MP 06W0000053

MITRE PRODUCT

Integration Manual

August 2006

Phil Brown
David Bodoh
James Finegan
Patricia Liguori
Matthew Pollack

Sponsor: The MITRE Corporation **Contract No.:** MITRE Technology Program
Dept. No.: F053 **Project No.:** 02MSR055H3

MITRE
Center for Advanced Aviation System Development
McLean, Virginia

Abstract

This document provides a high-level perspective of the components, requirements, and procedures, for integrating air-traffic related simulation systems (such as cockpit or air-traffic controller simulators) using the AviationSimNet Specification. It is designed to assist in estimating levels of effort and to guide potential participants through the process of joining AviationSimNet.

Table of Contents

List of Figures

1 Introduction

1.1 Background

AviationSimNet[1] was conceived in 2003 by The MITRE Corporation's Center for Advanced Aviation System Development (CAASD) and implemented through MITRE's internal research and development program. Extensive experience in distributed air traffic control (ATC) simulation led CAASD to identify a need to extend simulation capabilities beyond the limitations of a single laboratory and beyond the existing pair-wise establishment of distributed simulations.

In developing AviationSimNet CAASD drew on research and advances in ATC to leverage existing simulation assets from the aviation community, industry, and academia. This is further facilitated by advancements in network and communications technologies, as well as industry standards in simulation. By connecting those assets through such technologies and standards, AviationSimNet could expedite the promotion of concepts from the laboratory to the field. AviationSimNet is a collaborative effort with industry, academia, and government agencies.

1.2 Scope of the Document

This document is designed to guide potential AviationSimNet participants through the process of joining AviationSimNet. It describes the features, processes, and functions of the various components of AviationSimNet, addressing connectivity, performance, interfacing, and security. The document omits some details about the low-level operation of various components in order to maintain a focus on guiding participants directly toward successful integration.

1.3 Intended Audience

This document is intended for an audience already familiar with the high-level purpose and function of AviationSimNet. It emphasizes the technical aspects of creating a connection to and participating in AviationSimNet. Readers who require a detailed introduction to AviationSimNet should visit www.aviationsimnet.net .

1.4 Related Documents

The following related documents, as well as additional information on AviationSimNet in general, are available online at www.aviationsimnet.net:

---

1 AviationSimNet™ is a trademark of The MITRE Corporation.

- *AviationSimNet Specification* defines the technologies required to support data and voice intercommunication in a controlled, simulated environment that models ATC simulations.
- *AviationSimNet Federation Object Model (FOM)* presents all of the data objects and attributes and their units and valid ranges.

1.5 Organization of This Document

The present document includes the following sections and appendices:

- Section 2 presents some of the important components and issues relevant to integrating a simulation system with AviationSimNet.
- Section 3 summarizes requirements and recommendations for those computer systems directly connected to an AviationSimNet Simulation Center (SimCenter) that are responsible for exchanging data with the high-level architecture (HLA) run-time infrastructure (RTI) and the voice servers.
- Section 4 outlines the various steps needed to integrate into the AviationSimNet simulation.
- Section 5 describes how to build a federate application that satisfies the AviationSimNet specifications for data communications.
- Section 6 describes the delivery mechanisms for simulated radio-voice communications in AviationSimNet.
- Appendix A discusses how to build an AviationSimNet federate.
- Appendix B provides information on Simulation Center hosting.
- Appendix C contains supplemental contact information.
- Appendix D is a glossary of acronyms.

2 Overview

This section illustrates, from a high-level perspective, some of the components and issues relevant to integrating an existing simulation system with the AviationSimNet System.

2.1 AviationSimNet System Integration Overview

AviationSimNet is a specification describing the use of existing standards to connect disparate simulation environments into a single common simulation domain. From a technical developer's perspective, AviationSimNet can also be seen as a distributed simulation bridging environment. Figure 2-1 conceptually illustrates, at a very high level, how discrete simulation environments may be linked together over the Internet using AviationSimNet. To accomplish this, an AviationSimNet Simulation Center (SimCenter; see Appendix B) hosts voice and data servers that allow different sites to connect over the Internet and share data within the same domain. Through these servers participants can exchange voice and target data without explicit knowledge of the inner workings of the other participants. Each facility is responsible for developing a bridge application that translates from the AviationSimNet domain to its own local domain.

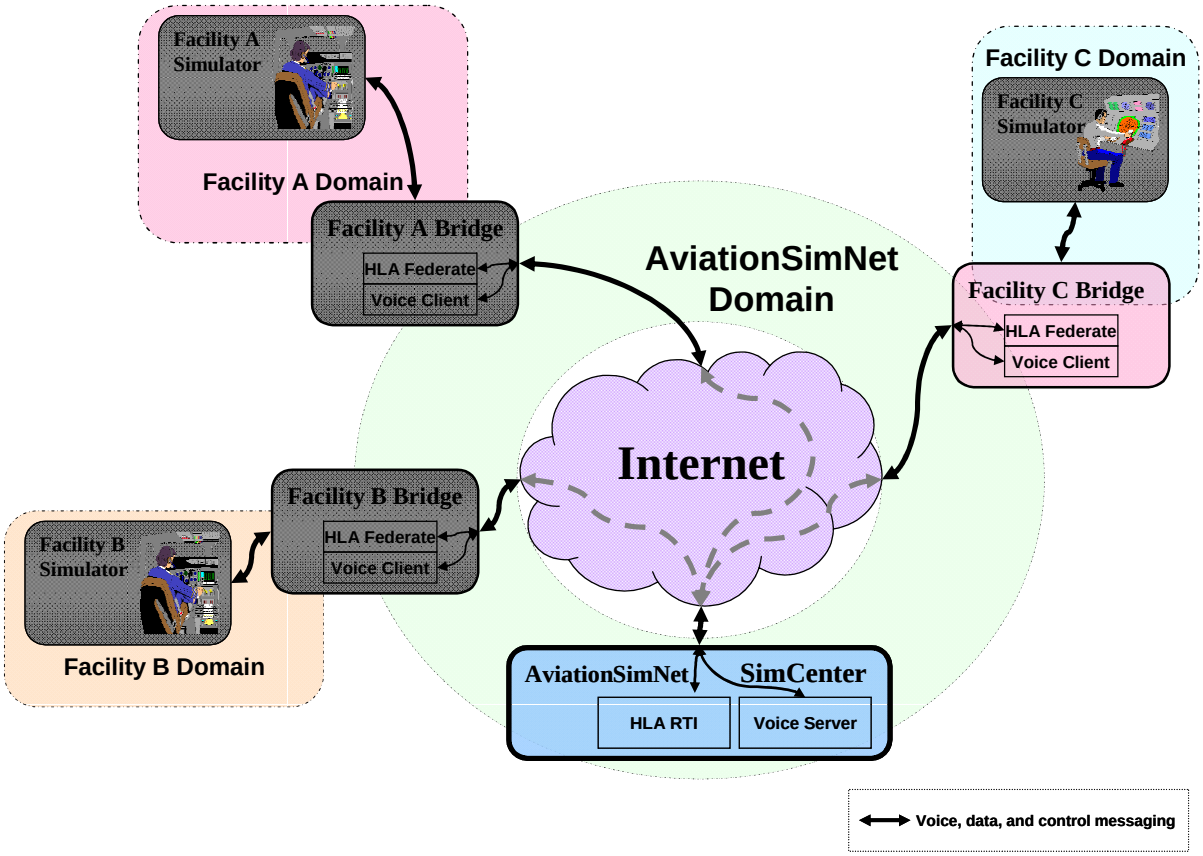

Figure 2-1. System Overview

2.2 Connectivity

Each facility participating in an experiment using AviationSimNet must have a bidirectional network path to the SimCenter. This path should have sufficient speed and capacity to support the flow of voice and target data, as well as any non-AviationSimNet network traffic with which it may have to compete for capacity. Network performance is unlikely to hamper intranet uses of AviationSimNet, since most local area networks (LANs) provide sufficient speed. However, when participants must access the SimCenter over the Internet they should give particular consideration to the quality of that connectivity. Section 2.6 discusses performance further.

2.3 Interfacing and Bridging

To operate in a shared simulation domain the participating organizations must create a bridge between AviationSimNet and the local simulation domain (see Figure 2-2). This bridge provides an interface with the local simulation and permits exchange of object data, voice streams, and control information with AviationSimNet. In essence, the bridge must be capable of speaking two protocols: that of the local environment, and that of AviationSimNet. It translates information to and from AviationSimNet to the protocols and formats defined by the local simulation environment. In creating this bridge, participants should pay close attention to the size, units, and encoding of any data items defined in the AviationSimNet FOM.

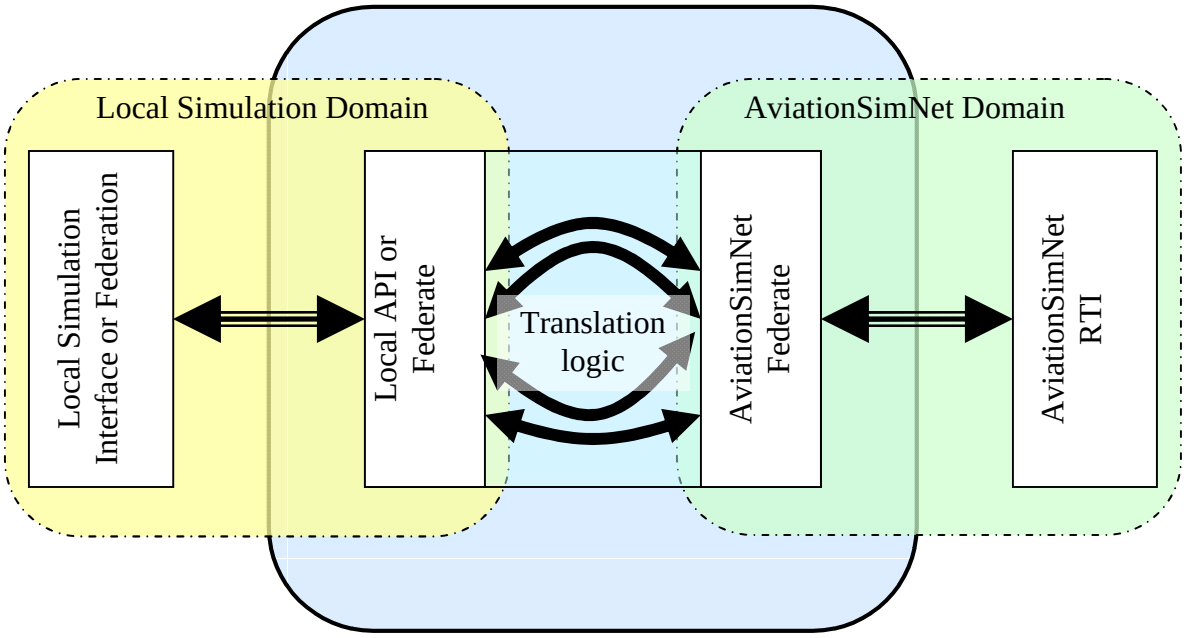

Figure 2-2. Gateway Bridging Illustration

The interface(s) with the local simulation must support access to local data and the integration of any external data such as objects created and updated by other AviationSimNet federates. This interface can take the form of:

- A software application programming interface (API) that provides read/write access to key data components.
- A local-domain federate that communicates with the local simulation (if using HLA to connect with an RTI of a non-AviationSimNet federation).

2.4 Data Communications

AviationSimNet requires the use of an HLA federation for the distribution of all target and simulation control data (see Figure 2-3). To become a part of this federation and exchange such data, participants must create a local federate that communicates with the HLA RTI. This federate subscribes to all the object classes about which it wishes to receive updates and informs the server of any data that it will publish. The *AviationSimNet FOM* provides the details of specific objects and their attributes. Additional specific information regarding data communications can be found in Section 5.

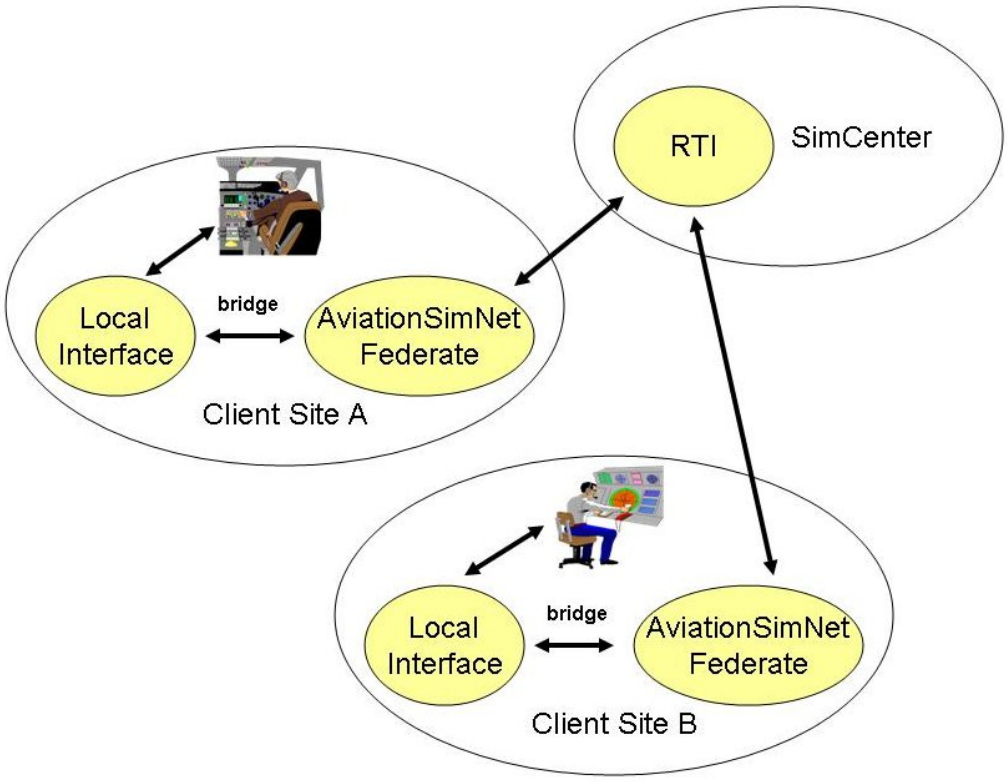

Figure 2-3. Data Communications Illustration

2.5 Voice Communications

AviationSimNet voice communications adhere to the Institute of Electrical and Electronics Engineers (IEEE) Distributed Interactive Simulation (DIS)-1278.1a standard. However, typical implementations of this standard expect communications to span only a single subnet. This alone does not satisfy the need for distributed voice architecture with participants potentially across the globe. Some means of rebroadcasting voice data from one network to another is required.

MITRE solved this problem by developing a client/server architecture for voice relay. The voice relay server (VRS) allows all participants to connect and share voice data (see Figure 2-4). Participants connect to the VRS through the voice relay client (VRC), which transfers data between the relay hub and the local subnet. Data received by the VRC from the VRS is broadcast onto the local subnet per the 1278.1a standard. Likewise, 1278.1a broadcasts are received by the VRC and relayed to the VRS where the data is distributed to other participants. The VRS and VRC are not required for voice communications as they are only one possible solution. Technical details are provided in Section 6.

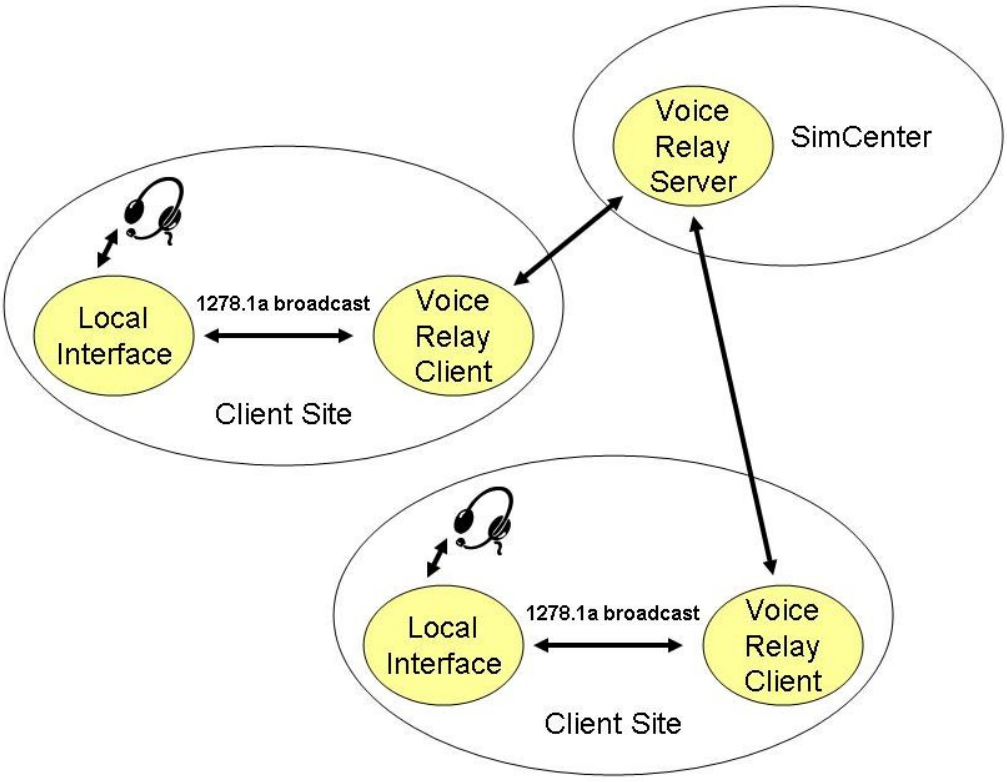

Figure 2-4. Voice Communications Illustration

2.6 Performance

Several factors can have an impact on overall simulation performance when conducting simulations over the public Internet, such as:

- The Central Processing Unit (CPU) speed and loading of:
    - The local simulation computer systems
    - The AviationSimNet federate's computer system
    - The SimCenter's computer system
- The available network bandwidth (relative to the specified connection speed)
- The efficiency of software design and implementation
- The number of simulation objects being modeled and shared

The varying impact of these factors makes it extremely difficult to accurately predict the peak capability of any particular distributed simulation configuration at any particular time without taking direct measurements. However, it is reasonable to establish nominal performance expectations based upon what is known about the connection specifications and some general assumptions.

The greatest limiting factor for experiments using AviationSimNet over the Internet will be bandwidth. Insufficient bandwidth can severely impact the operation of the distributed simulation over the Internet.

2.6.1 Achievable Performance

The best measure of performance is the number of simulation objects that can successfully be supported during an experiment, such as the number of aircraft targets propagated and voice channels in use. Figure 2-5 shows the theoretical limits for a given bandwidth and number of voice channels. The results in the graph are based upon the following assumptions:

- No more than 80% of the maximum bandwidth is available.
    - This could be less than 50% for non-dedicated network connections.
- Target packet size is estimated at around 100 bytes (800 bits).
- Targets are updated once per second (0.8 kbits/sec per target).
- Each voice channel uses about 64Kbps.

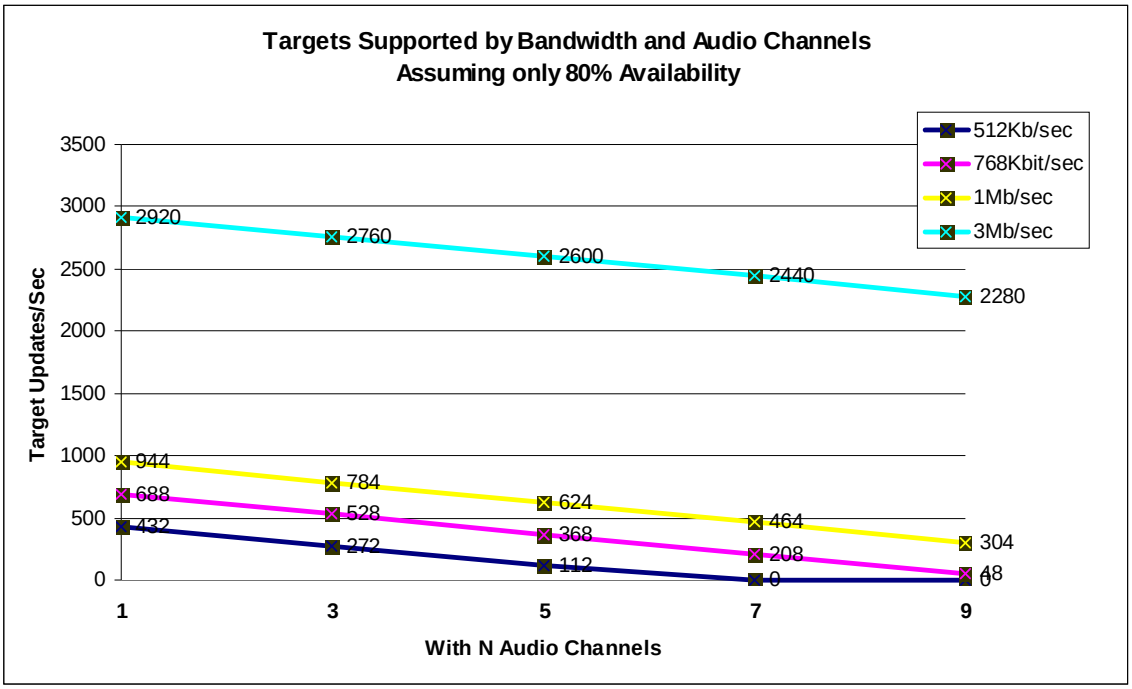

Figure 2-5. Potential Number of Targets Given Bandwidth Specifications

System and network performance characteristics should be carefully planned as early in the integration process as possible to ensure that sufficient resources will be available for an experiment. Therefore, given the results shown in Figure 2-5, if an experiment with 200 targets and 3 voice channels is desired, then a connection speed of 512 kb/sec may be sufficient. If the experiment uses a non-dedicated connection (that is, one potentially shared with other network traffic) then it may be necessary to have a higher rated connection.

For example, a site with a T1 (rated at around 1.5Mbps) connection to the internet that is shared by others at the site, could comfortably expect to support an experiment with as many as 700 active targets and three radio channels simultaneously in use. This assumes that each target is updated once per second. The actual required bandwidth is approximately 752 Kbps. To calculate peak required bandwidth, use the following formula:

Bandwidth = num_aircraft * 0.8Kbps + num_radio_channels * 64Kbps

MITRE has a performance assessment tool, called the AviationSimNet Performance Tester, which can be used to calculate the actual achievable throughput along the path from a participant's site to the RTIExec and back. This tool is available via the *AviationSimNet Software License Agreement.* Contact your IT department or Internet service provider (ISP) for more information on your connectivity performance statistics.

3 System Requirements and Recommendations

This section describes requirements and recommendations for those computer systems that are directly connected to an AviationSimNet SimCenter and are responsible for exchanging data with the RTI and voice servers.

3.1 Computer Systems

The following system configurations are requirements for systems running applications that are directly participating with AviationSimNet. This includes AviationSimNet federates and the VRS and VRC voice applications.

- Linux or Windows PC
- 2GHz processor
- 512Mb RAM

There are no other specific hardware requirements beyond these listed.

3.2 Software Components

3.2.1 High-Level Architecture Run Time Infrastructure

There are currently several verified HLA implementations available on the market; however, AviationSimNet has currently selected the MÄK RTI from MÄK Industries (www.mak.com), since it provides functionality conducive to a distributive architecture traversing the Internet. Specifically, its performance characteristics and those of the RTIexec allow federates to initiate the network connections. Contact MÄK Industries (refer to Section 4.3) to request an evaluation copy or to purchase a license.

Internally, a customer may use whichever simulation environment is appropriate for their needs, but it will be necessary to create a bridge from the internal environment to that of AviationSimNet.

3.2.2 AviationSimNet Voice Relay Software

MITRE has developed a voice data packet relay system that bridges 1278.1a protocols across multiple networks and allows participants to communicate with each other. This product is available as part of the *AviationSimNet Software License Agreement.*

3.2.3 AviationSimNet Performance Test Software (optional)

MITRE has developed some performance testing software that can verify the HLA connectivity and perform statistical throughput testing with the RTI. This product is available as part of the *AviationSimNet Software License Agreement.*

3.3 Networking

AviationSimNet exchanges data over a network, either by Internet or intranet. Therefore, a strong recommendation for becoming a part of AviationSimNet is that participants have a high-quality network connection to the data and voice servers. The quality of the connection can become a limiting factor for the types of simulations that may be performed, primarily because of the need to quickly exchange large amounts of data.

3.3.1 Bandwidth and Performance

To increase the possibilities of sharing capabilities the following minimum Internet connection speed ratings are highly recommended for a facility:

- 1 Mb/sec downlink speed
- 256 Kb/sec uplink (some Internet connections have different speeds for downlink and uplink)

A SimCenter has higher bandwidth requirements so the following minimums are highly recommended:

- 3 Mb/sec downlink speed
- 1 Mb/sec uplink speed

Actual bandwidth required is a function of the needs of any particular experiment, and participants should give special consideration to this issue during the planning phase. For example, a T1 Internet connection is rated at around 1.5Mb/sec.

3.4 Security Guidelines and Recommendations

Any computer systems that connect via the Internet are potentially exposed to a variety of security threats. However, common measures to secure the systems can be used to reduce the risks associated with a malicious attack. For example, participants can implement many of the system hardening recommendations made by the Center for Internet Security (www.cisecurity.org).

The following are a few examples where common hardening techniques have been successfully applied under AviationSimNet:

- Disable unnecessary services and daemons
- Lock system accounts
- Enable the integrated firewall and access control utilities
- Maximize file protections
- Employ intrusion prevention tools and techniques
- Perform backups

Participants should consult with their IT or network security professional to ensure that adequate steps have been taken to address the issues outlined in this section. Organizations such as CERT® (www.cert.org) have an abundance of information regarding these issues; thus, they are not addressed in this document.

4 Integration Process

This section describes the various steps needed to integrate into the AviationSimNet simulation environment.

4.1 Preparation

4.1.1 Identify Assets

AviationSimNet encompasses distributed human-in-the-loop ATC simulations. They can include a wide range of assets such as:

- Medium-fidelity flight simulators
- Simulated controller Display System Replacement (DSR) stations
- Software-based pilot stations
- Simulation management interfaces

AviationSimNet can be applied to a great many more types of simulation applications, such as tower simulators, airline operations centers, flight dispatcher stations, weather services, and others. Any asset that can participate in an ATC simulation environment is a candidate for integration with AviationSimNet.

4.1.2 Simulation Compatibility

Integrating an asset into AviationSimNet involves varying degrees of complexity, depending on the particular application. The specification supports constructs for conducting a simulation over a wide-area network. Certain activities, such as initialization and shutdown, have protocols to make those activities flow seamlessly. However, adhering to those requirements can be challenging if the assets are not equipped with the means to respond to external events that control a simulation. At a minimum, any asset will need a software interface that allows event and state information to pass between it and some remote process. Without that interface, the asset has no means to participate with other assets in an AviationSimNet simulation.

To assess the integration complexity, prospective users should determine how well each asset supports the following activities:

- Report to an external process that it is ready to commence a simulation.
- Suspend its initialization activities until some external process gives a signal to proceed.
- Instantly suspend and resume run-time activities upon receipt of a signal from an external process.
- Exchange target state information with external processes.

If the asset can satisfy these fundamental activities, then it is equipped to satisfy the AviationSimNet requirements for participating in a simulation. Unless all of these activities are supported any experiment using this asset may experience significant limitations.

4.2 Data Communications Bridge/Gateway

Even if the simulation assets are participating on a network that can access the RTI server, and can respond to external simulation control signals and share their internal simulation state information with other applications, they still need a way to translate those signals and messages between their representation and the HLA object model representation. This translation role is typically provided through a simulation gateway at the participating organization, especially if the simulation components existed prior to AviationSimNet. Gateways allow the asset's internal implementation to remain entirely or nearly unchanged, while relaying AviationSimNet messages to and from that internal implementation. Any gateway must be built explicitly to interface with the systems it is bridging.

Some simulation assets may provide developer interfaces to send and receive information directly to and from the internal model. This is particularly true for simulations that are software based or were developed locally. Other assets may not provide direct access, but have custom-designed protocols for communicating with external processes. In either case, an AviationSimNet participant may prefer to leave this asset unmodified, but still connect it to the AviationSimNet voice and data networks. This can be accomplished through the use of a bridge or gateway application.

In data communications, a bridge federate participates in both AviationSimNet protocols and local protocols understood by the asset. It translates events and messages in both directions, brokering the asset's participation in AviationSimNet. In fact, a bridge federate may perform brokering activities for multiple assets, but participate in an AviationSimNet federation as a single federate. Some example bridge federate scenarios are described below.

4.2.1 Example: Bridging an Air Traffic Management Lab to AviationSimNet

Consider a laboratory that conducts research and development in air traffic management and has a distributed communications architecture for conducting simulations across multiple computers on a single network. This architecture includes a simulation control application that ensures all other applications are properly connected. It also has a user interface that accepts commands from an operator to send *Start*, *Pause*, *Resume*, and *Shutdown* directives to all the participating applications. In addition, the architecture maintains a distributed database of aircraft state information for every simulated aircraft declared to exist.

A gateway represents the AviationSimNet federation manager, as defined in the specification. It interfaces with the simulation control application to relay the simulation operator's commands to the AviationSimNet system. Those commands are translated into HLA interactions for *StartSimTime*, *Resume*, *Pause*, and *Shutdown*. Other AviationSimNet federates receive those messages when the laboratory operator enters them into the interface. Likewise, the gateway

prevent the rest of the laboratory applications from starting the clock until all the other AviationSimNet federates have properly joined and are prepared to proceed. Furthermore, the gateway links with the laboratory's distributed aircraft database and can publish/subscribe select elements of the aircraft objects to the rest of the AviationSimNet federation.

4.2.2 Example: Bridging a Flight Simulator to the AviationSimNet Network of Simulations

Consider a flight simulator equipped with avionics hardware components that communicate with a flight management system (FMS) computer. Those components use an ARINC 429 databus to transmit and receive information about the state of the simulator while in operation.

A gateway built from hardware and software components interfaces with the FMS via the ARINC 429 databus to relay the state of the simulator to an AviationSimNet simulation. The hardware consists of an ARINC 429 receiver peripheral component interconnect (PCI) card installed on a separate computer. The software consists of drivers and libraries to interface with that PCI card and libraries for accessing the local RTI component (LRC), both linked within the same program.

Assuming the federation manager is somewhere else in the federation, the simulator's gateway application relays simulation commands from the manager to the simulator's operator, reflecting *Pause* and *Resume* activities. Similarly, the gateway relays the current state of the simulator to the rest of the federation by creating an aircraft object instance and updating the attributes of that aircraft as the data are received through the ARINC 429 card from the FMS.

4.2.3 Example: Bridging a Composite Federate to the AviationSimNet Network of Simulations

Consider an existing HLA federation composed of several simulated software pilot stations. The FOM for that federation has some objects for aircraft state and events. It is a different FOM from the AviationSimNet object model.

A gateway application is constructed to participate as a federate in both federations. Its tasks include:

- Relaying discovered objects from one federation by creating a corresponding object in the other
- Relaying attribute reflections from one federation as attribute updates in the other
- Forwarding interaction messages received from one federation by sending the corresponding message to the other.

The gateway's greatest challenge is to find the appropriate mapping between objects, attributes, and interactions between the two object models.

4.3 Voice Communications Bridge/Gateway

The standard for digital voice communications in AviationSimNet is the IEEE DIS-1278.1a. According to this standard, the local network receives broadcast packets of voice data as specified. In situations where the local simulation environment does not support this standard, some additional work may be needed to either adapt to this standard or to convert to the local format. In these situations the bridging may be as simple as repackaging the encoded voice data portion of the 1278.1a packet consistent with the protocol of the local simulation. However, it could also require recoding of the voice payload to the format used by the local simulation. Specific details of these types of conversions are beyond the scope of this document.

4.4 Sample Integration Plan

The following sample integration plan illustrates some important steps for successful integration:

1. Procure all required hardware and software:
    a. MÄK HLA libraries and licenses for building federates from MÄK
    b. VRC from MITRE
    c. Software example federates from MITRE. (optional, code examples, tests)
2. Establish and ensure Internet/network connectivity:
    a. Ensure firewall integrity and access through the participant's IT professional
    b. Ensure sufficient bandwidth
3. Develop data interfaces to local simulator providing access for:
    a. Simulator position/attitude data of locally modeled targets (if any)
    b. External target data injection, such as that produced by other participants.
4. Develop audio interfaces to controllers and/or pilots stations that allow:
    a. Channel selection
    b. Push to talk
    c. IEEE DIS-1278.1a standards conformance.
5. Develop federate(s) consistent with AviationSimNet Specification that:
    a. Interact with simulator data interface (above)
    b. Are capable of publishing and subscribing to target data to RTI
    c. Are capable of synchronizing start/stop with AviationSimNet Simulation.
6. Create data and voice bridges, as needed, from the local environment to RTI
7. Conduct an integration test with a SimCenter, either local or MITRE/CAASD's

5 Data Communications

This section describes the roles and responsibilities of an AviationSimNet federate.

5.1 HLA Interface

5.1.1 HLA Runtime Infrastructure

The *AviationSimNet Specification* describes a transmission control protocol (TCP) forwarder topology where a centralized server receives connections initiated by clients, after which bidirectional communication is possible. This topology is a requirement for the voice and data servers of AviationSimNet. Thus, the RTI implementation of network connections must satisfy the *AviationSimNet Specification*. However, as the HLA specification does not mandate how connections are established, the details are left to the RTI implementer.

Of all the commercial RTI implementations available, one that effectively supports that topology is the MÄK RTI. The MÄK RTI is a collection of software libraries and applications that can be configured to satisfy the HLA specification, and to support a multiple-client/single-server topology such as AviationSimNet. MÄK RTI includes a server process called the RTIexec, which can be configured to listen on a specific network port and act as a TCP forwarder for client connections.

On the client side, applications link with RTI libraries such that each application has its own Local RTI Component (LRC). The entire MÄK RTI implementation for AviationSimNet consists of multiple LRCs (one for each federate) and the RTIexec. If network access is available, each LRC establishes a TCP/Internet Protocol (IP) connection to the RTIexec, and the RTIexec coordinates bidirectional message forwarding among those socket connections. There are no direct LRC-to-LRC network connections, and all connections are established only from each LRC to the RTIexec server. Once those connections are established, data flow and communications can be conducted bi-directionally. This may be less efficient than a fully meshed network topology, but it is a more secure implementation.

The MÄK RTI is configured through a set of RTI initialization data (RID) parameters. Each federate and the RTIexec has its own set of RID parameters collected in a rid.mtl file. In order for the federate's LRC to know how to connect to the RTIexec, the rid.mtl file must have the following settings:

(setqb RTI_useRtiExec 1)

(setqb RTI_udpPort *12345*)

(setqb RTI_internalMsgReliable 1)

(setqb RTI_fomDataReliable 1)

(setqb RT_tcpForwarderAddr "*123.123.123.123*")

Note that the udpPort and the TCP forwarder address represent the host and port where the RTIexec is running. Actual values may differ from the example above, and would be established prior to any multi-organizational federation execution. Other parameters in the RID file can be adjusted according to the specific needs of the individual federate or the federation as a whole, and should not interfere with the RTI's conformance to the AviationSimNet specification.

When executing an AviationSimNet federation among multiple organizations, the RTIexec host and port must be accessible from each federate. If the federation is local to a single LAN, the host and port need only be accessible to federates on that LAN.

The MÄK RTI does not require a runtime license to execute the RTIexec process. However, the organization responsible for setting up the RTIexec must have purchased a runtime license in order to have access to certain MÄK shared libraries. The RTIexec only needs a rid.mtl file, but the LD_LIBRARY_PATH must be able to locate the MÄK shared libraries.

5.1.2 Simulation States

The *AviationSimNet Specification* describes a set of federate states that reflect the set of available actions a federate can perform during certain stages of the simulation. Some of those states represent the series of synchronization points that are coordinated during the initialization stage (pre-clock start). The other states represent the modes a federate can be in once the clock starts.

Federate developers are not required to create a state variable that tracks the events controlling the federate as it moves between these states. However, it may be helpful to identify the state of a federate at any time to diagnose problems or odd behavior.

The states for Initialization, Declaration, and Population are used to distinguish which AviationSimNet synchronization points have been achieved. The RTI manages synchronization across a federation at those points to prevent any federate from advancing beyond the readiness of others. The synchronization points are:

- *ReadyToDeclare*: a gate indicating that once all federates have achieved that point, they can be certain that it is safe to start declaring publication and subscription interests with the RTI.
- *ReadyToPopulate*: a gate indicating that once all federates have achieved that point, they can be certain that it is then safe to start creating and updating objects available in the FOM. By following these gates, the federates guarantee that they are not missing any updates.
- *ReadyToRun*: a gate that indicates when all federates are ready to start advancing time. The federation time keeper (described in the *AviationSimNet Specification*) relies especially strongly on this synchronization point, because it requires the assurance that all federates are ready to start advancing the simulation clock before the first clock Resume event is made. The *StartSimTime* interaction must be received prior to proceeding to this point.

Once the clock is started, federates alternate between paused and unpaused states according to the *Resume* and *Pause* events, as published by the time keeper. All federates should be prepared to receive a *Shutdown* interaction no matter what state they are in. That interaction consists of an instruction to stop the simulation activities immediately and resign from the federation. After a *Shutdown* interaction no further processing of data flowing to or from the AviationSimNet federation is allowed, and any attempt to continue processing will likely be ignored by the other federates. Once the federate does resign from the federation, it can continue to process any state information locally.

5.1.3 Producer Responsibilities

AviationSimNet federates will be producers of data, consumers of data, or both. Aircraft target information provides an example. As a producer of target information, the federate is first responsible for declaring its ability to publish target information to the RTI. This enables the federate later to register actual instances of aircraft targets. Declaration of aircraft target publication can happen anytime after the federation is synchronized at the *ReadyToDeclare* synchronization. The federate is responsible for creating an instance of each aircraft target it generates and registering it with the RTI. Registration can happen anytime after the federation is synchronized at the *ReadyToPopulate* synchronization point. The federate can then proceed to register and update any aircraft targets it wishes to share with the rest of the AviationSimNet federation.

According to the HLA specification, the registering federate has the option either to specify an instance name for the object it creates with the RTI or to let the RTI assign a name. Because the flight identification (ID) of each aircraft in the simulation must be unique, providing the flight ID as the object ID is a safe practice in AviationSimNet.

According to the AviationSimNet specification, the federate must remain the owner of all target attributes as long as it publishes that target. If the federate decides to relinquish ownership of the target, it must give up ownership of all the aircraft attributes, so that another federate can take ownership of the entire set. Once the federate determines that a target it owns will no longer be updated in the simulation, it can destroy the instance with the RTI.

If a federate is designated as the federation manager according to the AviationSimNet specification, it also has the responsibility of publishing simulation control interactions: *Pause*, *Resume*, and *Shutdown*. The manager must, at a minimum, send a *Resume* interaction once to start the clock, and a *Shutdown* interaction to stop the clock. Any number of *Pause*/*Resume* events can occur before sending the *Shutdown* interaction.

5.1.4 Consumer Responsibilities

A federate can also behave as a consumer of target information. Upon synchronization at *ReadyToDeclare*, a federate may register its interest as a subscriber of some or all aircraft attributes. Once the federation is synchronized at *ReadyToPopulate*, a federate must be prepared to start discovering aircraft objects and reflecting their attribute updates, as published by other

federates. Federates that are both consumers and producers of aircraft targets must be able to maintain a distinction between the two types of targets, so that its internal model does not accidentally attempt to register an instance of an aircraft target that is already being updated by another federate in the simulation.

Regardless of their interest in aircraft target information, all federates must subscribe to two simulation control interaction classes: *Resume* and *Shutdown*. Federates need only subscribe to the *Pause* interaction if they are capable of suspending internal state at the receipt of such an interaction from the federation manager. If the federate does not subscribe to the Pause interaction, the manager can query the RTI to identify that condition and use that knowledge in deciding whether to send a Pause or not.

6 Voice Communications

This section describes the delivery mechanisms for simulated radio-voice communications in AviationSimNet. Figure 6-1 illustrates how voice data is propagated from the radio headsets in one facility to the headsets at another facility through the voice relay server at the SimCenter.

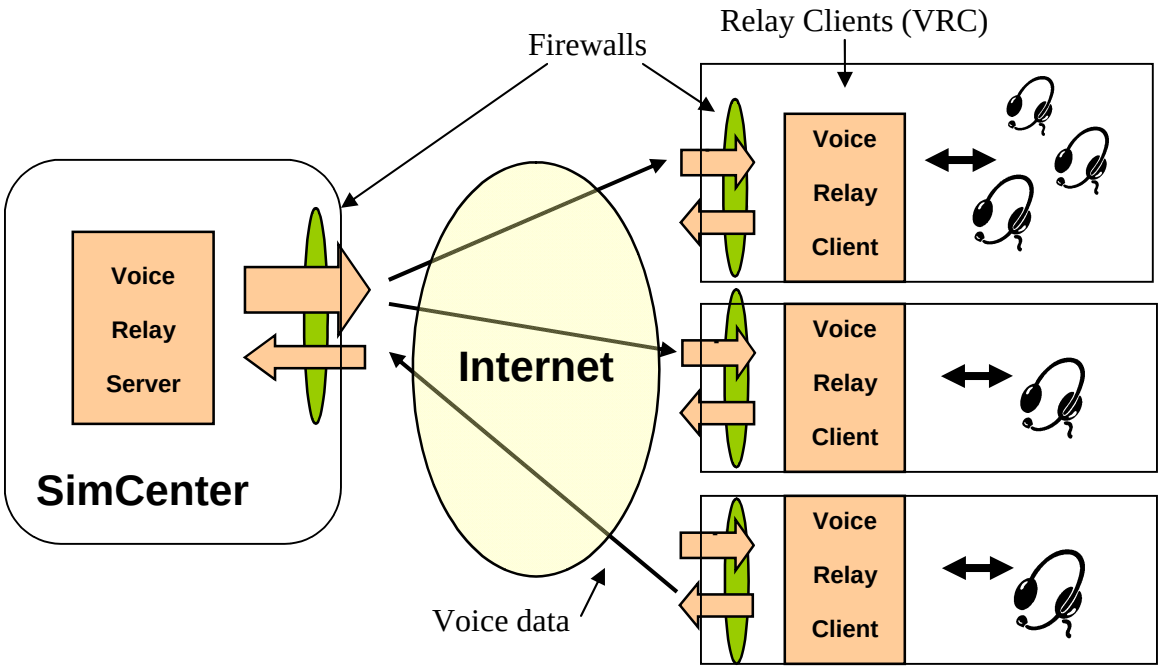

Figure 6-1. Voice Communications

6.1 Voice Architecture

The IEEE DIS-1278.1a standard describes the packaging of simulated radio receiver and transmitter data. In most configurations, DIS radio networks are set up as LANs. DIS radio transmitters convert audio data into digitized packets, potentially compress the data according to one of a few encoding schemes such as mu-Law and the Pulse Code Modulation (PCM), and then send the digitized audio along with additional identifying information to the local subnet as a broadcast. DIS radio receivers get the transmitted audio data and determine if the data should be decompressed and decrypted on the basis of transmission frequency or location. If the various parameters are correct, the audio data is converted back into an analog signal that can be sent to a computer's sound card.

MITRE/CAASD provides AviationSimNet data distribution software, the VRC, that listens for DIS audio broadcast messages on the local subnet and sends them to a relay server located on a publicly accessible network. The relay server distributes these messages to all other AviationSimNet participants. Upon receipt of a message from the relay server, the local

distribution client broadcasts the message onto the local subnet for the DIS receivers to pick up. This, in effect, allows multiple DIS radios to send and receive data over a wide area network.

6.2 Voice Configuration

Before an exercise starts, participants must agree on several configuration options. The following configurable parameters, taken from the *DIS 1278.1a Specification*, must be coordinated with all participating organizations:

- Exercise ID
- Site ID
- Entity ID
- Antenna position
- Codec (voice data encoding technique)
- Frequencies

The choice of network bridging systems will directly affect the integration effort. Use of the MITRE-built voice-relay software requires only a simple client application (the VRC) running on a Linux host on the participant's LAN. This host must have Internet access to the VRS at the SimCenter. Any firewalls between the host and the Internet must be configured to allow outbound traffic.

6.2.1 Data Rates

The IEEE DIS-1278.1a specification provides for configurable encoding schemes and sample rates. Undoubtedly, the highest fidelity audio comes with a sample rate no less than 8,000 samples/sec, and an uncompressed encoding such as 16-bit PCM. However, it has been found that a sample rate of 8,000 samples/sec with a 2:1 or 4:1 compression ratio, such as mu-Law or adaptive differential PCM (ADPCM), is also acceptable. Compressions beyond that tend to affect voice quality. Technically, an outbound bandwidth allowance of 64kbits/sec should be made for each connected radio. However, in practice there is generally only one radio per frequency in use at any particular time, as speakers tend to take turns while communicating.

6.3 Voice Hardware

The nature of audio communications systems makes it impractical to provide blanket guidelines for the connectivity between a participant laboratory and AviationSimNet. Each participant's current audio configuration will drive the integration effort.

For example, assume that Participant A currently has a laboratory with an analog communications system consisting of a set of headsets and speakers linked by a proprietary switching solution. For this participant to communicate in AviationSimNet, signals on this analog audio system must be digitized and the resultant digital audio must be formatted in a manner consistent with IEEE DIS-1278.1a. The participant can take two approaches from this decision point: first, a commercial off-

the-shelf (COTS) solution may either replace or be integrated with the current system; or second, a custom solution can be designed and implemented. Once the audio is presented as IEEE DIS-1278.1a-compliant data, the audio can be sent to the AviationSimNet audio server. It is important to be aware of potential problems, such as impedance matching and phantom power requirements, when connecting aviation headsets to a digitizing system.

6.4 Voice Software

As with the audio hardware, the audio software that would be responsible for converting analog audio into IEEE DIS-1278.1a-compliant data may vary. The software system setup will depend upon COTS vendor selection and/or internal system design and development.

6.4.1 Voice Relay Software

MITRE/CAASD provides the VRC, which is used to connect a local IEEE DIS-1278.1a-compliant voice system to the VRS run at the SimCenter. The client listens for broadcast voice packets on a specified port on the local network. Upon receipt of those packets it packages and sends the message to the server. The server, in turn, distributes this packet to all other connected clients. Command-line arguments to the client allow the server location (IP address and port number) and local port number to be specified. The client also provides some statistical feedback within the window in which it is run to indicate activity. Additional details about the use of this tool are provided with the software distributed under the *AviationSimNet Software License Agreement.*

Appendix A Building an AviationSimNet Federate

A software toolkit that encapsulates much of the functionality of an HLA federate is available to potential participants under the *AviationSimNet Software License Agreement*. This toolkit contains a set of C++ header files (for inclusion into a federate application), C++ source files, an archive library (to be linked into an executable), configuration files, and some supporting documentation. The document *Software Support for AviationSimNet Compatible Applications*, bundled with the software toolkit, gives some coding examples for creating an AviationSimNet federate using this toolkit.

In general, a federate has two major interface classes (called ambassadors) for operating in a federation: the RTI ambassador (*rtiAmb*), and the federate ambassador (*fedAmb*). The rtiAmb is an object that provides functionality for sending information or making queries to the RTI. The fedAmb is an object that is used by the rtiAmb to pass data received from the RTI to the federate. The data is received by the federate as a set of callback handlers. Figure A-1 illustrates how the rtiAmb communicates with the RTI through the LRC (described in Section 5.1.1) by calls, such as to *publish* or *update* a target, made with the rtiAmb. Likewise, the fedAmb receives callbacks from the RTI through the LRC, such as target updates from another federate.

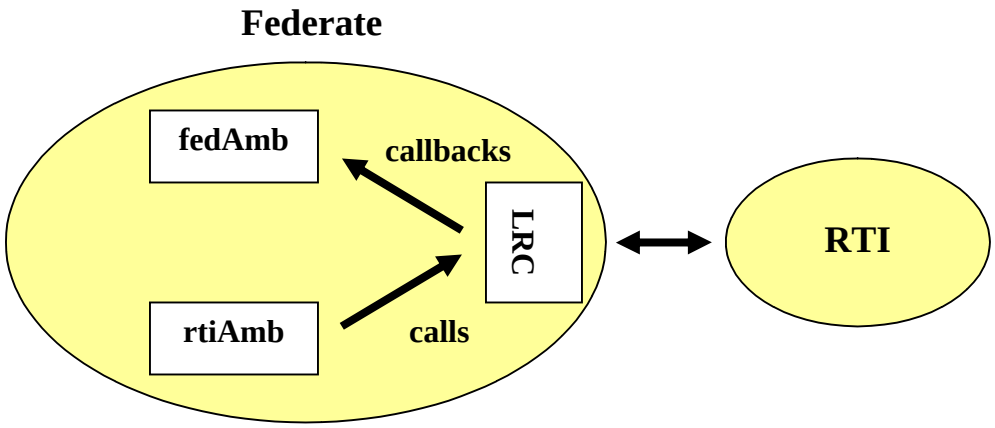

Figure A-1. Illustration of the RTI and Federate Ambassadors

The functions in the rtiAmb are defined and implemented as part of the HLA interface. By contrast, the developer of the federate must implement the callback functions of the fedAmb.

Some example rtiAmb calls are:

- *createFederationExecution()* – establish the federation if it does not already exist.
- *joinFederationExecution()* – join or connect to the federation.
- *resignFederationExectuion()* – shut down the federation.

- *subscribeObjectClassAttributes()* – indicate that the participant wishes to receive updates for the specified attributes all objects of a specified class.
- *publishObjectClass()* – indicate that the participant will be publishing/creating objects of the specified type, such as aircraft targets.
- *registerObjectInstance()* – create an instance of an object, e.g., create a new aircraft.
- *updateAttributeValues()* – update the values of the object's attributes.
- *deleteObjectInstance()* – delete an object from the federation, e.g., delete an existing aircraft.
- *tick()* – very important function that calls an event checking loop in the rtiAmb. Events such as new objects, updated objects, interactions and synchronization points are all received (when present) as a result of this call. Additionally, callbacks made to the fedAmb are initiated exclusively through this call, thus it should be called periodically, such as once every second.

This document does not include details about the parameters of the function calls, but those details may be found in the *HLA Specification*.

Some example callbacks made to the fedAmb are:

- *receiveInteraction*() – called when the RTI receives an interaction created by another federate.
- *discoverObjectInstance*() – called when a federate creates a new object, such as when another federate calls *registerObjectInstance()*.
- *reflectAttributeValues()* – called when an existing object has one or more of its attributes updated, such as when another federate calls *updateAttributeValues()*.
- *removeObjectInstance()* – called when an object is deleted, such as when another federate calls *deleteObjectInstance()*.

Appendix B AviationSimNet SimCenter Hosting

The ability to host an AviationSimNet experiment is not exclusive to MITRE. This section describes the components and procedures for hosting a SimCenter within a LAN, as well as over the Internet.

Components of a SimCenter include, but are not limited to:

- MÄK RTIexec – server execution component for data communications (see Section 5.1.1).
- MITRE's VRS (or equivalent service) – for distribution of voice data to participants (see Section 6.4).

Internet-based SimCenters, those with participants external to the LAN, need to take into account the following special considerations:

- Firewall configuration – to allow connections to the RTIexec and VRS while preventing undesired connections.
- Virus protection software – to ensure the safety of the SimCenter and the systems of all connecting participants.

The following procedures illustrate typical activities necessary to prepare a SimCenter for hosting:

- Follow security guidelines outlines in Section 3.5.
- Acquire components listed above and become familiar with their operation.
- Select network port addresses for the RTI and voice servers.
- Communicate configuration settings (port address and server host IP address) to participants so that their federates and voice software can reach the servers.
- Configure the RTIexec via the rid.mtl file.
- Start RTI and voice servers.
- Locally test connectivity and function of servers to ensure basic operation.
- If possible, coordinate a test with any potential external participants to ensure connectivity and basic operation.

Appendix C Supplemental Contact Information

HLA RTI Vendors:

MÄK Technologies

68 Moulton Street - Cambridge, MA 02138
Phone: 617.876.8085
Fax: 617.876.9208
info@mak.com

Voice solution vendors:

SimPhonics, Inc.
3226 N. Falkenburg Road - Tampa, FL 33619

Phone: (877) 205-4901

Fax: (813) 623-5119

info@simphonics.com

Appendix D Glossary

ADPCM	adaptive differential pulse code modulation
API	application programming interface
ATC	air traffic control
CAASD	Center for Advanced Aviation System Development
COTS	commercial off-the-shelf
CPU	central processing unit
DIS	distributed interactive simulation
DSR	display side replacement
FMS	flight management system
FOM	federation object model
HLA	High-level architecture
ID	identification
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
KBPS	kilobits per second
LAN	local area network
LRC	local RTI component
MBPS	megabits per second
PCI	peripheral component interconnect
PCM	pulse code modulation
RID	RTI initialization data
RTI	run-time infrastructure
TCP	transmission control protocol
UDP	user datagram protocol
VRC	voice relay client
VRS	voice relay server