APPLICABILITY OF MOBILE AGENTS TO QUALITY OF SERVICE MANAGEMENT FOR MILITARY DIGITAL AVIONICS SYSTEMS

1Lt Jason T. Lawson, Charles Satterthwaite, Air Force Research Laboratory, Wright-Patterson AFB, OH

Abstract

Based on current Internet trends, mobile agents have been envisioned as a solution for future technical challenges such as information overload, mobile devices and users, and user-driven customization. A potential military application of mobile agents deals with providing increasingly complex functionality with acceptable levels of quality of service (QoS) for digital avionics systems. Providing increasingly complex functionality with acceptable levels of QoS for digital avionics systems is often difficult, because these systems are required to compute and communicate within hard real-time deadlines and operate in highly resource-constrained environments. To begin addressing this issue, an existing agent-based application is presented as a partial solution to these challenges. This application, known as the Guardian Agent, is designed to act as an off-board surrogate for digital avionics systems that reside on a multitude of tactical assets. Hence, the computational load associated with new complex functionality, such as data aggregation, filtering, and fusion, is shifted to a host node that is far less resource constrained compared to the tactical asset.

Given this assumption, consideration needs to be afforded to the allocation, execution, and management of the Guardian Agents on the associated host node. The thesis of this paper is that incorporating mobility into the Guardian Agent application, along with a QoS management framework on associated host nodes, will increase the scalability of the Guardian Agent application. Specifically, these enhancements will enable the migration of Guardian Agents to potential host nodes that can provide the desired complex functionality with acceptable levels of QoS. Benefits from this enhanced application include implicit load balancing across potential host nodes, improved real-time performance of the Guardian Agent, and user-driven customization based on QoS and other real-time constraints. In addition, operational requirements such as the dynamic integration of tactical assets, systems-of-systems fault tolerance, and efficient use of tactical datalink bandwidth, will also drive the need for mobility of the Guardian Agent.

1.0 Introduction

Providing increasingly complex functionality with acceptable levels of QoS for military digital avionics systems has been a difficult problem to solve due to operational requirements for real-time computing and communication via constrained resources, such as embedded processors and datalinks. Typically, computation and communication resources that exist on tactical assets are maximized to their limits, just to provide mission-essential functionality. Thus, adding additional on-board functionality is often not a feasible option for increasing the military effectiveness of these existing tactical assets. Regardless of these technical limitations, new requirements are continually being thrust upon future and legacy military digital avionics systems.

For example, initiatives such as the Joint Battlespace Infosphere (JBI) [22] envision the creation, processing, and sharing of information in real-time across a multitude of strategic assets and tactical military digital avionics systems. These new requirements can generally be classified as a combination of hard and soft real-time processing, which is governed by a variety of QoS constraints that define the timeliness requirements for specific computations or communication [9]. As such, being able to provide acceptable levels of QoS for this new increasingly complex functionality will likely have a tremendous impact on battlespace effectiveness of tactical assets.

A potential solution to this problem, known as the Guardian Agent application [4], has been developed under the Insertion of Embedded

U.S. Government work not protected by U.S. Copyright

5.D.5-1
Infosphere Support Technology (IEIST) initiative. The IEIST initiative is funded by the Air Force Research Laboratory (AFRL), and is being implemented by the Boeing Company using the Bold Stroke open avionics software architecture. Specifically, the Guardian Agent acts as a surrogate for digital avionics systems that reside on a multitude of tactical assets. The Guardian Agent is responsible for aggregating, filtering, integrating and delivering decision-quality information in a timely fashion to a proxy. This proxy, known as the Host Agent, executes on the associated digital avionics system and distributes the decision-quality information to the pilot-vehicle interface (PVI) on the tactical platform, as depicted in Figure 1.

Tactical Node Architecture

- **Host Agent (HA)**
  - Defines tactical system information needs
  - Sets priorities and deadlines for QoS logic
  - Responsible for receiving information updates needed by the weapon system from GA
  - Located on the weapon system
  - Interacts with Pilot-Vehicle Interface (PVI)

- **Force Templates**
  - Information required by a specific platform
  - Information sent by a specific platform
  - Access to F-15 GA Fuzets (F-15 GA Personality)

- **Tactical Communications Manager**
  - Converts HA needs, priorities & deadlines into content needed to control tactical datalink(s)
  - Provides network capabilities to GA

- **Guardian Agent (GA)**
  - Determines needed/available information
  - Adjusts QoS via Quality Object Middleware
  - Specific to a platform
  - Located off-board the platform
  - Scalable and relocatable
  - Consists of executable processes – fuselets
  - Retains state via JBI information objects

Figure 1.

One basic assumption implicit in the design of the Guardian Agent is that new information processing functionality will reside off-board the tactical asset. This new and complex functionality will likely be hosted on a strategic asset, such as an Airborne Warning And Control System (AWACS). Currently, computation and communication resources on such strategic assets are not as constrained as similar resources that exist on their tactical counterparts. Fielding the Guardian Agent application will present an opportunity for strategic assets to host numerous Guardian Agents, potentially numbering in the hundreds. Over the next two decades, Unmanned Air Vehicles (UAVs) will likely dominate the battlespace in numbers that are potentially an order of magnitude greater than those of current tactical assets, and will likely introduce information exchange requirements similar to those of current tactical assets. As such, careful consideration needs to be given to the computation and communication resources that reside on strategic assets. Without careful consideration of these resources, strategic assets could eventually become as constrained as their tactical counterparts.

During the design and initial development of the Guardian Agent application, focus was placed on static allocation of Guardian Agents to strategic assets. The thesis of this paper is that incorporating the concept of mobility into the Guardian Agent application, along with a QoS management framework on associated strategic assets, will increase the scalability of the Guardian Agent concept. Specifically, these enhancements will...
enable the migration of Guardian Agents to strategic assets that can provide the desired JBI-like functionality with acceptable levels of QoS. In this context, a key QoS parameter that will be defined is the ability of a strategic asset to provide timely delivery of raw and partially fused tactical data. Raw and partially fused tactical data are the building blocks upon which the Guardian Agent performs its work and is then able to deliver timely decision-quality information to the warfighter, who is utilizing this information via the PVI on the associated military digital avionics system. Hence, this paper describes extensions to the Guardian Agent application that would incorporate the concepts of mobility and QoS management, and are planned to be demonstrated utilizing commercial-off-the-shelf (COTS) technology.

Currently, potential suitable COTS technology is being matured as part of the Real-Time Java™ for Embedded Systems (RTJES) initiative [21]. The RTJES initiative is also being funded by the Air Force Research Laboratory, and implemented by the Boeing Company. Java™ and Real-Time Java™ technologies have the potential to provide a foundation for this new military capability for a number of reasons.

First, Java™ technology provides the platform-independent computational model that is needed for any type of application that deals with code mobility. [2] In addition, Java™ provides a highly compact code representation that is tailored to reduce footprint and for transmission over networks, both of which are attractive features for implementation in embedded systems with highly constrained computation and communication resources. [10] Second, Real-Time Java™ technology can provide support for deterministic and hard real-time computation and execution of a Guardian Agent. [3] Another attractive feature of Real-Time Java™ technology is the scheduling framework, which could potentially support the requirements of soft real-time computation, such as dynamic scheduling, QoS management, adaptive resource management, etc.

In addition to capitalizing on the use of COTS technology, an underlying theme in the application of the mobile Guardian Agent, and mobile agents in general, is the parallel to open systems approaches. Fundamentally, mobile agents can be considered an open systems approach because the mobile agent concept subscribes to the paradigm of modular design. Modular designs are characterized by discretely partitioned functionality, reusable components and modules, rigorous interface definitions, and technology transparency [20], all attributes that are characteristic of mobile agents in general. As an example, a mobile Guardian Agent would represent discretely partitioned functionality. This functionality is reusable across numerous military digital avionics systems, and would require rigorous interface definitions to achieve such a level of cross-platform interoperability. Benefits of incorporating open systems approaches, such as increased cross-platform interoperability, have specific impacts and implications that need to be analyzed in terms of military effectiveness.

Other impacts to military effectiveness, pertaining to the mobile Guardian Agent concept, are also deserving of reflection. Infusing the Guardian Agent with mobility can potentially provide the needed flexibility to adapt to a battlespace that is constantly changing. Scenarios in which mobile Guardian Agents, and mobile code in general, would be particularly suited include the dynamic integration of additional tactical assets, system-of-systems fault tolerance, and efficient use of tactical datalink bandwidth, as depicted in Figure 2. An example of system-of-systems fault tolerance is an AWACS aircraft exiting the JBI, which implies the transfer of control and execution of currently instantiated Guardian Agents. Other scenarios could be pursued, but these three have been previously identified as important to enabling incorporation and transition of the Guardian Agent application into existing legacy systems.

The remainder of this paper is structured as follows: The Related Work section describes other applicable research in the mobile agent and mobile code domains and provides specific background regarding the Guardian Agent application. The Implementation section describes the proposed mobile Guardian Agent application, and the underlying QoS framework that the mobile Guardian Agent will monitor and utilize as the basis for adaptation and migration. The Conclusion section presents concluding remarks regarding mobile Guardian Agent applications.

5.D.5-3
Mobile Guardian Agent Scenario

Challenges To Be Addressed
- Interoperability between C++ and Java subsystems
- Distribution
- Mobility and portability
- Resource management
  - Dynamic Admission control
  - Multiple asset scheduling

Figure 2.

2.0 Related Work

2.1 Overview of Mobile Agents, Mobile Code and QoS Management

There are a number of research efforts in the domain of mobile agents and mobile code that provide solutions to portions of the problem defined thus far in the paper. At the present time, no complete solution exists in the literature that addresses the aforementioned issues involved with the use of mobile agents or mobile code, within the context of military digital avionics systems. Increased complex functionality, with acceptable levels of QoS, to military digital avionics systems must be provided. As will be discussed later in this section, there has been some initial research done in developing generalized frameworks for the use of mobile agents and mobile code to manage QoS across different types of networks and systems.

An excellent generalized overview of code mobility is presented in [8]. As will be described later, the strong mobility computational model that was developed by Fugetta, Picco and Vigna, is chosen to represent the mobile Guardian Agent. The strong mobility computational model was chosen because mobile Guardian Agents are envisioned to require mobility of their execution state, including data space. Data space, which can be a significant portion of the execution state, is defined as the JBI Information Objects that the Guardian Agent operates upon. It is these Information Objects that are the mechanism to accomplish the following: provide decision-quality information back to the PVI that resides on the associated military digital avionics system; publish sensor data from the tactical asset; and in general provide a description of the common operating picture within the battlespace. [20] While this interpretation may fall somewhat short of the strict definition of strong mobility given in [8], it certainly implies an execution state that is far more involved than simple initialization data or state variables, which separates strong mobility from weak mobility.

Given that mobile Guardian Agents will possess the capability to decide when and where to migrate, an entity or mechanism will be required that can communicate and describe constraints and loads on the current strategic asset and other potential host nodes that reside in the battlespace. A framework that embodies such mechanisms is described in [5]. This framework employs the concept of a domain, which is defined as a "logical boundary used to delimit nodes, agents, and
resources into manageable and distinct entities". As such, a domain agent is "a static agent that supervises the activities that occur within a domain".

Potential services that a domain agent could provide include a migration service for mobile agents, an authentication and validation service for mobile agents, mediating mobile agent access to system resources that exist on the strategic asset, and advertising available sources of raw data and associated QoS levels that can be provided for mobile agents. Alternatives to implementing QoS and resource management functionality, which may be more suitable to meeting the needs of hard and soft real-time applications, will be introduced later in this section.

The research effort that is described in [11] is very similar to the proposed mobile Guardian Agent application. In this case, the mobile agents are utilized for three purposes: encapsulating and routing communication that is required by an individual warfighter in the battlespace; migrating intensive, multi-step computations and queries to the sources of data required for execution; and for providing system-to-system interoperability between a command post and units deployed in the battlespace. Of specific interest for this paper is the use of mobile agents to migrate intensive, multi-step computations and queries around the network. In essence, this is one of the applications of the proposed mobile Guardian Agent that has tremendous potential for increasing the overall military effectiveness of static Guardian Agents.

Several research efforts have begun to look at QoS management and provisioning via utilization of intelligent agents. A particularly interesting framework for wireless networks is addressed in [1]. The framework described is focused on the management of QoS for mobile users in a wireless domain. This is similar to the environment that military digital avionics systems are required to operate. Specifically, the aforementioned framework delegates responsibility to mobile agents for performing load balancing, assuring QoS criteria for certain classes of users, and identifying available resources. In addition, mobile agents are utilized as a tool for predicting user migration. The paper also introduces the concept of "best QoS", in regards to the connection established by the mobile user to a server or mobile host that can provide the highest quantitative level of QoS.

Other approaches and frameworks have been developed to effectively manage and provision for QoS, without the incorporation of mobile agents or mobile code. One such framework that is interesting and practical, from the viewpoint of military digital avionics systems, is the Quality Object (QuO) middleware extensions described in [15]. The QuO middleware extensions provide for the development of the following components to facilitate the management of and provisioning for QoS in middleware-based applications: contracts, delegates, and system condition objects.

Contracts are used to specify levels of QoS desired from a remote object or are provided by the client. In addition, contracts are used to specify operating regions that define boundaries for measured levels of QoS and the actions to take when those boundaries have been exceeded. Delegates act as local proxies for clients and remote objects, and provide locally adaptive behavior based on current system-level QoS. System condition objects utilize interfaces to system resources such as CPU cycles, memory and remote objects. Also included are potentially quantitative measurements of relevance to the JBI, such as the number of subscriptions to particularly raw or partially fused tactical data. These system condition objects are used by the QuO contracts to measure system-level QoS, which may be used to drive adaptation in the system.

For the purposes of this paper, it is envisioned that adaptation in the system translates into Guardian Agents migrating to other host nodes, which can provide the desired levels of QoS for raw and partially fused tactical data, or other potential complex functionality. As described in [15], the QuO middleware is already being utilized to manage QoS in regards to the transmission of imagery and other tactical data between military digital avionics systems. Hence, the QuO middleware extensions are an excellent candidate for implementing the functionality of the domain agent described earlier in this section.

Defining desired levels of QoS for raw and partially fused tactical data must be accomplished in a manner that is flexible and can adapt to a battlespace that is constantly changing. An
The approach that will be adopted for use in this paper is described in [16]. This approach, known as force templates, is a methodology to provide a description of an entity or force that exists in the battlespace, and in a format that is readily accessible via electronic means. The purpose of a force template is to enable the rapid integration of the entity or force into the JBI, and define information exchange and service resource requirements.

The force template concept is already integrated within the existing Guardian Agent application, and defines the information exchange requirements that a given tactical asset must meet to successfully integrate and interact with the JBI. For the purposes of this paper, the force template concept will be expanded to define QoS requirements for individual pieces and streams of raw and partially fused tactical data. In addition, this expansion of the force template concept is envisioned to eventually support the automatic generation of QuO contracts, which can be used to dynamically monitor and adapt to changes in system-level QoS.

2.2 Guardian Agent Overview

The Guardian Agent, as depicted in Figure 1, identifies and accesses information of interest across the JBI, evaluates the tactical utility of the accessed information, and transmits the information to the tactical asset using available communications, such as legacy tactical datalinks. Similarly, the Guardian Agent extracts information of interest from the tactical asset and publishes it to interested entities across the JBI. The Guardian Agent also brokers cooperative actions between its associated tactical asset and other tactical assets, with specific focus on aiding the associated asset in establishing and executing cooperative actions. This capability supports the controlled, but flexible and rapid, decision making among tactical assets that is important in the pursuit of Time Critical Targets (TCTs). The Guardian Agent is scaleable to the tactical need and relocatable anywhere within the JBI. The Guardian Agent design is generic in nature allowing reuse across a wide range of systems. The Guardian Agent design provides communications between the Guardian Agent and all other JBI nodes through a Common Object Request Broker Architecture (CORBA) interface, with reliance on the eXtensible Markup Language (XML) standard for system flexibility and reuse. The CORBA interface lies above and is compatible with existing datalink layers. The Guardian Agent concept also includes use of JBI services and fuselets that will tailor information captured and deliver that information to the tactical asset.

The Host Agent, as depicted in Figure 1, is a relatively small software entity, which resides on the tactical asset and operates in conjunction with the Operational Flight Program (OFP) software, which controls the PVI. The Host Agent provides an interface between an existing military digital avionics system and the Guardian Agent, using legacy tactical data links for communications. The Host Agent includes QoS management capability similar to the Guardian Agent and also satisfies any operator interface requirement associated with additional JBI functionality.

The Navigation and Discovery process, as depicted in Figure 1, defines how the Guardian Agent navigates the JBI and discovers potential sources of data or information. In order to support this functionality, each asset registers with the JBI when it becomes available and whenever it has a major change of state. For example, a strategic asset, such as an AWACS aircraft, would initially register with the JBI and provide its planned mission and sensor coverage. As other tactical assets register, they would evaluate the capabilities of the strategic asset against their specific mission needs. If the strategic asset shows potential for supplying tactically significant information, the registering asset’s Guardian Agent would subscribe to the strategic asset. Subsequently, the Guardian Agent would evaluate each sensor report from the strategic asset for impact to the tactical asset. Only those reports, which are of significance to the tactical asset, would be passed on. This methodology for Navigation and Discovery requires joint evolution between tactical assets and the JBI.
3.0 Proposed Implementation

3.1 Proposed Infrastructure for Mobile Guardian Agents

A fundamental requirement imposed by a mobile agent or mobile code application is the infrastructure to support the transmission and execution of code and its associated state. The design and implementation of this infrastructure needs to be given serious thought, as certain design and implementation decisions may limit the applicability and performance of the mobile agent or mobile code application. A first step in the design process is to choose a sufficient computational model to represent the desired mobility mechanism.

As mentioned in the previous section, [8] provides several models of computation that can be used to classify various mobility mechanisms. The computational model that is most applicable to the mobile Guardian Agent application is strong mobility. As defined in [8], strong mobility is the ability of a Mobile Code System (MCS, also called strong MCS) to allow migration of both the code segment and the execution state of an Executing Unit (EU) to a different Computational Environment (CE). The mobile Guardian Agent application can be classified under strong mobility, because the Guardian Agent would require access to its corresponding data space, in spite of any migrations to other CEs that have occurred or might need to occur in the future.

Furthermore, a particular Guardian Agent that is processing information within the context of hard real-time deadlines, will potentially require that its specific execution state is transmitted along with the code segment and data space. This could also support potential host nodes in accomplishing schedule feasibility analysis at run-time, which would otherwise be completed statically using traditional scheduling techniques.

The mobile Guardian Agent application can also be classified along the lines of proactive migration [8], since the Guardian Agent is envisioned to autonomously determine the time and destination for migration. This point will be clear in the next section, since the proposed migration of the Guardian Agent is tied specifically to levels of QoS that are being provided by the host node.

Another issue that needs to be resolved, in terms of the computational model, is data space management. As described in [8], there are two classes of problems that need to be addressed in regards to dataspace management for migrating EUs, such as mobile Guardian Agents. The first issue is resource relocation. Three categories of relocatable resources are defined in [8]: Free Transferable, Fixed Transferable, and Fixed Not Transferable. Respectively, these categories refer to the status of resources being free to move across the network versus remaining fixed at a specific CE, and whether it is feasible to actually move a particular resource across the network.

As mentioned previously, JBI Information Objects are an important resource within the data space of the Guardian Agent. In general, JBI Information Objects could be classified under any of the previous definitions. For the purpose of this paper, JBI Information Objects, associated with mobile Guardian Agents, are classified as Free Transferable. This classification is made because the mobile Guardian Agent will utilize Information Objects to accumulate results of the aggregation, filtering and fusion that the Guardian Agent is performing across the JBI. Thus, it is logical for associated Information Objects to be transmitted along with the Guardian Agent code segment and execution state.

A second issue in dataspace management that needs to be addressed is binding reconfiguration. Three categories of resource bindings are identified in [8]: By Identifier, By Value, and By Type. The titles of these categories are sufficiently self-explanatory. Given that JBI Information Objects are utilized by Guardian Agents for the purposes previously described, a logical classification for Information Objects is “By Identifier”. Information Objects will likely be bound to specific Guardian Agents, since Information Objects will be utilized for data aggregation, filtering and fusion that is specific to a particular Guardian Agent. In addition, this binding classification also supports the previous resource classification of JBI Information Objects as Free Transferable. This classification does not preclude a particular Information Object from being shared across the JBI. On the contrary, this still
could be accomplished via publish and subscribe mechanisms supported by the Guardian Agent. What this classification does provide is a simplification of the binding reconfiguration process described in [8]. Under these assumptions, almost no binding removal or re-binding would need to be accomplished to allow for a particular JBI Information Object to migrate along with the code segment and execution state of a Guardian Agent.

Mobile Guardian Agent Infrastructure

Figure 3.

Typically, the next step in the design process would be to select a design paradigm to follow, based in large part upon the computational model chosen. Given the nature of the mobile Guardian Agent application, the design paradigm that has been chosen is Mobile Agent. Regardless, it is worthwhile mentioning the other design paradigms as a reference for the reader who is interested in general techniques. Aside from Mobile Agent, other design paradigms defined in [8] include: Client-Server (CS), Remote Evaluation (REV) and Code on Demand (COD). The Mobile Agent design paradigm is defined in the context of a component that has knowledge of how to interact with another component or a service, but is required to be hosted at that site or elsewhere based on resource constraints or other requirements in the execution environment. Nonetheless, the mobile Guardian Agent application possesses characteristics similar to other design paradigms, such as Client-Server and Remote Evaluation. As an example, a previously mentioned operational benefit of the mobile Guardian Agent application is reduced consumption of tactical datalink bandwidth. This same benefit could potentially be achieved by strictly migrating only the Guardian Agent code segment to a data-rich location, such as a large ground-based database. Thus, the mobile Guardian Agent application in this case is treated as an instance of the Remote Evaluation design paradigm.

For the purpose of this paper, these other design paradigms are not specifically utilized because conditions in the execution environment are driving the necessity for mobility on the part of the Guardian Agent. The other design paradigms would be more appropriate if the availability of local data or local knowledge was driving the necessity for mobility of the Guardian Agent.
After consideration of all the issues previously mentioned, the final step in the design process is choosing an appropriate technology for the purpose of application development and implementation. Currently, numerous software technologies exist that can support both the implementation and execution of mobile agents and code. Examples include Agent Tcl, Ara, Java™, Java™ Aglets, Sumatra, TACOMA, etc. For the purpose of this paper, a combination of Java™ and Real-Time Java™ is envisioned to form the foundation for the mobile Guardian Agent application.

As mentioned previously, these two technologies are chosen because of their support for cross-platform interoperability and execution of hard real-time and soft real-time processing. In addition, the Java™ Virtual Machine (JVM) provides basic mechanisms for secure execution of local and mobile code. [2] Utilizing Java™ technology also simplifies the previous discussions regarding management of execution state and data space. Given that code loaded by the JVM is executed from scratch, transmitting execution state is not required in the traditional sense. As described in this section, the mobile Guardian Agent will require local access to the JBI Information Objects that will be utilized. Thus, dataspace management will still be required at some level.

Figure 3 provides a general overview of a potential implementation of the mobile Guardian Agent application and infrastructure.

3.2 Proposed QoS Framework for Mobile Guardian Agents

Given the previously described infrastructure for the mobile Guardian Agent application, the remaining task is to define the QoS framework and associated management that will potentially drive the migration of Guardian Agents. A key concept that will be implied is the autonomy provided to the mobile Guardian Agent, in terms of allowing it to monitor QoS at various system levels on the host node. This information will be used to initiate a migration from the current host to another host, which can satisfy the constraints and requirements that are established by the Guardian Agent.

Before defining a QoS framework for mobile Guardian Agents, several underlying assumptions need to be clarified. First, given that Guardian Agents represent tactical nodes in the JBI, a logical assumption to be made is that a given strategic asset will be required to provide access to or act as a conduit for real-time data that the tactical asset requires or can provide. In most cases, which are representative of sensor-actuator applications in general [9], the real-time data also tends to be periodic in nature. Given that this periodic data is processed and transmitted at sufficient rates, for example in the range of 10-20Hz, it can be argued that this periodic real-time data can be classified as streaming data. As such, it is not altogether different from other forms of streaming data that will be transmitted throughout the JBI, such as streaming video, audio, etc., and can be treated similarly from the standpoint of theory and implementation.

A second assumption that needs to be clarified is the requirement for the existence of a middleware infrastructure on strategic assets that are potential host nodes for mobile Guardian Agents. As will be discussed later, the QuO technologies assume the existence of a middleware infrastructure, specifically one that is CORBA-based. While it may seem premature to choose an implementation technology at this stage in the design process, there are good reasons to do so. For example, other similar avionics applications, as described in [15], have been implemented utilizing CORBA-based technology, along with QoS management extensions that are implemented using QuO technologies. Other constraints, such as support for hard real-time processing, also limit the choice of suitable COTS technology for use in the proposed implementation.

As mentioned in the Introduction, a principal QoS parameter for mobile Guardian Agents is the ability of the host node to deliver raw and partially fused tactical data in a timely fashion. This will potentially be a problem in the future for strategic assets, since these assets are envisioned to support n-number of Guardian Agents and the various on-board command and control systems that already exist. Hence, the Guardian Agents being hosted will have to compete for various system resources on the strategic asset, to potentially include JBI data streams that the strategic asset has access to or will act as a conduit.
Specific QoS parameters for raw and partially fused tactical data, such as JBI data streams, can be defined in a fashion suited to the particular type of data or data stream. For example, QoS parameters regarding data streams could include frame rate, latency, processing time, etc. All of these related QoS parameters are of interest to the mobile Guardian Agent, and a combination of any number of them could influence the Guardian Agent's decision to migrate to another host node. As such, a priority or other classification scheme must be included with specific QoS parameters, to denote their relative influence in a Guardian Agent's decision to migrate. Determining which specific QoS parameters are important to a particular tactical asset is intended for future research efforts.

Before entering the battlespace, specific QoS parameters will likely be assigned to tactical assets by the JBI Information Staff [22], and should correspond with the practices and requirements established for given tactical assets and their associated Guardian Agents. The problem that needs to be resolved in assigning QoS parameters is the use of a methodology that enables seamless run-time integration with the JBI. Force templates is a technique that is already integrated within the existing Guardian Agent application, and defines the information exchange requirements that a given tactical asset establishes to successfully integrate and interact with the JBI. As described in [16], force templates provide a description of an entity or force that exists in the battlespace, and in a format that is readily accessible via electronic means. As previously mentioned, the Guardian Agent application is already utilizing XML-based technology to implement such force templates. Thus, adding the capability to assign and specify QoS parameters within a force template is a matter of providing additional fields in the XML-compliant force template document. The following force template excerpt is an example of how this can be accomplished for a tactical asset that is publishing raw or partially fused tactical data:

```xml
<mil.JBI.Force_Template.Guardian_Agent.UAV.xml>
<brmil.Video.meta>
<REGION>Afgahistan</REGION>
<PRODUCER>RQ-1 Predator</PRODUCER>
<QOS_PARAMETERS>
  <FRAME_RATE>20</FRAME_RATE>
  <LATENCY>2</LATENCY>
  <PROCESSING_TIME>10</PROCESSING_TIME>
  <COMPRESSION_RATIO>50:1</COMPRESSION_RATIO>
</QOS_PARAMETERS>
</mil.Video.meta>
</mil.JBI.Force_Template.Guardian_Agent.UAV.xml>
```

It is assumed that QoS management components, such as Quo contracts, will be initially developed using the Quality Description Language (QDL) and other techniques, as described in [17]. While this is acceptable for the purposes of research and concept demonstrations, the required QoS management components will eventually need to be generated in an automated fashion. Potentially, force templates could be used to automatically generate Quo contracts and other components that the Guardian Agent would utilize to define how it would monitor QoS on the strategic asset.

To illustrate the proposed QoS management framework, the interaction of various Quo components will be discussed in the context of Figure 4. Each individual fuselet or other application, being managed by the Guardian Agent, will subscribe to JBI data or data streams via a Quo application delegate. Likewise, JBI data or data streams will publish to Guardian Agents, or other command and control systems, via a Quo application delegate. This allows for in-band adaptation [15], where a Guardian Agent or other on-board command and control system can modify the flow of data based upon tolerances in the associated Quo contract. A similar argument can be made for Guardian Agents that are publishing raw and partially fused tactical data on behalf of their associated tactical asset. In this case, an additional Quo contract will be established to monitor progress and system resources reserved for the data stream being published out to the JBI.

In addition, it is likely that Guardian Agents will be prioritized based on mission need, and with respect to other on-board command and control systems. This will determine which Guardian Agent or other on-board command and control system will be allowed to modify the flow of data from a shared JBI data stream. As examples, this capability would be useful if a given fuselet application is being overloaded with data, or if the Guardian

5.D.5-10
Agent itself is not receiving sufficient system resources, such as processor time or memory, to execute on the strategic asset. Hence, the strategic asset can reclaim system resources, via adaptive resource managers, that are reserved for JBI data streams or other purposes, and are not being utilized at their stated capacity.

**QoS Management Framework for Mobile Guardian Agents**

![QoS Management Framework Diagram](image)

**4.0 Conclusion**

In conclusion, this paper has described a general infrastructure to develop mobile Guardian Agent applications, which can provide increasingly complex functionality with acceptable levels of QoS for military digital avionics systems. Given that the Guardian Agent approach will be pursued as the de facto standard for integrating JBI-like functionality on future and legacy military digital avionics systems, solutions must be developed to deal with the shift in computational complexity and load that needs to be accommodated by strategic assets or other potential host nodes. If such solutions are not developed, strategic assets could eventually become as resource constrained as their tactical counterparts.

The authors would like to thank Mr. David Sharpe, Boeing Company, Mr. Joseph Loyall, BBN Technologies, and Mr. Kenneth Littlejohn, Air Force Research Laboratory, for reviewing and providing comments on this paper.

**5.0 References**


[4] Corman, Dr. David E., Thomas Herm, Dr. Kirby Keller, Charles Satterthwaite, 2001, Transforming Legacy Systems To Obtain Information Superiority,
Proceedings of 6th International Command and Control Research and Technology Symposium.


5.D.5-12