time
PreDepartureRerouteAvgDelay	Average flight time difference per pre-departure reroute
InflightRerouteCount	The number of in-flight reroutes
InflightRerouteDelay	The total minutes of flight time added for in-flight reroutes that increase flight time
InflightRerouteNegativeDelay	The total minutes of flight time savings for in-flight reroutes that reduce flight time
InflightRerouteAvgDelay	Average flight time difference per in-flight reroute
TotalDelayCount	Total number of delay events
TotalDelay	Total minutes of flight time delay
AvgDelay	Average minutes of delay per departure
FuelBurn	Total fuel burn (pounds)

Date	Date of experiment
Runtime	Runtime of experiment
WxDate	WSI Archive directory
Descr	Descriptive test to track experiments
Revision	Software revision

SECTOR INFORMATION REPORT

The Sector Information Report maps the sector column in the AtGate Sector Loading, Enroute Sector Loading, WxMap Sector Capacity, and MxMap Sector Capacity Reports to a sector name.

ATGATE SECTOR LOADING REPORT

The AtGate Sector Loading Report is generated by the FDS Analysis. This report contains the sector loading for every time interval of runtime.

Field	Description
Interval	Interval value (0-95)
Sector1	Number of aircraft that are at the gate and contribute to this sector's loading
Sector …	Number of aircraft that are at the gate and contribute to this sector's loading
Sector 9999	Number of aircraft that are at the gate and contribute to this sector's loading

ENROUTE SECTOR LOADING REPORT

The Enroute Sector Loading Report is generated by the FDS Analysis. This report contains the sector loading for every interval of runtime.

Field	Description
Interval	Interval value (0-95)
Sector1	Number of aircraft that are enroute and contribute to this sector's loading
Sector …	Number of aircraft that are enroute and contribute to this sector's loading
Sector 9999	Number of aircraft that are enroute and contribute to this sector's loading

DELAY REPORT

The Delay Report is generated by the FDS Analysis. This report contains aircraft delay information.

Field	Description
flightIndex	Internal PNP id to track flights
aircraftId	Aircraft flight identifier
orig	Departure airport
dest	Arrival airport
bin	The departure interval based on wheels off time (0- 95).
predictedCost	Predicted cost
maxGPCvalue	Gross Predicted Cost
BNSI1	Bin number sector for the maxGPCIndex = 0
BNSI2	Bin number sector for the maxGPCIndex = 1
BNSI3	Bin number sector for the maxGPCIndex = 2
sectorload	Number of aircraft in departure sector
delayInMinutes	Total minutes of flight time delay
routeDelayInMinutes	The total minutes of flight time added for pre-departure reroutes that increase flight time
DelayCount	The number of times this flight has been delayed

delayType	Sector or airport delays

DEPARTURE REPORT

The Departure Report is generated by the FDS Analysis. This report contains aircraft departure information.

Field	Description
flightIndex	Internal PNP id to track flights
aircraftId	Aircraft flight identifier
continuousBin	The continuous departure interval based on wheels off time (0 - ∞).
bin	The departure interval based on wheels off time (0 - 95).
predictedCost	Predicted cost
maxGPCvalue	Gross Predicted Cost
BNSI1	Bin number sector for the maxGPCIndex = 0
BNSI2	Bin number sector for the maxGPCIndex = 1
BNSI3	Bin number sector for the maxGPCIndex = 2
BNSILoading	Bin number sector for the maxGPC Loading
flightTime	Minutes of flight time
wheelsOffTime	Predicted airborne time

orig	Departure airport
dest	Arrival airport

REROUTE REPORT

The Reroute Report is generated by the FDS Analysis. This report contains aircraft reroute information.

Field	Description
flightIndex	Internal PNP id to track flights
aircraftId	Aircraft flight identifier
orig	Departure airport
dest	Arrival airport
routeDelayInMinutes	The total minutes of flight time added/subtracted for pre-departure reroutes that increase/decrease flight time

MXMAP SECTOR CAPACITY REPORT

The MxMap Sector Capacity Report is generated by the FDS Analysis. This report contains the Monitor Alert Parameter (MAP) values for every sector for every interval of runtime.

Field	Description
Interval	Interval value (0-95)
Sector1	Sector MAP Value
Sector …	Sector MAP Value

Sector 9999	Sector MAP Value

WXMAP SECTOR CAPACITY REPORT

The WxMap Sector Capacity Report is generated by the FDS Analysis. This report contains the Weather Reduced Monitor Alert Parameter (MAP) values for every sector for every interval of runtime.

Field	Description
Interval	Interval value (0-95)
Sector1	Sector Weather Reduced MAP Value
Sector …	Sector Weather Reduced MAP Value
Sector 9999	Sector Weather Reduced MAP Value

AIRPORT CAPACITY REPORT

The Airport Capacity Report is generated by the FDS Analysis. This report contains the Airport Capacities for every interval of runtime.

Table 1 Airport Capacity Report

Field	Description
Interval	Interval value (0-95)
Airport	Airport name
AirportCapacity	Airport capacity value
WeatherReducedAirportCapacity	Airport weather reduced capacity value
AirportLoading	Airport loading for this interval

TAF REPORT

The Terminal Area Forecast Report is generated by the FDS Analysis when the TAF parameter is enabled, See

Field	Description
icaoSiteId	The ICAO station identifier
origbin	The interval when the report is received (0- 95).
startbin	The interval when the report takes affect (0- 95).
stopbin	The interval when the report expires (0- 95).

visibility	The number of miles of visibility
ceiling	The altitude of the ceiling
windSpeed	The Speed of the wind in knots
windDirection	The direction of the wind relative to true north
code	The Weather Reporting Notations
timestamp	The timestamp of the report message

METAR REPORT

The METAR Report is generated by the FDS Analysis when the METAR parameter is enabled.

Field	Description
icaoSiteId	The ICAO station identifier
origbin	The interval when the report is received (0- 95).
startbin	The interval when the report takes affect (0- 95).
stopbin	The interval when the report expires (0- 95).
visibility	The number of miles of visibility
ceiling	The altitude of the ceiling
windSpeed	The Speed of the wind in knots

windDirection	The direction of the wind relative to true north
code	The Weather Reporting Notations
timestamp	The timestamp of the report message

RUNNING PNP

EXPERIMENTS

PNP experiments are file driven, using four types of files: the Simulation Parameter, Client Parameter, Experiment Definition and Schedule Definition Files. Note that the Schedule Definition files are used to run PNP in a batch mode, and are not necessary to make a single run.

SIMULATION PARAMETER FILES

Simulation parameters files contain the PNP parameters listed in the PNP Server Parameters section. These parameters instruct PNP on how to run.

CLIENT PARAMETER FILES

Client parameter files contain the parameters specific to each client.

EXPERIMENT DEFINITION FILES

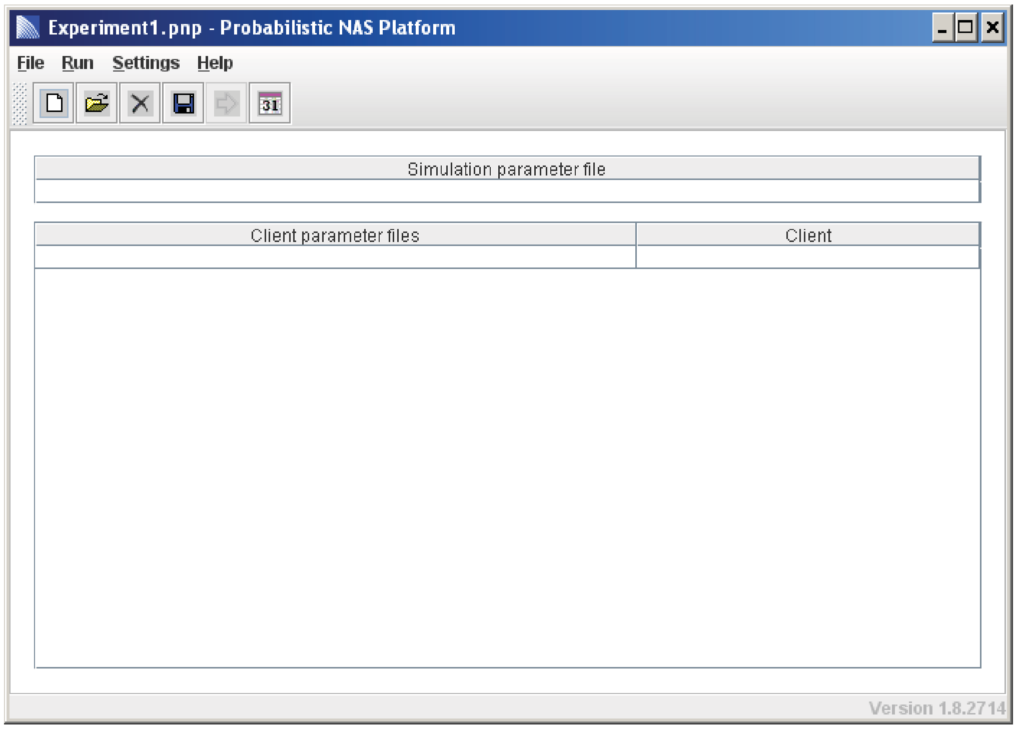
Experiment definition files contain the information necessary to run PNP and its clients, including the simulation parameter file name and client parameter file names.

SCHEDULE DEFINITION FILES

Schedule Definition Files contain the experiment definition files names to be run chronologically, in a batch mode.

INITIAL STARTUP

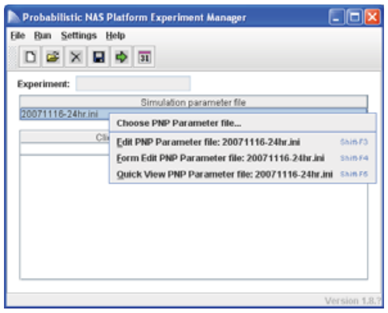
Upon initialization the PNP Window displays as follows:


Experiment1.pnp - Probabilistic NAS Platform
File Run Settings Help
Simulation parameter file
Client parameter files
Client
Version 1.8.2714

NAVIGATION: FILE SELECTION

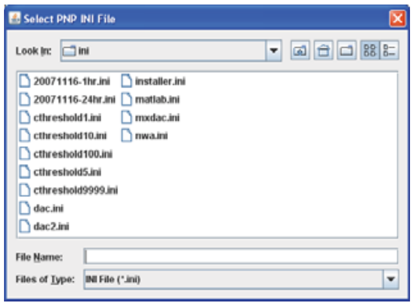
There are three ways to select a parameter file from an entry box.

1. Then perform one of the three files selection techniques.
    1. Left Mouse Button
        i. Double left mouse click
    2. Right Mouse Button
        i. Right mouse click and select **Choose file** menu option.


Probabilistic NAS Platform Experiment Manager
File Run Settings Help
Experiment:
Simulation parameter file
20071116-24hr.ini
Choose PNP Parameter file...
Edit PNP Parameter file: 20071116-24hr.ini Shift F3
Form Edit PNP Parameter file: 20071116-24hr.ini Shift F4
Quick View PNP Parameter file: 20071116-24hr.ini Shift F5
Version 1.8.7

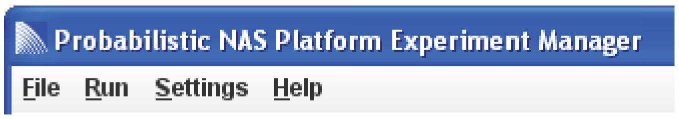
3. Insert Key
   i. Single left mouse click entry box to select it
   ii. Press ***Insert*** Key

2. Choose a file by double clicking the filename in the file chooser dialog



NAVIGATION: MENU USAGE

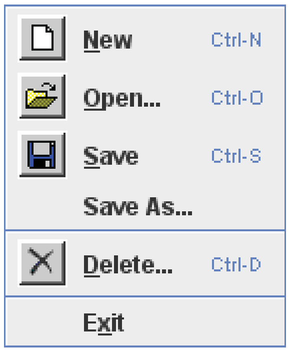
PNP provides a standard windows menu bar interface to navigate through the PNP application.



FILE MENU OPTIONS

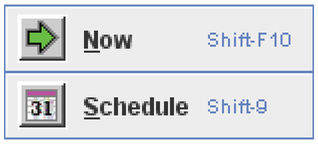
The file menu provides the following actions:

1. New – Creating a new experiment
2. Open – Open an existing experiment
3. Save – Save the current experiment
4. Save As – Save the current experiment with a new experiment name
5. Delete – Delete the current experiment



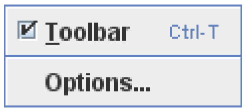
RUN MENU OPTIONS

1. Now – Executes the current experiment
2. Schedule – Opens the **Schedule Manager** Window



SETTINGS MENU OPTIONS

1. Toolbar – Toggles the toolbar on and off
2. Options – Opens the **PNP Options** Window

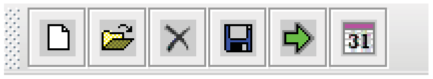


HELP MENU OPTIONS

1. Tips – Displays **PNP Navigation Tips** Window
2. About – Display **PNP Overview** Window

NAVIGATION: TOOLBAR USAGE

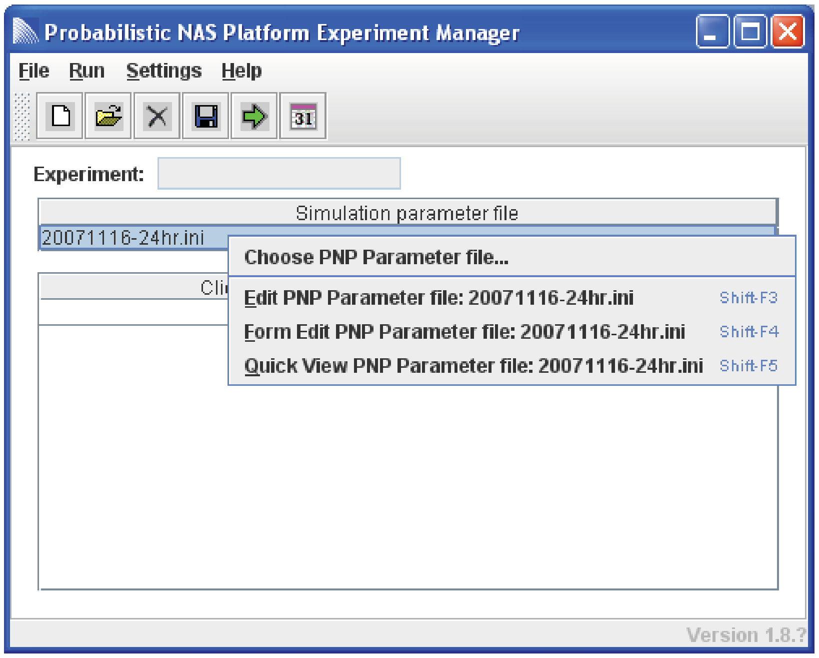
The toolbar provides an alternative method to the menu bar allowing quick single click access to the New, Open, Delete, Save, Run, and Schedule menu options.



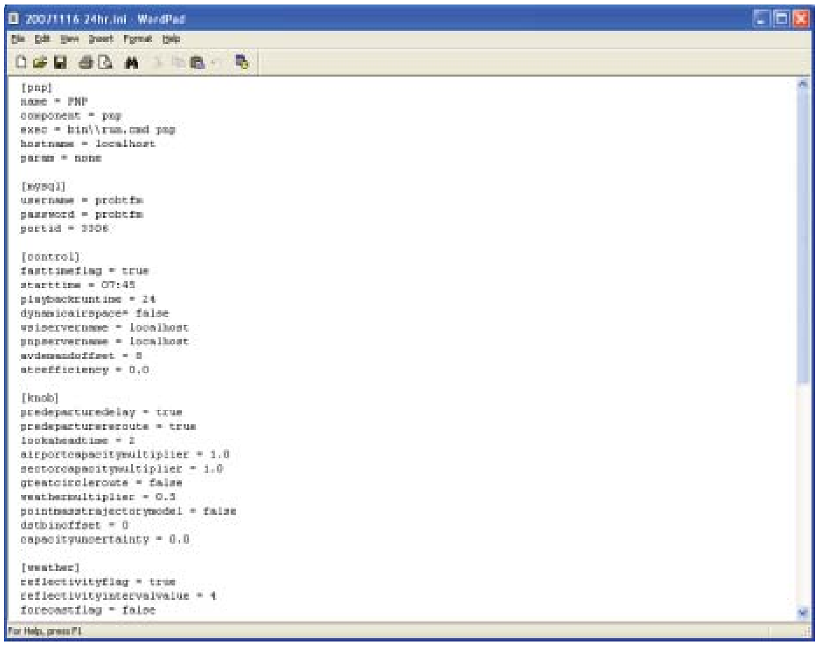
NAVIGATION: FILE MANIPULATION

There are several ways to view/edit/delete a parameter file from an entry box.

1. Right mouse click and select **Choose file** menu option.


Probabilistic NAS Platform Experiment Manager
File Run Settings Help
Experiment:
Simulation parameter file
20071116-24hr.ini
Cli
Choose PNP Parameter file...
Edit PNP Parameter file: 20071116-24hr.ini Shift-F3
Form Edit PNP Parameter file: 20071116-24hr.ini Shift-F4
Quick View PNP Parameter file: 20071116-24hr.ini Shift-F5
Version 1.8.?

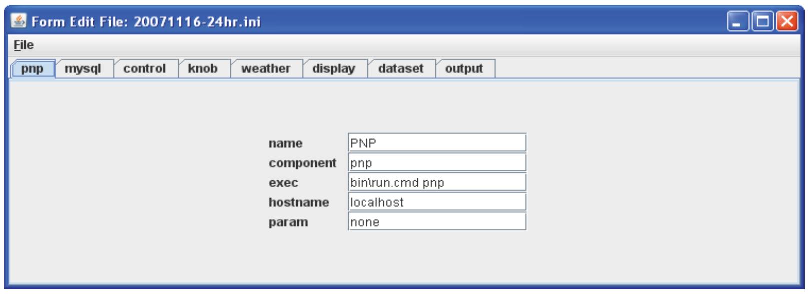
2. Press ***Shift-F3*** Key to edit the file in the user defined editor



20071116-24hr.ini - WordPad

```
[pnp]
name = PNP
component = pnp
exec = bin\\run.cmd pnp
hostname = localhost
param = none

[mysql]
username = probtfm
password = probtfm
portid = 3306

[control]
fasttimeflag = true
starttime = 07:45
playbackruntime = 24
dynamicairspace= false
wsiservername = localhost
pnpservername = localhost
avdemandoffset = 8
atcefficiency = 0.0

[knob]
predeparturedelay = true
predeparturereroute = true
lookaheadtime = 2
airportcapacitymultiplier = 1.0
sectorcapacitymultiplier = 1.0
greatcircleroute = false
weathermultiplier = 0.3
pointmasstrajectorymodel = false
dstbinoffset = 0
capacityuncertainty = 0.0

[weather]
reflectivityflag = true
reflectivityintervalvalue = 4
forecastflag = false
```

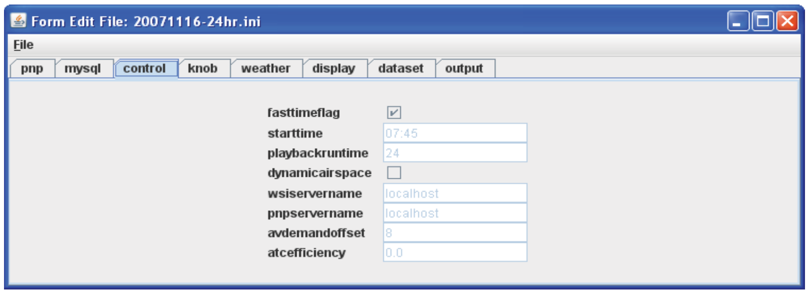
3. Press ***Shift-F4*** Key to edit the file in the form editor



Form Edit File: 20071116-24hr.ini

name	PNP
component	pnp
exec	bin\run.cmd pnp
hostname	localhost
param	none

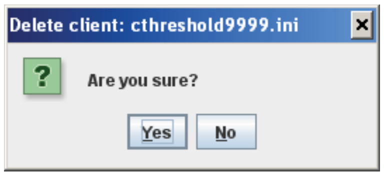
4. Press ***Shift-F5*** Key to edit the file in the form viewer



Form Edit File: 20071116-24hr.ini

fasttimeflag	✓
starttime	07:45
playbackruntime	24
dynamicairspace	
wsiservername	localhost
pnpservername	localhost
avdemandoffset	8
atcefficiency	0.0

5. Press Delete Key to detele the selected entry



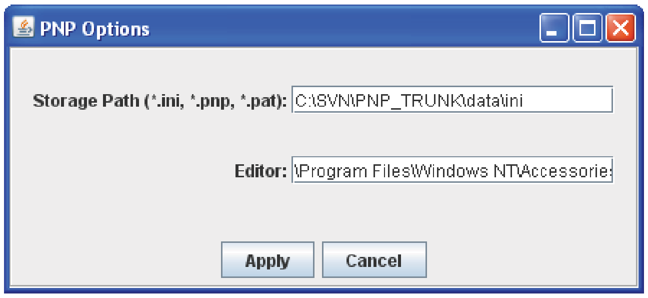
Delete client: cthreshold9999.ini

Are you sure?

SETTING THE USER DEFINED EDITOR

PNP allows the user to specify their editor by updating the **Editor** entry box in the **PNP Options Window**. The default editor is MS Windows XP wordpad.exe.

(\Program Files\Windows NT\Accessories\wordpad.exe)

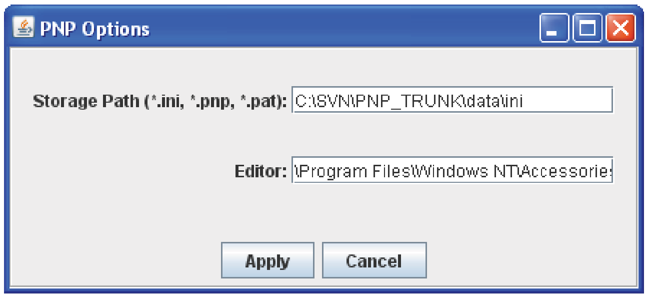

PNP Options

Storage Path (*.ini, *.pnp, *.pat): C:\SVN\PNP_TRUNK\data\ini

Editor: \Program Files\Windows NT\Accessorie

SETTING THE USER DEFINED PATH FOR PNP FILES

PNP allows the user to specify the path to the PNP files by updating the **Storage Path** entry box in the **PNP Options Window**. The default path is C:\Program Files\Sensis\Probabilistic NAS Platform\data\ini


PNP Options

Storage Path (*.ini, *.pnp, *.pat): C:\SVN\PNP_TRUNK\data\ini

Editor: \Program Files\Windows NT\Accessorie

CREATING A NEW EXPERIMENT

To create an experiment a user can use the menu to select ***File...New*** *or* press the hotkey: **Ctrl-N** *or* click the 🗋 icon on the toolbar.

1. Select a **Simulation parameter file**

   This file determines the data sets, weather, and configuration of the experiment.

2. Select **Client parameter files**

   These files determine what clients will be part of the current experiment.

**Note: An experiement must contain a Simulation parameter file.*

3. Save the experiment.

OPENING AN EXPERIMENT

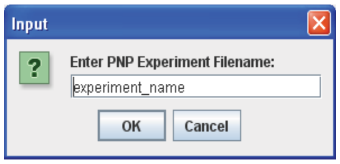
To open an experiment a user can use the menu to select ***File...Open…*** *or* press the hotkey: **Ctrl-O** *or* click the icon on the toolbar.

1. Choose experiment definition file from the file chooser dialog. The selected experiment definition is loaded into the Experiment Manager dialog.

SAVING AN EXPERIMENT

To save an experiment, a user can use the menu to select ***File...Save, File...Save As...*** or press the hotkey: **Ctrl-S** *or* click the icon on the toolbar.

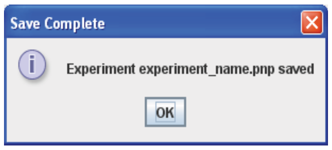
1. if the experiment entry box is empty, the user will be prompted for the experiment name.


Input

Enter PNP Experiment Filename:

experiment_name

2. Once the experiment has been saved a confirmation window will appear.

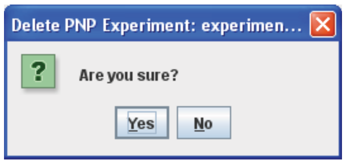

Save Complete

Experiment experiment_name.pnp saved

DELETING AN EXPERIMENT

To delete an experiment a user can use the menu to select ***File...Delete*** *or* press the hotkey: **Ctrl-D** *or* click the ✕ icon on the toolbar or press in the **Delete** Key.

1. A confirmation window will appear, press **Yes** button to delete current experiment.



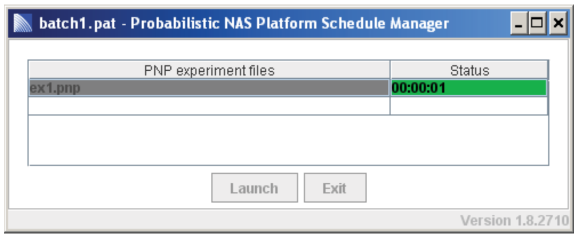
Delete PNP Experiment: experimen...

Are you sure?

RUNNNING AN EXPERIMENT

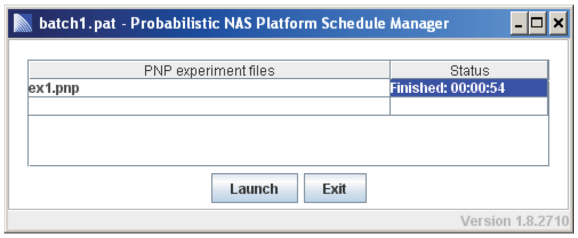
To run an experiment a user can use the menu to select ***Run…Now*** *or* press the hotkey: **Shift-F10** *or* click the icon on the toolbar.

1. The Schedule Manager Window will appear. As the run transgresses the runtime is updated in the status field adjacent to the experiment name. Once the run has completed the status will reflect this status as Finished with the total runtime.


batch1.pat - Probabilistic NAS Platform Schedule Manager

PNP experiment files	Status
ex1.pnp	00:00:01

2. After the run completes or is shuitdown, the status is upated to *finished* with the total runtime. The user can exit the run scheduler by press the **Exit** button. If the user wants to run the experiment again, press the **Launch** Button.

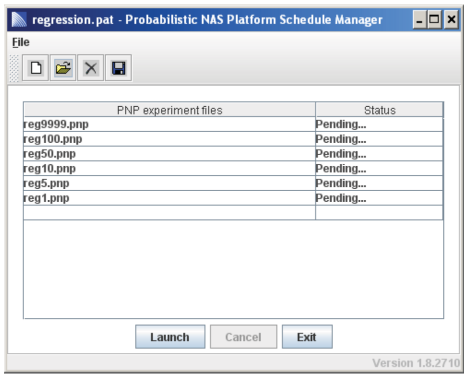

batch1.pat - Probabilistic NAS Platform Schedule Manager

PNP experiment files	Status
ex1.pnp	Finished: 00:00:54

SCHEDULING EXPERIMENTS

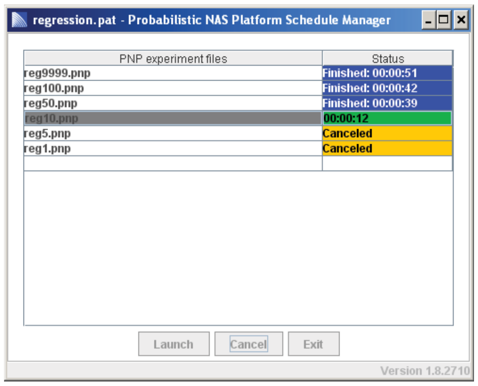
To schedule experiments a user can use the menu to select ***Run…Schedule*** *or* press the hotkey: **Shift-F9** *or* Click the icon on the toolbar.

1. Select experiment file to be run.
2. Save schedule if you intend to run it again in the future.
3. Use the **Launch** button to start the schedule experiment runs.


regression.pat - Probabilistic NAS Platform Schedule Manager
File
Launch
Cancel
Exit
Version 1.8.2710

PNP experiment files	Status
reg9999.pnp	Pending...
reg100.pnp	Pending...
reg50.pnp	Pending...
reg10.pnp	Pending...
reg5.pnp	Pending...
reg1.pnp	Pending...

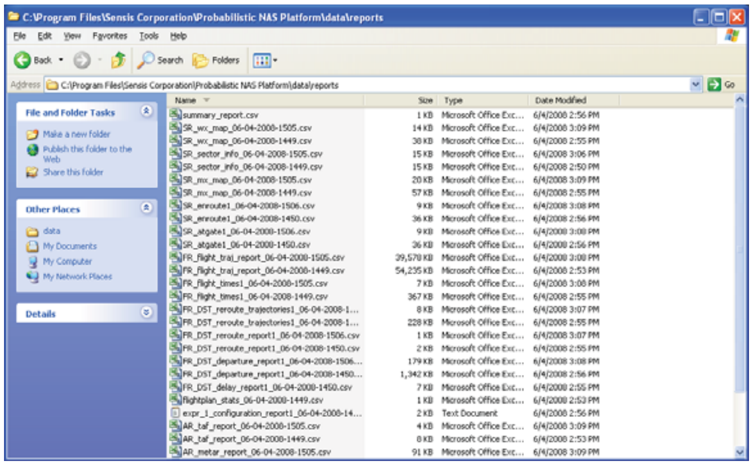
4. As the run transgresses the runtime is updated in the status field adjacent to the experiment name. Once the run has completed the status will reflect this status as *finished* with the total runtime. The **Cancel** button will cancel the next experiment in the queue and will be reflected in the experiments status as *canceled*.


regression.pat - Probabilistic NAS Platform Schedule Manager
Launch
Cancel
Exit
Version 1.8.2710

PNP experiment files	Status
reg9999.pnp	Finished: 00:00:51
reg100.pnp	Finished: 00:00:42
reg50.pnp	Finished: 00:00:39
reg10.pnp	00:00:12
reg5.pnp	Canceled
reg1.pnp	Canceled

ANALYZING EXPERIMENTS

At the completion of an experiment run, reports can be found in the output directory defined in the PNP configuration files.


C:\Program Files\Sensis Corporation\Probabilistic NAS Platform\data\reports
File
Edit
View
Favorites
Tools
Help
Back
Search
Folders
Address
C:\Program Files\Sensis Corporation\Probabilistic NAS Platform\data\reports
Go
File and Folder Tasks
Make a new folder
Publish this folder to the Web
Share this folder
Other Places
data
My Documents
My Computer
My Network Places
Details

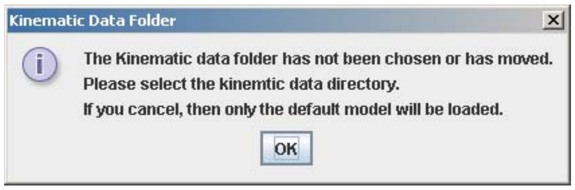
Name	Size	Type	Date Modified
summary_report.csv	1 KB	Microsoft Office Exc...	6/4/2008 2:56 PM
SR_wx_map_06-04-2008-1505.csv	14 KB	Microsoft Office Exc...	6/4/2008 3:09 PM
SR_wx_map_06-04-2008-1449.csv	38 KB	Microsoft Office Exc...	6/4/2008 2:55 PM
SR_sector_info_06-04-2008-1505.csv	15 KB	Microsoft Office Exc...	6/4/2008 3:06 PM
SR_sector_info_06-04-2008-1449.csv	15 KB	Microsoft Office Exc...	6/4/2008 2:50 PM
SR_mx_map_06-04-2008-1505.csv	20 KB	Microsoft Office Exc...	6/4/2008 3:09 PM
SR_mx_map_06-04-2008-1449.csv	57 KB	Microsoft Office Exc...	6/4/2008 2:55 PM
SR_enroute1_06-04-2008-1506.csv	9 KB	Microsoft Office Exc...	6/4/2008 3:08 PM
SR_enroute1_06-04-2008-1450.csv	36 KB	Microsoft Office Exc...	6/4/2008 2:56 PM
SR_atgate1_06-04-2008-1506.csv	9 KB	Microsoft Office Exc...	6/4/2008 3:08 PM
SR_atgate1_06-04-2008-1450.csv	36 KB	Microsoft Office Exc...	6/4/2008 2:56 PM
FR_flight_traj_report_06-04-2008-1505.csv	39,570 KB	Microsoft Office Exc...	6/4/2008 3:08 PM
FR_flight_traj_report_06-04-2008-1449.csv	54,235 KB	Microsoft Office Exc...	6/4/2008 2:53 PM
FR_flight_times1_06-04-2008-1505.csv	7 KB	Microsoft Office Exc...	6/4/2008 3:08 PM
FR_flight_times1_06-04-2008-1449.csv	367 KB	Microsoft Office Exc...	6/4/2008 2:55 PM
FR_DST_reroute_trajectories1_06-04-2008-1...	8 KB	Microsoft Office Exc...	6/4/2008 3:07 PM
FR_DST_reroute_trajectories1_06-04-2008-1...	228 KB	Microsoft Office Exc...	6/4/2008 2:55 PM
FR_DST_reroute_report1_06-04-2008-1506.csv	1 KB	Microsoft Office Exc...	6/4/2008 3:07 PM
FR_DST_reroute_report1_06-04-2008-1450.csv	2 KB	Microsoft Office Exc...	6/4/2008 2:55 PM
FR_DST_departure_report1_06-04-2008-1506...	179 KB	Microsoft Office Exc...	6/4/2008 3:08 PM
FR_DST_departure_report1_06-04-2008-1450...	1,342 KB	Microsoft Office Exc...	6/4/2008 2:56 PM
FR_DST_delay_report1_06-04-2008-1450.csv	7 KB	Microsoft Office Exc...	6/4/2008 2:55 PM
flightplan_stats_06-04-2008-1449.csv	1 KB	Microsoft Office Exc...	6/4/2008 2:53 PM
expr_1_configuration_report1_06-04-2008-14...	2 KB	Text Document	6/4/2008 2:56 PM
AR_taf_report_06-04-2008-1505.csv	4 KB	Microsoft Office Exc...	6/4/2008 3:09 PM
AR_taf_report_06-04-2008-1449.csv	8 KB	Microsoft Office Exc...	6/4/2008 2:53 PM
AR_metar_report_06-04-2008-1505.csv	91 KB	Microsoft Office Exc...	6/4/2008 3:09 PM

APPENDICES

APPENDIX A: TRAJECTORY MODEL INITIALIZATION

The trajectory model uses BADA V3.6 aircraft performance models.

1. Upon startup of the Kinematic trajectory model, the following window will appear if the BADA path has not been found:

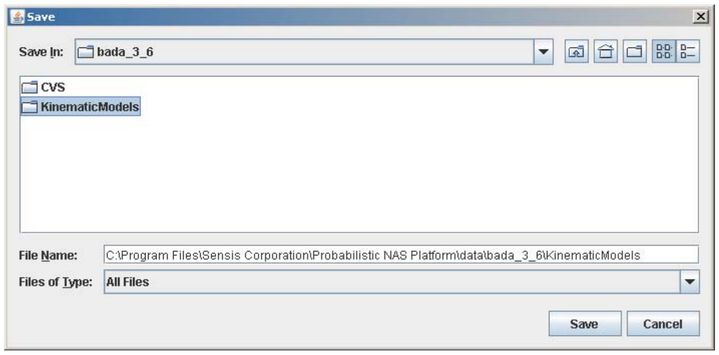

Kinematic Data Folder

The Kinematic data folder has not been chosen or has moved.
Please select the kinemtic data directory.
If you cancel, then only the default model will be loaded.

2. Press OK button and a Save dialog window will appear.

3. Select the following directory:

C:\Program Files\Sensis Corporation\Probabilistic NAS Platform\data\bada_3_6\KinematicModels



File Name: C:\Program Files\Sensis Corporation\Probabilistic NAS Platform\data\bada_3_6\KinematicModels

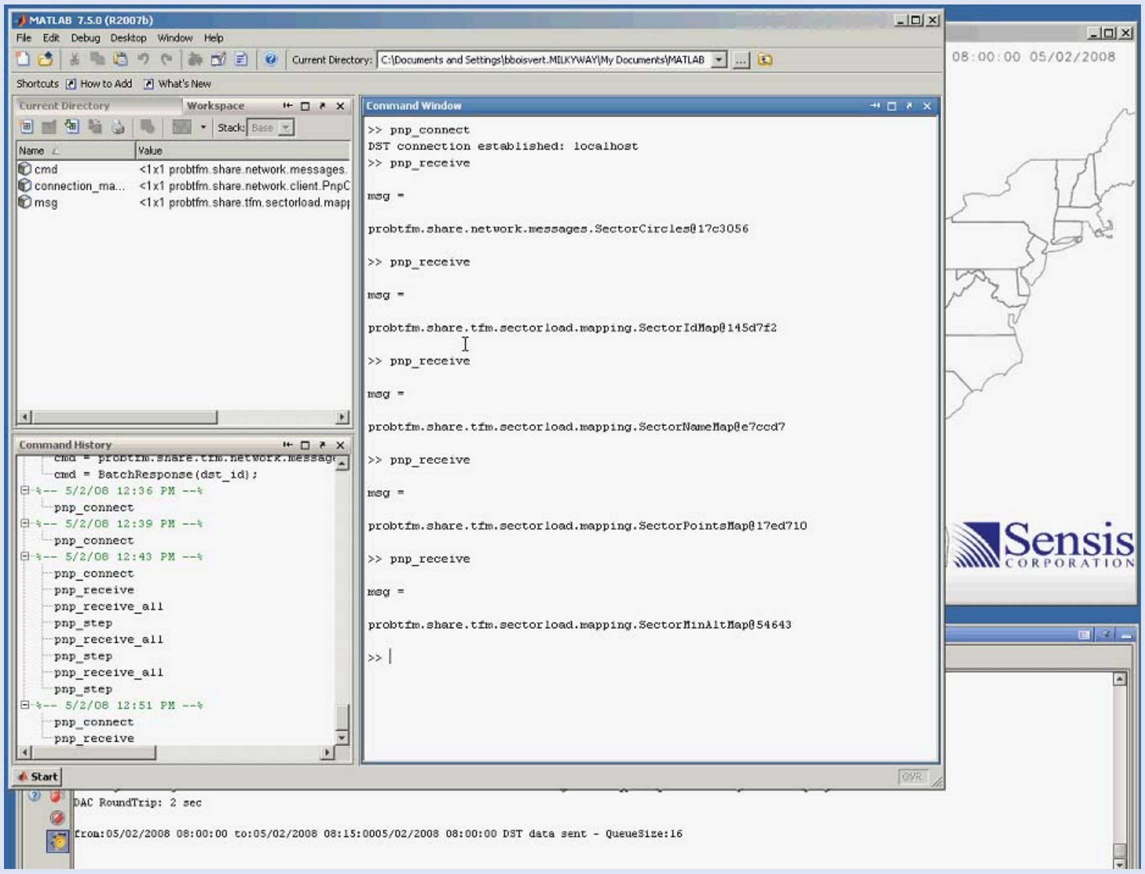
Files of Type: All Files

4. Press Save to set BADA directory path.

APPENDIX B: MATLAB SUPPORT

The PNP server can directly interface with MATLAB. Three simple scripts were written to connect, disconnect, and step PNP via the MATLAB command interface. These scripts are available in the following directory:

C:\Program Files\Sensis Corporation\Probabilistic NAS Platform\MATLAB


MATLAB 7.5.0 (R2007b)
File Edit Debug Desktop Window Help
Current Directory: C:\Documents and Settings\bboisvert.MILKYWAY\My Documents\MATLAB
Shortcuts How to Add What's New
Current Directory Workspace
Stack: Base
Command History
Command Window
Start
OVR
08:00:00 05/02/2008
Sensis CORPORATION

Name	Value
cmd	<1x1 probtfm.share.network.messages.
connection_ma...	<1x1 probtfm.share.network.client.PnpC
msg	<1x1 probtfm.share.tfm.sectorload.mapp

```
  cmd = probtfm.share.tfm.network.message
  cmd = BatchResponse(dst_id);
%-- 5/2/08 12:36 PM --%
  pnp_connect
%-- 5/2/08 12:39 PM --%
  pnp_connect
%-- 5/2/08 12:43 PM --%
  pnp_connect
  pnp_receive
  pnp_receive_all
  pnp_step
  pnp_receive_all
  pnp_step
  pnp_receive_all
  pnp_step
%-- 5/2/08 12:51 PM --%
  pnp_connect
  pnp_receive
```

```
>> pnp_connect
DST connection established: localhost
>> pnp_receive

msg =

probtfm.share.network.messages.SectorCircles@17c3056

>> pnp_receive

msg =

probtfm.share.tfm.sectorload.mapping.SectorIdMap@145d7f2

>> pnp_receive

msg =

probtfm.share.tfm.sectorload.mapping.SectorNameMap@e7ccd7

>> pnp_receive

msg =

probtfm.share.tfm.sectorload.mapping.SectorPointsMap@17ed710

>> pnp_receive

msg =

probtfm.share.tfm.sectorload.mapping.SectorMinAltMap@54643

>> |
```

```
DAC RoundTrip: 2 sec

from:05/02/2008 08:00:00 to:05/02/2008 08:15:0005/02/2008 08:00:00 DST data sent - QueueSize:16
```

APPENDIX C: CLEANUP

There is a utility to cleanup the system reports, files, and logs. The command file ***cleanup.cmd*** is located in the ***bin*** directory. Simple *double-click* this file and the system files will be deleted.

**Note: It is convenient to create a shortcut so ty can be used from your desktop.*

Probabilistic NAS Platform

v1.2

Plan View Display User Guide

8/16/2011
Sensis Corporation

TABLE OF CONTENTS

INTRODUCTION

The Probabilistic NAS Platform (PNP) is both a research tool and a decision support platform to analyze air traffic and traffic flow management (TFM) issues. Real-time PNP provides real time meteorological data and traffic advisories in which an operator or Air Traffic Manager can make more informed and accurate decisions. Playback PNP evaluates current or future TFM concepts using projected or actual traffic demands sets with recorded historical meteorological data.

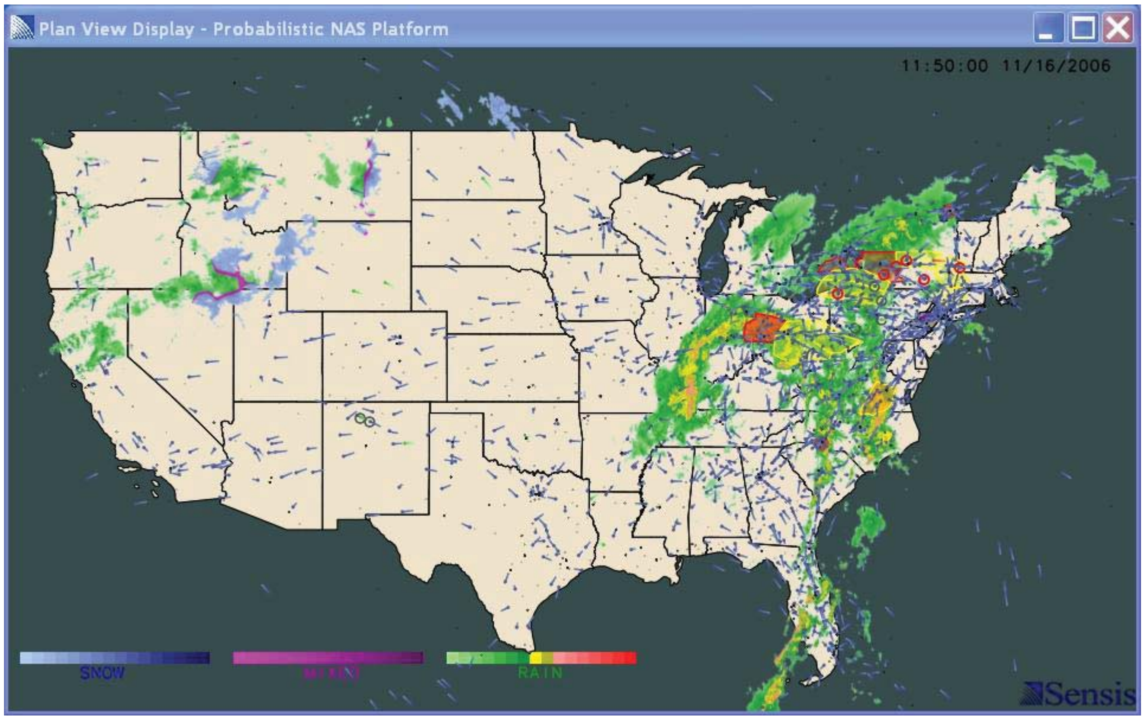
SCOPE

This document guides the PNP user in the usage of the Plan View Display (PVD).

Topics covered include:

- ❖ Plan View Display

PLAN VIEW DISPLAY (PVD)



The Plan View Display allows the user to visualize the NAS. Toggles have been built in to control the data as the PVD may become cluttered during a session when aircraft counts typically exceed 40,000. The PVD has been developed for debugging and marketing purposes and is still under construction.

OPERATIONS

MOUSE FUNCTIONS

SCROLL BUTTON

The Scroll Button on the mouse zooms in and out of the PVD.

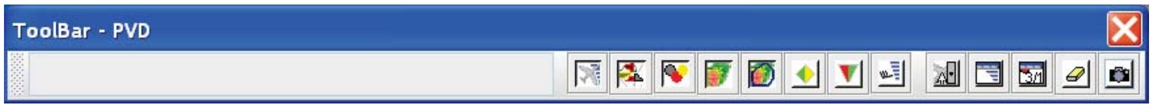
RIGHT BUTTON

The Right Button on the mouse re-centers the map on the PVD.

LEFT BUTTON

The Left Button selects sectors, airports, or aircraft based upon the selection criteria enabled.

TOOLBAR



The PVD has a toolbar which can be dragged off the PVD. The toolbar provides the user with buttons, toggles, and dialogs to control what is displayed on the PVD. The text box displays status messages.

SNAPSHOT BUTTON

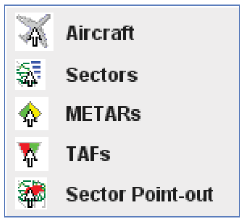
The Snapshot Button takes a snapshot of the PVD using the current time to create a jpeg in the output directory.

CLEAR DISPLAY BUTTON

The Clear Display Button removes selected aircraft and sectors from the PVD.

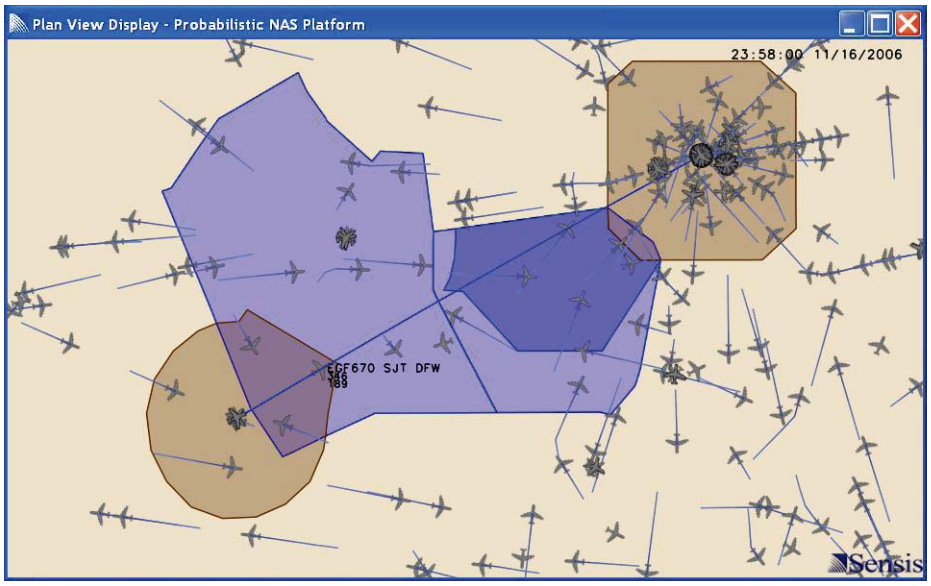
CURSOR SELECT

The Cursor Pull-down Menu toggles the cursor selection between aircraft , sectors, sector point-out , METARs , and TAFs.



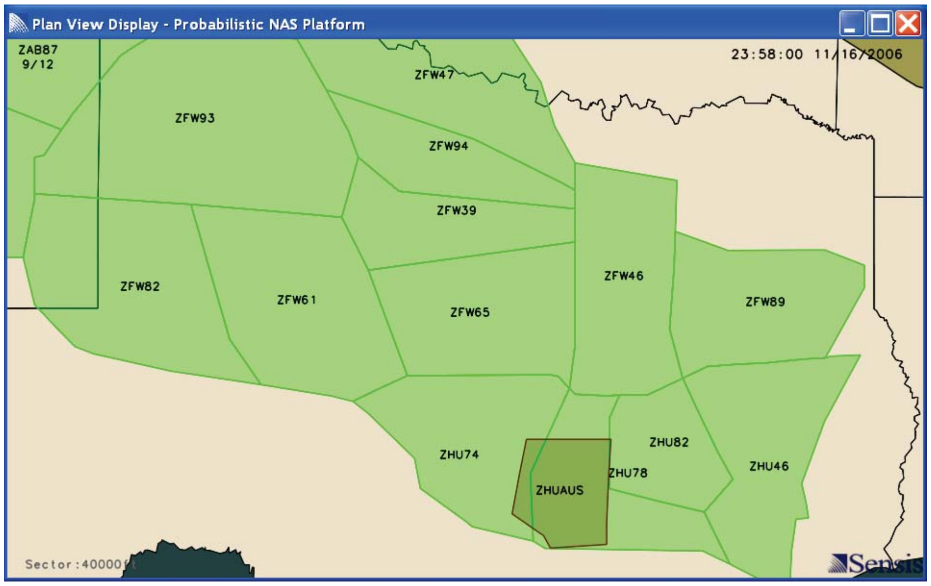
CURSOR SELECT: AIRCRAFT

Clicking on an aircraft icon will cause aircraft information to display based on the aircraft display filter, when the aircraft cursor in enabled.



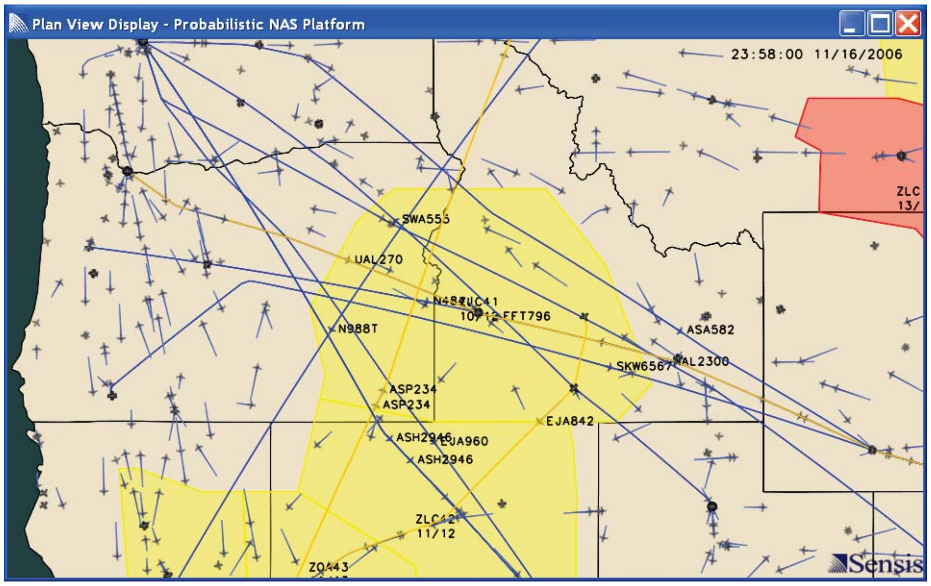
CURSOR SELECT: SECTORS

Clicking on the map will display sectors at the cursor location based on sector display filter, when the sector cursor is enabled.



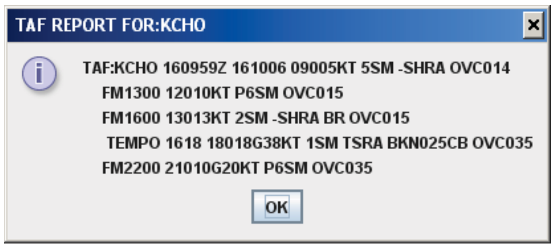
CURSOR SELECT: SECTOR-POINT OUT

Clicking on an over-loaded sector will display all the flights schedule to be in that sector, when the sector point-out cursor is selected.



CURSOR SELECT: TAF

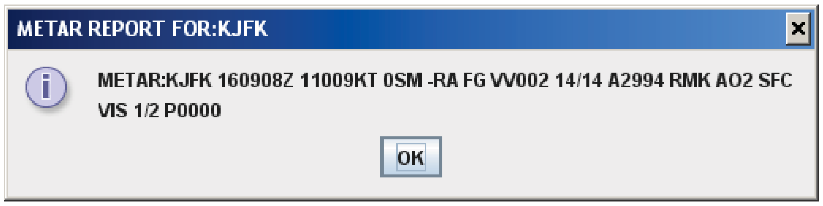
Clicking on a TAF report symbol will cause the report to display in a pop-up window, when the TAF cursor is enabled.


TAF REPORT FOR:KCHO

```
TAF:KCHO 160959Z 161006 09005KT 5SM -SHRA OVC014
   FM1300 12010KT P6SM OVC015
   FM1600 13013KT 2SM -SHRA BR OVC015
    TEMPO 1618 18018G38KT 1SM TSRA BKN025CB OVC035
   FM2200 21010G20KT P6SM OVC035
```

CURSOR SELECT: METAR

Clicking on a METAR report symbol will cause the report to display in a pop-up window, when the METAR cursor is enabled.

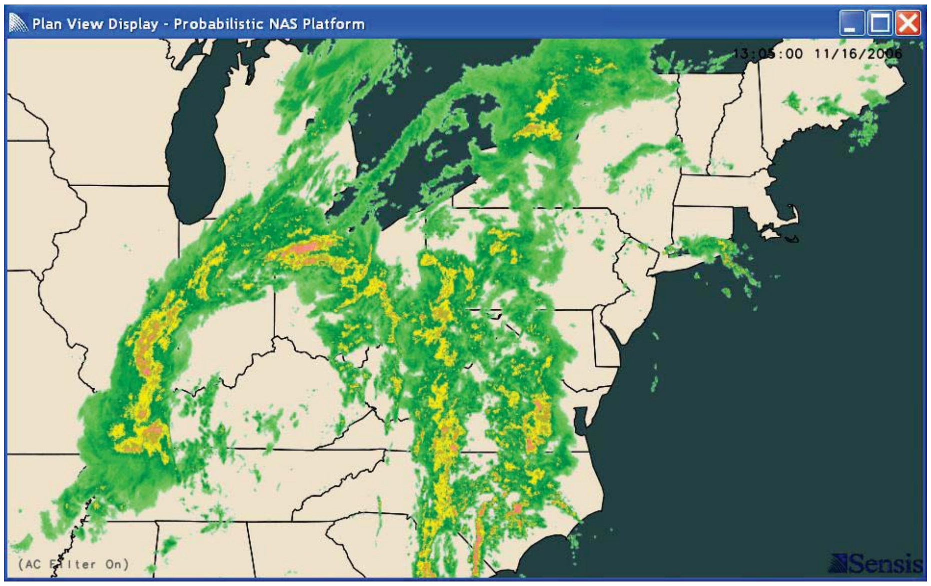

METAR REPORT FOR:KJFK

```
METAR:KJFK 160908Z 11009KT 0SM -RA FG VV002 14/14 A2994 RMK AO2 SFC
VIS 1/2 P0000
```

DISPLAY TOGGLES

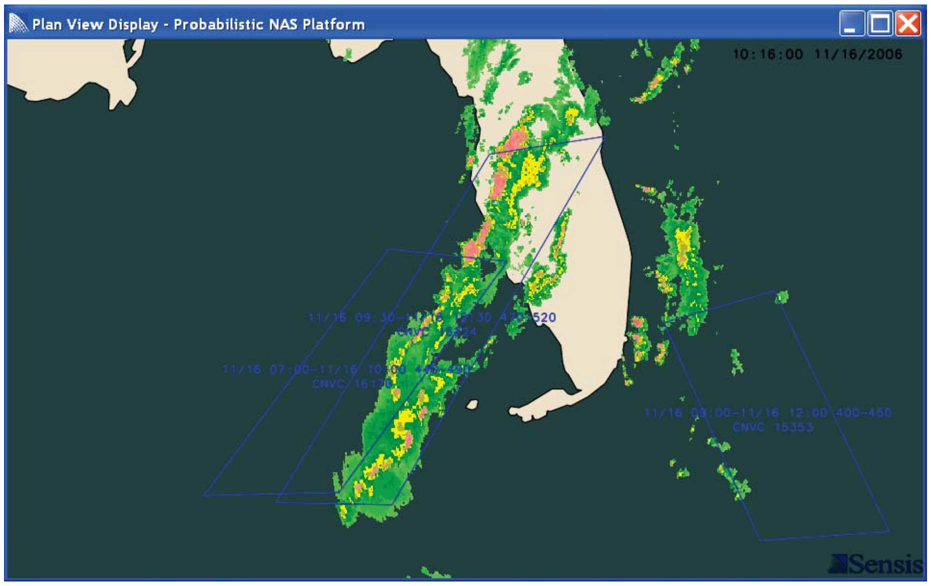
REFLECTIVITY TOGGLE

The Reflectivity Toggle turns on and off the reflectivity display data.



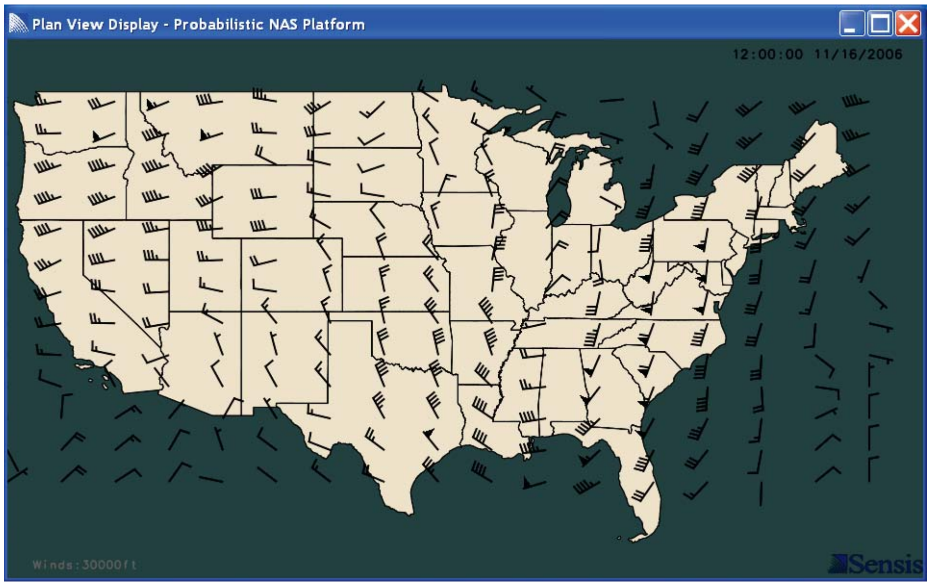
FORECAST TOGGLE

The Forecast Toggle turns on and off the forecast display data.



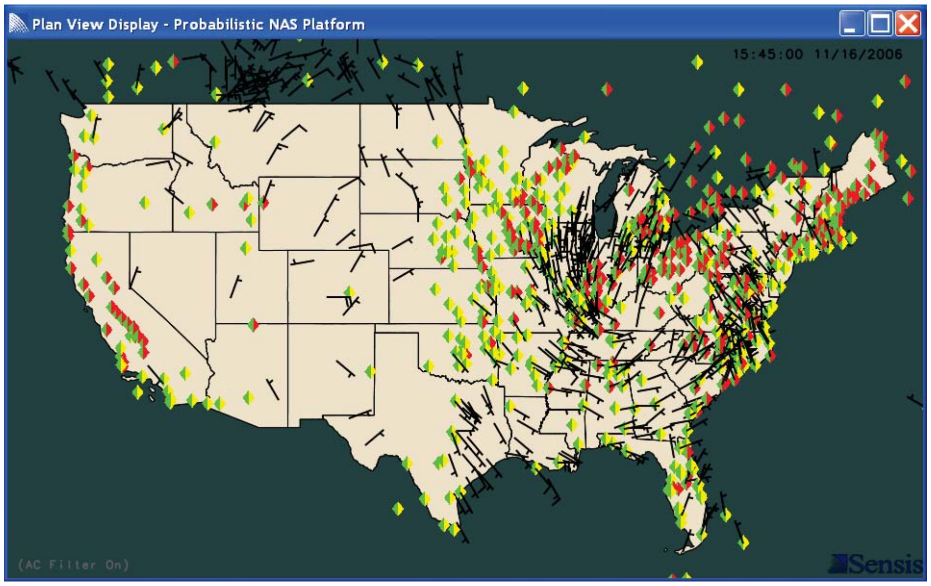
WINDS TOGGLE

The Winds Toggle turns of and off the wind display data.



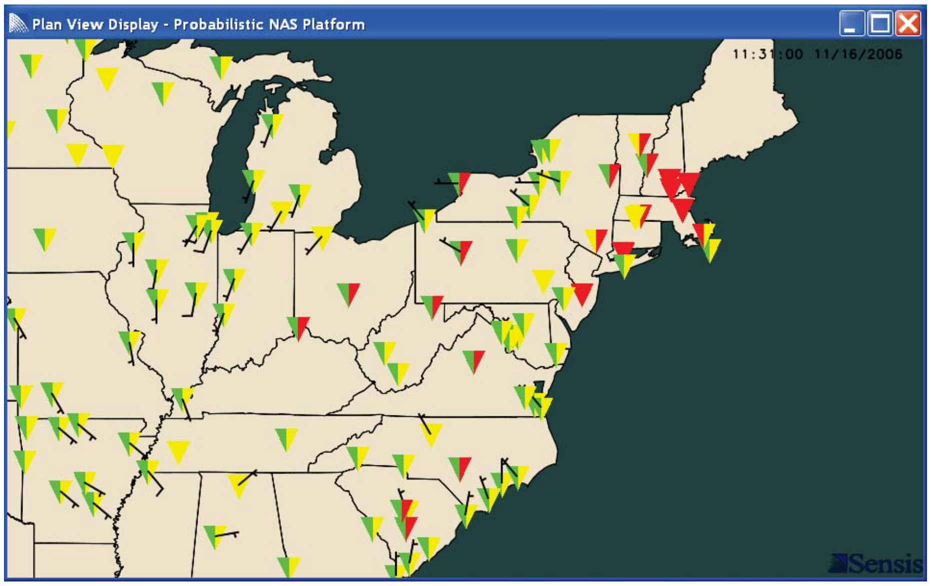
METAR TOGGLE

The METAR Toggle turns on and off the METAR forecast data.



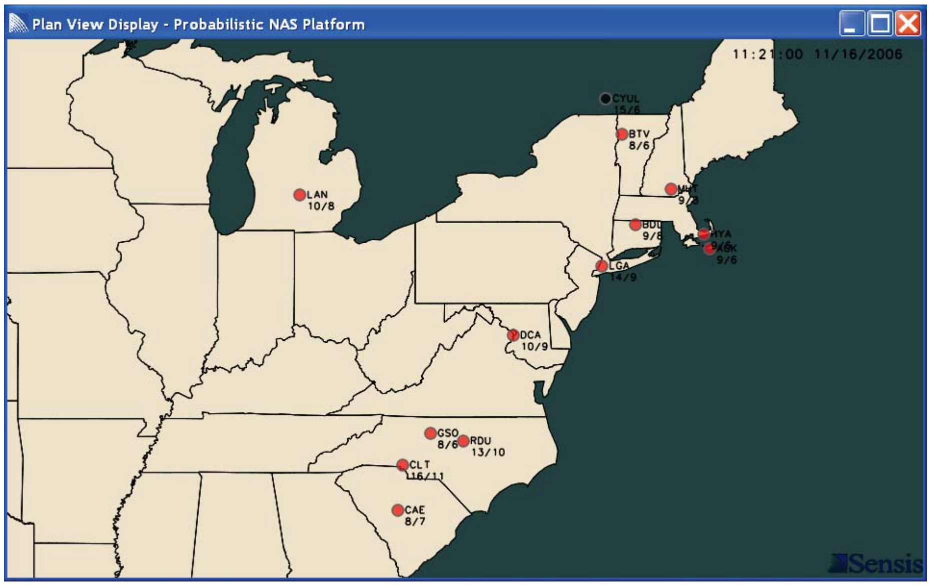
TAF TOGGLE

The TAF Toggle turns on and off the TAF forecast data.



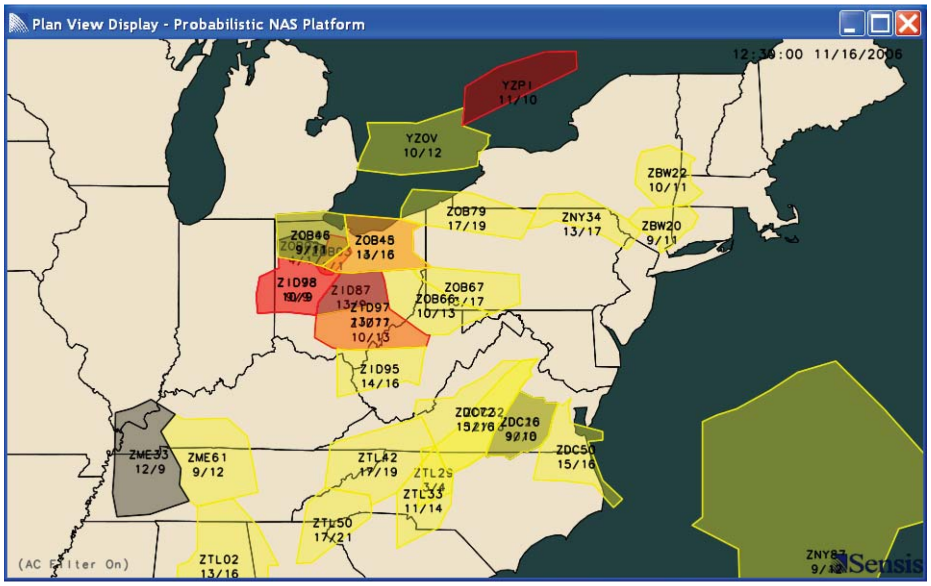
AIRPORT CONGESTION TOGGLE

The Airport Congestion Toggle toggles the display of airport congestion. Congested airports appear as yellow, red, or black circles, as shown in the figure below.



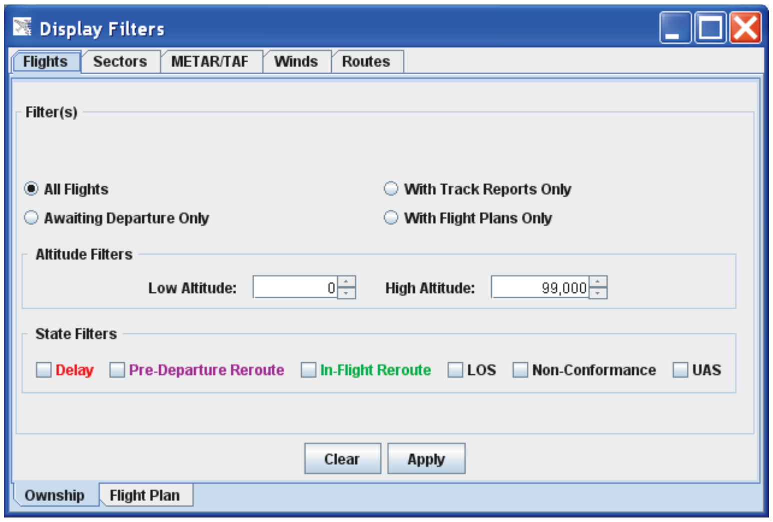
SECTOR CONGESTION TOGGLE

The Sector Congestion Toggle toggles the display of sector congestion. Congested sectors appear in yellow, red, or black, as shown in the figure below.



DISPLAY FILTERS

The Display Filters Button displays the Display Filters Window. The Display Filters Window contains tabs to change the display filters associated with flights, sectors, METAR/TAF reports, winds and routes.


Display Filters

Flights | Sectors | METAR/TAF | Winds | Routes

Filter(s)

[● All Flights] ○ With Track Reports Only
○ Awaiting Departure Only ○ With Flight Plans Only

Altitude Filters

Low Altitude: 0 High Altitude: 99,000

State Filters

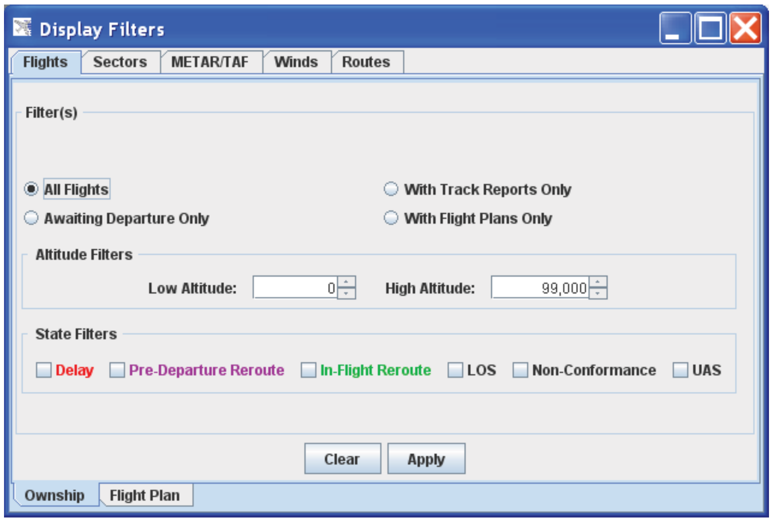
☐ Delay ☐ Pre-Departure Reroute ☐ In-Flight Reroute ☐ LOS ☐ Non-Conformance ☐ UAS

Clear | Apply

Ownship | Flight Plan

FLIGHTS

The flights filter window allows the user to display windows based on the altitude. The ***Apply*** button will display the flights based on the specified filters. The ***Clear*** button will clear the filters.



Display Filters

Flights | Sectors | METAR/TAF | Winds | Routes

Filter(s)

[● All Flights] ○ With Track Reports Only
○ Awaiting Departure Only ○ With Flight Plans Only

Altitude Filters

Low Altitude: 0 High Altitude: 99,000

State Filters

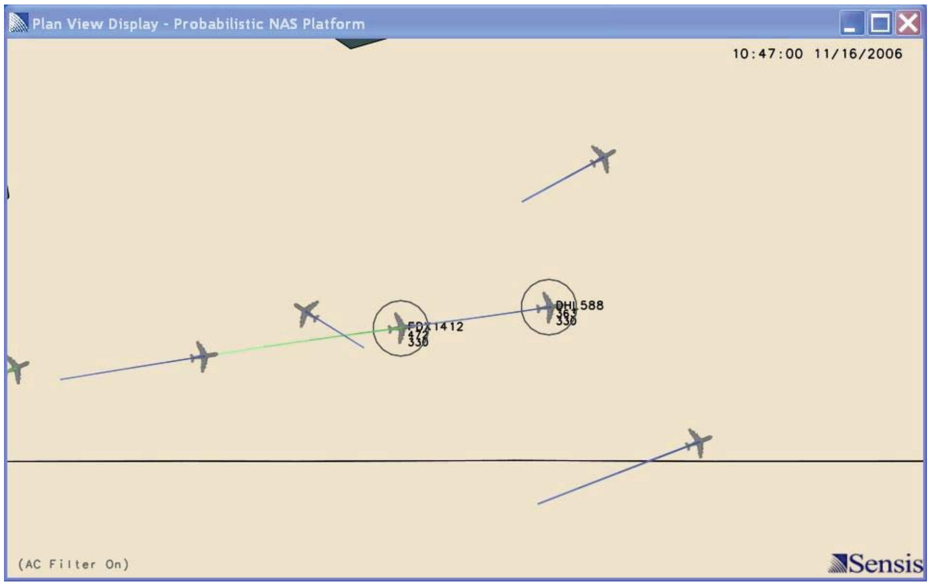
☐ Delay ☐ Pre-Departure Reroute ☐ In-Flight Reroute ☐ LOS ☐ Non-Conformance ☐ UAS

Clear Apply

Ownship | Flight Plan

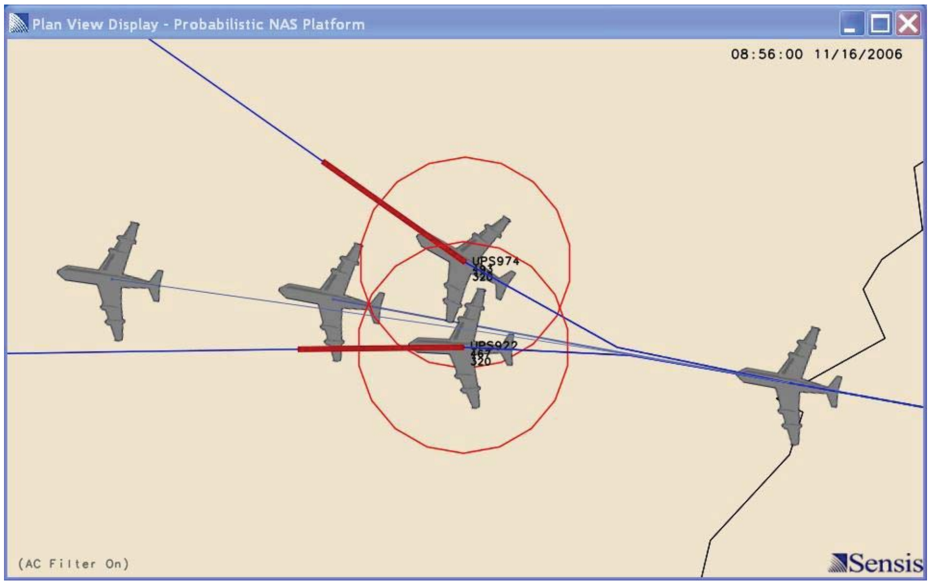
FLIGHTS: CONFLICTS

Flights that have a future loss of separation event are highlighted by black circles.



FLIGHTS: LOSS OF SEPARATION

Flights are in a loss of separation event are highlighted by red circles. The red line highlights where on the trajectory the loss of separation occurs.



FLIGHT FILTERS

OWNSHIP FILTER: ALL FLIGHTS

The ***All Flights*** checkbox allows a user to specify whether the all flights will be displayed.

OWNSHIP FILTER: AWAITING DEPARTURE

The ***Awaiting Departure*** checkbox allows a user to specify only flights awaiting departure are displayed.

OWNSHIP FILTER: WITH TRACK REPORTS ONLY

The ***With Track Reports Only*** checkbox allows a user to specify only flights with track ports are displayed.

OWNSHIP FILTER: WITH FLIGHT PLAN ONLY

The ***With Flight Plan*** checkbox allows a user to specify only flights with flight plans are displayed.

OWNSHIP FILTER: LOW ALTITUDE

The ***low altitude*** spinner sets the low bound of the altitude filter of the display.

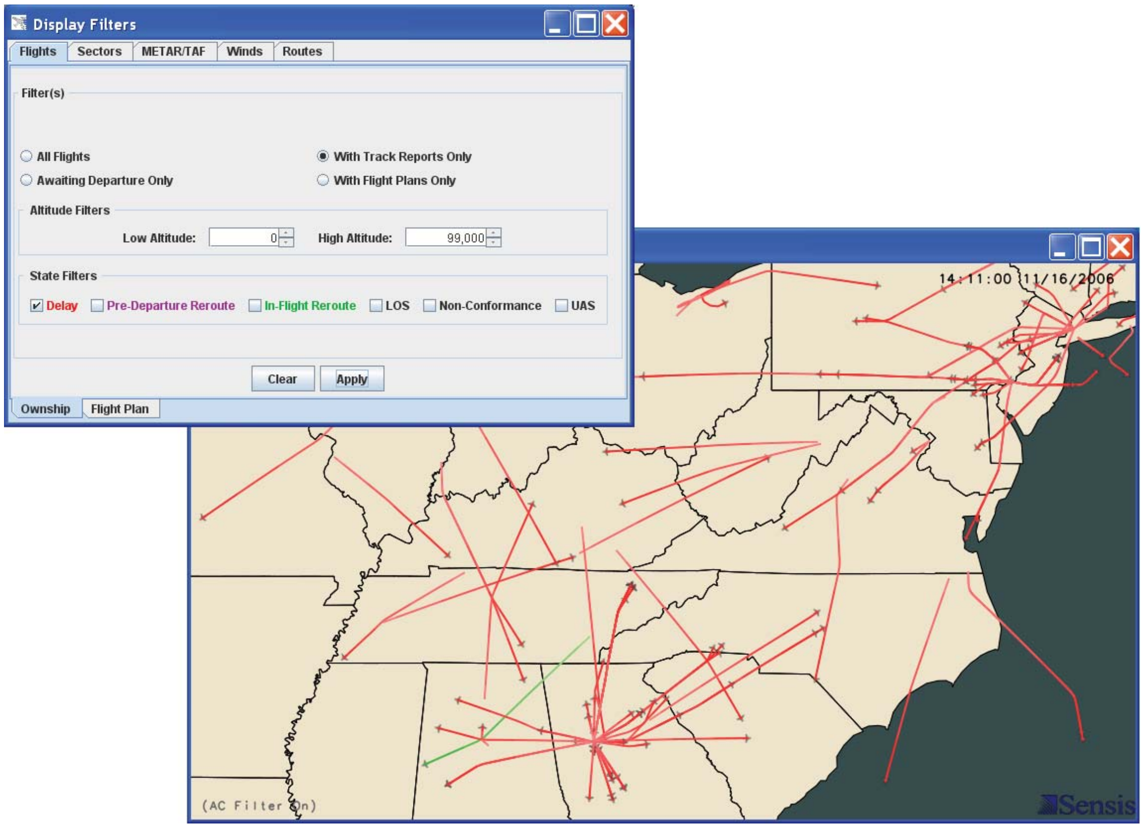
OWNSHIP FILTER: HIGH ALTITUDE

The ***high altitude*** spinner sets the upper bound of the altitude filter of the display.

STATE FILTERS

STATE FILTERS: DELAY

The ***DELAY*** checkbox selects flights with pre-departure delays. Pre-departure delayed flights have red historical waypoints.



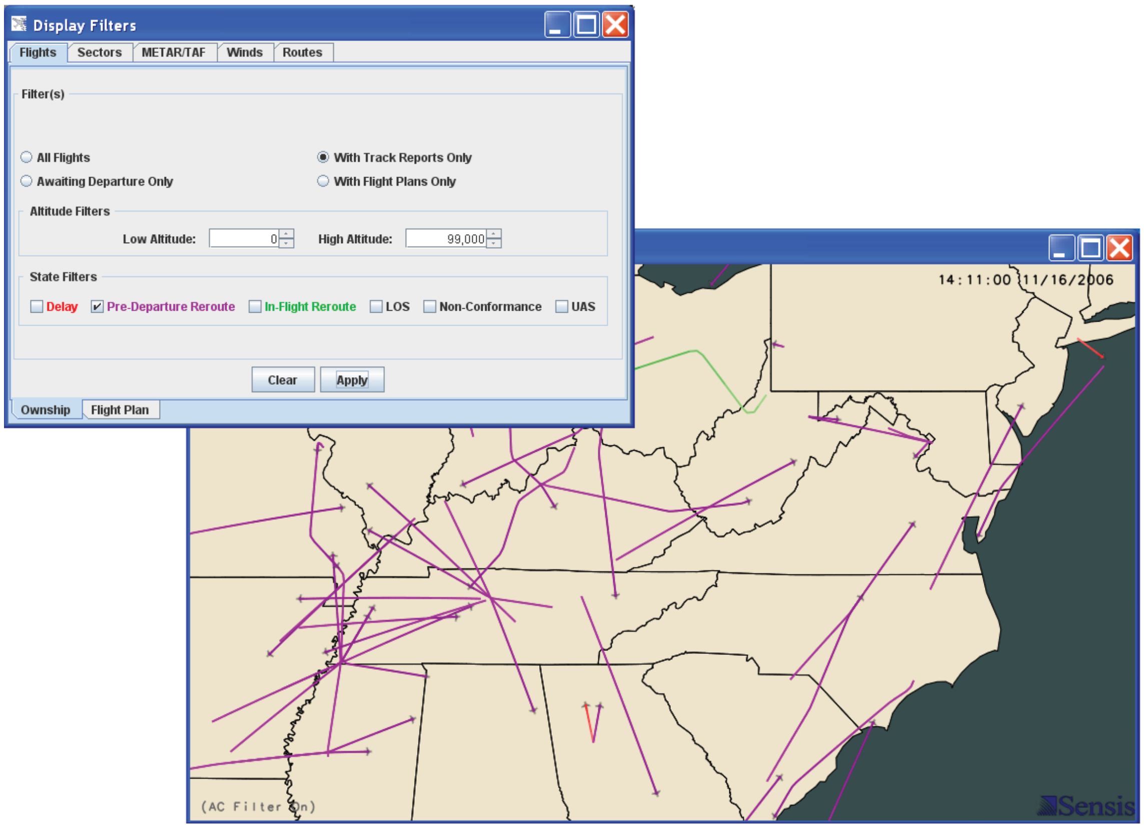
○ All Flights
○ Awaiting Departure Only
● With Track Reports Only
○ With Flight Plans Only

Altitude Filters
Low Altitude: 0
High Altitude: 99,000

State Filters
☑ Delay ☐ Pre-Departure Reroute ☐ In-Flight Reroute ☐ LOS ☐ Non-Conformance ☐ UAS

STATE FILTERS: PRE-DEPARTURE REROUTE

The ***PRE-DEPERTURE REROUTE*** checkbox selects flights with pre-departure reroutes. Pre-departure reroute flights have purple historical waypoints.


Display Filters

Flights | Sectors | METAR/TAF | Winds | Routes

Filter(s)

( ) All Flights [● With Track Reports Only]
( ) Awaiting Departure Only ( ) With Flight Plans Only

Altitude Filters

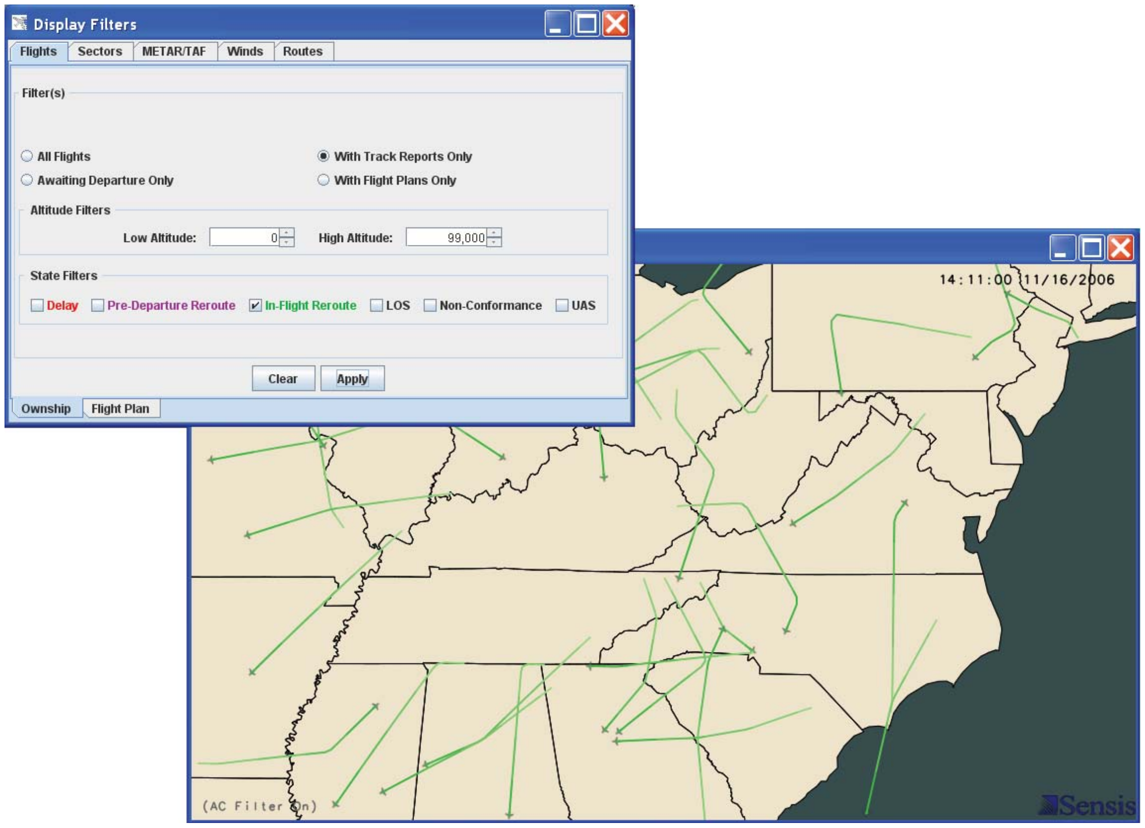
Low Altitude: 0 High Altitude: 99,000

State Filters

☐ Delay [✓ Pre-Departure Reroute] ☐ In-Flight Reroute ☐ LOS ☐ Non-Conformance ☐ UAS

STATE FILTERS: IN-FLIGHT REROUTE

The ***IN-FLIGHT REROUTE*** checkbox selects flights with in-flight reroutes. In-flight reroute flights have green historical waypoints.



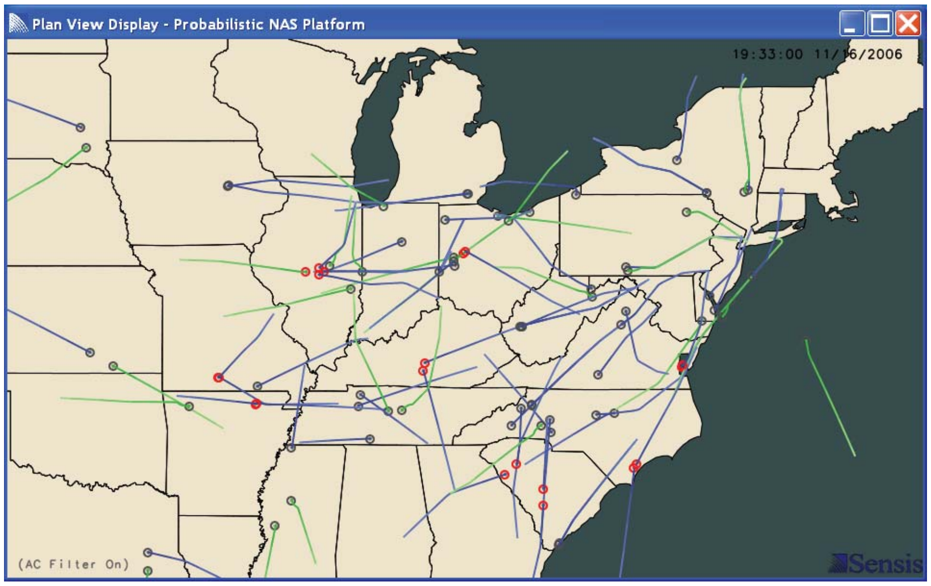
Filter(s): ○ All Flights ● With Track Reports Only ○ Awaiting Departure Only ○ With Flight Plans Only

Altitude Filters: Low Altitude: 0 High Altitude: 99,000

State Filters: ☐ Delay ☐ Pre-Departure Reroute ☑ In-Flight Reroute ☐ LOS ☐ Non-Conformance ☐ UAS

STATE FILTERS: LOS

The ***LOS*** checkbox selects flights with loss of separation events.

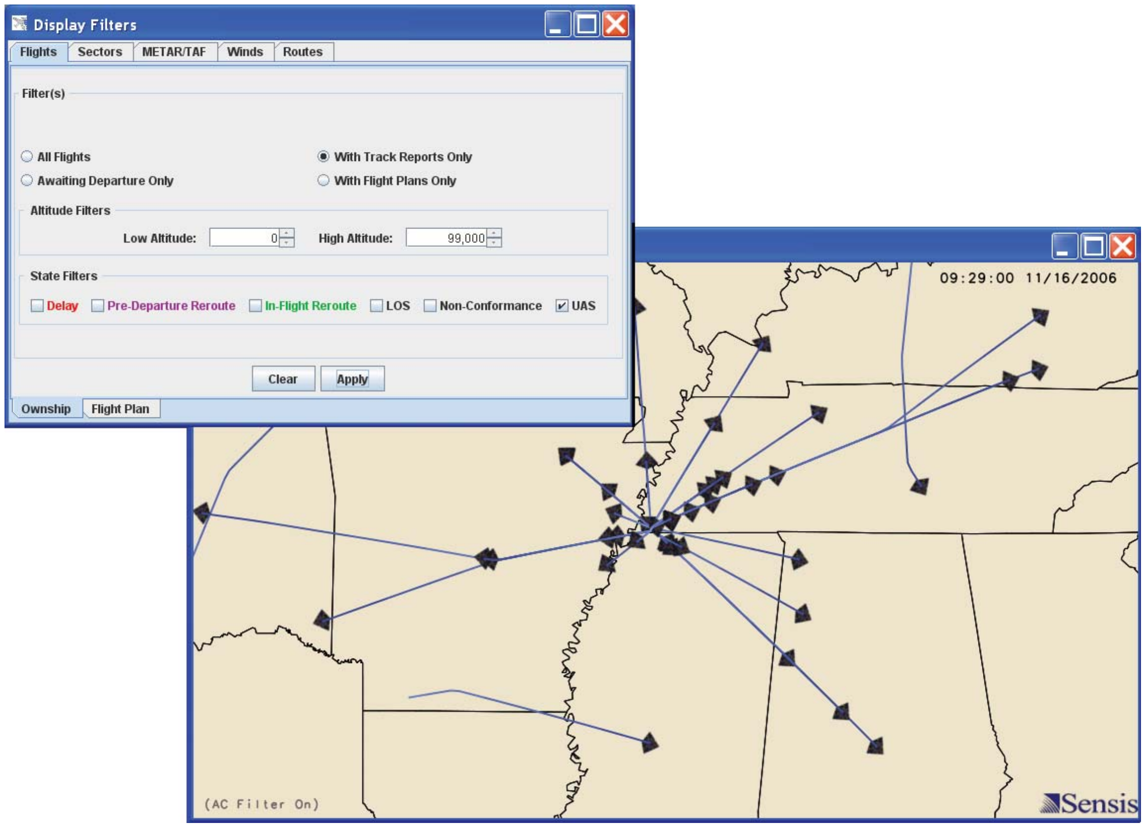


STATE FILTERS: NON-CONFORMANCE

The ***Non-Conformance*** selects flights with non-conformance events. ****Work In Progress***

STATE FILTERS: UAS

The ***UAS*** checkbox selects unmanned aircraft systems.



( ) All Flights
( ) Awaiting Departure Only
(●) With Track Reports Only
( ) With Flight Plans Only

Altitude Filters

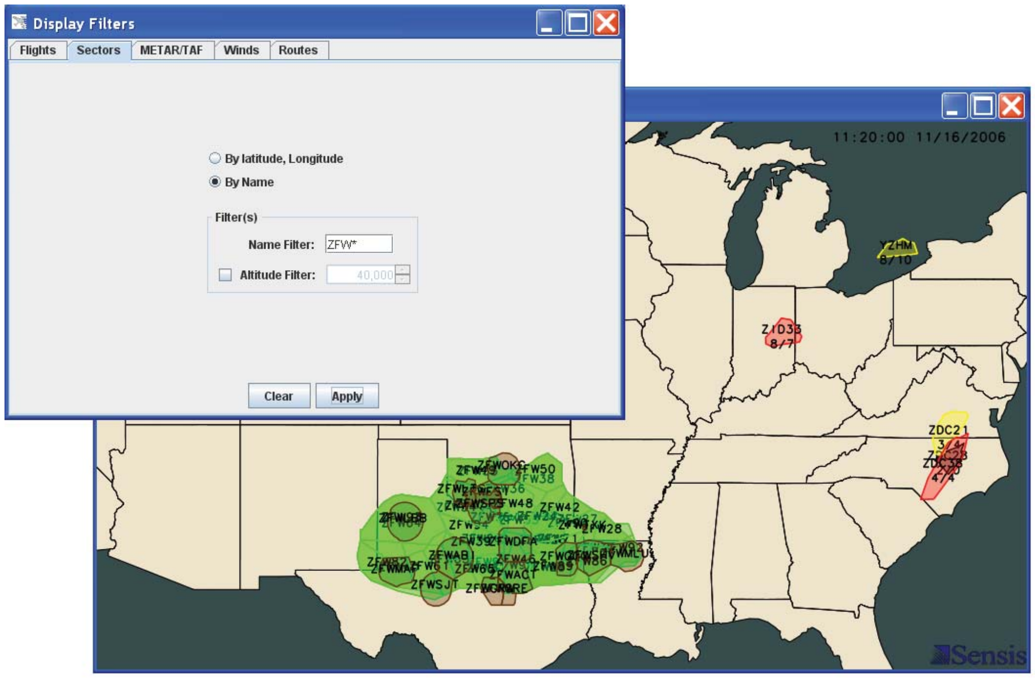
Low Altitude: 0
High Altitude: 99,000

State Filters

☐ Delay ☐ Pre-Departure Reroute ☐ In-Flight Reroute ☐ LOS ☐ Non-Conformance ☑ UAS

SECTOR

The sector filter window allows the user to display sectors based on latitude/longitude, altitude, or name. The ***Apply*** button will display the sectors based on the specified filters. The ***Clear*** button will clear the sectors from the display.



By latitude, Longitude
[✓ By Name]

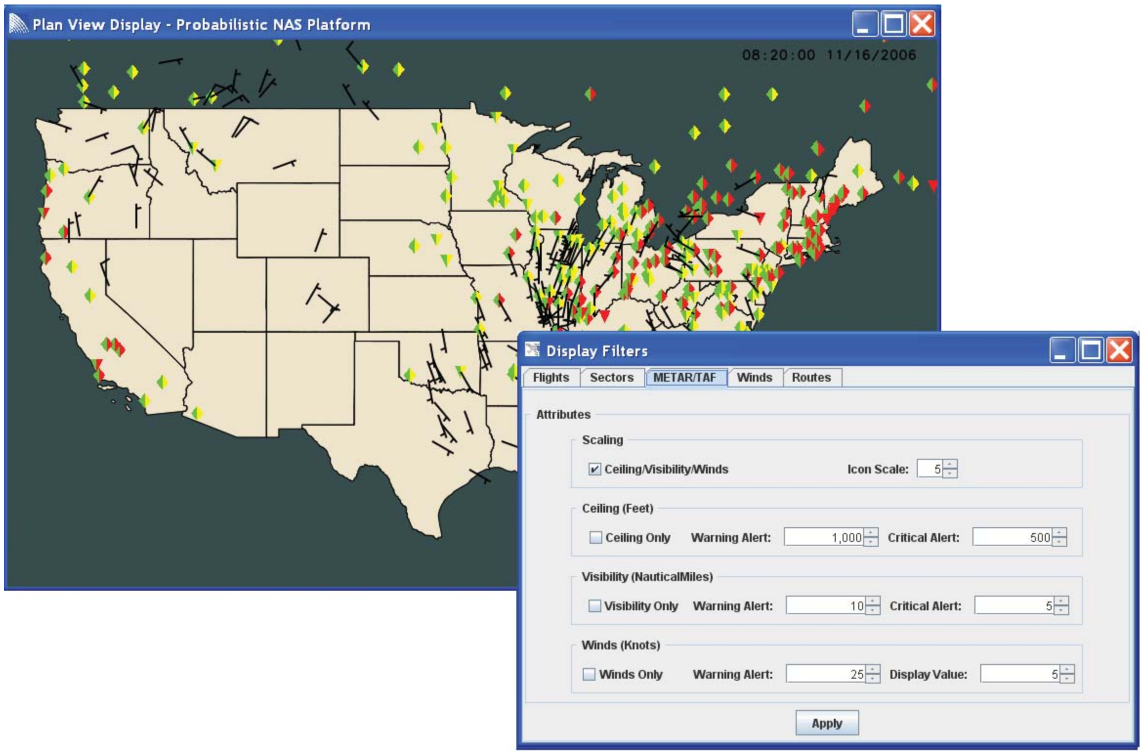
Filter(s)

Name Filter: ZFW*

Altitude Filter: 40,000

METAR/TAF

The METAR/TAF filter window allows the user to display METAR/TAF icons based on filter criteria. The ***Apply*** button will display the METAR/TAF icons based on the specified filters.



Attributes

Scaling
☑ Ceiling/Visibility/Winds Icon Scale: 5

Ceiling (Feet)
☐ Ceiling Only Warning Alert: 1,000 Critical Alert: 500

Visibility (NauticalMiles)
☐ Visibility Only Warning Alert: 10 Critical Alert: 5

Winds (Knots)
☐ Winds Only Warning Alert: 25 Display Value: 5

METAR/TAF FILTER: CEILING/VISIBILITY/WINDS

The ***Ceiling/Visibility/Winds*** checkbox allows a user to specify whether all METAR/TAF report icons will display.

METAR/TAF FILTER: ICON SCALE

The ***Icon Scale*** spinner allows a user to specify the scale of the METAR/TAF icons on the display.

METAR/TAF FILTER: CEILING ONLY

The ***Ceiling Only*** checkbox allows a user to specify whether the ceiling criterion only is used to display METAR/TAF report icons.

METAR/TAF CEILING LIMITS FILTER: CEILING WARNING ALERT

The ***Warning Alert*** spinner allows a user to specify the ceiling warning limits.

METAR/TAF CEILING LIMITS FILTER: CEILING CRITICAL ALERT

The ***Critical Alert*** spinner allows a user to specify the ceiling critical limits.

METAR/TAF FILTER: VISIBILITY ONLY

The ***Visibility Only*** checkbox allows a user to specify whether the visibility criterion only is used to display METAR/TAF report icons.

METAR/TAF CEILING LIMITS FILTER: VISIBILITY WARNING ALERT

The ***Warning Alert*** spinner allows a user to specify the visibility warning limits.

METAR/TAF CEILING LIMITS FILTER: VISIBILITY CRITICAL ALERT

The ***Critical Alert*** spinner allows a user to specify the visibility critical limits.

METAR/TAF FILTER: WINDS ONLY

The ***Winds Only*** checkbox allows a user to specify whether the winds criterion only is used to display METAR/TAF report icons.

METAR/TAF CEILING LIMITS FILTER: WINDS WARNING ALERT

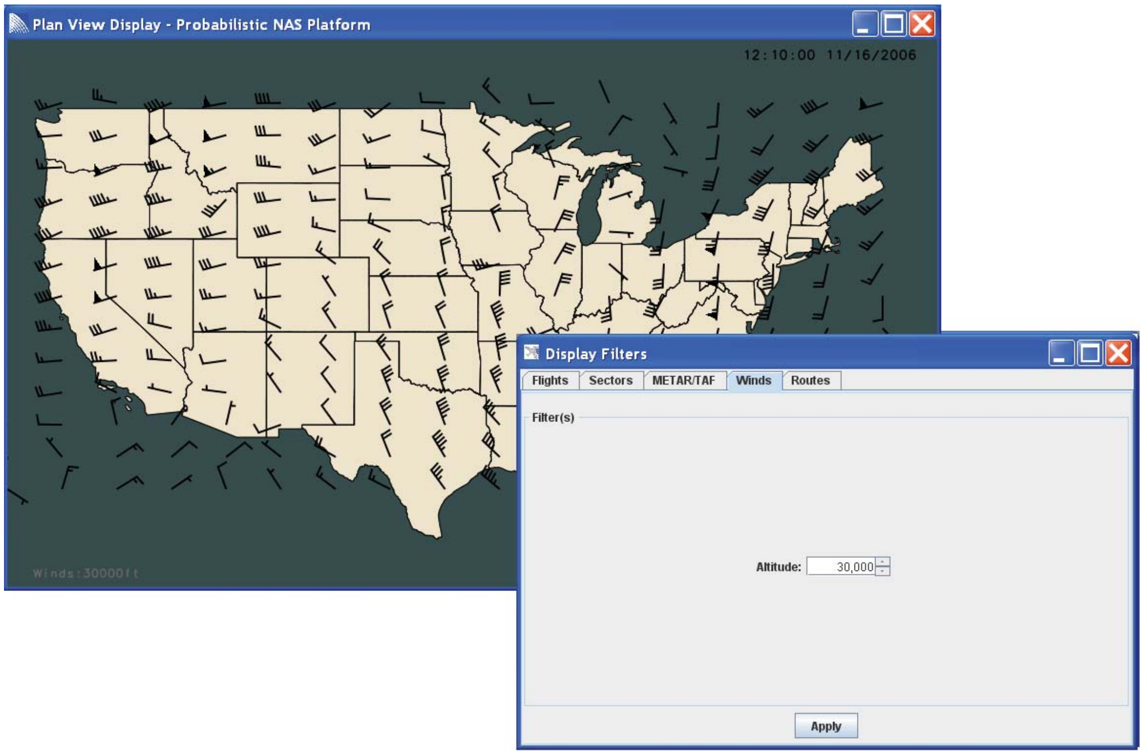
The ***Warning Alert*** spinner allows a user to specify the winds warning limits.

METAR/TAF CEILING LIMITS FILTER: WINDS CRITICAL ALERT

The ***Critical Alert*** spinner allows a user to specify the winds critical limits.

WINDS

The winds filter window allows the user to display winds based on the altitude. The ***Apply*** button will display the winds for the specified altitude.

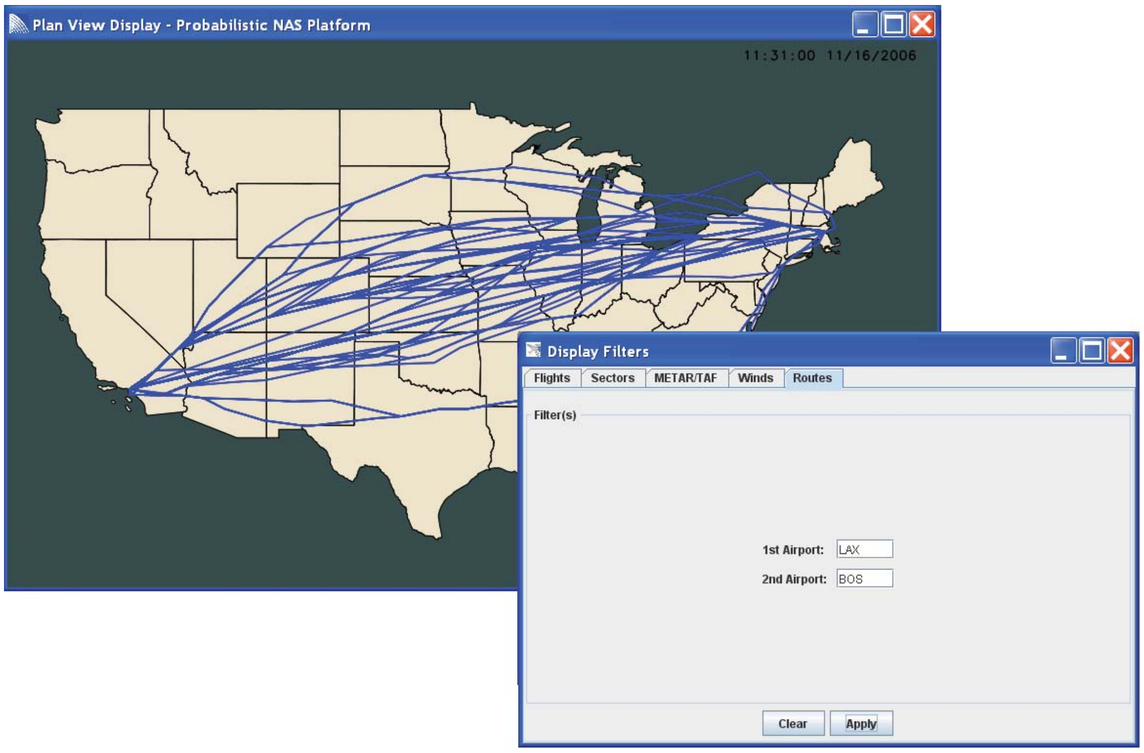


WINDS FILTER: ALTITUDE

The ***Altitude*** spinner allows a user to specify the altitude of the winds to be displayed.

ROUTES

The routes filter window allows the user to display routes based on the city pair inputs. The ***Apply*** button will display all the routes associated with the entered airports. The ***Clear*** button will clear the text from the input boxes and clear the routes from the display.



1st Airport: LAX

2nd Airport: BOS

ROUTES FILTER: 1ST AIRPORT

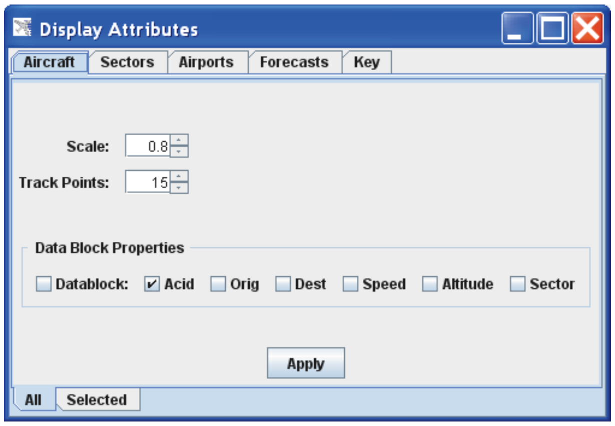
The ***1st Airport*** input box allows a user to specify the fist city pair.

ROUTES FILTER: 2ND AIRPORT

The ***1st Airport*** input box allows a user to specify the fist city pair.

DISPLAY ATRTIBUTES

The Display Attributes Button displays the Display Attributes Window. The Display Attributes Window contains tabs to change the display attributes associated with aircraft (selected and all), sectors, airports, forecasts, and the reflectivity key.



Scale: 0.8

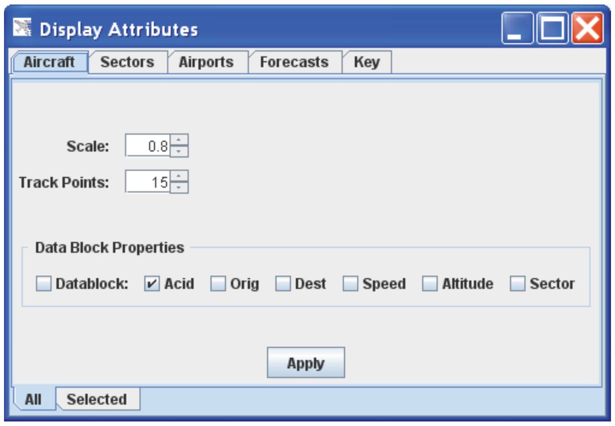
Track Points: 15

Data Block Properties

Datablock: [ ] Acid [✓] Orig [ ] Dest [ ] Speed [ ] Altitude [ ] Sector [ ]

AIRCRAFT

The aircraft display attribute window for ***all*** aircrafts allows the user to toggle the display of the aircraft attributes on the screen.



Scale: 0.8
Track Points: 15

Data Block Properties

Datablock: [ ] Acid [✓] Orig [ ] Dest [ ] Speed [ ] Altitude [ ] Sector [ ]

AIRCRAFT ATTRIBUTE: SCALE

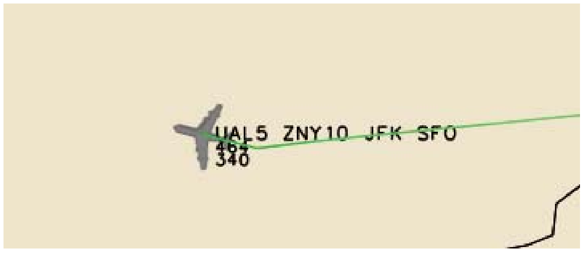
The ***Scale*** spinner allows a user to specify the scale of the aircraft icons on the display.

AIRCRAFT ATTRIBUTE: TRACK POINTS

The ***Track Points*** spinner allows a user to specify the number of historical waypoints per aircraft on the display.

DATABLOCK ATTRIBUTES

The ***Datablock*** checkbox allows a user to specify what flight tags will display.



DATABLOCK ATTRIBUTE: ACID

The ***Acid*** checkbox allows a user to specify whether the aircraft identifier will display.

DATABLOCK ATTRIBUTE: SECTOR

The ***Sector*** checkbox allows a user to specify whether the aircraft's current sector will display.

DATABLOCK ATTRIBUTE: ORIG

The ***Orig*** checkbox allows a user to specify whether the aircraft departure airport will display.

DATABLOCK ATTRIBUTE: DEST

The ***Dest*** checkbox allows a user to specify whether the aircraft arrival airport will display.

DATABLOCK ATTRIBUTE: SPEED

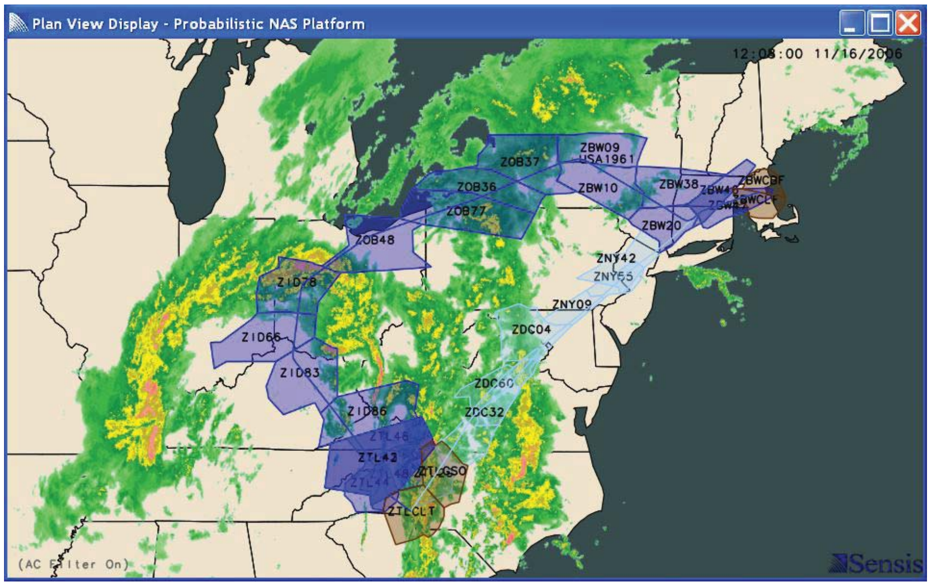
The ***Speed*** checkbox allows a user to specify whether the aircraft speed will display.

DATABLOCK ATTRIBUTE: ALTITUDE

The ***Altitude*** checkbox allows a user to specify whether the aircraft altitude will display.

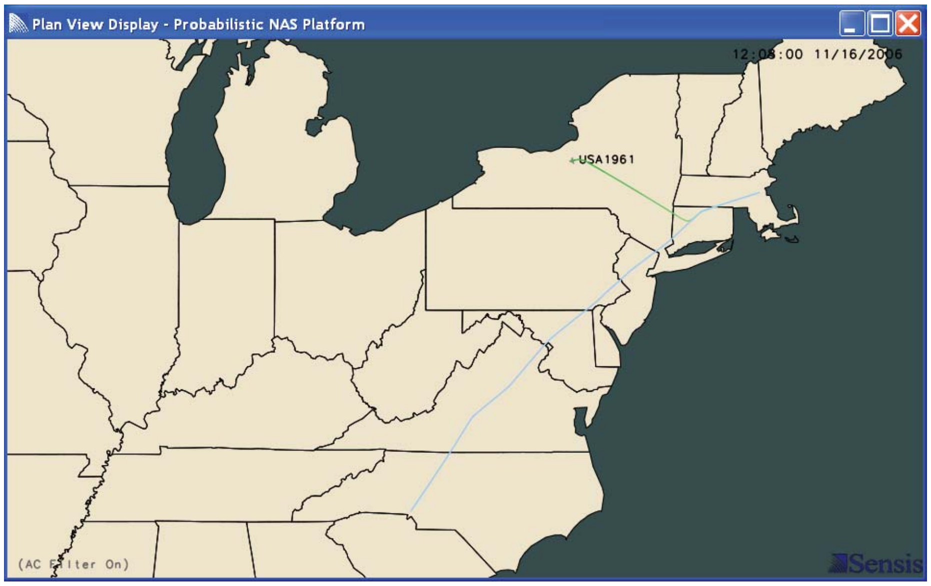
AIRCRAFT | SELECTED

The aircraft display attribute window for ***selected*** aircrafts allows the user to toggle the display of the selected aircraft attributes on the screen.



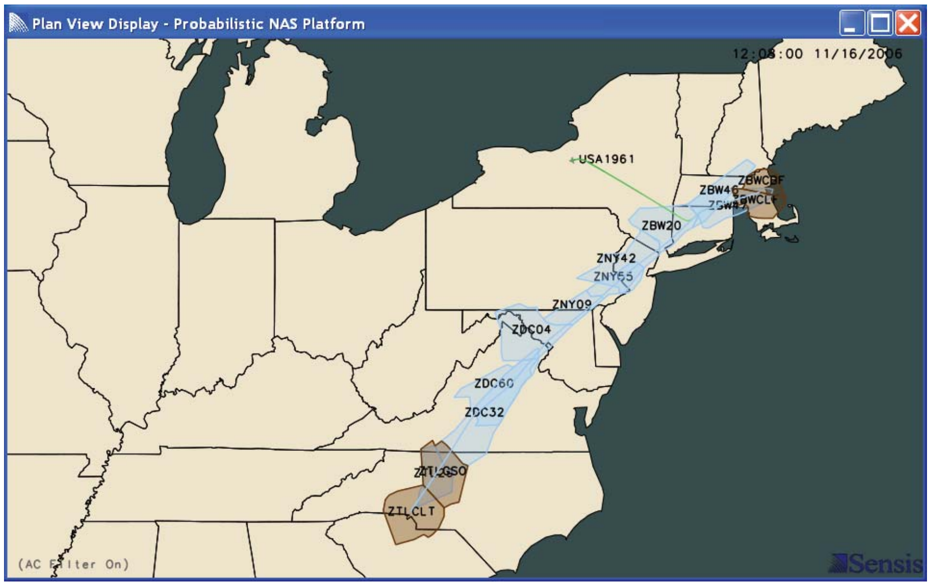
FLIGHT PLAN TRAJECTORY

The original flight plan trajectory is displayed as a blue line. If the line is cyan, then a reroute is in effect.



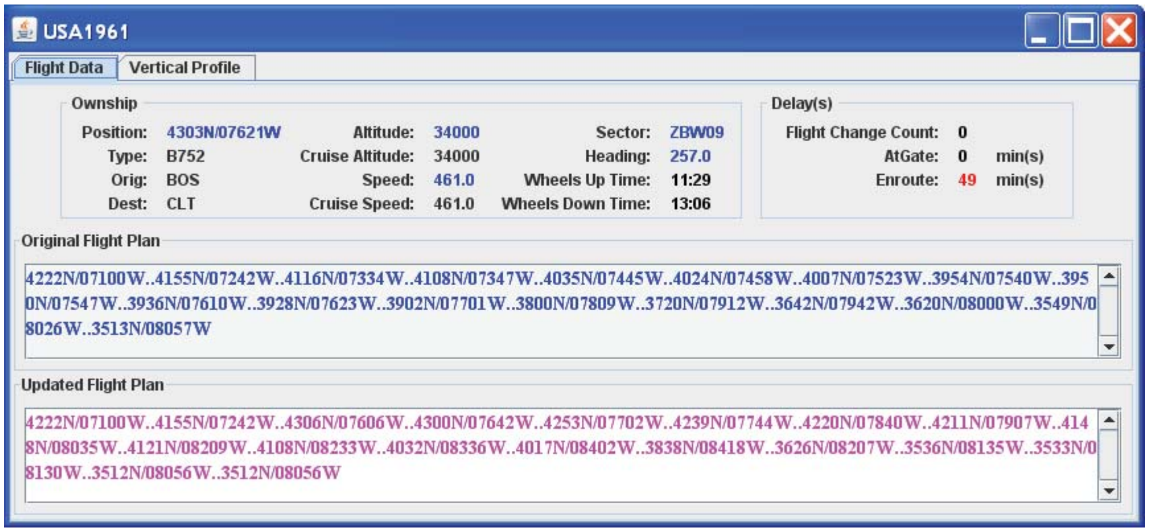
FLIGHT PLAN SECTORS

The flight plan sectors are displayed as translucent blue polygons. If the sectors are cyan, then a reroute is in effect. *Note*: TRACONs are brown.



FLIGHT DETAILS WINDOW

The flight details window displays detailed information about a selected flight in a pop-up window.


USA1961

Flight Data | Vertical Profile

Ownship

```
Position:  4303N/07621W        Altitude:  34000              Sector:  ZBW09
    Type:  B752         Cruise Altitude:  34000             Heading:  257.0
    Orig:  BOS                   Speed:  461.0     Wheels Up Time:  11:29
    Dest:  CLT            Cruise Speed:  461.0   Wheels Down Time:  13:06
```

Delay(s)

```
Flight Change Count:  0
             AtGate:  0    min(s)
            Enroute:  49   min(s)
```

Original Flight Plan

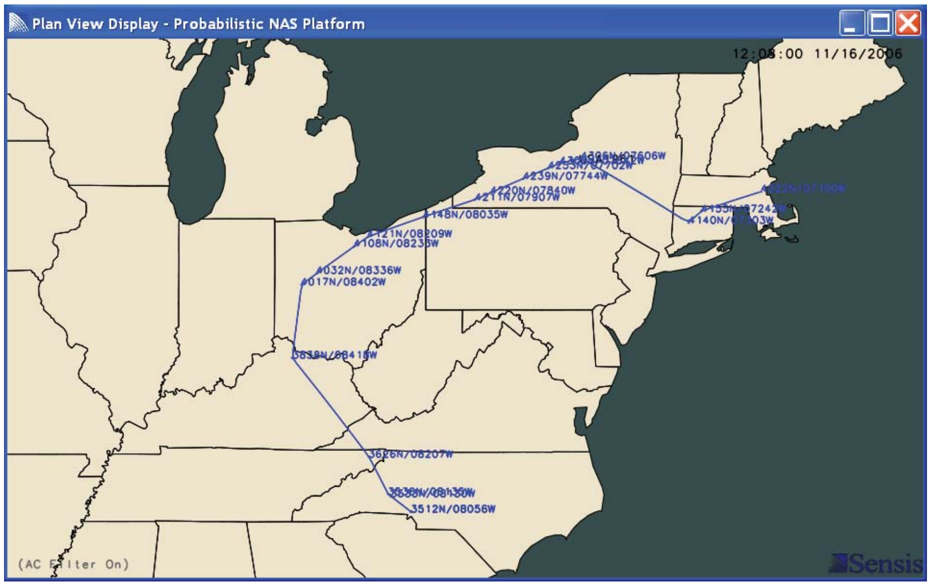
```
4222N/07100W..4155N/07242W..4116N/07334W..4108N/07347W..4035N/07445W..4024N/07458W..4007N/07523W..3954N/07540W..395
0N/07547W..3936N/07610W..3928N/07623W..3902N/07701W..3800N/07809W..3720N/07912W..3642N/07942W..3620N/08000W..3549N/0
8026W..3513N/08057W
```

Updated Flight Plan

```
4222N/07100W..4155N/07242W..4306N/07606W..4300N/07642W..4253N/07702W..4239N/07744W..4220N/07840W..4211N/07907W..414
8N/08035W..4121N/08209W..4108N/08233W..4032N/08336W..4017N/08402W..3838N/08418W..3626N/08207W..3536N/08135W..3533N/0
8130W..3512N/08056W..3512N/08056W
```

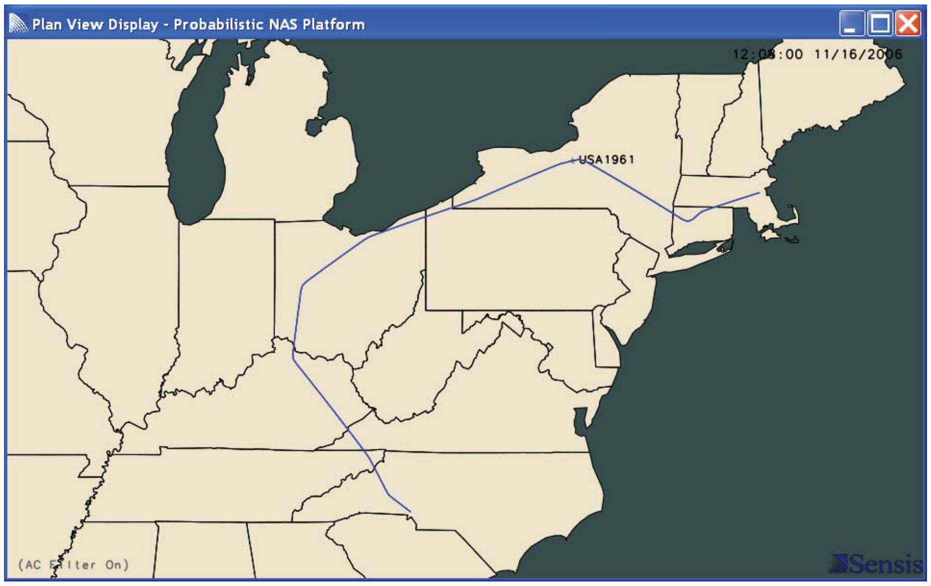
FLIGHT PLAN WAYPOINT NAMES

The flight plan waypoint names are displayed on the current flight plan line.



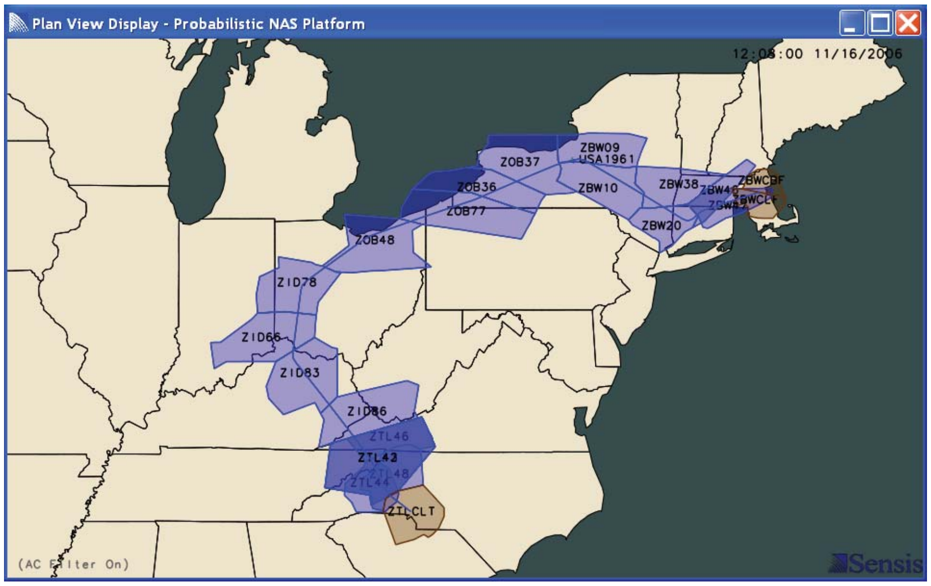
REROUTE TRAJECTORY

The reroute trajectory is display as a blue line.



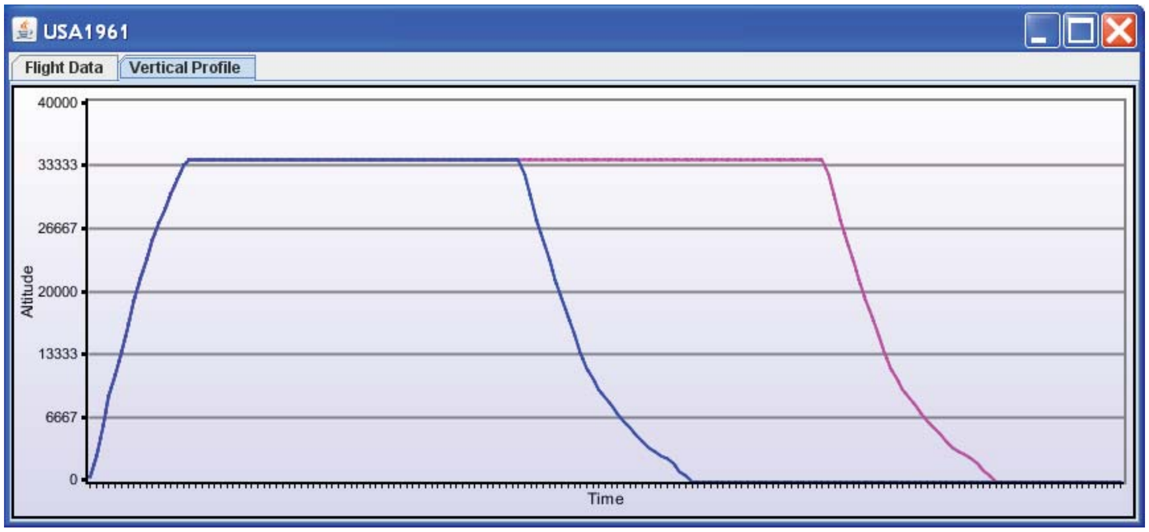
REROUTE PLAN SECTORS

The flight plan sectors are displayed as translucent blue polygons.



VERTICAL PROFILE WINDOW

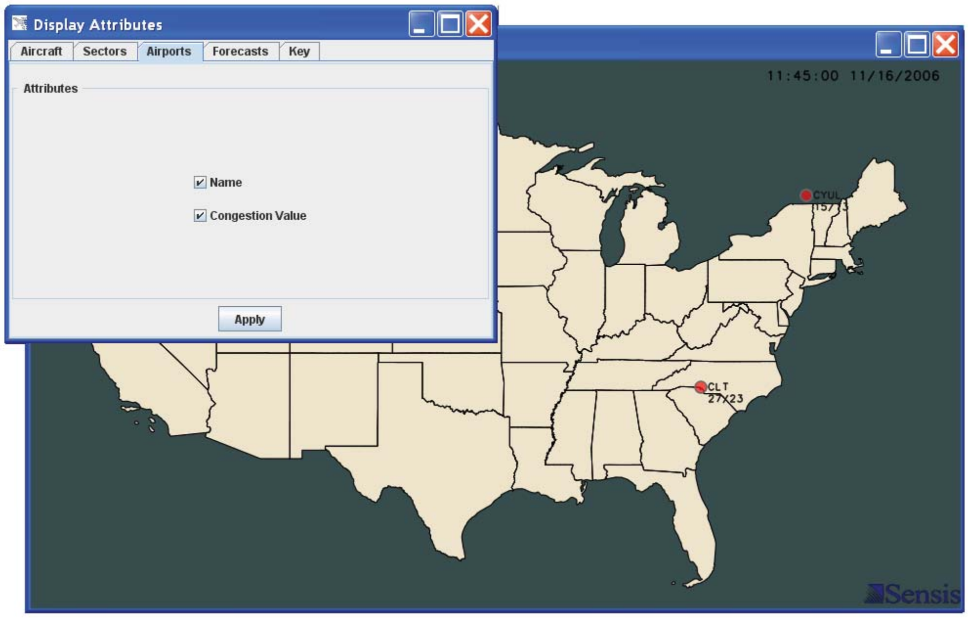
The vertical profile tab in the flight details window details the altitude profile of the flight. The blue line represents the original flight plan while the magenta line represents a reroute.



AIRPORT CONGESTION

The airport congestion display attribute window allows the user to toggle the display of the airport name and congestion value on the screen. The congestion value consists of two parts:

- Number of aircraft overloading an airport
- Airport capacity



☑ Name

☑ Congestion Value

AIRPORT CONGESTION ATTRIBUTE: NAME

The ***Name*** checkbox allows a user to specify whether airport names will display.

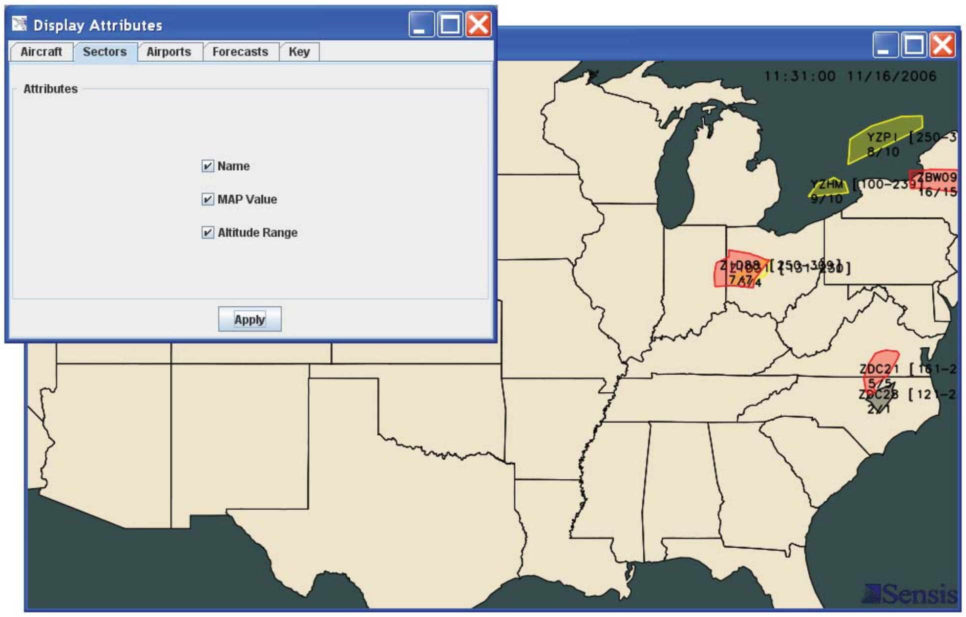
AIRPORT CONGESTION ATTRIBUTE: CONGESTION

The ***Congestion Value*** checkbox allows a user to specify whether airport congestion values will display.

SECTOR CONGESTION

The sector congestion display attribute window allows the user to toggle the display of the sector name, altitude range, and MAP value on the screen. The sector value consists of two parts:

- Number of aircraft overloading a sector
- Sector capacity



☑ Name
☑ MAP Value
☑ Altitude Range

SECTOR CONGESTION ATTRIBUTE: NAME

The ***Name*** checkbox allows a user to specify whether sector names will display.

SECTOR CONGESTION ATTRIBUTE: MAP VALUE

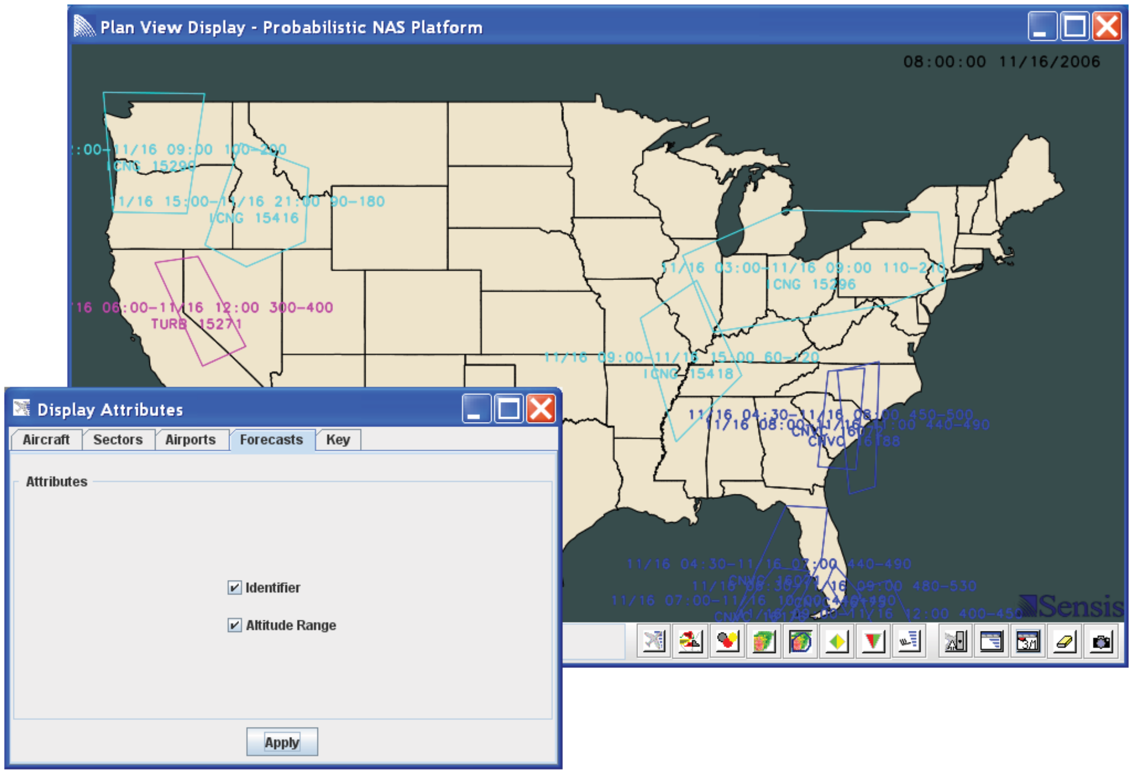
The ***MAP Value*** checkbox allows a user to specify whether the sector MAP values will display.

SECTOR CONGESTION ATTRIBUTE: ALTITUDE RANGE

The ***Altitude Range*** checkbox allows a user to specify whether the sector altitude range will display.

FORECASTS

The forecast display attribute window allows the user to toggle the display of the forecast identifier and altitude range on the screen.



☑ Identifier

☑ Altitude Range

FORECAST ATTRIBUTE: IDENTIFIER

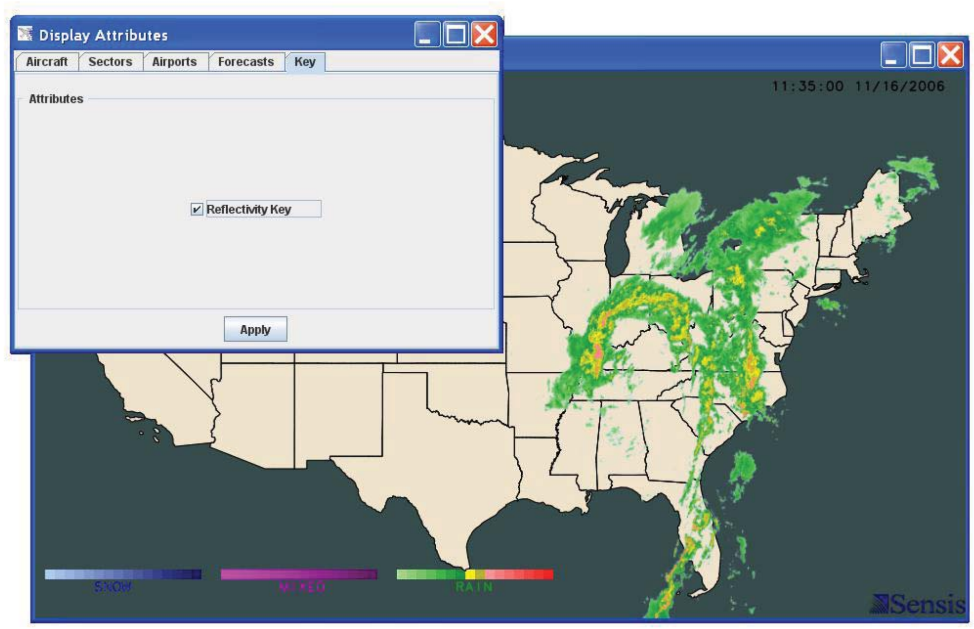
The ***Identifier*** checkbox allows a user to specify whether the forecast identifier will display.

FORECAST ATTRIBUTE: ALTITUDE RANGE

The ***Altitude Range*** checkbox allows a user to specify whether the forecast altitude range will display.

KEY

The key display attribute window allows the user to toggle the display of the reflectivity key on the screen.



KEY ATTRIBUTE: KEY

The ***Reflectivity Key*** checkbox allows a user to specify whether the reflectivity key will display.

APPENDICES

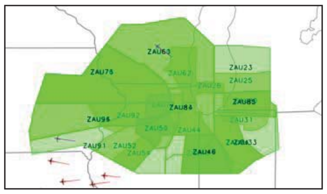
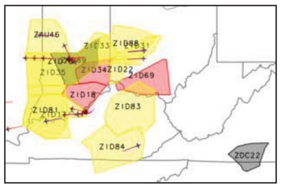
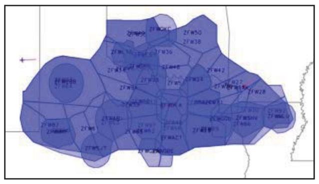
APPENDIX A: CONGESTION COLOR KEY

Congestion is color-coded on the PNP PVD. Each color signifies a minimum level of congestion. The following table defines the significance of each color used to indicate congestion.

Color	*Meaning*
Yellow	The congested NAS element is at or above 75% of capacity.
Red	The congested NAS element is at or above full capacity.
Black	The congested NAS element is at or above 125% of capacity.

APPENDIX B: SECTOR COLOR KEY

Displays the name of sectors. Sectors highlighted in green are selected by the cursor, blue highlighted are directly from the server or client to display, yellow, black, and red highlighting is indicative of the congestion threshold overage.

14809-02

SA Developer's Guide

Sensis Corporation

Prepared for:

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681

Under:

NASA BOA: NNL08AA17B

August, 2011

TABLE OF CONTENTS

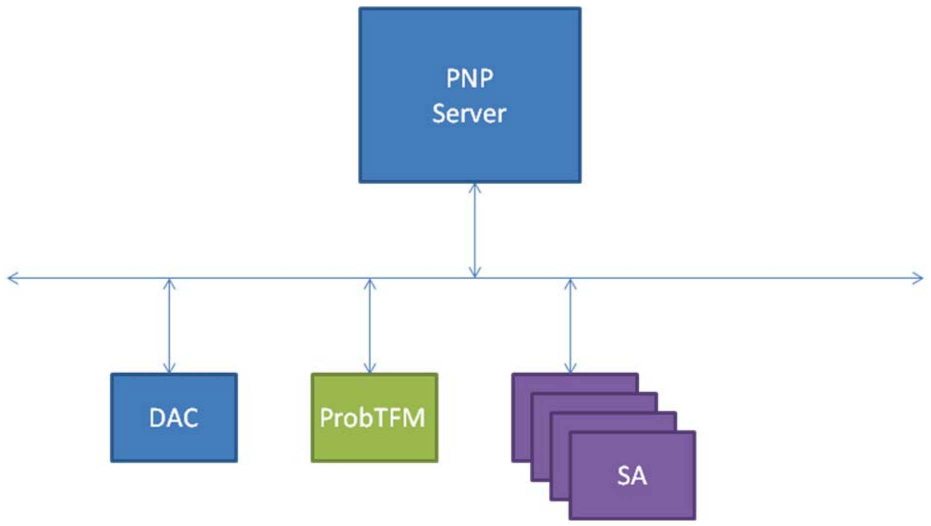
INTRODUCTION

The SA Developer's Guide provides developers with a guide to understand the SA client, PNP API, and software factory patterns available to add separation assurance algorithms.

SA OVERVIEW

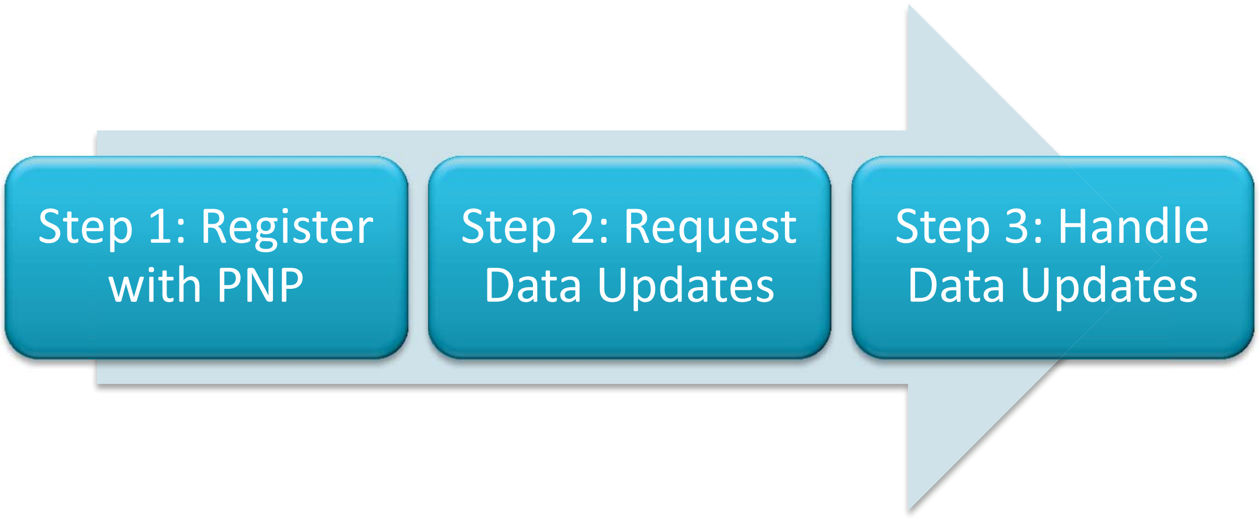
SA ARCHITECTURE

The SA is a Java client in the PNP architecture. Clients can dynamically enter and exit an experiment, using the PNP interface. The SA Client may request a variety of data messages, and can return to PNP modifications, including flight plan modifications and airspace (sector) redesigns.



SA CLIENTS

PNP clients model NAS components. By connecting to the server and registering with PNP, a client can request the data it needs to model NAS components and decision making. The SA client models conflict detection and resolution decision making in the NAS. The SA client has implemented the SimObjects API and performs three basic steps:


Step 1: Register with PNP
Step 2: Request Data Updates
Step 3: Handle Data Updates

1. Register with PNP Server

The SA client registers with PNP server as a client specifying its configuration parameters.

EXAMPLE

The following example registers the SA client with PNP:

```
ClientConfiguration config = new ClientConfiguration("SA");
config.setDistanceFilter((int) Wini.instance().getMinimumHorizontalSeparationInNm());
config.setAltitudeFilter(Wini.instance().getMinimumAltitudeBound());
config.setLookAheadTimeFilter(Wini.instance().getSeparationAssuranceTime());

init(Wini.instance().getPnpServerName(), config, flightDetailsDescr);
```

2. Request Updates

The SA client requests for Enroute aircraft, simulation commands, and health checks based on an adaptable interval value.

EXAMPLE

The following example requests a list of all Enroute flights every 15 minutes:

```
   int interval = Wini.instance().getServerRequestInterval();

   MessageRegistryManager.instance().register(PnpEnrouteFlightDetails.class, interval,
inFlightActionListener);
   MessageRegistryManager.instance().register(Heartbeat.class, interval,
heartbeatActionListener);
```

```
   MessageRegistryManager.instance().register(CommandRequest.class, interval,
commandActionListener);
```

3. Handles Data Updates

The SA client receives the data based on the adaptable interval value and manages the Enroute aircraft. The SA client processes all the aircraft and sends back a health check message.

EXAMPLE

The following example are message listeners associated to the requested messages.

```
InFlightActionListener inFlightActionListener = new InFlightActionListener();
HeartbeatActionListener heartbeatActionListener = new HeartbeatActionListener();
CommandActionListener commandActionListener = new CommandActionListener();
```

THE PNP COMMUNICATIONS CYCLE

The PNP server communicates with clients in a synchronous cycle. As PNP advances time in the simulation, it will periodically reach a time at which messages need to be sent to its clients. At each time point when the PNP server needs to send messages, it will send the appropriate messages, send a heartbeat message to the clients, and wait for the receiving clients to respond. Only after all receiving clients have responded will the PNP server continue to advance time. This asynchronous communications cycle ensures repeatability of the simulation, and is illustrated in Figure 2.

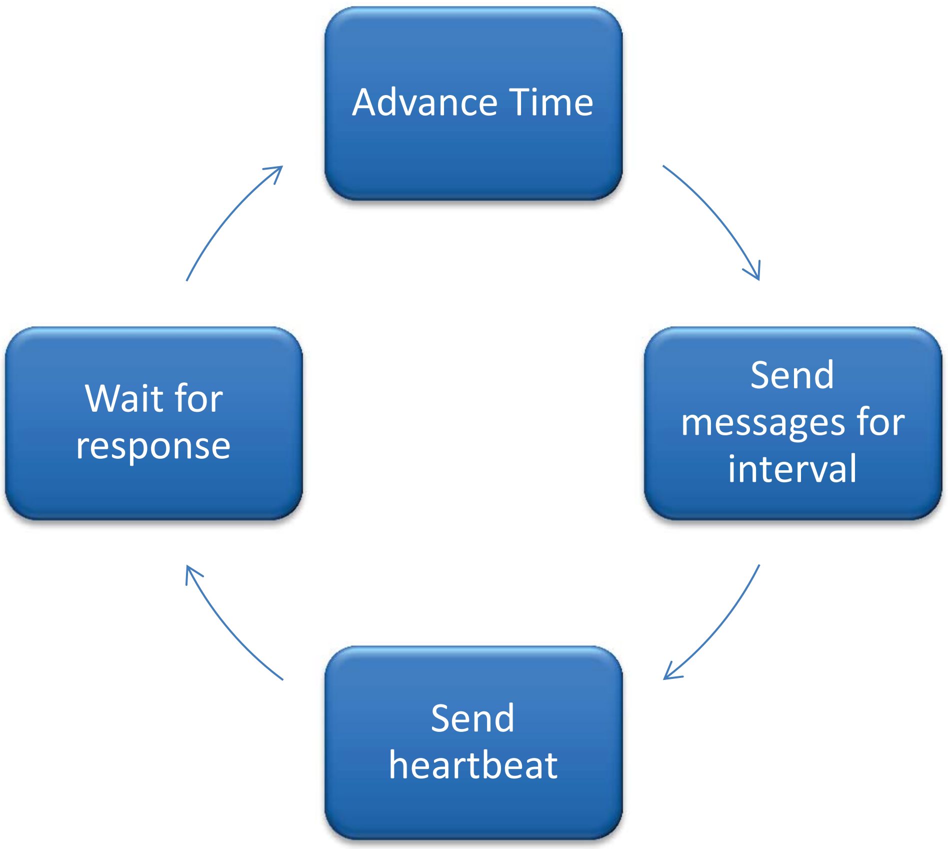


Figure 5. The PNP Communications Cycle

SIMOBJECTS API

SimObjects is the PNP API. It uses TCP/IP to transfer messages between PNP and its clients.

MESSAGING

The SimObjects API consists of over forty messages, which are used to transfer data between PNP and its clients.

Messages in PNP exist as ***`Serializable`*** Java classes named after the data they contain. `PnpFlightDetails`, for example, contains a list of flights and information pertaining to those flights. All message classes exist in the package ***pnp.share.network.messages***.

Please refer to the SimObjects API documentation for details on API messages.

DYNAMIC CLIENTS

Clients in PNP are referred to as dynamic because they can enter the simulation at any time. When connecting to PNP, each client specifies its configuration parameters, along with its data requirements. This provides for a flexible client/server architecture, where the server does not require advance knowledge of its clients, but can provide targeted data to each client based on its requirements.

CREATING A PNP JAVA CLIENT

REQUIRED LIBRARIES

The following libraries need to be included by a Java client.

Library	Use
pnp_common.jar	Contains shared API messages
commons-logging-api.jar	Used for error logging
commons-logging-av.jar	Used for error logging
log4j-1.2.9.jar	Used for logging
seagull-common.jar	Contains many utility functions utilized by several API messages
seagull-position.jar	Contains classes for defining position data
seagull-units.jar	Used by position classes for specifying position units

CONNECTING TO PNP FROM A JAVA CLIENT

Connecting a client to PNP is easy and requires only three steps.

STEP 1: REGISTER FOR MESSAGES

In order to connect and register with PNP, a client must create and send a list of message requests to PNP. These requests specify the messages PNP should send to the client, as well as how often each message should be sent.

To create a message request, simply register for messages using the *MessageRegistryManager* singleton. Three parameters are required to register a message:

- The class of the API message object
- The interval, in minutes of simulation time, at which PNP is to send the message to the client
- An object implementing the ApiListener interface to invoke when the requested message type is received

The ApiListener interface contains only one method, which will be called when the requested message is received. The only parameter of the *actionPerformed* method is the message object. The signature of the method is:

```
void actionPerformed(Object o);
```

EXAMPLE

The following example requests a list of all enroute flights every 15 minutes:

```
/**
 * Register for messages desired to be received from PNP
 */
private void registerForMessages()
{
    // We'd like to receive all messages at 15-minute intervals
    int interval = 15;

    MessageRegistryManager registry = MessageRegistryManager.instance();

    // Register for messages
    registry.register(PnpEnrouteFlightDetails.class, interval, m_MessageHandler);
}
```

In this example, *m_MessageHandler* is an *ApiListener* object. In your client, you may use the same *ApiListener* to handle many messages, or you may use different *ApiListener*s for different messages or sets of messages. It is up to you.

STEP 2: CREATE A CLIENT CONFIGURATION OBJECT

Now, you need to create a *ClientConfiguration* object to tell PNP a little bit about your client. Some rudimentary information is required, such as the name of your client, so that PNP knows how to list your client in the System Monitor. Other settings in the configuration object are optional, and you may use these as required by your client.

A full description of the methods in *ClientConfiguration* is available in the Javadoc for the API, available in the ***doc/*** directory of your PNP release.

To create a basic *ClientConfiguration* object, simply call the constructor of *ClientConfiguration* with the name of your client as the only parameter.

```
// Create a configuration object for this client
ClientConfiguration config = new ClientConfiguration("My First Client");
```

STEP 3: INITIALIZE THE CONNECTION

Once the client configuration object has been created, the connection with the server can be initiated. To start the connection, simply create a *PnpFlightDetailsDescr* object and use it to call the *init* method to start the connection.

The *PnpFlightDetailsDescr* object simply gives the PNP server some guidance as to what information to include in flight data sent from the server to the client. This allows the server to optimize network usage by not sending data to the client that the client doesn't need.

EXAMPLE

The following example creates a *PnpFlightDetailsDescr* that requires that only the sector schedule data be filled out for flight objects sent by the server.

```
// Set up the flight details descriptor
PnpFlightDetailsDescr flightDetailsDescr = new PnpFlightDetailsDescr();
flightDetailsDescr.setSectorScheduleFlag(true);
```

At this point, we are ready to connect to the PNP server. To do this, simply call *init*:

```
// Connect to PNP
init(pnp_server_name, config, flightDetailsDescr);
```

Here, *pnp_server_name* is the network name of the host computer on which PNP is running, and *config* is the *ClientConfiguration* object.

SENDING DATA FROM YOUR CLIENT

Sending data from a client to the PNP server is easy. The OCMM contains a *send* method which will send any message to PNP. The message will be a *Serializable* object from the ***pnp.share.network.messages*** package of PNP.

Using the class from the previous section, the following method could be used to send messages back to PNP:

```
private void send(Object message)
{
        m_Client.send(message);
}
```

RESPONDING TO PNP HEARTBEAT MESSAGES

Once the PNP server has sent all requested messages to your client for the given interval, it will send a `Heartbeat` message to your client, and will await a response before moving the simulation forward. The purpose of this is to keep the simulation synchronous and repeatable. The client should only respond after it has processed all of the data it has received for that interval.

To respond to the heartbeat message, the client need only send a `HeartbeatResponse` message to the server:

```
send(new HeartbeatResponse())
```

INTERACTING WITH PNP

A client can interact with PNP by modifying flights. PNP clients can delay or reroute flights, and the following sections explain how to do just that.

THE BATCHRESPONSE MESSAGE

All delays and reroutes are performed using the BatchResponse message. The BatchResponse message allows a client to specify which (if any) flights to delay or reroute. This message should be sent to PNP once the client has received a Heartbeat message and has done all of its processing.

The BatchResponse message contains the following constructor:

```
public BatchResponse(int dstId,
                     Map<Integer, Short> delayMap,
                     Map<Integer, ReroutePlan> rerouteMap,
                     Map<Integer, ReroutePlan> inflightRerouteMap)
```

```
    {
        m_DstId = dstId;
        m_DelayMap = delayMap;
        m_RerouteMap = rerouteMap;
        m_InflightRerouteMap = inflightRerouteMap;
    }
```

DELAYING FLIGHTS

The delay map in the BatchResponse is a map that is ordered by flight ID. Each flight ID maps to a *short* value specifying the number of minutes by which to delay the flight. PNP will apply the specified delay to each flight in the *delayMap* provided.

REROUTING FLIGHTS

Flight reroutes fall into two categories:

- Pre-departure reroutes, applied to flights that have not yet departed
- Inflight reroutes, applied to flights that are already enroute

As with delays, the maps specified for reroutes are keyed by flight ID. The ReroutePlan provided simply contains the new list of 3-dimensional positions, listed at one-minute intervals, for each flight. The *rerouteMap* specifies any pre-departure reroutes requested by the client. The *inflightRerouteMap* specifies any inflight reroutes requested by the client.

All reroutes requested by the client are implemented by PNP.

BEST PRACTICES FOR DEVELOPING PNP CLIENTS

In order to make the best use possible of PNP and SimObjects, the following have been identified as best practices to follow when creating clients for PNP:

- **Create separate threads for processing vs. connection management**
  A PNP client performs whatever simulation tasks are appropriate for it. While it is performing this processing, it may need to send or receive messages from PNP. Keeping the communications handling code in a separate thread from the processing code allows messages to be sent and received while the processing code is running. This added flexibility significantly enhances the capability of any client developed in this way.

- **Send all processing results to PNP before sending a `HeartbeatResponse`**
  A `HeartbeatResponse` tells PNP that your client has finished its processing. Once PNP has received this message, it continues advancing time, and will send a new set of messages to your client for processing. If the client has not finished sending its results to PNP before it receives the new data, the repeatability of the simulation may be affected, as the order in which messages are sent and received may change if the same simulation is run again. Additionally, if the client's internal data are not cleared before the new data are received, the client may end up with incorrect data. To ensure against this, the client should wait until after sending all its results before sending the `HeartbeatResponse` message.

FACTORIES

The SA client is software architecture is expandable through software factory patterns. The SA client has three factories built to allow for the implementation of different algorithms.

SEPARATION ASSURANCE IMPLEMENTATION FACTORY

The Separation Assurance Implementation Factory allows the user to create an interface that handles the processing of aircraft and CDR algorithms.

```
public interface SeparationAssuranceInterface
{
    public void process(ConflictDetectionInterface conflictDetection,
                        ConflictResolutionInterface conflictResolution,
                        int timestepInSeconds,
                        long ts);
    public void clear();
    public Map<Integer, ReroutePlan> getReroutes();
    public LosList getLosList();
    public ConflictList getConflictList();
}
```

The process method passes the adapted CDR interfaces, trajectory time step between waypoints, and current timestamp. The clear method allows the implementation to clear its state data. The getRerotues method returns the reroutes to be passed on to the PNP server. The getLosList method returns the loss of separation events to be passed on to the PNP server for display on the PVD. The getConflictList method returns the loss of separation events to be passed on to the PNP server for display on the PVD.

N-SQUARED INTERFACE

The N-Squared interface compares every aircraft to every other aircraft using the adapted conflict detection and resolution factory interfaces.

OWNSHIP

The ownship interface compares the current aircraft (ownship) to every other aircraft (target aircraft) using the adapted conflict detection and resolution factory interfaces.

SURVEILLANCE

Surveillance separation assurance implementation was provided by the government and beyond the scope of this contract.

CONFLICT DETECTION FACTORY

The Conflict Detection Factory allows the use to implement conflict detection algorithms.

```
public interface ConflictDetectionInterface
{
    public List<DetectedConflict> detectConflicts(int ownshipFlightId,
                                                  List<FlightDetails> flights,
                                                  long ts);
}
```

The detectionConflicts method compares the trajectories of the ownship flight to that of the other flights over time based on the current timestamp.

SENSIS CONFLICT DETECTION

The Sensis Conflict Detection algorithm is a simple geo-distance comparison. The trajectory waypoints are compared based on position and time for an adaptable look a-head period.

STRATWAY CONFLICT DETECTION

The Stratway Conflict Detection algorithm is described in the Stratway v1.0 User Manual (NASA/TM-2011-000000).

CONFLICT RESOLUTION FACTORY

The Conflict Resolution Factory allows the use to implement conflict resolution algorithms.

```
public interface ConflictResolutionInterface
{
    public List<Resolution> computeConflictResolutionOptions(
                                FlightDetails ownship,
                                FlightDetails nextship,
                                DetectedConflict conflict,
                                long ts);
}
```

The computeConflictResolutionOptions method compares the trajectories of the ownship and target flight updating the DetectedConflict object and returning a list of resolutions based on the current timestamp.

SENSIS CONFLICT RESOLUTION

The Sensis Conflict Resolution algorithm is a simple four resolution approach. The algorithm attempts a speed deceleration, altitude decent, right turn, and left turn resolution.

STRATWAY CONFLICT RESOLUTION

The Stratway Conflict Resolution algorithm is described in the Stratway v1.0 User Manual (NASA/TM-2011-000000).

GLOSSARY

Term	Definition
API	Application Programming Interface
NAS	National Airspace System
OCMM	Object Client Message Manager. Utility for managing a client's data connection with PNP.
PNP	Probabilistic NAS Platform
SimObjects	The PNP API

APPENDIX A: CONFLICT DETECTION AND RESOLUTION

Some client applications detect airspace conflicts, or imminent losses of separation between aircraft. The PNP Plan View Display (PVD) is capable of depicting conflicts, as well as losses of separation, when data on such events are supplied by a client.

CONFLICT-RELATED API MESSAGES

The PNP API contains two messages that may be used by clients for pointing out airspace conflicts and losses of separation. These are:

- ***ConflictList***
  The *ConflictList* message contains a list of *Conflict* objects, each representing an imminent loss of separation between two aircraft.

- ***LosList***
  The *LosList* message contains a list of *LossOfSeparation* objects, each representing a current loss of separation between two aircraft.

Both of these messages are documented in the API JavaDoc documentation contained in the ***doc/*** folder of your PNP release.

DISPLAY OF CD&R INFORMATION

When a conflict or loss of separation is sent from the client to the server, the server will display the conflict on the PVD for the duration of the time bin. This information is cleared by the server at the end of the time bin, at which it is allowed to expire if not re-sent by the client. This behavior exists because conflicts or losses of separation may be resolved during the time bin. Therefore, a client that detects conflicts and loss of separation (LOS) events must re-send all conflict and LOS information at every time bin during which the conditions are active.

Separation Assurance Client

v1.1

User Guide

8/18/2011
Sensis Corporation

Table of Contents

The Separation Assurance Client (SA) is a platform to experiment with different conflict detection and resolution algorithms. The SA provides an interface for both conflict detection and resolution (CDR) algorithms. Currently two CDR algorithms have been integrated. One simple/fast CDR implementation provided by Sensis and the other by NASA Langley called Stratway.

SCOPE

This document guides the SA user in the design and execution of PNP simulations with an SA client.

Topics covered include:

- Software Overview
- Configuring SA
- Running SA

SOFTWARE

ARCHITECTURE

The SA is a client to the PNP server. The SA is dependent upon the PNP server for data and heartbeat messages.

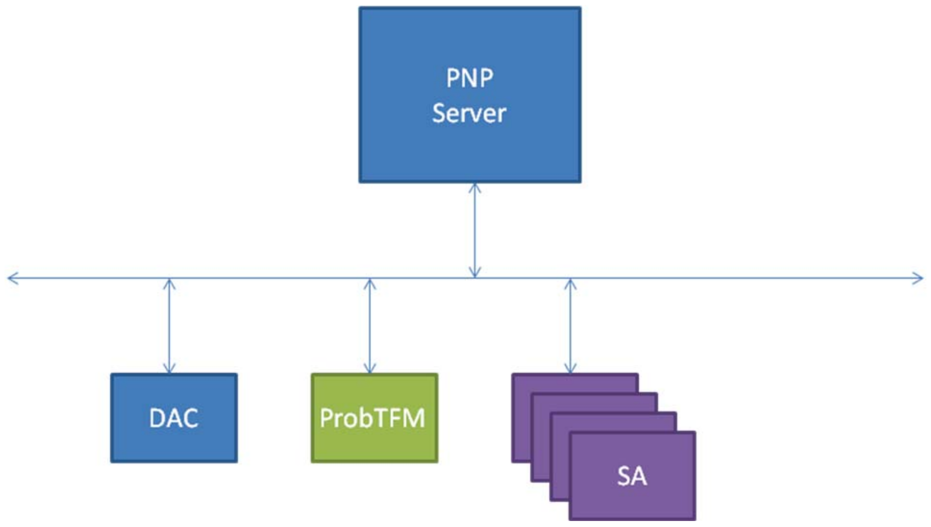


Figure 6. PNP/SA Architecture

DESIGN

SA

The SA has four main responsibilities: communications, CDR, trajectory management, and statistics. The SA executive manages flights and calls the CDR algorithms. The SA communications thread establishes and maintains communications with the PNP server. The communications thread registers for messages with the PNP server. The communications thread requests the interval it would like to receive messages.

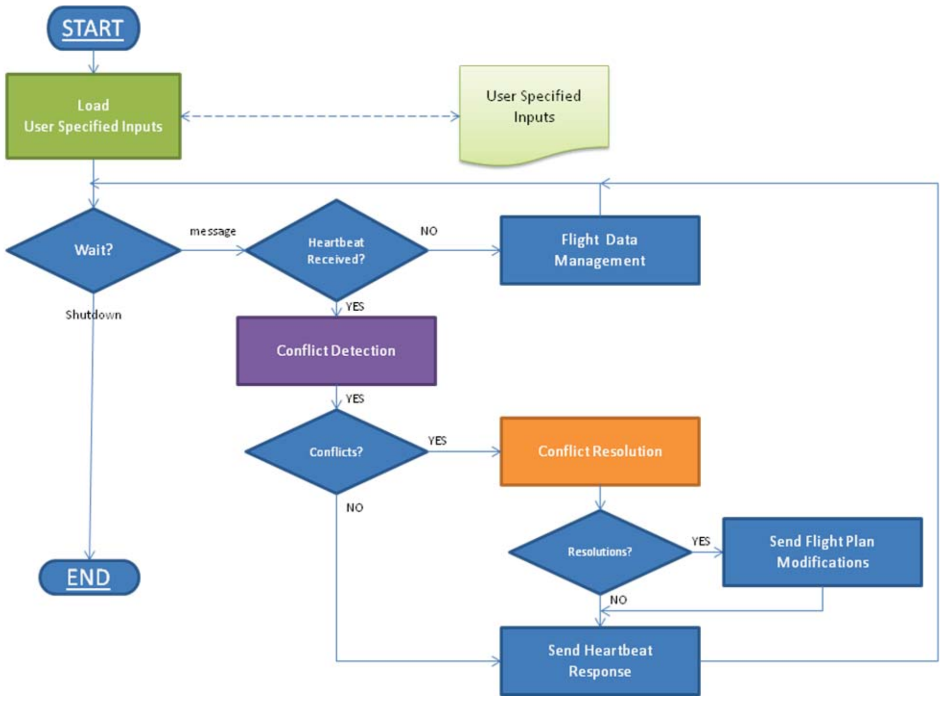


Figure 7. SA Design

MESSAGES

The messages in the system are serialized objects. The following section identifies the current messages in the system.

REQUEST MESSAGES

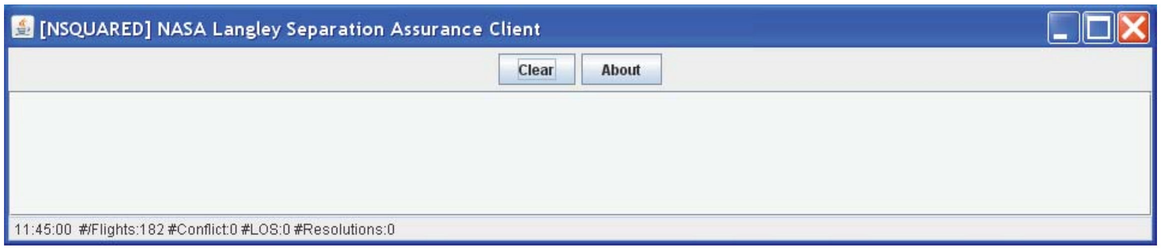
Request Messages	
Message Type	**Source**
PNP Enroute Flight Details	**Server**
Command Requests	**Server**
Heartbeat	**Server**

RESPONSE MESSAGES

Response Messages	
Message Type	**Source**
LosList	**Client**
ConflictList	**Client**
ReroutePlan	**Client**
Heartbeat Response	**Server**

EXPERIMENTS

In order to include the SA in PNP experiment a SA configuration file must be created as described below.



11:45:00 #Flights:182 #Conflict:0 #LOS:0 #Resolutions:0

DATA SETS

SA requires the following data sets.

DIRECTORIES

INI

The ***ini*** subdirectory contains initialization files used to configure the SA.

FILES

SA PARAMETERS

The SA parameters control the runtime attributes of the experiment. The parameters customize the CDR algorithms.

The various SA parameters are specified in the SA initialization (.ini) file and are broken up into parameter sets. The following sections describe the SA parameter sets found in the SA initialization file.

PNP PARAMETERS

PNP parameters provide basic startup and initialization information for PNP launcher. These parameters do not typically change between experiments, and are meant to be used to standardize a set of runs and ensure that some basic parameters are used across several sets of runs.

PNP Parameters

Field Name	Field Description	Default Value
name	The name parameter specifies the name of the client	SA
component	Specifies the type of component in the experiment.	client
exec	Specifies how PNP launcher will start the client.	bin\\run.cmd sa
hostname	Specifies the component's hostname	localhost
param	Specifies any additional parameters passed to the PNP launcher.	none

CONTROL PARAMETERS

Control parameters provide basic connection and initialization information for PNP Server. These parameters do not typically change between experiments, and are meant to be used to standardize a set of runs and ensure that some basic parameters are used across several sets of runs.

Control Parameters

Field Name	Field Description	Default Value
pnpservername	The pnpservername parameter specifies the hostname of the PNP server.	localhost
conflictdetectionimplementation	The conflictdetectionimplementationparameter specifies the implementation name for the conflict detection algorithm.	sensis
conflictresolutionimplementation	Specifies the implementation name for the conflict resolution algorithm.	sensis

policy	Specifies the flight request policy instructing the PNP how to package flights. polygon – Instructs PNP to package flights in an area defined by separationassurancepolygon. Fleet Name – Instructs PNP to package flights by an Aircraft Id filter (example. – UAL). Ownship - Instructs PNP to package flights by Aircraft Id (example – UAL1076).	nsquared
serverrequestinterval	Specifies in minutes how often you want the requested message data.	1

KNOB PARAMETERS

Knobs represent a series of parameters that can be adjusted to tweak the behavior of the simulation.

Knob Parameters

Field Name	**Field Description**	**Default Value**
separationassurancetime	Specifies in minutes how far to look ahead in the trajectory .	2
minimumaltitudebound	Specifies the minimum altitude of interest in which an aircraft is considered for SA.	1000
minimumhorizontalseparation	Specifies the minimum horizontal separation distance in nautical miles between two aircraft	5
minimumverticalseparation	Specifies the minimum vertical separation distance in feet between two aircraft.	1000
departuredistance	Specifies the distance in nautical miles away from the departure airport in which an aircraft is considered for SA.	100

arrivaldistance	Specifies the distance in nautical miles away from the arrival airport in which an aircraft is considered for SA.	100
trajectorytimestepinseconds	Specifies the trajectory time step in seconds.	10
hitl	Specifies the debug mode that allows a user to visualize the resolution before they are intrumented.	false
separationassurancepolygon	Specifies the polygon used when the policy is configured to polygon. (example - 34.8964/-95.03756,30.8964/-95.03756,30.8964/-99.03756,34.8964/-99.03756)	

DATASET PARAMETERS

The Dataset parameters control which files PNP uses as its source for weather and traffic data. Note that all directory definitions use the double-backslash (\\) to separate directories. Double backslashes are necessary for the proper operation of PNP.

Dataset Parameters

Field Name	Field Description	Default Value
conflictdataset	The Input File parameter is the directory and filename of the Flight Data Set (FDS) file containing all the planned flights for the debug simulation.	\\ARCHIVE\\demandsets\\conflicts.csv

OUTPUT PARAMETERS

The Output parameters specify what data should be output by PNP.

Output Parameters

Field Name	Field Description	Default Value
description	A description of the experiment	
outputpath	Specifies where the output data is to be written	data\\reports

DEBUG PARAMETERS

The Output parameters specify what data should be output by PNP.

Output Parameters

Field Name	Field Description	Default Value
debugCdr	Specifies the debug flag to output cdr debug information	false

OUTPUTS

The SA generates several reports that can be used for analysis and post processing by other applications. These reports are described in the following sections.

REPORTS

SUMMARY REPORT

The Summary Report is generated by the SA at the conclusion of an experiment. This report contains the following fields:

```
return "Conflicts,SolvedConflicts,UnresolvedConflicts,AvgTimeInConflict,%Success";
```

Field	Description
Experiment	The name of the experiment
#RightTurnResolutions	Total number of right turn resolutions
#LeftTurnResolutions	Total number of left turn resolutions
#SpeedControlResolutions	Total number of speed control resolutions
#AltitudeChangeResolutions	Total number of altitude change resolutions
PosDelayMinutes	The total minutes of flight time savings for in-flight reroutes that increase flight time
negDelayMinutes	The total minutes of flight time savings for in-flight reroutes that reduce flight time
Conflicts	Total number of conflicts.
SolvedConflicts	Total number of resolutions accepted by SA algorithm.
UnresolvedConflicts	Total number of conflicts after implementing resolutions.
AvgTineInConflict	Average time to solve a conflict.

%Success	Resolution algorithm percentage solved.

CONFLICT REPORT

The SA Conflict Report is generated by the SA after every SA session that there are conflicts. This report contains the following fields:

Field	Description
timestamp	The timestamp of the SA session
#PreConflicts	Total number of conflicts before performing SA.
#PreLOS	Total number of loss of separation events before performing SA.
#Resolutions	Total number of resolutions accepted by SA algorithm.
#SolvedConflicts	Number of solved conflicts.
#PostConflicts	Total number of conflicts after implementing resolutions.
#PostLos	Total number of loss of separation events after implementing resolutions.
%resolved	Running percent conflicts resolved
TotalConflicts	Total running conflicts

DETECTED CONFLICT REPORT

The Detected Conflict Report is generated by the SA as conflicts are encountered. This report contains the following fields:

Field	Description
timestamp	Timestamp of detected conflict
Aircraft Id	Aircraft identifier
Flight Index	Unique flight identifier assigned by PNP server flight manager.
AcType	Aircraft type
Orig	Departure airport
Dest	Arrival airport
Departure Time	Timestamp when aircraft leaves the gate
Wheels Up time	Timestamp when aircraft takes off
PosIdx	Current position Index of 4D trajectory
ConflictType	Future conflict or loss of separation event
SecondsToLOS	Seconds to loss of separation
SecondsToPCA	Seconds to point of closest approach
SecondsToGOS	Seconds to gain of separation
Delay	Pre-departure delays
#Maneuvers	Number of manevers

RESOLUTION REPORT

The Resolution Report is generated by the SA to perform resolution analysis. This report contains the following fields:

Field	Description
Aircraft Id	Aircraft identifier
Flight Index	Unique flight identifier assigned by PNP server flight manager.
Status	Status of resolution: attempt, failed, n/a, or accepted.
#Conflicts	Number of conflicts after implementing this resolution
resolution	Actual resolution

DELAY REPORT

The Delay Report is generated by the SA. This report contains aircraft delay information.

Field	Description
Aircraft Id	Aircraft identifier
Flight Index	Unique flight identifier assigned by PNP server flight manager.
ResolutionType	RIGHT_TURN, LEFT_TURN, SPEED_CONTROL, ALTITUDE_CHANGE.
PosDelayMinutes	The total minutes of flight time savings for in-flight reroutes that increase flight time
negDelayMinutes	The total minutes of flight time savings for in-flight reroutes that reduce flight time

UNRESOLVED CONFLICT REPORT

The Unresolved Conflict Report is generated by the SA when the CDR algorithm cannot solve a conflict. This report contains aircraft delay information.

Field	Description
timestamp	Timestamp of detected conflict
Ownship Flight Index	Unique flight identifier assigned by PNP server flight manager.
Target Flight Index	Unique flight identifier assigned by PNP server flight manager.

APPENDICES

APPENDIX A: HITL 3D DEBUG PROCEDURE

1. *Use conflict report to select specific conflict*
2. *Create a file demand set* **(\ARCHIVE\demandsets\conflict1.csv**) *with the two conflicting aircraft. Verify/update WheelsUpTime as a delay may have been assigned by PNP before departure.*
3. *Create sa_conflict1.ini file to include:*

```
[knob]

hitl = true

[dataset]

conflictdataset = \\ARCHIVE\\demandsets\\conflict1.csv
```

4. *Run SA standalone with sa_{description}.ini*
5. *Use mouse toolbar buttons to manipulate 3D display.*

Required Time of Arrival (RTMX) Client

v1.1

User Guide

6/26/2012
Saab Sensis Corporation

TABLE OF CONTENTS

INTRODUCTION

The Required Time of Arrival Client (RTMX) is a platform to experiment with different spacing concepts. The departure spacing implementation is performed on pre-departures using airport capacities. RTA assignment is performed en-route based on the distance to the metering fix. When an aircraft enters the freeze horizon a RTA is assigned. The RTMX client continuously checks for RTA conformance as flights approach the metering fix. RTA reassignments occurs based on non-conformance.

SCOPE

This document guides the RTMX user in the design and execution of PNP simulations with a RTMX client.

Topics covered include:

- ❖ Software Overview
- ❖ Configuring RTMX
- ❖ Running RTMX

SOFTWARE

ARCHITECTURE

The RTMX is a client to the PNP server. The RTMX is dependent upon the PNP server for data and heartbeat messages.

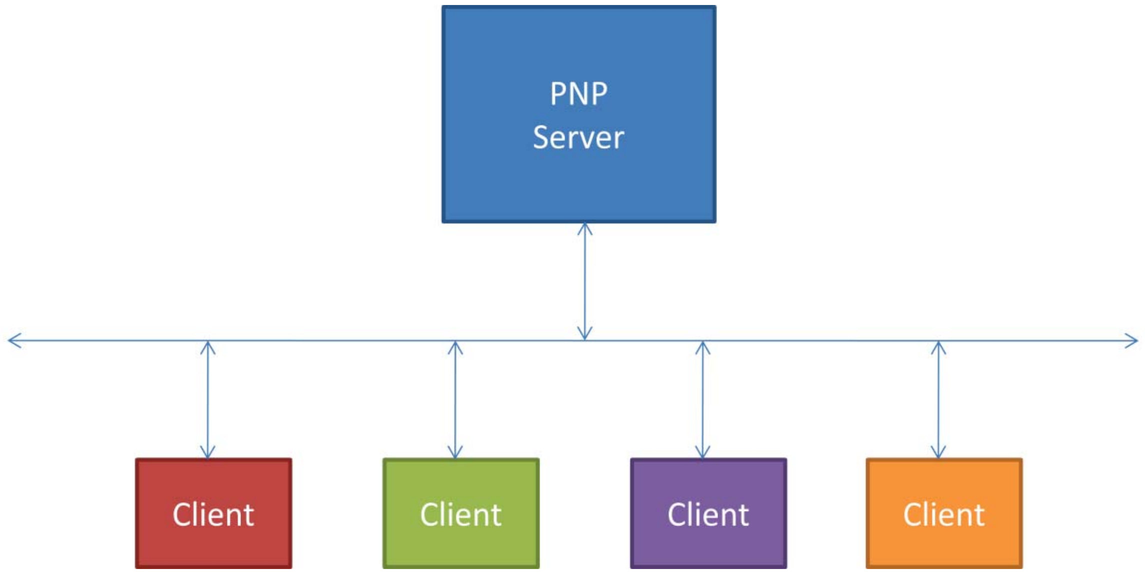


Figure 8. PNP/ RTMX Architecture

DESIGN

RTMX

The RTMX has four main responsibilities: communications, departure spacing, RTA assignment, and statistics. The RTMX executive manages flights and calls the spacing algorithms. The RTMX communications thread establishes and maintains communications with the PNP server. The communications thread registers for messages with the PNP server. The communications thread requests the interval it would like to receive messages.

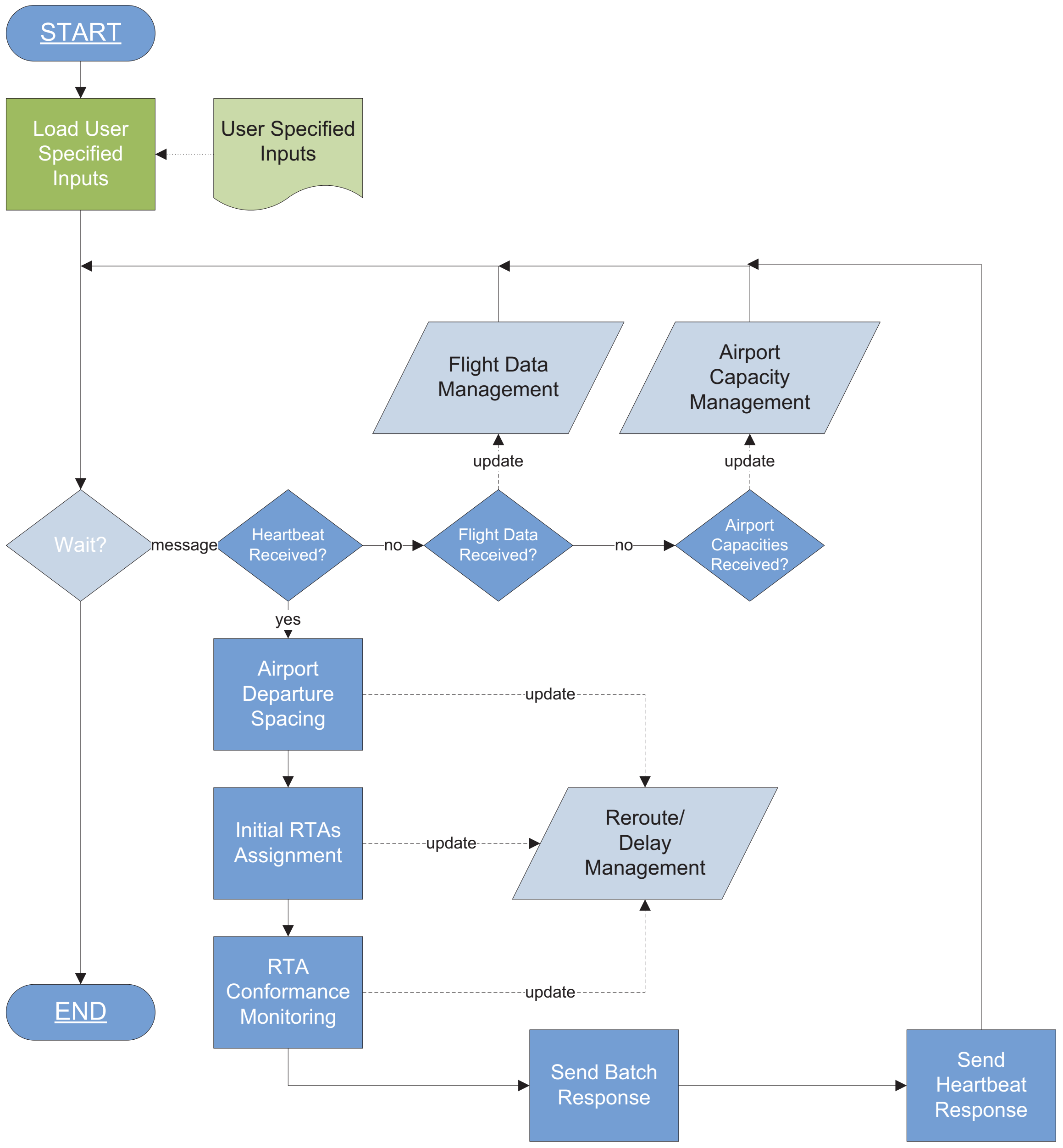


Figure 9. RTMX Design

MESSAGES

The messages in the system are serialized objects. The following section identifies the current messages in the system.

REQUEST MESSAGES

Request Messages	
Message Type	**Source**
PNP AtGate Flight Details	**Server**
PNP Enroute Flight Details	**Server**
AirportCapacities	**Server**
Command Requests	**Server**
Heartbeat	**Server**

RESPONSE MESSAGES

Response Messages	
Message Type	**Source**
BatchResponse	**Client**
Heartbeat Response	**Server**

EXPERIMENTS

In order to include the RTMX in PNP experiment a RTMX configuration file must be created as described below.

DATA SETS

RTA requires the following data sets.

DIRECTORIES

INI

The ***ini*** subdirectory contains initialization files used to configure the RTA.

FILES

RTA PARAMETERS

The RTA parameters control the runtime attributes of the experiment. The parameters customize the weather avoidance concepts.

The various RTA parameters are specified in the RTA initialization (.ini) file and are broken up into parameter sets. The following sections describe the RTA parameter sets found in the RTA initialization file.

PNP PARAMETERS

PNP parameters provide basic startup and initialization information for PNP launcher. These parameters do not typically change between experiments, and are meant to be used to standardize a set of runs and ensure that some basic parameters are used across several sets of runs.

PNP Parameters

Field Name	Field Description	Default Value
name	The name parameter specifies the name of the client	RTA
component	Specifies the type of component in the experiment.	client

exec	Specifies how PNP launcher will start the client.	bin\\run.cmd RTA
hostname	Specifies the component's hostname	localhost
param	Specifies any additional parameters passed to the PNP launcher.	none

CONTROL PARAMETERS

Control parameters provide basic connection and initialization information for PNP Server. These parameters do not typically change between experiments, and are meant to be used to standardize a set of runs and ensure that some basic parameters are used across several sets of runs.

Control Parameters

Field Name	Field Description	Default Value
pnpservername	The pnpservername parameter specifies the hostname of the PNP server.	localhost
serverrequestinterval	Specifies in minutes how often the client wants the requested message data.	15

KNOB PARAMETERS

Knobs represent a series of parameters that can be adjusted to tweak the behavior of the simulation.

Knob Parameters

Field Name	Field Description	Default Value
speedcontrolspacing	Specifies whether speed control maneuvering is enabled.	true

decentprofilespacing	Specifies whether altitude change maneuvering is enabled.	true
doglegspacing	Specifies whether dogleg rerouting is enabled	true
lookaheadminutes	Specifies the look ahead in minutes for flights as candidates	120
rtalookaheadminutes	Specifies the RTA look ahead in minutes for candidates	1440
rtatoleranceseconds	Specifies the seconds of tolerance for a RTA.	5
freezerangehorizon	Specifies the freezerange horizon distance in nautical miles.	300
meteringspacingseconds	Specifies the seconds between flights at the metering fix.	45

DATASET PARAMETERS

The Dataset parameters control which files PNP uses as its source for weather and traffic data. Note that all directory definitions use the double-backslash (\\) to separate directories. Double backslashes are necessary for the proper operation of PNP.

Dataset Parameters		
Field Name	**Field Description**	**Default Value**

OUTPUT PARAMETERS

The Output parameters specify what data should be output by PNP.

Output Parameters

Field Name	Field Description	Default Value
description	A description of the experiment	RTA
outputpath	Specifies where the output data is to be written	data\\reports

DEBUG PARAMETERS

The Output parameters specify what data should be output by PNP.

Output Parameters		
Field Name	**Field Description**	**Default Value**

OUTPUTS

The RTA generates several reports that can be used for analysis and post processing by other applications. These reports are described in the following sections.

REPORTS

SUMMARY REPORT

The Summary Report is generated by the RTA at the conclusion of an experiment. This report contains the following fields:

Field	Description
Experiment	The name of the experiment
TotalDepartures	Total number of departures
PreDepartureDelayCount	The number of pre-departure reroutes
PreDepartureDelay	The total minutes of pre-departure delays
AvgPreDepartureDelayCount	Average delay per pre-departure delay
PreDepartureRerouteCount	The number of pre-departure reroutes
PreDepartureRerouteDelay	The total minutes of flight time added for pre-departure reroutes that increase flight time
PreDepartureRerouteNegativeDelay	The total minutes of flight time savings for pre-departure reroutes that reduce flight time
PreDepartureRerouteAvgDelay	Average flight time difference per pre-departure reroute
InflightRerouteCount	The number of in-flight reroutes
InflightRerouteDelay	The total minutes of flight time added for in-flight reroutes that increase flight time

InflightRerouteNegativeDelay	The total minutes of flight time savings for in-flight reroutes that reduce flight time
InflightRerouteAvgDelay	Average flight time difference per in-flight reroute
TotalDelayCount	Total number of delay events
TotalDelay	Total minutes of flight time delay
AvgDelay	Average minutes of delay per departure
Date	Date of experiment

MANEUVER REPORT

The Maneuver Report is generated by the RTMX every interval session. This report contains the following fields:

Field	Description
Experiment	The name of the experiment
DecentProfileCount	The number of speed control maneuvers.
DecentProfilePositiveDelay	The positive delay of speed control maneuvers.
DecentProfileNegativeDelay	The negative delay of speed control maneuvers.
CruiseCasCount	The number of altitude change maneuvers.
CruiseCasPositiveDelay	The positive delay of altitude change maneuvers.
CruiseCasNegativeDelay	The negative delay of altitude change maneuvers.
DogLegCount	The number of dogleg reroute maneuvers.
DogLegEnrereoutePositiveDelay	The positive delay of dogleg reroute maneuvers.
DogLegNegativeDelay	The negative delay of dogleg reroute maneuvers.

INITIAL RTA REPORT

The Initial RTA Report is generated by the RTMX every interval session. This report contains the following fields:

Field	Description
timestamp	The timestamp of the RTA session
flightIndex	Unique flight identifier assigned by PNP server flight manager.
AircraftId	Aircraft Identifier
Orig	Flight origination
Dest	Flight Destination
meteringFix	Assigned metering fix
totalflightMinutes	Total flight length in minutes
minutesFlown	Flight minutes flown
wheelsUpTime	The time when wheels are off the tarmac
initialRta	Initial RTA assignment
earliestRta	Earliest RTA assignment available for this maneuver
latestRta	Latest RTA assignment available for this maneuver
maneuver	Maneuver description
implementationTime	Time when RTA maneuver is implemented
rtaAssignment	RTA assignment selected
newEta	New Estimated time of arrival (wheels down)
rtaDelay	RTA Delay imposed by RTA assignment

etaDelay	ETA Delay imposed by RTA assignment
status	SLOTTED – RTA assignment accepted assigned INFEASIBLE_REQUEST – aircraft not in aircraft window INITIAL_RTA_FORCED_SPACING – skip this slot to create space for future needs SUCCESS - RTA assignment found NO_SPOT_AVAILABLE – RTA assignment failure due to insufficient slots

DEPARTURE SPACING REPORT

The Departure Spacing Report is generated by the RTMX to perform resolution analysis. This report contains the following fields:

Field	Description
timestamp	The timestamp of the RTA session
flightIndex	Unique flight identifier assigned by PNP server flight manager.
AircraftId	Aircraft Identifier
Orig	Flight origination
Dest	Flight Destination
meteringFix	Assigned metering fix
flightMinutes	Total flight length in minutes
spacingSeconds	Spacing seconds for origination airport
wheelsUpTime	The time when wheels are off the tarmac
rta	Initial RTA assignment
wheelsDownTime	The time when wheels are on the tarmac
delay	ETA Delay imposed by departure spacing
status	DEPARTURE_SPACING– departure spacing accepted assigned without delay DEPARTURE_FORCED_SPACING – departure spacing accepted assigned with delay

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 01- 06 - 2013	**2. REPORT TYPE** Contractor Report	**3. DATES COVERED** *(From - To)*

4. TITLE AND SUBTITLE Upgrades to the Probabilistic NAS Platform Air Traffic Simulation Software	**5a. CONTRACT NUMBER** NNL08AA17B
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Hunter, George; Boisvert, Benjamin	**5d. PROJECT NUMBER**
	5e. TASK NUMBER NNL10AC94T
	5f. WORK UNIT NUMBER 305295.02.90.07.02.01
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Langley Research Center Hampton, Virginia 23681	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001	**10. SPONSOR/MONITOR'S ACRONYM(S)** NASA
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) NASA/CR-2013-218008

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 03
Availability: NASA CASI (443) 757-5802

13. SUPPLEMENTARY NOTES

Langley Technical Monitor: Ty V. Marien

14. ABSTRACT

This document is the final report for the project entitled “Upgrades to the Probabilistic NAS Platform Air Traffic Simulation Software.” This report consists of 17 sections which document the results of the several subtasks of this effort. The Probabilistic NAS Platform (PNP) is an air operations simulation platform developed and maintained by the Saab Sensis Corporation. The improvements made to the PNP simulation include the following: an airborne distributed separation assurance capability, a required time of arrival assignment and conformance capability, and a tactical and strategic weather avoidance capability.

15. SUBJECT TERMS

Air traffic simulation; Separation assurance; Terminal area modeling; Traffic flow management

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER ***(Include area code)***
U	U	U	UU	371	(443) 757-5802