Flexile Middleware: A Proposed New Class of Adaptive Middleware and its Utility for Time-Critical Applications

A Thesis
Presented to
the Department of Electrical and Computer Engineering
Villanova University

In Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy (PhD)

Thomas A. DuBois

May 2012

Declaration

STATEMENT BY AUTHOR

Acknowledgements

I would like to acknowledge my team of advisors led by Dr. Richard Perry (Villanova University), and including Dr. Sarvesh Kulkarni (Villanova University), Dr. David Russell (The Pennsylvania State University), and Dr. Oleg Sokolsky (University of Pennsylvania). These individuals kept me focused on the most fruitful aspects of the research and gave me the confidence to complete this work. Praises are due to the many other researchers who presented their results and ideas in the study of adaptive middleware. This dissertation builds upon their accomplishments. Thanks also to colleagues from The Boeing Company, who helped me with scenario and modeling work, including Brendan Blanton, and Zachary Prindle. Finally, I would like to thank Gustavo Luiz Bastos Baptista, a research colleague at Pontifícia Universidade Católica do Rio de Janeiro, who helped with the middleware trade study contained in an appendix to this thesis.

DEDICATION

This dissertation is dedicated to my parents, Thomas and Barbara, and my children, Michele and Thomas, Jr.

Contents

List of Figures

List of Tables

Glossary

API	Application Program Interface. 11, 76
ATNS	Advanced Tactical Network System. 65
AWACS	Airborne Warning And Control System. 10
BLOS	Beyond Line Of Sight. 62
CLIP	Common Link Integration Processing. 10
CONOPS	Concept Of Operations. 62
CORBA	Common Object Request Broker Architecture. 8
CORE	Common Open Research Emulator. 55, 58
DBE	DDS Benchmarking Environment. 89
DBMA	Data Base Management System. 78
DCOM	Distributed Component Object Model. 8
DCPS	Data Centric Publish-Subscribe. 69, 87
DDS	Data Distribution Service. 3, 7, 8, 26, 68, 70, 73, 78
DIACAP	Defense Information Assurance Certification Accreditation Process. 53
DISR	Defense Industry Standards Registry. 53
DLRL	Data Local Reconstruction Layer. 72

DoDAF Department of Defense Architecture Framework. 51

ESB Electronic Service Bus. 83

FBCB2 Force Battle Command Brigade and Below. 9

GESP GIG Enterprise Service Profile. 53

GIG Global Information Grid. 53

HTTP Hypertext Transfer Protocol. 11

IA Information Assurance. 53

IaaS Infrastructure as a Service. 14

IDE Integrated Design Environment. 72

IDL Interface Description Language. 69

IOC Initial Operating Capability. 50

IP Internet Protocol. 11, 65

ISP Information Support Plan. 48

JBC Joint Battle Command. 9

JCIDS Joint Capabilities Integration and Development System. 48, 51

JEFX-10 Joint Effects Forces Experiment 2010. 62

JITC Joint Interoperability Test Certificate. 48

JMS Java Messaging Service. 11

JTRS Joint Tactical Radio System. 64

JVM Java Virtual Machine. 8

LCS Littoral Combat Ship. 9

LPD Landing Platform Dock. 9

MANET Mobile Ad hoc Network. 70, 93

MoCA Mobile Collaboration Architecture. 20

NIST National Institute of Standards and Technology. 13

NR-KPP Net-Ready Key Performance Parameter. 48

OMG Object Management Group. 8, 69, 73

OpenMP Open Multi-Processing. 33

ORB Object Request Broker. 16

OSI Open System Interconnect. 11

PaaS Platform as a Service. 14

QoS Quality of Service. 67, 70, 74, 87

RMI Remote Method Invocation. 16

RTI Real Time Innovations. 10, 79

RTOS Real-Time Operating System. 78

RTPS Real-Time Publish-Subscribe. 70

SaaS Software as a Service. 14

SAR Search-And-Rescue. 28

SATCOM Satellite Communications. 62

SCADA Supervisory Control And Data Acquisition. 73

SCTP Stream Control Transmission Protocol. 11

SOAP Simple Object Access Protocol. 11, 71

SOSCOE System of Systems Common Operating Environment. 16

SQL Software Query Language. 69

STK Satellite Toolkit. 36, 41, 55, 58

SysML System Modeling Language. 51

TRL Technical Readiness Level. 80

UAV Unmanned Air Vehicle. 27, 55, 56, 58, 62

UDP User Datagram Protocol. 11

UML Unified Modeling Language. 51

WAN Wide Area Network. 83

1

Introduction

As is most often the case, necessity is the cause for invention, and the same is true in the case of Flexile Middleware. The need arises from missions that have real-time constraints and using an ad hoc, heterogeneous network. The dynamic nature of these missions and the network itself make defining the processing requirements most difficult. Fortunately, the emergence of net-based applications provides the ability to deploy useful functionality without a detailed understanding of both predictable and unpredictable events that may occur during the mission. The challenge is loading the best applications on the various system processors that are collaborating to accomplish a mission while using an ad hoc wireless network. Clearly, there is a technical gap that needs to be filled.

Middleware provides a programming layer that sits between the operating system and the applications on the software architecture of networked systems, and as such would be most suitable for defining the functionality needed to address this technical gap. Middleware is composed of services used by applications to interact with other applications and the network, and additionally it includes programming executive functions that can provide the means to achieve application-level adaptability. Towards the goal of addressing the processing gap just introduced, research began with a survey of middleware, which is contained in Appendix A. There are various forms of middleware, including different ways of classifying middleware. There is also much literature on adaptive middleware (e.g. [1], [2], [3], [4]). It would be desirable to employ a middleware that is adaptive and dynamic to domain rules, and both host processing and network limitations. Since critical missions can change significantly while in progress, the domain rules themselves may change. Current middleware implementations do not contain all the functionality needed to address this technical gap. Flexile Middleware is a form of adaptive middleware that does not fit existing taxonomies either, and variations on its implementation gives rise to a new class of adaptive middleware. In this thesis, the term Flexile Middleware is used to describe both an implementation and the class of middleware defined by the characteristics assigned to Flexile Middleware.

Flexile Middleware can be defined as middleware that adapts in several different ways. It is this multi-dimensional adaptability and the fact that it can be generalized that gives rise to its existence as a class of middleware unto itself. Similar to other forms of adaptive middleware, Flexile Middleware uses a priori information to set an operational context which can vary from mission to mission, and it also contains the ability to change the behavior of applications that are using its services. Middleware that have these characteristics are referred to as both static and adaptive middleware. Flexile Middleware has both of these features. However, Flexile Middleware extends adaptability by taking processing and networking resources into consideration to determine an informed recommendation of applications that should be operating on processing systems collaborating to accomplish a mission. Flexile Middleware can dynamically change the subset of applications to adapt to changes in missions constrained by available resources. Domain specific rules are defined for this purpose, and Flexile Middleware also includes a learning feature where the rules themselves can change. Flexile Middleware recognizes that applications use services which are part of the middleware, and as the set of active applications changes, so does the set of active middleware services. Some core services in Flexile Middleware will always be running to ensure it behaves as defined, but it can mutate from mission to mission or during a mission. All this adaptability is why the word 'flexile' was chosen to describe this type of middleware. Chapter 4 is dedicated to describing the details of how Flexile Middleware works.

Since there are many different ways to implement Flexile Middleware, the functionality described in Chapter 4 actually defines a class of middleware. In fact, one would expect very different adaptation rules depending on the missions for its intended use. Intended use is a very important concept in the design of Flexile Middleware. When, how, and how often Flexile Middleware can adapt may vary from application to application as well. All this adaptability introduces processing overhead. The benefits have to be worth the added overhead. In applications where memory, processing, and network bandwidth are not issues, Flexile Middleware would probably not be beneficial, so the use of Flexile Middleware is confined to missions that have these constraints. Sometimes, these constraints may not be realistic, but contrived for utility. For example, it may be possible to run numerous applications, but outputs from these applications could introduce too much cognitive workload, which could reduce the probability of mission success. One way to address this concern is to artificially limit the memory available for net-based applications, and let Flexile Middleware attempt to figure out how to select the applications to fill that memory.

To illustrate the value of Flexile Middleware, this thesis contains the description of a representative critical mission suitable for generalization. Chapter 5 describes the example mission, rationale for selected parameters, estimates of expected mission effectiveness improvements from Flexile Middleware, a sensitivity analysis, and a discussion on general mission value extrapolated from the example mission and the sensitivity analyses.

A common characteristic of critical missions discussed in this thesis is the use of wideband Mobile Ad hoc Networks (MANETs). These missions are highly mobile and involve the use of two or more of the following: rotorcraft, fixed wing aircraft, Unmanned Air Vehicles (UAVs), ground vehicles (manned and unmanned), and ground personnel. These missions will often be in areas where network connectivity is complicated by line-of-sight considerations between entities involved in the operation. An assumption put forth is that operational video can provide improved situational awareness, which increases operational effectiveness. Video drives the bandwidth, which impacts line-of-sight [5]. It is possible to mitigate line-of-sight by using digital VHF/UHF radios, but radios of this type that are suitable for use on rotorcraft and UAVs can achieve at most 64 kbps, which is insufficient to support video communications. Data rates of at least 250 kbps (preferably 500 kbps) per video source would be necessary to achieve the situational awareness value attributable to video. Networked communication systems (i.e. radios, amplifiers, and antennas) to support these higher bandwidth requirements need to be either directional for ranges up to 100 nautical miles, or omnidirectional for ranges up to about 15 nautical miles. In either case, the frequencies used by radios to support these rates depend on line-of-sight between connected nodes. Another way to attempt to mitigate line-of-sight issues is to use Beyond-Line-Of-Sight (BLOS) connections (e.g. Satellite Communications). This consideration was also address by DuBois, et al. [5], where the conclusion is that it is more effective to use line-of-sight connections instead of attempting to solve the difficult problem of wideband BLOS on rotorcraft, which is complicated principally by modulation from rotor blade interference. There are additional integration concerns associated with space, weight, and power constraints cause by BLOS antennas and electromechanical processing.

The value of Flexile Middleware is assessed by comparing it to other methods and then calculating the probability of mission success for all methods. Four methods, including Flexile Middleware, were used for the analysis. Characteristics for each of the methods was used to define operational parameters and probabilities. A mission-based mathematical model was prepared to calculate probability of success. Parameters used in the model were defined as a result of running numerous mission profiles using a commercially available constructive simulation tool. The critical mission is a search-and-rescue operation with complications from a radiation cloud resulting from a nuclear accident. This mission was inspired from the tsunami that devastated Japan in March of 2011. Time is critical for this mission as is often the case for search-and-rescue operations. There are two major decision points based on how quickly information reaches the helicopter pilot charged with the rescue. A first decision can be made if the pilot receives predictive information about the movement of the radiation cloud, and then chooses to modify the flight route based on that information. The second decision can be made only if there is no decision at the first decision point. This represents receipt of more firm information about the radiation cloud, which would be much closer to the helicopter at this point requiring a recovery route to avoid a high probability of contamination.

The example base mission case is generalizable. Time is a parameter common to most missions. Though probability of contamination will be used to calculate the probability of mission success, other major events, such as an accident, or in the case of a military operation, detection from an enemy necessitating abortion of the mission can be alternatively considered. It is basically a major event impacting mission success. Time represents a more fluid parameter. The example contains two major decision points, but it is reasonable to assume that results can be extrapolated for more major decisions that might occur during a mission. This will be discussed further in Chapter 5. A sensitivity analysis was used to assess impact on probability of mission success using different values for mission time and probability of contamination. The conclusions are that: (1) Flexile Middleware offers significant improvement over voice-only communications, simple data link connectivity, and marginal improvement over current state-of-the-art network communications; (2) The improved value of Flexile Middleware is most significant when there is not much margin time for accomplishing the mission; and (3) The probability of a mission-ending event does not significantly change the improvement offered by Flexile Middleware, because decisions to avoid that event are much more important. Chapter 5 contains more details showing the derivation of these conclusions.

In summary, the thesis is organized into the following chapters:

- **Chapter 2 – Aims of the Research:** This chapter describes operational environments where Flexile Middleware is most useful, and introduces a modeling approach to show the value of Flexile Middleware. It also presents key net-ready applications that middleware needs to support and the processing architectures in which it will operate. Motivation for the research is discussed with an outline describing how this research intends to advance the state-of-the-art in middleware.

- **Chapter 3 – Middleware Discussion:** This chapter defines middleware, discusses its history, and summarizes the various classes of middleware. The latest research in adaptive middleware is presented. The Data Distribution Service (DDS$^{\mathrm{TM}}$) standard is described along with references to a trade study of commercial tools to support its use. The trade study itself is included as an appendix.

- **Chapter 4 – Flexile Middleware:** This section introduces and defines Flexile Middleware. A functional block diagram of Flexile Middleware is presented along with detailed descriptions of the nine modules contained in the diagram. Many of the unique features of Flexile Middleware are contained in a special reconfiguration module. This module makes use of rule-based inferencing, prioritized scheduling, and proposes an implementation to exploit the characteristics of multi-core processing architectures.

- **Chapter 5 – Metrics and Measurements:** This section will describe details on the models used to measure the value of Flexile Middleware, including experimentation results.

- **Chapter 6 – Motivation and Background:** This chapter provides more details on the operational environment in which Flexile Middleware is expected to show utility. The background discussion shows how advancements in this area of science can be transitioned for production use. Detailed mission descriptions are presented to illustrate how Flexile Middleware can support activities that are necessary for these missions.

- **Chapter 7 – Summary:** This chapter summarizes the key ideas in the thesis and suggests areas for future research.

2

Aims of the Research

This thesis intends to show that when computer-based intelligence is applied to middleware that it will result in improved network operations and computational efficiency. In accomplishing this aim, the state-of-the-art in adaptive middleware will be advanced and new ways to measure the operational value of middleware will be presented and calculated.

2.1 Principal Aim of the Research

The principal aim of this research is to perform critical missions better through advancements in middleware assuming network connectivity among participants. The approach is to employ networked systems, and then improve functions and decisions by implementing applications and middleware to assist in accomplishing the missions. This dissertation focuses on ensuring that the subset of available middleware services is best suited to support networked applications used for critical missions, and especially for those missions that are time-critical. The research led to the definition of a new class of adaptive middleware, which will be defined and measured in representative critical operations. Not only does this middleware adapt to mission context and roles, but it also adapts to processing and network capabilities and constraints. The adaptive nature of this new middleware offers the potential to provide a higher level of control over network-based services and applications.

2.2 Advancing the State-Of-The-Art in Middleware

A new form of adaptive middleware is presented in this dissertation as a means of addressing the desire to perform critical missions better. It is called Flexile Middleware. The definition and characteristics of Flexile Middleware will be defined in detail later in this thesis (Chapters 3 and 4). It represents a unique hybrid form with both static and dynamic features that are salient during configuration and runtime. This thesis contains a taxonomy that partitions the various forms of middleware and illustrates how Flexile Middleware has a classification of its own.

The advancement of the state-of-the-art in middleware is accomplished by setting up a framework and methodology that allows middleware to adapt in terms of domain-specific rules with due consideration of available processing and network resources. The concept of Flexile Middleware emerged based on the observation that, in critical missions, there were both "disadvantaged" computing devices that could not host all the applications that may be needed, and others with sufficient computing resources were performing too many functions that caused distractions during attempts to accomplish a mission. There is an operational need to have the best set of applications running on networked systems collaborating to accomplish a mission.

Recognizing that adaptation may be difficult to predict if defined too generally (and this is not acceptable for some critical missions), a proposed new class of middleware emerged that seeks to balance adaptation with predictable, yet dynamic, behavior. This is the genesis of Flexile Middleware. It was originally designed with critical applications in mind, but after reviewing its characteristics, it appeared to define its own class of adaptive middleware. Details about this assessment are contained in Chapter 3. Achieving balanced adaptation is the challenge. To address that challenge demands the means to evaluate performance, and a proposed methodology is proposed and applied in Chapter 5. Results from Chapter 5 show that an implementation of Flexile Middleware can improve performance in critical missions, hence advancing the state-of-the-art in middleware.

2.3 Measuring the Operational Value of Middleware

Another aim of this research is to define and present more operationally relevant measures of middleware performance. Parameters, such as bandwidth and throughput, are contributing measures, but this research seeks to quantify performance in terms of mission success. In the context of critical operations, a more useful measure would be probability of survival, or probability of winning a battle. For measures of this form, it is necessary to model missions with network connectivity. Embedded in these models are rules associated with how the middleware performs. Mission decisions based on information provided by network middleware services are contained in the models, so when simulations are run, the behavior of the middleware can be recognized and measured.

Many critical operations are performed in a collaborative manner. Those who perform these operations are beginning to recognize the value of networking to improve how their operations are accomplished. Much has been published on the topic of middleware and its various forms (e.g. [1, 6, 7]). However, the metrics used to measure the performance of middleware appear more directly related to computer-based parameters, such as bandwidth, latency, and other Quality of Service measures. These are useful, but they are not directly related to operational goals or objectives. Admittedly, higher bandwidth and throughput can lead to more timely situational awareness information. A more direct measure of utility is how that timely information is used to accomplish mission-related goals and objectives.

By modeling networked operations with an understanding of the behavior of middleware, it is possible to more directly connect the functionality of middleware to mission-related goals and objectives. General purpose network use is too broad to derive any conclusions, but critical operations (e.g. military, disaster relief) have goals defined well enough where there is a more direct connection between middleware functionality and the ability to accomplish missions. These domain-specific applications give rise to the creation of rules related to mission constraints and contingencies, and these rules can be used to improve the performance of the middleware itself for the applications. Example rules are presented in Chapter 4. The rules are used to determine which applications should be running, and by extension, which middleware services are needed. Middleware is designed to be independent of the underlying applications, but applications use middleware services. It is this connection that provides an opportunity to improve the state-of-the-art in middleware that leads to improved performance in network-enabled, time-critical missions.

3

Middleware Discussion

This chapter summarizes the latest research in the field of middleware. The first section contains a description of middleware, including its most common perceptions. Next, there is a section briefly describing the history of middleware, leading up to current products and standards. One particularly relevant standard is the Data Distribution Service[8], DDS^{TM} which will be described in detail, and that section includes a list of applications. The next section discusses the connection between middleware and Cloud Computing. The last section of this chapter discusses the various ways to classify middleware. One particular classification is chosen, mainly due to the high degree of independence between each of the classification categories. This last section serves as an introduction to the following chapter on Flexile Middleware, because Flexile Middleware is presented as a new class of middleware with characteristics not adequately described by current classifications.

3.1 Description and Perceptions of Middleware

In the software architecture of a networked system, middleware is software that resides between the applications and operating system services[9]. It is software that consists of a set of services that allow multiple processes running on one or more machines to interact across a network. It provides publish-subscribe capabilities for networked users to share information, discovery, message translation, information assurance, and other basic services needed by network-based applications, i.e. net-ready applications. The best middleware conforms to open standards, such as the Data Distribution Service (DDS)[8] standard.

Adaptive middleware has emerged as a topic of considerable interest motivated by the understanding that most ad hoc distributed systems have limited resources, intermittent connectivity, and network heterogeneous systems. When middleware can adapt to the context of the distributed system, efficiencies can be realized. Context-aware adaptive middleware can address quality of service requirements, improve the perception of quality, and enhance the ability to deliver service. According to Loyall, et al.[7], automatically determining user intent is a hard problem, but if successful it could be used effectively in the context of adaptive middleware.

3.2 Middleware History, Products, and Standards

This section briefly describes the history of middleware, and the latest middleware trend towards compliance with the Object Management Group's (OMG®) Data Distribution Service (DDS) standard.

3.2.1 History

Viewing middleware as an intermediate layer in a software architecture means that its implementation from a programming perspective is in the form of library calls (subroutines) from a programming language. Its application to facilitate interoperability across collaborating systems provides a context for the subroutines contained in the middleware library while introducing an architecture for communication (e.g. publish-subscribe, message-passing, etc.). Middleware essentially evolved from object-oriented programming with services taking the form of interoperable objects. If a programmer wants to share information among multiple distributed programs, objects can be used to free the programmer from needing to know details about the variables and structures in all of the other programs. It also represents a development improvement by avoiding the complexities of socket-level programming. This approach to programming and middleware emerged in the 1980s.

In the 1990s, Microsoft® developed their own form of middleware called Distributed Component Object Model (DCOM)[10, 11], and JavaBeans from Sun Microsystems implements similar functionality. In the case of DCOM, interoperability was limited to software running on Windows platforms. JavaBeans was not restricted to a particular operating system, but it would only run on Java Virtual Machines (JVMs) using the Remote Method

Invocation for interfacing between different Java code running on different JVMs. The first widely accepted middleware standard with these characteristics implemented the Common Object Request Broker Architecture (CORBA)[12, 13]. These approaches have matured to the present with improvements, such as COM+[14], Enterprise JavaBeans[15], and revised versions of the CORBA specification[16, 17]. These options are still available today each with their own strengths and weaknesses depending on the application and operating environment[6].

3.2.2 The Data Distribution Service Standard

Middleware has emerged as a topic of considerable interest, and the Data Distribution Service (DDS) standard has been proposed to allow one form of middleware to interoperate with others. There are many details associated with the DDS specification[8], but the key interoperability aspect of DDS is the definition of an object structure that all middleware can use to share information. The DoD Unmanned Air System (UAS) Control Segment (UCS) provides an executive summary on DDS: DDS is widely deployed in some of the nation's most mission-critical systems. DDS is the backbone of the Navy's Open Architecture initiative, integrating shipboard subsystems and weapons on most all new ships. It powers major Army programs under development like JBC-P (Blue Force Tracker/FBCB2 upgrade). It also forms the advanced architectures for many new air systems, manned and unmanned. All major US prime system integrators and most US defense research laboratories are users. It also has a growing footprint in commercial telecommunications, train, automotive, medical, science and financial applications. There are over 300,000 commercial licenses running on or designed into well over $500 billion in equipment[18]. The UCS report[18] also describes how DDS is used for the many military and commercial applications, such as Littoral Combat System (LCS), DDG 1000 Destroyer, General Atomics Ground Control Station, Airborne Warning And Control System (AWACS), Air Force Common Link Integration Processing (CLIP), UK MoD Generic Vehicle Architecture, Wind River Real-Time Operating Systems, CAE Flight Simulator, Air Traffic Control, Grand Coulee Dam, Medical Imaging, and Cancer Treatment.

These applications show that DDS is the standard to use for middleware when real-time processing requirements need to be addressed. There are several commercial products that can be used to create DDS-based middleware, for example RTI[19], PrismTech

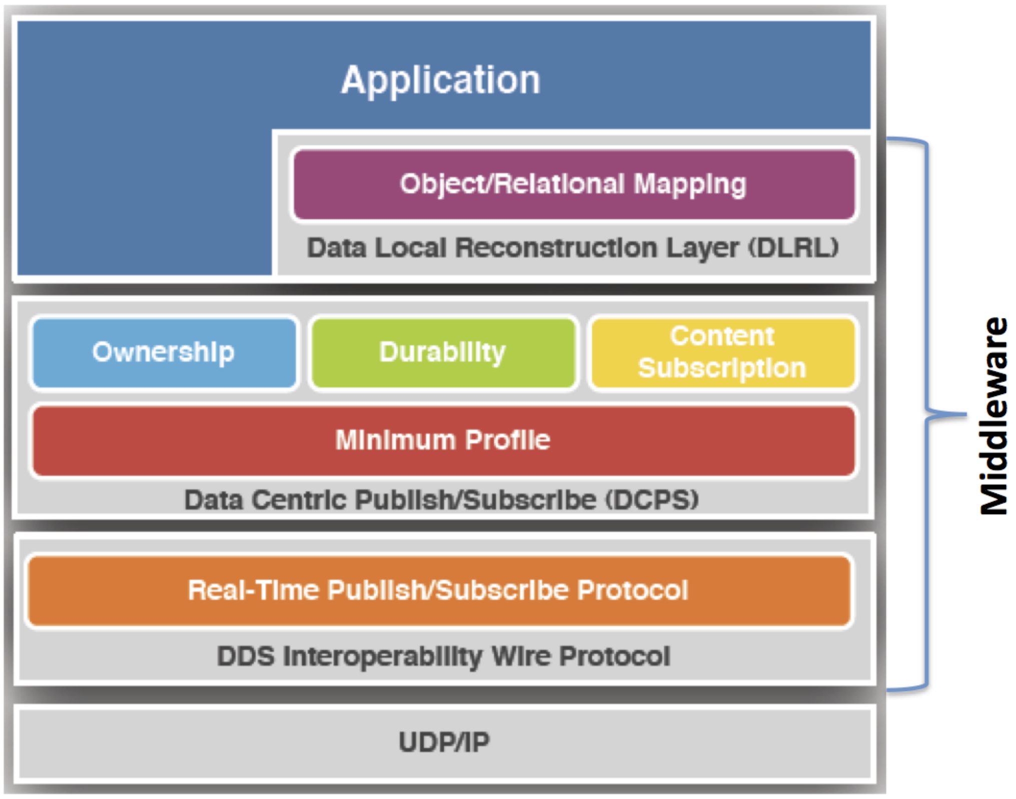

Application
Object/Relational Mapping
Data Local Reconstruction Layer (DLRL)
Ownership
Durability
Content Subscription
Minimum Profile
Data Centric Publish/Subscribe (DCPS)
Real-Time Publish/Subscribe Protocol
DDS Interoperability Wire Protocol
UDP/IP
Middleware

Figure 3.1: DDS Software Architecture

OpenSplice®[20], Twin Oaks Computing[21], and IBM JMS API[22].

Numerous tutorials on DDS are available[23, 24, 25], and the current specification[8] can be referenced as well. In summary, DDS can be used for interoperability by designing three software models: Communications Model, Object Model, and Architecture Model. As an implementation, these models conform with other protocol standards at Levels 3, 4, and 5 of the Open System Interconnect (OSI) communication model (e.g. IP, UDP, SCTP, SOAP, HTTP). Figure 3.1 is a software architecture illustrating DDS organization and functionality.

The DDS Communication Model operates within a publish-subscribe architecture. It contains a Shared Queue Space and a Global Data Space that are accessible to all application subscribers and publishers. This architecture can be implemented within the same computer or across a network. The Shared Queue Space contains cached message queues, and exchange modules to pass messages to the right queues. Every message has a key that is used to determine the proper queue. Subscribers are organized to receive messages from only one

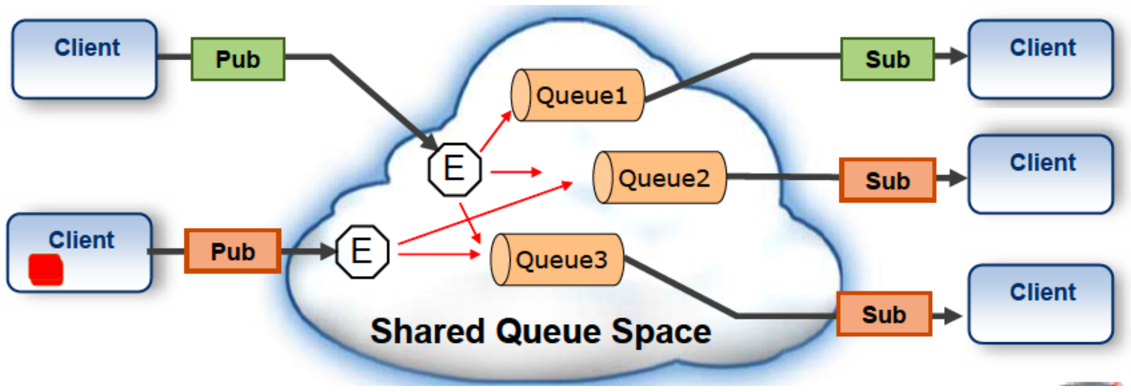


Figure 3.2: DDS Shared Queue Space[24]

queue. This interaction is illustrated in Figure 3.2. The Global Data Space contains objects for data, where the objects are organized by domain, topic, and key. Quality of Service is managed in the Global Data Space, and it is used for automatic discovery and configuration. Publishers and subscribers are organized into domains, and interact by either publishing to or subscribing from a topic.

The DDS Object Model defines the structure for data content and other attributes that are needed for information sharing using this standard. In DDS, a publisher is a data writer to a defined topic. The publisher can also offer Quality of Service associated with that data. A subscriber is a data reader for a topic, and can request a particular Quality of Service for the data. By conforming to the DDS object structure, data type complications are removed, while creating the means for data discovery. Organizing the data into domains and topics contains the discovery algorithm to only those applications that have an interest in the data. This algorithmic efficiency, in addition to brokered Quality of Service gives DDS its performance strengths, and preference as a middleware for real-time networking applications. Pardo-Castellote contains a chart in his tutorial [24] comparing the latency performance of DDS, JMS, and Notification Service. This comparison shows that latency with DDS is much better than JMS and Notification Service.

The DDS Architecture Model is unbrokered peer-to-peer publish-subscribe, which is implemented by commercial commercial DDS development products listed earlier in this section.

3.3 Relationship Between Middleware and Cloud Computing

There are two current, widely accepted definitions of Cloud Computing. The first from the National Institute of Standards and Technologies (NIST)[26] states that Cloud Computing is, “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”. A second definition from Gartner[27] states that Cloud Computing is, “a style of computing where massively scalable IT-enabled capabilities are delivered ’as a service’ to external customers using Internet technologies.” The key point from these definitions is that Cloud Computing presents a model of interacting with a network. It is a style of networked computing. Middleware is a set of services used by network applications for the sharing of data and information. Middleware would be used to implement a Cloud Computing environment.

Within Cloud Computing, there are three different service models referred to as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [26]. With SaaS, users (or clients) do not have the applications on their machines, but they make a request for information from the “cloud”. This request is accomplished by sending data across a network to a server that is hosting the application needed to perform the processing. Processing is done on the server and the requested information is returned to the client. In this case, middleware would be used to format the client information request, send it to the cloud server while enforcing quality of service requirements, and return the data to the requesting client. Commercial examples of SaaS include Salesforce.com[28], Google Apps[29], and IBM Smart Cloud[30]. With PaaS, the clients have the applications on their systems and the cloud is used as a distributed database to ensure that collaborations across the network are using consistent and timely information. A network of nodes all implementing DDS middleware is an implementation of a PaaS Cloud Computing model. Commercial examples of PaaS include Microsoft®Azure[31], Google App Engine[29], and Rackspace® Cloud Sites[32]. With IaaS, the clients have control over infrastructure resources, such as server storage, processors, and deployed applications. In this case, not only does the client request information, but also requests applications and infrastructure components to perform enduring processing functions. The “cloud” would manage and allocate all such requests.

IaaS middleware would be the implementation of the client requests from infrastructure resources. Commercial examples of IaaS include Amazon EC2[33] and Rackspace® Cloud Servers[32].

NIST also discusses four different deployment models, referred to as Private Cloud, Community Cloud, Public Cloud, and Hybrid Cloud. This aspect of Cloud Computing refers to scope of interaction across a network, and does not have any added relationships with middleware beyond what was discussed in the previous two paragraphs.

3.4 Classifying Middleware

Sadjadi and McKinley[34] performed a survey of adaptive middleware referencing three published taxonomies, two of which are introduced by the author, and a third proposed by Schmidt[35]. Schmidt categorizes adaptive middleware into four layers: host-infrastructure, distribution, common-services, and domain-services. Since the value of middleware depends on its use, this taxonomy would be unwieldy for purposes of this thesis, because Flexile Middleware is adaptive across multiple layers. One of Sadjadi's taxonomies suggests partitioning by the application domain. Initially, this appears like a useful approach, but the application partitions do not uniquely represent operational applications. For this partitioning, Sadjadi suggests QoS-oriented systems, dependable systems, and embedded systems. The critical missions most usefully serviced by Flexile Middleware contain all of these applications. Sadjadi's other categorization partitions adaptive middleware into four types: configurable, customizable, tunable, and mutable. The definitions of these terms create exclusive partitions. With these types of adaptive middleware, the services are determined a priori, but service functionality is adaptive. Mutable middleware allows for the most significant changes in middleware behavior that can occur while it is executing. Schmidt[35] developed a form of mutable middleware, called ADAPTIVE Service eXecutive (ASX), which was developed for use in distributed computing applications. Using the ASX framework, the services in the applications may be updated and extended without modifying, recompiling, relinking, or restarting the applications at run-time. Flexile Middleware is proposed as a fifth type of adaptive middleware referencing this one of Sadjadi's taxonomies as shown in Figure 3.3.

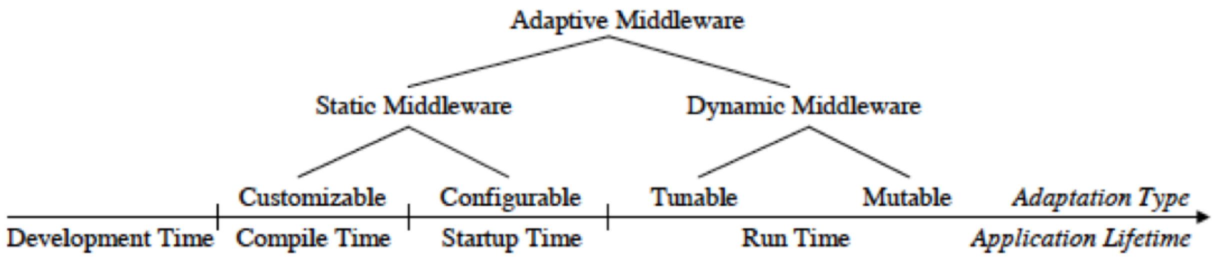


Figure 3.3: Sadjadi[34] Taxonomy of Adaptive Middleware

3.4.1 Static Middleware

There are five types of traditional static middleware: (1) Message-Oriented Middleware; (2) Transaction Processing Monitors; (3) Remote Procedure Calls; (4) Distributed Computing Environment; and (5) Object-Oriented Middleware[36]. Message-Oriented Middleware is characterized by point-to-point connections, asynchronous communications, message queues, and notifications. Messages have priority, sequencing, and expiration. Quality of service can be integrated using these characteristics. Transaction Processing Monitors are most commonly associated with banking and financial applications. Transactions implement business logic and are most commonly associated with distributed data bases. Remote Procedure Calls can be considered middleware because of their middle position in the software architecture of distributed systems. Remote Procedure Calls simply implement client-server functionality by allowing a client application to request processing and data on a server system, where the two are connected with a network. The programmer does not need to know too much of the networking details, but application details force a high degree of coordination between client and server programming. Distributed Computing Environments are similar to Remote Procedure Calls, but additional abstractions are included for interaction threads, file structures, and security. Object-oriented technologies provide the means to add a higher degree of abstraction to networked interoperability. This relationship gave rise to the creation of Object-Request Brokers (ORBs), and through standardization efforts DCOM, RMI, and CORBA emerged as implementations of ORBs as middleware.

The most significant static middleware development for critical applications was done by the U.S. Army for the Future Combat System (also known as Brigade Combat Team Modernization) program. Known as the System Of Systems Common Operating Environment

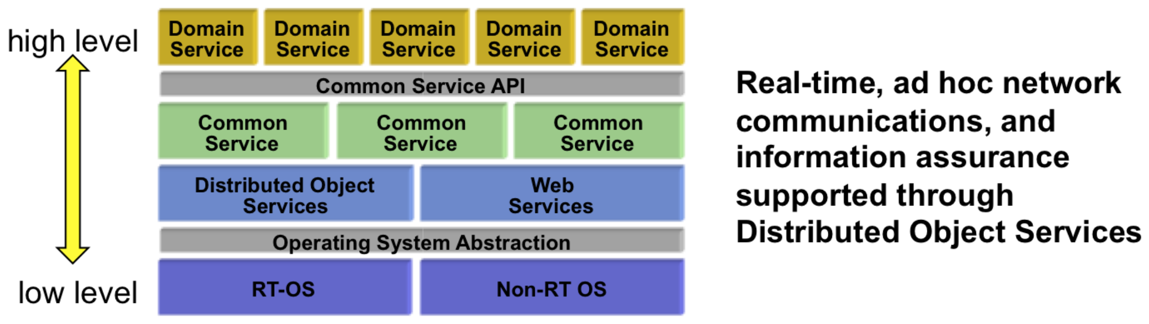


Figure 3.4: Software Architecture for SOSCOE

(SOSCOE)[37], it is an object-oriented middleware designed to support applications needed for new Army operational concepts that take advantage of networked connectivity. There are multiple editions of SOSCOE depending on user needs[38]:

- **Standard Edition:** used by applications similar to desktop and larger systems.
- **Real-Time Edition:** provides services similar to the standard edition but executes on a real-time OS and has more demanding real-time processing requirements.
- **Micro Edition:** used by applications that will execute on small platforms such as digital signal processors and handheld devices.

Figure 3.4 shows how SOSCOE is a set of services that architecturally sits between the operating system and the applications.

The services that comprise SOSCOE include the following:

- **Administrative Services:** Manages basic operations of the computer host, including logon, logoff, power management, and zeroize.
- **Knowledge Services:** Services and tools that enable domain applications and services to represent, manage, query, search, and interpret domain knowledge, and provide common representations across domains.
- **Information Services:** Services for the storage, retrieval and maintenance of geographic, environmental, track, symbology, entity characteristics, and mission information.

- **Communication Services:** Services that allow applications to communicate the right information at the right level of detail at the right time, based on the context of each user node, available bandwidth, information assurance constraints, quality-of-service, and current user policies. Includes discovery, naming, and dissemination services (e.g. publish/subscribe), and delivery mechanisms.
- **Data Store Services:** Services that support the creation and maintenance of virtual persistent data environment whereby any node (e.g. unit, vehicle, or footprint) has access to both local and remote data objects or links to those objects.
- **Information Assurance:** Provide services to protect, secure, and assure information used by applications and services within/between nodes (e.g. unit, vehicle, or footprint). Includes identification, authorization, authentication, encryption, intrusion detection and prevention, and anti-virus services.
- **Interoperability:** Facilitates information dissemination in the right format and over the right communications path. Includes parsers for message format translations.
- **System Management Services:** Supports management of middleware and applications. Enables the ability to automatically update to new versions.
- **Analysis Services:** Provide services for automatic text processing of very large volumes of formatted and free-text messages. Also includes data mining services.
- **System Services:** Includes basic operational services, such as recording, playback, time distribution, and event handling.
- **Configuration and Control:** Provide services for application configuration and system configuration. Includes services to monitor, control, and manage system resources (e.g. health and usage monitoring) and processes as allowed by policy.
- **I/O Services:** Services for higher level control of device drivers.
- **Web Services:** Abstraction of services to enable and use data from internet web-based sources.

- **User Profile and Policy Services:** Services that allow for the management of user profiles and system policy information based upon preference and role. As the role or preferences are modified, the user profile is updated to coordinate the information received by the user.

- **OS Abstraction:** Provide an operating system (OS) abstraction that encapsulates OS services used by services and applications. Abstraction includes inter-process communication (IPC), process/thread, concurrency mechanisms, memory management, and graphical interfaces.

This list shows that there is a broad range of service functionality applicable to middleware for critical applications. One example of the computational complexity of middleware services is SOSCOE Interoperability Services. These services must be able to interpret many different messaging standards from legacy systems that are needed to conduct operations effectively. Figure 3.5 shows the various systems for which SOSCOE needs to provide interoperability.

3.4.2 Adaptive Middleware

One form of adaptive middleware adjusts to the context of the operational situation. According to da Rocha, et al.[4], middleware needs to satisfy 12 requirements to support ubiquitous context-awareness. They are:

1. Distributed context management
2. Support for context evolution
3. Dynamic context discovery
4. Scope of context perception
5. Multiple mechanisms and policies for accessing context
6. Extensible abstractions for accessing and using context
7. Management of application dynamic loading
8. Abstract handling of context interest

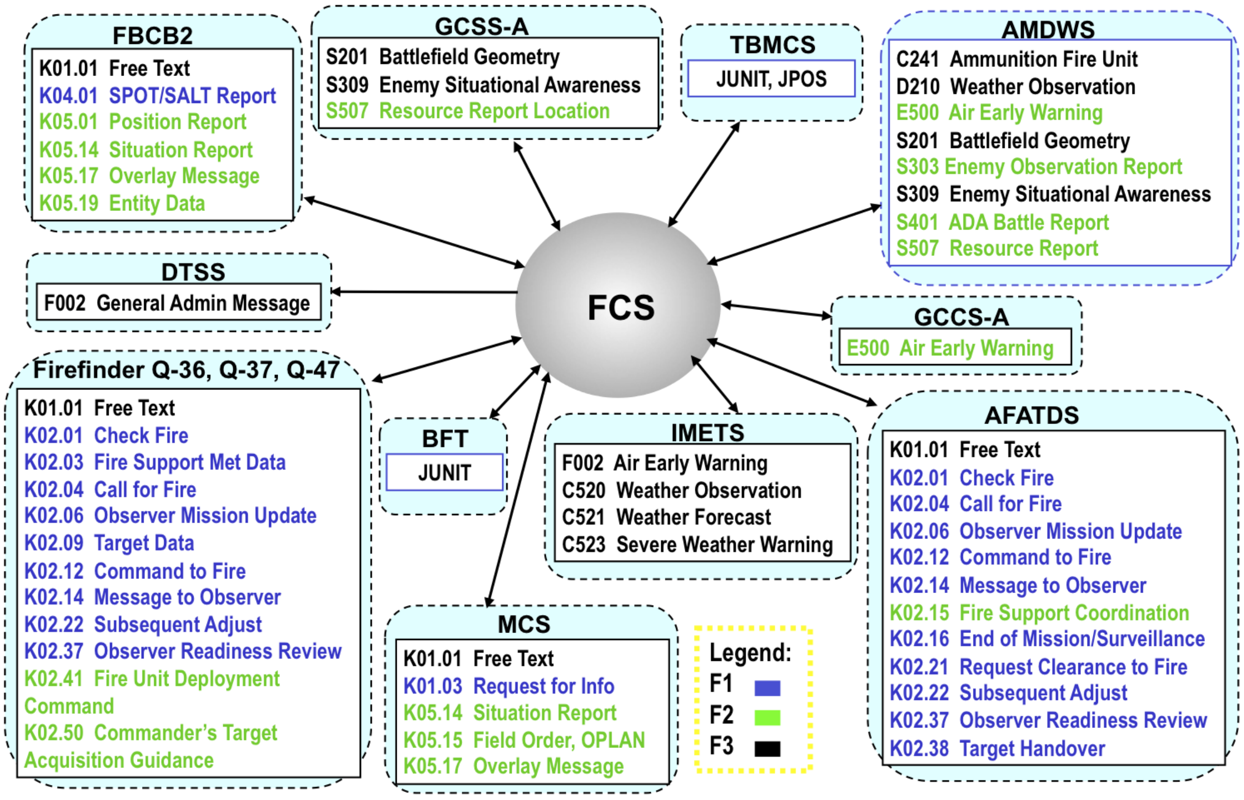


Figure 3.5: SOSCOE Interoperability Services

9. Architectural independence
10. Decoupling between context management and inference mechanisms
11. Easy incremental deployment, distributed administration and standardization
12. Suitable programming tools for context discovery

Sacramento, et al.[39, 40], describe Mobile Collaboration Architecture (MoCA), which is a middleware for developing and deploying context-aware collaborative applications for mobile users. It addresses the 12 requirements and contains adaptive services to optimize application performance based on context information, for example geographic proximity. It comprises client and server Application Program Interfaces (APIs), core services for monitoring and inferring the mobile devices context, and an object-oriented framework for instantiating customized application proxies. It contains several adaptive middleware services that change their behaviors based on inferred user intent. Those services are: Context Information Service (CIS), Discover Service (DS), Configuration Service (CS), and Location Inference Service (LIS).

Some middleware are considered adaptive because they adjust their performance to the form of the network. Yu, et al.[3] describe an adaptive middleware that is used by a distributed sensor application. The sensors are placed randomly and need to communicate with each other. The adaptive middleware contains a set of eight components with functionality that can adapt to the environment and configuration of the network to optimize performance. One of the components is a prediction model. By focusing on quality of service with user preferences, another related form of adaptive middleware has been described. The application in this case is a distributed visual tracking system. Li, et al.[1] present an adaptive middleware architecture as a way of providing quality of service where it is not an integral part of the networking. The architecture consists of middleware modules that take user preferences and monitors the network to create an adaptive behavior for the application to optimize performance. This form of adaptability uses information about the application domain to determine how to adapt to changes in the network structure itself. Flexile Middleware has a performance monitoring module to provide a similar form of adaptability based on changes in network characteristics. This will be discussed more in the following chapter.

Other adaptive middleware adjust performance to the characteristics of the users. One example of this is online virtual-reality style gaming. Balan, et al.[41] developed four middleware components that can be used by game developers which comprise a system called, Matrix. These components adapt to the configuration and usage profiles of the gamers playing against each other during a session. Flexile Middleware also contains the ability to provide some adaptation to user characteristics through the inclusion of a module on usage statistics. This will be discussed more in the following chapter.

Another form of adaptive middleware uses quality of service to determine how to modify network services. Othman, et al.[2] discuss this type of adaptive middleware which uses load balancing across a network as the quality of service objective function for adaptation. Othman describes the use of this adaptive middleware for general distributed applications, such as server transparency, decentralized load balancing, stateful replicas, diverse load monitoring granularity, fault tolerant load balancing, extensible load balancing, and on-demand replica activation. In this case, these functions are being designed into commercial CORBA[16] products with stated application to other middleware systems. An application of this CORBA-based adaptive middleware has been tested in a program called Cygnus[42], which has been shown to reduce network overhead when scaling up applications.

4

Flexile Middleware

Flexile describes something that is able to flex or bend easily[43]. Synonyms would be malleable, flexible, and elastic. This thesis introduces a new class of autonomic middleware called Flexile Middleware. It has both static and dynamic features. Flexile Middleware is static by containing a set of core services that must be present in all processors across the network. It is dynamic in multiple ways. First, knowledge of the operating environment and mission objectives will be used to set parameters used by the middleware. The services themselves would not change, but they would have adaptive qualities resulting from domain knowledge embodied in parameters used by the middleware. A second level of adaptive behavior occurs when the services themselves need to change based on user intent and the ability to fit applications with associated services within the hosts' processing capabilities. It is this second type of adaptive behavior, which is the inspiration for the term, Flexile Middleware. Middleware that is flexile has the following characteristics:

1. Adaptive in both performance and size of executable module;

2. Service content determined by prioritized net-ready applications;

3. Domain-specific rules that determine application priorities;

4. A performance monitor that runs periodically to assess performance versus accomplishment of mission objectives;

5. Ability to reconfigure the mix of middleware services based on usage statistics, learning rules, inferred user intent, and/or explicit user requests.

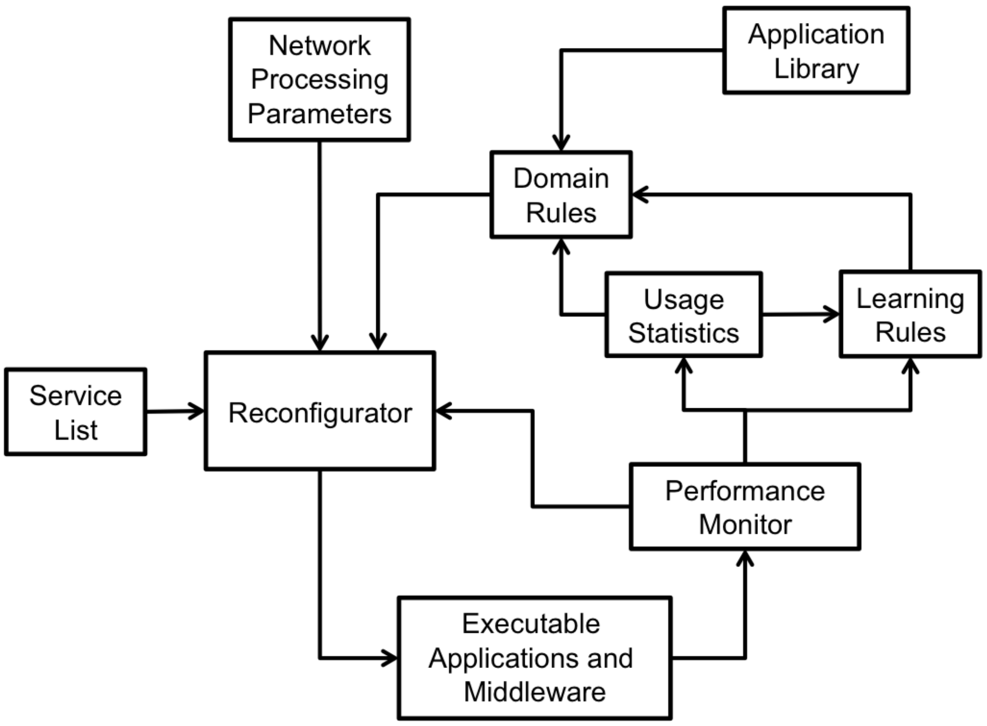


Figure 4.1: Functional Block Diagram of Flexile Middleware

A functional block diagram of Flexile Middleware is shown in Figure 4.1. The blocks in this figure are the outline for the rest of this section. Summarized below is a brief description of each of the components of Flexile Middleware:

- **Network Processing Parameters** – These parameters are specific to the allocation of resources on host hardware to support network functions. They are used by *Domain Rules* to determine the applications to install on the host hardware.

- **Application Library** – This is a database containing the complete set of available net-ready applications. It may reside in secondary storage on the host hardware, or it may reside on a server reachable across the network.

- **Domain Rules** – This is a knowledge base designed to determine which applications are the best ones to install on the host hardware. The rules themselves may change based on how the user intends to use the network. This information may be explicitly provided by the user or inferred based on usage and heuristically determined user intent.

- **Service List** – This is a database containing the complete set of middleware services available to all of the net-ready applications.

- **Executable Applications and Middleware** – This is an executable image of the net-ready applications and middleware services. The *Domain Rules* use a priori information about the applications and middleware to ensure that, when the executable is created, it will fit in the available memory, execute within real-time requirements, and incur allowable bandwidth utilization.

- **Performance Monitor** – This module monitors system usage and occasionally calls the *Reconfigurator*. The call to the *Reconfigurator* can be done periodically, or it can be triggered based on certain events.

- **Usage Statistics** – This is a database created by the *Performance Monitor*. It is used to determine how the system is being used and to ascertain automatically user intent for network usage.

- **Learning Rules** – This is a knowledge base governing the extent to which the *Domain Rules* can change, and specifically how to change those rules.

- **Reconfigurator** – This module uses *Network Processing Parameters* and executes the *Domain Rules* to define a new set of applications for the device. It is triggered by the *Performance Monitor*, and can operate periodically or by events. The *Reconfigurator* maintains a list of the currently running applications and middleware services, and executes the code needed to change the active software. It also references a database called the *Service List*, which contains a mapping the services need for each of the applications. This module implements much of the functionality that gives Flexile Middleware its unique capabilities.

With reference back to Section 3.4 and Figure 3.3, this more detailed understanding of Flexile Middleware provides the basis for the assertion that it deserves its own classification of adaptive middleware. Within Sadjadi's taxonomy, Flexile Middleware would be a new subclass of Dynamic Middleware. Flexile Middleware has both static and dynamic features, which is common among all dynamic middleware. The other two subclasses of Dynamic Middleware are Tunable and Mutable. Tunable Middleware dynamically changes in multiple stages with administrator fine tuning. As such, its dynamic features are not automatic like Flexile Middleware and would require administrative review which is not practical for real-time applications. Furthermore, with Tunable Middleware, there is no built-in understanding of an operational context. If this type of information were used, it would need to come from the administrator, which is very different from Flexile Middleware. Mutable Middleware relies on "meta models", which are pre-defined. Events, or a logical combination of events, can cause Mutable Middleware to change states according to one of the meta models. In contrast, Flexile Middleware does not use pre-defined meta models. Furthermore, the dynamic nature of Mutable Middleware is deterministic. Flexile Middleware has a learning component that provides more variability to how it dynamically changes. If the learning element of Flexile Middleware is not used, it is still not limited to a specific set of predefined meta models to change states. These fundamental differences justify Flexile Middleware as its own class of Adaptive Middleware. Relative to Sadjadi's taxonomy, it would be a third subclass of Dynamic Middleware.

Flexile Middleware flow of control is shown in Figure 4.2. The *Performance Monitor* monitors data collected by applications running on the System. When events are matched, then data is provided to the *Usage Statistics* module. This module calculates statistics and

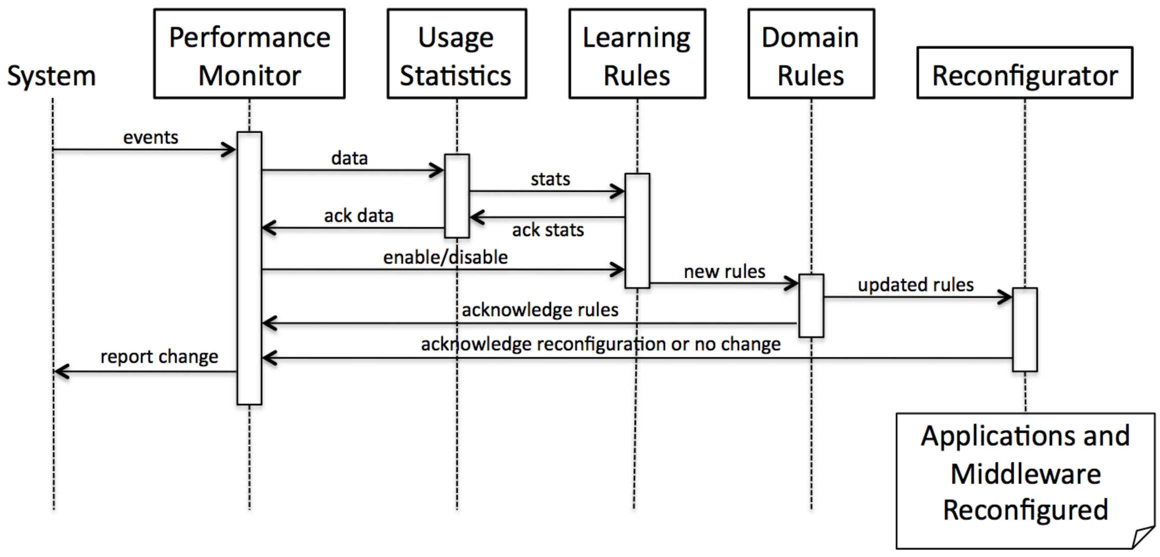


Figure 4.2: Sequence Diagram of Flexile Middleware

provides that data to the *Learning Rules* module, and acknowledges completion back to the *Performance Monitor*. The *Performance Monitor* knows whether *Learning Rules* can be run during operations, and provides a command to either create new rules and send them to the *Domain Rules* module or simple call *Domain Rules* without any updates. Either way, the *Domain Rules* module calls the *Reconfigurator* module which executes the instructions to implement updates to the mix of applications and middleware running on the System, and then acknowledges completion to the *Performance Monitor*, so it can be ready to continue monitoring the System. If the *Reconfigurator* changes the mix of applications, the System is notified. If it is a user-based System, that notification could be in the form of a display change.

4.1 Network Processing Parameters

Network Processing Parameters are used by the *Reconfigurator* to determine the set of applications that can be run on the host system. This would be the application load. Example network processing parameters are:

- **Host Memory** – These parameters contain the maximum amount of memory that can be allocated to network processing, and how much is currently allocated.

- **Latency** – This parameter relates average response time to applications. It is implemented as a two-dimensional matrix of application versus latency. Other Flexile Middleware applications can change the values of this matrix dynamically during operation. In cases where latency is unacceptable, application priorities would need to be reassessed.
- **Bandwidth** – Each application is expected to have an impact on bandwidth. These parameters define the amount of average bandwidth that can be expected to be used by the current load of applications, and what is available for the possible addition of other applications.
- **Network Role** – In an ad hoc network, it is possible for nodes to be both intermediate and end nodes in the resulting topology. If the network is mobile, this role can change dynamically. However, depending on other parameters, it may be necessary to assign a node to be predominantly intermediate or an end node.
- **Video Support** – Streaming video across a network is a major concern, and it should be treated specially. Video support parameters would state how many video streams the host can handle simultaneously for display without degraded performance, and if acting as an intermediate node, it would define how many simultaneous streams can be routed through it. Degraded video performance is a subjective measure itself, which can also be a parameter. For uses when high quality video is required, degraded video performance can be defined to be less than 500 kbps for nodes displaying that video.

Many of these parameters can be implemented as DDS^{TM} Quality of Service features. For modeling purposes, the effect of *Network Processing Parameters* is embedded in assumptions about which applications are running on which nodes. To differentiate Flexile Middleware from other forms of adaptive middleware, this will be seen in the inability of adaptive middleware nodes to add in new and desirable applications dynamically, which are critical to mission accomplishment because of resource limitations. Flexile Middleware would not have an effect on resources, but could use *Network Processing Parameters* to prioritize applications leading to the most useful set of applications to maximize the probability of mission success.

4.2 Application Library

The *Application Library* is a database of network applications with associated parameters. Four applications were discussed in Section 3.5: Video Dissemination, Networked Weather, Tactical White-Boarding with Chat, and Common Operating Picture. Other potential applications are as follows:

- **Uncharted Objects** – For critical missions, it is reasonable to assume that there will be new objects constructed or destroyed within the operational area. Sensors on some vehicles have the ability to sense objects with geographic precision, but such sensing capability is not available with all assets collaborating to accomplish a mission. The ability to share uncharted objects across a network could improve operational effectiveness.

- **Logistics Management** – Critical missions often depend on supplies. Knowing which supplies are where will have an impact on operational effectiveness. This real-time tracking is best done through networks, and it would include people in addition to cargo.

- **9-Liners** – This term refers to specially formatted messages traditionally sent using 9 lines. As an application, it is intended to address surprises that could occur during critical missions, such as an unforeseen search-and-rescue or injury. information in the 9-liner contains the type of event, where it happened, and other details associated with the type of event that occurred.

- **Connectivity Optimizer** – This application is intended for UAVs or other assets whose primary mission is to provide connectivity among the nodes collaborating to accomplish a mission. This application knows the changing network topology and creates flight paths for the host to maximize connectivity between any two nodes on the network. It is a dynamic algorithm. Connectivity can be accomplished through multiple hops, and this application would be periodically updated with network topology information towards the goal of maximizing the number of possible pairwise connections. This application is particularly useful with line-of-sight communications in an operational environment with obstructions impeding line-of-sight. For this application to work properly, real-time networking information is necessary.

- **Event-Driven Imagery Dissemination** – In cases where video is not practical due to host resource limitations or the desire to not significantly impact human workload watching the video, this application can provide utility. Users of this application can set predetermined points where the onboard system will take a snapshot and post the result to the network for others to see. Other users can subscribe to these postings. In addition to predetermined locations, imagery can be posted by user selection or from some other event germane to accomplishing the mission. Captured imagery can be automatically disseminated or stored on the host awaiting human "mark ups" prior to dissemination.

- **Networked Fires** – Some critical missions require the use of lethal and non-lethal weapons. This application can use information collected from multiple sources to provide a more precise firing solution or allow greater standoff ranges between the targeting asset and the firing asset.

In addition to containing the executable code for the networking applications, the *Application Library* also contains parameters that the *Network Processing Parameters* database needs for dynamically updating the networking state on the host. Each application also references middleware services it needs to operate properly. These references are used to create the Service List. From a modeling perspective, these applications will cause discrete decisions to be made relative to how the mission is performed. Branch points in the models will be used to represent the effects of these applications running on the host.

4.3 Domain Rules

Domain Rules indirectly impact the middleware by controlling which applications are in memory. When *Domain Rules* are triggered by events, the list of running applications could change. When this happens, Flexile Middleware will access the *Applications Library* to determine which middleware services need to be available. This information is passed to the *Reconfigurator*, which will implement the change. Example *Domain Rules* inspired by the evaluation scenarios described in this thesis are as follows:

- If (Radiological Sensor exceeds a Safe Level) then (Load Flight Re-Planner Application) and (Publish Message)

- If (User Commands State Change to Relay) then (Load Communications Relay Application)
- if (On-Board Sensors Detect Environmental Changes) then (Load Network Weather Application)
- if (Search-And-Rescue Message Received) then (Load SAR 9-Liner Application) and (Load Flight Re-Planner Application)
- if (User Commands Person or Cargo Pick-Up) then (Load Logistics Application)

Notice that the results of the *Domain Rules* are all load actions. The *Reconfigurator* has the job of actually loading the applications. It is done this way because the *Reconfigurator* has knowledge of the processing capabilities of the host and maintains a priority listing of the applications. Results from the *Domain Rules* are used by the *Reconfigurator* to adjust the application priorities, and perform the reconfiguration.

4.4 Service List

All middleware-based networking applications require services. The services provide needed isolation between the applications and network stack on the host. Services implement publish, subscribe, broadcast, quality of service, and other capabilities listed in Chapter 3 for middleware. The *Service List* is constructed from a mapping contained in the *Application Library*. There is no direct modeling of the *Service List*, but its functionality is essentially modeled as part of the *Reconfigurator*.

4.5 Executable Applications and Middleware

Executable Applications and Middleware is the code and memory that is run to obtain desired network performance and functionality. With Flexile Middleware, this will change during missions as determined by *Domain Rules*, *Learning Rules*, and the *Reconfigurator* using information provided by the other Flexile Middleware modules. Multiple processing cores can be used to maintain both an active operational system and one that represents a different operational system designed for better performance. This is discussed more in the *Reconfigurator* section.

4.6 Performance Monitor

The *Performance Monitor* normally runs periodically, but can also run asynchronously based on critical events. Those events could be matched to some set of monitored mission parameters or direct user input. Monitoring data is collected in the *Usage Statistics*, and information is also passed from the *Performance Monitor* to the *Learning Rules*. The *Performance Monitor* triggers the *Reconfigurator*, which in turn could cause the Flexile Middleware to change operational states. It is important that the cycle time for the *Performance Monitor* is always greater than the cycle time for executing the sequence of *Learning Rules*, *Domain Rules*, and *Reconfigurator*.

4.7 Usage Statistics

Usage Statistics are collected each cycle by the *Performance Monitor*. This information is used by the Learning Rules to improve the performance of Flexile Middleware. Performance is associated with the parameters of Flexile Middleware itself, not with the domain application or rules.

4.8 Learning Rules

Learning Rules are defined to improve the performance of Flexile Middleware over time. Time in this case could be dynamically changing during operation or between operations. Example conditions for learning rules are as follows:

- *Reconfigurator* changes states too often or not often enough.
- Applications in memory not used enough or at all for a given operational state.
- A non-core application functions more like a core application for the same type of mission.
- *Reconfigurator* frequently alternates between two different operational states before settling on a sustained state.

If a learning rule triggers during operation, its impact on other modules is a change to a Flexile Middleware operational parameter, such as *Performance Monitor* or *Reconfigurator* frequency. Since *Learning Rules* use information that is collected over multiple missions, its data must be loaded and saved from a non-volatile source. This information can be reviewed by human operators to make direct changes to Flexile Middleware parameters, such as the *Application Library*, *Service List*, and the *Learning Rules* themselves. It is possible, but not desirable, for *Learning Rules* to cause changes in the logic of *Domain Rules*. If such an adaptation is desired, it will be necessary to recompile the *Domain Rules*, including a topological sort of the rule base. The result of this compilation would be used to recalculate the cycle time for the *Performance Monitor*. This is necessary to ensure that the cycle time of the *Performance Monitor* is always more than the cycle time for executing the sequence of *Learning Rules*, *Domain Rules*, and *Reconfigurator*.

4.9 Reconfigurator

At the core of Flexile Middleware is the *Reconfigurator* module. This module executes the *Domain Rules*, and if the result demands a change in operational state, it sets the *Executable Middleware and Applications* to the set defined for that state. Information about which middleware services and applications are needed for each operational state is calculated by the *Reconfigurator* using information contained in the *Application Library* and *Service List*. The *Reconfigurator* knows which applications and middleware are currently executing, and how to make the transition to a new memory state. A flowchart illustrating the logic of the *Reconfigurator* is shown in Figure 4.3.

The *Reconfigurator* contains a rule-based inference engine to execute the *Domain Rules*. Since the goal is to minimize processing latency, the best implementation for this inference engine is to compile it directly into executable code. The rules can be sorted topologically so that only needed rules are executed each time there is a change to a rule condition. The topological sort on the rules also determines a maximum rule chain length, which can be used to determine the latency for execution of the *Domain Rules*. This value is necessary because it contributes to the cycle time for the *Performance Monitor*.

A prioritized list of applications is the result of executing the *Domain Rules*. The *Reconfigurator* first compares the new list to the list of currently executing applications. Order is not important for this check. If the new list is different, then the *Reconfigurator* calculates

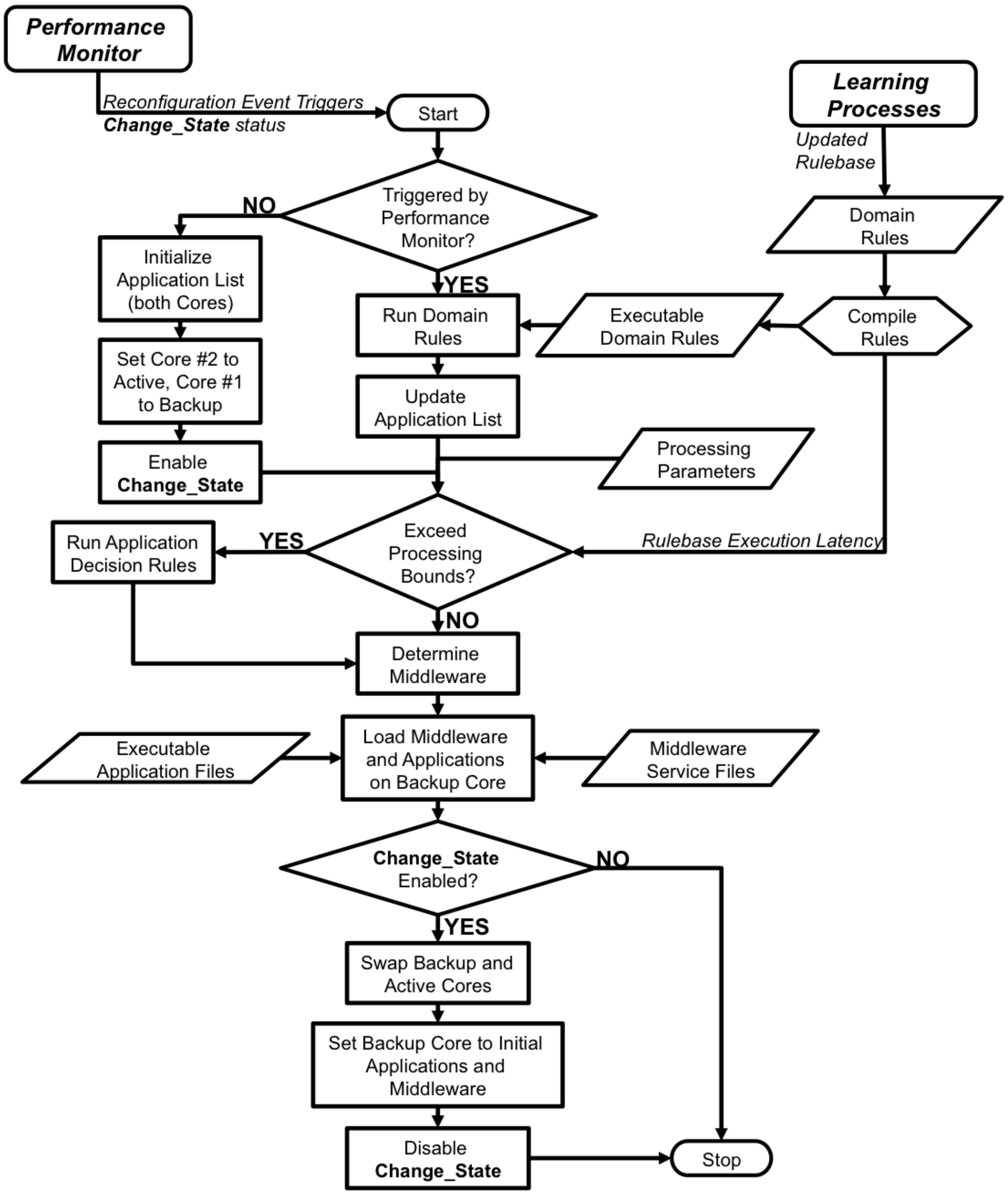

Performance Monitor
Reconfiguration Event Triggers
Change_State status
Start
Learning Processes
Updated Rulebase
Triggered by Performance Monitor?
NO
YES
Domain Rules
Initialize Application List (both Cores)
Run Domain Rules
Executable Domain Rules
Compile Rules
Set Core #2 to Active, Core #1 to Backup
Update Application List
Enable Change_State
Processing Parameters
Run Application Decision Rules
YES
Exceed Processing Bounds?
Rulebase Execution Latency
NO
Determine Middleware
Executable Application Files
Load Middleware and Applications on Backup Core
Middleware Service Files
Change_State Enabled?
NO
YES
Swap Backup and Active Cores
Set Backup Core to Initial Applications and Middleware
Disable Change_State
Stop

Figure 4.3: Reconfigurator Flow Chart

a new list of applications in priority order while ensuring that the cumulative impact of each application does not exceed parameters defined in the *Network Processing Parameters* database. If the addition of an application causes a parameter to exceed its limit, then the rest of the applications are checked in order to see if any others could fit. Special decision rules can be implemented for the last stage of building a new application list. These special decision rules will be used to determine if it is better to add one higher priority application compared to multiple lower priority applications. These are best implemented with an understanding of the mission, and could also be dynamically changed as a result of executing the *Learning Rules.*

By using certain *Domain Rules*, the *Reconfigurator* can be used to provide additional redundancy for networked systems. For example, suppose several nodes in a network do not have the resources to run a complicated application and these nodes rely on one other node that can run the complicated application and provide the results that the other nodes need. If the complicated application is critical to accomplishing the mission and the node running that application becomes disconnected, a rule for reconfiguration could be used to load the critical application on another node that is connected to the network. It may also be desirable to ensure that at least two nodes on a network can run mission-critical applications, so there is less of a chance that a reconfiguration is needed to bring a critical application back into the network.

The *Reconfigurator* uses a variable called the Application Load, which is the list of applications currently running on the network host. This list changes dynamically with a Flexile Middleware implementation. From a modeling perspective, the models will reference this list and select predetermined branch points depending on whether the application is running or not, or if it is running, whether it has the right information to perform its expected functionality. The *Reconfigurator* also maintains a variable which is the list of currently loaded middleware services called the Service Load. It is the subset of services from the *Service List* that needs to run on the host to support the Application Load. The Service Load will impact *Network Processing Parameters*, because they effect memory, latency, and bandwidth, as well as other quality of service features. Similar to the *Service List*, there is no direct modeling of the Service Load, but its functionality is essentially modeled as part of the Application Load.

Real-time performance is a requirement for critical applications. Accordingly, it is not practical to cause a processing delay by swapping all the applications and middleware in memory. *Reconfigurator* takes advantage of multi-core architectures where one core contains the active code (Active Core) and the other contains the code for the transition state (Backup Core). This provides a seamless transition to a new state without interrupting the processing. When the state transition is made, the two cores simply swap roles and the previous active core is reloaded with the applications and middleware services common to all operational states. The Active Core interacts with users and the network. The Backup Core is running in the background, but it does not have direct access to users and the network. However, the Backup Core can also process user and network data, if it is the same information needed by the Active Core. This methodology minimizes the latency when Active and Backup Cores swap roles. The OpenMP® organization has an Application Program Interface [44] that supports this type of parallel processing in multi-core architectures. The *Reconfigurator* determines the content for each core and determines which core is the active one. Numerous compiler vendors and communities support the OpenMP® API, such as Gnu, IBM, Oracle, Intel, Microsoft®, and Cray. Supported languages include C, C++, and Fortran.

5

Metrics and Measurements

This thesis presents a new class of middleware intended to improve operational performance in critical missions. Operational performance is measured by probability of successfully accomplishing the mission. The challenge is to identify and measure parameters directly associated with accomplishing a critical mission. An obvious choice is time. For example, it is reasonable to assume that if a person who is stranded and injured is not provided initial medical care within an hour, then a search-and-rescue mission can be declared unsuccessful. Also, there is a probability that some mission-ending event may take place, which is not associated with the actual time of the mission. This chapter will show how these parameters will be measured to show the operational value of Flexile Middleware.

5.1 Measurement Methods

The most important contributor to operational performance is situational awareness. The cognitive processing of information creates individual situational awareness. Endsley [45] defines a term called Team Situational Awareness which represents the collective situational awareness of a group collaborating to accomplish a mission. Since there are differences in how information is communicated during networked operations, there is variation in situational awareness, ultimately leading to differences in operational performance applicable to both individuals and teams. Therefore, methods to measure situational awareness are useful to measure operational performance.

Endsley et al. [46] summarizes and compares numerous methods of situational awareness. A subjective evaluation was used to compare methods using questionnaires. Other approaches have attempted to use more objective measures, such as those connected to image quality ("blobby shading" and Delaunay triangulation) [47]. These methods have been proposed to measure individual situational awareness, but others, e.g. [48], have extended situational awareness measurement to entire theaters of operation involving numerous military units.

An individual approach is used to measure the operational value of Flexile Middleware compared to other methods of networked communication, because if one individual shows improvement in operational performance based on the type of networking then all others collaborating to accomplish the mission will show improvement. Purely objective methods may be applicable to a small segment of situational awareness, but the involvement of cognitive processes makes objective measures unreliable and inconclusive for measuring mission-level situational awareness. Otherwise, there would be a generally accepted measure of mission-related situational awareness, which there is not.

Conducting numerous missions with multiple different networking methods and then interviewing users afterwards may provide a reasonable subjective measure of situational awareness but not practical. Simulations with subsequent questionnaires similar to how Endsley [45] measured subjective situational awareness may also be possible but also impractical for lack of operational experts. Overby et al. [49] suggest a new approach to measuring situational awareness based on temporal and spatial metrics. Their metrics use subjective assessments to complement temporal and spatial calculations. This approach shows promise for the purpose of measuring the operational value of Flexile Middleware compared to other methods of networking however, the calculation of spatial and temporal metrics require multiple "peers" within a distance threshold and a measure of perceived position to actual position. There may not be enough peers to calculate a reasonable metric and positional measurements depend on many random factors, such as GPS and elevation data accuracies.

A combination of subjective and objective measurements appears to be a good approach, but unlike Overby et al. [49], the approach used here employs subjective assessments of mission routes, decisions, and expected information content of the various comparative networking methods. Subjectivity is used only during mission simulations. The different ways to accomplish a simulated mission were analyzed in terms of time and occurrence of a mission-ending event. This data was sufficient to collect statistics about probability of accomplishing the mission within a certain period of time and the probability that a mission-ending event could occur. The availability and quality of information was used to distinguish each of the networking methods based on reasonable assumptions related to the particular method. These assumptions will be described in the sections that follow. Essentially, the selected measurement methodology consists of: (a) mission modeling, (b) assumptions about information availability and quality, (c) conducting the same mission multiple times for each networking type, (d) grouping route segments by similarity, (e) building a statistical model of mission time and problematic events, and then (f) calculating an expected probability of mission success. The situational awareness aspect of operational performance is contained in information assumptions and expert decisions based on those assumptions.

5.2 Introduction to the Selected Measurement Method

The first step of this modeling is to define cases to show successive improvement by adding features leading up to capabilities associated with the use of Flexile Middleware. The initial baseline case would be communications using voice alone. This is the most common form in use today. The next increasing level of communication capability involves simple data connectivity in addition to voice. This could be through accessing a data link or other point-to-point connection between collaborating assets. The next increasing level of capability is a network. For this case, a network would be wireless and ad hoc, including a middleware implementation that could be adaptive in the classical sense as described in Chapter 3. Flexile Middleware is the fourth case, and it would have more context-sensitive capabilities compared to conventional middleware. This approach was first presented by DuBois, et al. [50]. Relative to that paper, Flexile Middleware represents an example of Advanced Networking.

The next step of modeling involves the assignment of assumptions relative to each of the four cases. Fundamentally, the methodology seeks to calculate the probability of making a decision based on information times the probability of receiving that information. Simulations and expert judgment are used to estimate these values relative to each of the communication methods. These assumptions need to be reasonable given the mission, and are described generally as follows:

(A) **Voice-Only** – In this case, most communications are associated with updating progress according to an original plan. If something significant happens during the mission, there is a low probability that the information will be used quickly enough to make the changes necessary to deal with that event. Voice messages are subject to misinterpretation and errors, more so than the other three cases. For this case, the expectation is that there is a lower probability of receiving information compared to the others , and also a lower probability of acting on that information.

(B) **Simple-Connectivity** – In this case, information is mainly limited to simple communication between pre-configured nodes on a point-to-point basis. Useful mission information that is known by other than these connected nodes are subject to the same deficiencies as the Voice Only case. In contrast to Voice Only, there is a reasonable chance of getting useful information in time to do something about it, and a low probability of misinterpretations or errors. For this case, the expectation is that there may be a slightly higher probability of receiving useful information compared to Voice-Only, and a slightly higher probability of acting on that information because it would be in a more actionable format.

(C) **Network-Connectivity** – In this case, data can pass from any sender to any receiver provided that a path of connections between them exists. It is reasonable to assume that with the right applications and middleware that some domain-aware processing is possible to help make better choices. However, there is some risk that unpredictable events in the operating environment will create the situation where the best applications may not be available to certain network entities that would use them. For this case, the expectation is that network connectivity makes the probability of receiving new information relatively high, and the digital format would make it useful for making decisions. However, there is still a possibility that the right applications may not be present to convert that information into something actionable.

(D) **Flexile-Middleware** – This case is similar to standard Network Connectivity, but with a higher probability that the right software applications are running on the network entities that need them. It would exhibit more context-aware processing compared to standard Network Connectivity based on the premise that processing and network capabilities are limited. For this case, the expectation is that the probability of receiving information would be the same as Network-Connectivity, but it would have a higher comparative probability of acting on that information, because of a higher probability that the right applications were available to process that information into an actionable form.

The third step is to construct an example to test the measurement methodology and then extrapolate that example to a more generic situation to make predictions about the operational performance improvements offered by Flexile Middleware.

5.3 Example Setup

An example is used to construct a measurement methodology. Consider a search-and-rescue mission with potential complications associated with a spreading radiation cloud from a nuclear accident. A helicopter needs to transport injured people from a position near the accident area to a hospital. There are unmanned air vehicles carrying radiological sensors that are available to participate. The team on the helicopter is equipped with medical staff capable of stabilizing the injuries for transport from the pick-up location to a hospital. Based on history, the helicopter team knows that their best chance for success is to meet the injured people within 60 minutes from the start of the mission. To obtain a reasonable set of assumptions to be used in calculations, this mission was tested using the STK® constructive modeling software.

Several observations were evident from running the STK® mission model. Approximately 20 minutes from the start of the mission, information about the spreading radiation cloud might be available to establish a better route to the injured people. For Case A, it is doubtful that this information would be available to the helicopter crew. For Case B, this information may be available about half the time, but the capability of preparing an optimal re-plan is probably not possible. The indication would be only that there is risk of approaching too close to the cloud. For Case C, there is a very good chance of getting the information, but video data showing the progression of the cloud may not be available, because processing this data has a significant impact on computing and network resources. Case D has a high probability of figuring out a best re-plan to avoid the cloud and make the pick-up in time. A second decision point occurs if the first opportunity to re-plan is missed. As an event, this is motivated by receipt of a warning message from an on-board radiological sensor or another message from a different off-board source. This decision typically occurs about 10 minutes after the first decision point. At the second decision point, four different recovery re-plans are plausible, each taking a different amount of time, where the longer routes may contain additional re-plans based upon more detailed updates. Each of the four cases have different probabilities of taking different recovery routes. Some routes are at higher risk of contamination from the radiation cloud. The decision criteria for mission success is that the mission time took less than or equal to 60 minutes and the helicopter was not contaminated.

STK® model analysis and observations used to establish a metric methodology are defined by the following steps:

- **Step 1**: Generate initial travel time of approximately 20 minutes using a normal distribution ($\mu = 20$, $\sigma = 2$) to the first decision point. Approximately 68% of the time that decision point will occur anywhere from 18 to 22 minutes, and 95% of the time it will occur from 16 to 24 minutes into the mission.

- **Step 2**: This the first decision point with 3 possible alternatives: No Decision; Re-plan to OK-Route; and Re-Plan to Best-Route. This decision is determined by the following rules relative to the four cases: (A) 75% Continue-Original, 20% Re-Plan to OK-Route, 5% Re-Plan to Best-Route; (B) 25% Continue-Original, 50% Re-Plan to OK-Route, 25% Re-Plan to Best-Route; (C) 10% Continue-Original, 30% Re-Plan to OK-Route, 60% Re-Plan to Best Route; and (D) 5% Continue-Original, 10% Re-Plan to OK-Route, 85% Re-Plan to Best-Route. Then go to Step 3a for Best-Route, Step 3b for OK-Route, or Step 3c for Continue-Original.

- **Step 3a (Best-Route)**: Generate additional travel time of approximately 30 minutes using a normal distribution ($\mu = 30$, $\sigma = 2$). Set probability of contamination to 0%. Go to Step 5.

- **Step 3b (OK-Route)**: Generate additional travel time of approximately 45 minutes using a normal distribution ($\mu = 45$, $\sigma = 5$). Set probability of contamination to 0%. Go to Step 5.

- **Step 3c (Continue-Original)**: Generate additional travel time of approximately 10 minutes using a normal distribution ($\mu = 10$, $\sigma = 1$), and then make a decision based on the following rules relative to the four cases: (A) 20% OK-Recovery, 45% Marginal-Recovery, 30% Difficult-Recovery, 5% Best-Recovery; (B) 10% Best-Recovery, 50% OK-Recovery, 25% Marginal-Recovery, 15% Difficult-Recovery; (C) 50% Best-Recovery, 25% OK-Recovery, 20% Marginal-Recovery, 5% Difficult-Recovery; and (D) 70% Best-Recovery, 15% OK-Recovery, 10% Marginal-Recovery, 5% Difficult-Recovery. Then go to Step 4a for Best-Recovery, Step 4b for OK-Recovery, Step 4c for Marginal-Recovery, or Step 4d for Difficult-Recovery.

- **Step 4a (Best-Recovery)**: Generate additional travel time of approximately 25 minutes using a normal distribution ($\mu = 25$, $\sigma = 3$). Set probability of contamination to 1% (PC_{Best}). Go to Step 5.

- **Step 4b (OK-Recovery)**: Generate additional travel time of approximately 30 minutes using a normal distribution ($\mu = 30$, $\sigma = 5$). Set probability of contamination to 5% (PC_{OK}). Go to Step 5.

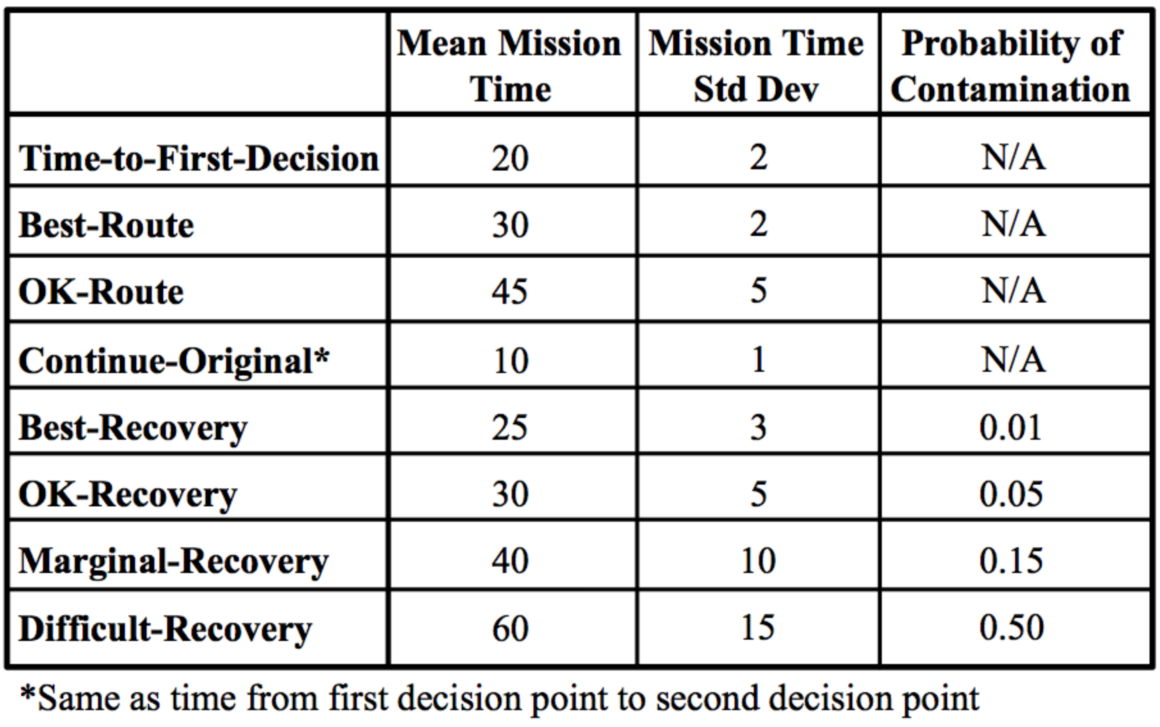
- **Step 4c (Marginal-Recovery)**: Generate additional travel time of approximately 40 minutes using a normal distribution ($\mu = 40$, $\sigma = 10$). Set probability of contamination to 15% ($\text{PC}_{\text{Marginal}}$). Go to Step 5.

- **Step 4d (Difficult-Recovery)**: Generate additional travel time of approximately 60 minutes using a normal distribution ($\mu = 60$, $\sigma = 15$). Set probability of contamination to 50% ($\text{PC}_{\text{Difficult}}$). Go to Step 5

- **Step 5: Show timing results**. For each case (A, B, C, D), show the probability of mission success, which is defined to be No Contamination and Cumulative Mission Time is less than or equal to 60 minutes.

A summary of the statistical parameters used for the mission segments is shown in Table 5.1.

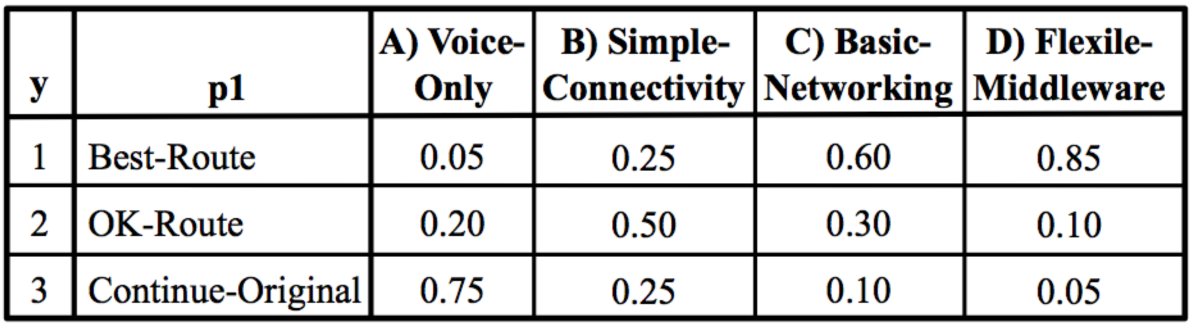
The numerical values discussed in steps 1 through 5 and shown in Figure 5.1 agree with the expectations discussed in Section 5.2. Those expectations refer to the probability of making a decision based on receiving new information times the probability of receiving that information. With agreement between expectations and simulated values, there is confidence that subsequent calculations will yield reasonable estimates of operation performance for comparative purposes.



	Mean Mission Time	Mission Time Std Dev	Probability of Contamination
Time-to-First-Decision	20	2	N/A
Best-Route	30	2	N/A
OK-Route	45	5	N/A
Continue-Original*	10	1	N/A
Best-Recovery	25	3	0.01
OK-Recovery	30	5	0.05
Marginal-Recovery	40	10	0.15
Difficult-Recovery	60	15	0.50

*Same as time from first decision point to second decision point

Table 5.1: Mission Segment Statistical Parameters



y	p1	A) Voice-Only	B) Simple-Connectivity	C) Basic-Networking	D) Flexile-Middleware
1	Best-Route	0.05	0.25	0.60	0.85
2	OK-Route	0.20	0.50	0.30	0.10
3	Continue-Original	0.75	0.25	0.10	0.05

Table 5.2: Decision Point p1 Probabilities

5.4 Calculations

As shown in this one example, it is possible to derive a metric methodology to predict the operational value of networking for a critical mission example. It relies on assumptions derived from running numerous models in a constructive simulation tool (in this case, STK®) to understand the choices that can be made as the mission unfolds. Operational performance is measured in comparison to conducting the mission without the benefit of networking, and the improved decision-making that comes with it. Using the search-and-rescue mission example, we can define a set of statistical equations for each of the four cases. The methodology defined in Steps 1 through 5 can be used to derive equations for each case of this example as follows.

Let random variable T_0 represent the time it takes to reach the first decision point. In all four cases, T_0 = N(20,2), where N is a normal probability density function. Three choices occur at the first decision point, so define the numerical variable $p1_{X,y}$ to represent the probability that for case X, choice y is taken. In this example, X can range from A to D, and y can range from 1 to 3. The values of p1 are shown in Table 5.2.

If y = 1 or 2, then the remaining time to complete the mission is defined by normal distribution functions for Best-Route and OK-Route, respectively. Step 3a defines Best-Route as N(30,2), and Step 3b defines OK-Route as N(45,5). The probability of choosing Best-Route is $p1_{X,1}$, and the probability of choosing OK-Route is $p1_{X,2}$. So, we can define the random variable T_1 to represent the time it takes to complete the mission after a choice is made at the first decision point. In this case, $T_1 = p1_{X,1}N(30,2) + p1_{X,2}N(45,5)$. Since

y	p2	A) Voice-Only	B) Simple-Connectivity	C) Basic-Networking	D) Flexile-Middleware
1	Best-Recovery	0.05	0.10	0.50	0.70
2	OK-Recovery	0.20	0.50	0.25	0.15
3	Marginal-Recovery	0.45	0.25	0.20	0.10
4	Difficult-Recovery	0.30	0.15	0.05	0.05

Table 5.3: Decision Point p2 Probabilities

the routes are independent, T_1 is a normal distribution with mean $\mu = 30\mathrm{p}1_{\mathrm{X},1} + 45\mathrm{p}1_{\mathrm{X},2}$, and variance $\sigma^2 = 2^2\mathrm{p}1_{\mathrm{X},1} + 5^2\mathrm{p}1_{\mathrm{X},2}$.

If y = 3, which means Continue-Original, or do not make a decision at the first decision point, then additional time is incurred before the second decision point. As stated in Step 3c, this additional time is given by N(10,2), which may be represented by random variable T_2. Similar to the first decision point, a numerical variable $\mathrm{p}2_{\mathrm{X,y}}$ is defined to represent the probability that for case X, choice y is taken at the second decision point. X ranges from A to D, and y ranges from 1 to 4 associated with Best-Recovery, OK-Recovery, Marginal-Recovery, and Difficult-Recovery routes, respectively.

Define the random variable T_3 to represent the time to complete the mission after the second decision point. Using the same logic as the first decision, $\mathrm{T}_3 = \mathrm{p}2_{\mathrm{X},1}\mathrm{N}(25{,}3) + \mathrm{p}2_{\mathrm{X},2}\mathrm{N}(30{,}5) + \mathrm{p}2_{\mathrm{X},3}\mathrm{N}(40{,}10) + \mathrm{p}2_{\mathrm{X},4}\mathrm{N}(60{,}15)$. This covers all the possibilities, so we can define random variable T_X to represent the time to accomplish the mission for case X as,

$$T_\mathrm{X} = T_0 + T_1 + p1_{\mathrm{X},3} * (T_2 + T_3).$$

Since all time segments are independent, this expands as a normal distribution with mean,

$$\mu_\mathrm{X} = 20+30p1_{\mathrm{X},1}+45p1_{\mathrm{X},2}+10p1_{\mathrm{X},3}+25p1_{\mathrm{X},3}p2_{\mathrm{X},1}+30p1_{\mathrm{X},3}p2_{\mathrm{X},2}+40p1_{\mathrm{X},3}p2_{\mathrm{X},3}+60p1_{\mathrm{X},3}p2_{\mathrm{X},4},$$

and variance,

$$\sigma^2{}_\mathrm{X} = 2^2+2^2p1_{\mathrm{X},1}+5^2p1_{\mathrm{X},2}+p1_{\mathrm{X},3}+3^2p1_{\mathrm{X},3}p2_{\mathrm{X},1}+5^2p1_{\mathrm{X},3}p2_{\mathrm{X},2}+10^2p1_{\mathrm{X},3}p2_{\mathrm{X},3}+15^2p1_{\mathrm{X},3}p2_{\mathrm{X},4}.$$

	A) Voice-Only	B) Simple-Connectivity	C) Basic-Networking	D) Flexile-Middleware
μ	70.44	61.63	55.60	51.95
σ^2	98.41	35.79	18.20	11.52
σ	9.92	5.98	4.27	3.39

Table 5.4: Means, Variances, and Standard Deviations for the Four Cases

Using the data contained in Tables 5.2 and 5.3 generates means, variances, and standard deviations for the four cases as shown in Table 5.4.

The Normal probability density functions defined by these means and standard deviations can be used to calculate the probability that the mission time will take 60 minutes or less. This is done by calculating the integral under the Normal curve from negative infinity to 60. Microsoft® Excel has a formula for this calculation called NORMDIST. Let PT_{X} be the probability that for Case X the mission time takes 60 minutes or less. In Excel, PT_{X} = NORMDIST(60,μ_{X},σ_{X},TRUE). Specific Excel-calculated values are $\mathrm{PT}_{\mathrm{A}} = 14.64\%$, $\mathrm{PT}_{\mathrm{B}} = 39.3\%$, $\mathrm{PT}_{\mathrm{C}} = 84.88\%$, $\mathrm{PT}_{\mathrm{D}} = 99.12\%$.

All routes following the situation when a decision is not made at the first decision point have some risk of contamination. Let PC_{X} represent the probability of contamination for Case X. This is calculated as the weighted sum of probability of contamination for each of the four routes multiplied by the probability of not making a decision at the first decision point. So,

$$PC_{\mathrm{X}} = p1_{\mathrm{X},3} * (p2_{\mathrm{X},1} * PC_{\mathrm{Best}} + p2_{\mathrm{X},2} * PC_{\mathrm{OK}} + p2_{\mathrm{X},3} * PC_{\mathrm{Marginal}} + p2_{\mathrm{X},4} * PC_{\mathrm{Difficult}})$$

Using data contained in Tables a and b results in the following probabilities of contamination: $\mathrm{PC}_{\mathrm{A}} = 17.1\%$, $\mathrm{PC}_{\mathrm{B}} = 3.46\%$, $\mathrm{PC}_{\mathrm{C}} = 0.73\%$, and $\mathrm{PC}_{\mathrm{D}} = 0.27\%$.

We now have enough information to calculate probability of mission success for all four cases. Let PS_{X} be the probability of mission success for Case X. $PS_{\mathrm{X}} = PT_{\mathrm{X}} * (1 - PC_{\mathrm{X}})$. Calculated values are $\mathrm{PS}_{\mathrm{A}} = 12.13\%$, $\mathrm{PS}_{\mathrm{B}} = 37.93\%$, $\mathrm{PS}_{\mathrm{C}} = 84.27\%$, and $\mathrm{PS}_{\mathrm{D}} = 98.85\%$. Table 5.5 summarizes the calculations.

	A) Voice-Only	B) Simple-Connectivity	C) Basic-Networking	D) Flexile-Middleware
Mean Mission Time	70.44	61.63	55.60	51.95
Mission Time Std Dev	9.92	5.98	4.27	3.39
Prob Msn Time <= 60	14.64%	39.30%	84.88%	99.12%
Prob of Contamination	17.10%	3.46%	0.73%	0.27%
Probability of Success	12.13%	37.93%	84.27%	98.85%

Table 5.5: Metric Calculations

5.5 Rationale for Assumptions

The mathematical results presented here are based on a series of experiments combining expert opinion with statistics resulting from simulations. When expert judgments were made, conservative assumptions were used so as not to overestimate the value of Flexile Middleware. Many trials of the same scenario were run using the STK® software to derive the starting estimates, which are contained in the p1 and p2 parameters, as well as mean and standard deviation values associated with route times. Since these numbers determine the end results, each one will be discussed with its rationale:

Time to First Decision Point (p1)

After studying the mission in STK®, experimentations showed that it would be possible to receive an indication about a spreading radiation cloud approximately 20 minutes after the start of the mission, with times as low as 16 and as high as 22, so the time to the first decision point is represented as a normal distribution with a mean of 20 and a standard deviation of 2.

Best-Route

Assuming that a decision is made at p1, STK® runs showed how it was possible to find alternate routes to the pick-up area, which on average is about 30 minutes after that decision is made. Operationally, this appears reasonable, because the goal is to make a pick-up within 60 minutes of receiving the distress call. A good, representative mission may have around

10 minutes of margin, which could be exploited en route with the right information. Those STK® runs showed how Best-Routes could be done in as few as 26 minutes, so Best-Route is represented as a normal distribution of mean 30 and a standard deviation of 2.

OK-Route

Several STK® routes had timelines that averaged about 45 minutes after the p1 decision. This was because those routes got too close to a dynamic radiation cloud from changing wind directions. These routes required ad hoc changes causing additional mission time compared to the Best-Routes. These groupings of routes showed times as low as 35 minutes, but rarely lower given the added re-planning time requirements. This defines a normal distribution with a mean of 45 and standard deviation of 5.

Time to Second Decision Point (p2)

The assumption is that the original mission route would intersect with a radiation cloud. If an early indication does not cause a decision at p1, then there would surely be a warning about 10 minutes later about the risk of getting too close to the cloud. This is modeled as a normal distribution with a mean of 10 and standard deviation of 1.

Best-Recovery

STK® runs showed that if a decision was not made at p1, then it was still possible to accomplish the mission within 60 minutes using a well-selected recovery route. These routes effectively avoid the radiation cloud while minimizing time to the pick-up area. Times as low as 19 minutes for the recovery route were possible, but averages were closer to 25 minutes. This is modeled as a normal distribution with a mean of 25 and a standard deviation of 3. Most times, these routes had no risk of radiation cloud contamination, but with a big change in wind speed it was possible. Accordingly, this risk was set to 1%.

OK-Recovery

STK® showed another set of route re-plans that averaged 30 minutes. These routes had one or two changes of path due to straying too close to the changing radiation cloud, but times as low as 20 minutes were possible. This is modeled as a normal distribution with a mean of 30 and a standard deviation of 5. Risk of contamination was still very low, but more than the Best-Recovery routes. Based on STK® runs, this risk is estimated at 5%.

Marginal-Recovery

STK® showed some routes that required a few en route changes to avoid the radiation cloud. These changes increased the average time to 40 minutes, but as low as 20 minutes was still possible. This is modeled as a normal distribution with a mean of 40 and a standard deviation of 10. In STK®, these routes showed risk of contamination about 3 times more likely than the OK-Recovery routes, so it is estimated at 15%.

Difficult-Recovery

Many of the recovery routes shown in STK® involved significant changes to the path or many small changes due to surprise information about the progression of the radiation cloud. These routes averaged about 60 minutes with a low close to 30. So, if average time was expended to the second decision point (30 minutes), then it is still possible to have a lucky Difficult-Recovery route that arrives at the pick-up location within 60 minutes. Difficult-Recovery routes are modeled as a normal distribution with a mean of 60 and a standard deviation of 15. These routes showed twice the risk of contamination as the Marginal-Recovery routes, so it is estimated at 30%.

Case A (Voice Only)

For the first decision point, the values are 75% of the time no decision, 20% OK-Route, and 5% Best-Route. The scenario assumes that the earliest time to get an indication of a radiation cloud is about 20 minutes from the start of the mission. For Voice-Only, this would require foresight on the part of those monitoring radiation levels as to the position of the helicopter en route for the search-and-rescue mission. With voice-only communications, the probability of the right information getting to the helicopter at the right time is very low. If it does (25% of the time), then it is reasonable to assume that there is a one in five chance of choosing the best route to accomplish the rest of the mission.

For the second major decision point, the values are 5% Best-Recovery, 20% OK-Recovery, 45% Marginal-Recovery, and 30% Difficult-Recovery. The difference between each of these recovery missions is expected mean of time to complete the mission with a corresponding standard deviation. With voice-only communications, there is very little information about all the options for choosing a recovery route, so this decision would be more reactive to the developing situation. There is still a chance (5%) of choosing a best recovery route accidentally, but without any real-time data on the spreading radiation cloud, it will be a hit-or-miss situation. The STK® model showed many routes from which to choose, and of course, changes can be made on the fly as well. Pilot intelligence can be considered, so there is only a 30% chance of choosing a route that has the lowest chance of meeting the time limit to accomplish the mission. Marginal-Recovery is more probable than OK-Recovery given very little information about the radiation cloud.

Case B (Simple Digital Connectivity)

For the first major decision point, the values are 25% of the time no decision, 50% OK-Route, and 25% Best-Route. 25% was selected for the No-Decision case at p1, because it is doubtful that a specific connection between the remote sensing device for the radiation cloud and the helicopter would be in place. It would rely on human relays. It is still better than the 5% for Case A, because there may be a digital feed from the sensing device to the helicopter or from a command-and-control connection to the helicopter which had access to the relevant information. If a decision is made, and lacking any event-specific applications, the chances of selecting Best-Route (25%) is much lower than selecting OK-Route (50%).

For second major decision point, the values are 10% Best-Recovery, 50% OK-Recovery, 25% Marginal-Recovery, and 15% Difficult-Recovery. Since there is a connection between the helicopter and the command-and-control station, there would still be a better than 50/50 chance of recovering from not making a good decision at the first decision point. However, lacking detailed real-time information about the propagation of the radiation, cloud it is much more probably that the selected route would be more OK than Best, but 10% of the time a luck best route may be selected. With limited information in communications, there is still a reasonable probability that riskier routes may be selected.

Case C (Basic Network Connectivity)

For the first major decision point, the values are 10% of the time no decision, 30% OK-Route, and 60% Best-Route. Basic networking includes a full functioning ad hoc wireless network running applications that may have adaptive features loaded before the mission began. Since video-based applications have major impacts on processing and bandwidth, only pre-defined video connections are assumed for Case C, and there is no way for the applications to reconfigure themselves based on the mission context. These features would be unique to Flexile Middleware, which is represented by Case D. Therefore, there is a very high probability of making a decision at p1 (90% of the time), and a good chance that the selected route would be the best option (60%). However, without the advantage of video feeds that were not pre-established or to load additional video-based applications, the chances of coming up with best route at p1 are not as high as in Case D.

For the second major decision point, the values are 50% Best-Recovery, 25% OK-Recovery, 20% Marginal-Recovery, and 5% Difficult-Recovery. If the decision is not made at p1, the

Mission Time	Voice-Only	Simple-Connectivity	Basic-Networking
50	1627.48%	1023.78%	200.11%
55	1539.79%	528.54%	84.50%
60	714.62%	160.57%	17.30%
65	312.23%	44.74%	1.85%
70	149.37%	12.38%	0.49%
75	77.64%	4.63%	0.46%
80	44.51%	3.41%	0.46%
85	29.50%	3.31%	0.46%
90	23.30%	3.30%	0.46%

Table 5.6: Time Sensitivity Values

capability of making a good recovery is still possible, but the closer proximity to the radiation cloud makes information from video-based applications more important for the recovery route selection.

Case D (Networking with Flexile Middleware)

For the first major decision point, the values are 5% of the time no decision, 10% OK-Route, and 85% Best-Route, and for p2, the values are 70% Best-Recovery, 15% OK-Recovery, 20% Marginal-Recovery, and 5% Difficult-Recovery. These values are small improvements over Case C to represent the added mission value from being able to load and process video-based networking applications dynamically.

5.6 Sensitivity Analysis

The mission success time for the base case was set to 60 minutes to correspond to the "golden hour" operational maxim for search-and-rescue operations. Since time is an important parameter for critical missions, it would be desirable to assess the sensitivity of expected performance improvements offered by Flexile Middleware relative to time. For this analysis, mission success time is allowed to vary from 50 minutes to 90 minutes using 5-minute increments. For each case and each time setting, percent improvement of the Flexile Middleware case (Case D) relative to the other three cases is calculated. Results are shown in Figure 5.1, and the values for this graph are shown in Table 5.6.

This analysis shows that performance improvements comparing Flexile Middleware to Voice-Only is significant. Even when mission time is increased to 90 minutes, Flexile Mid-

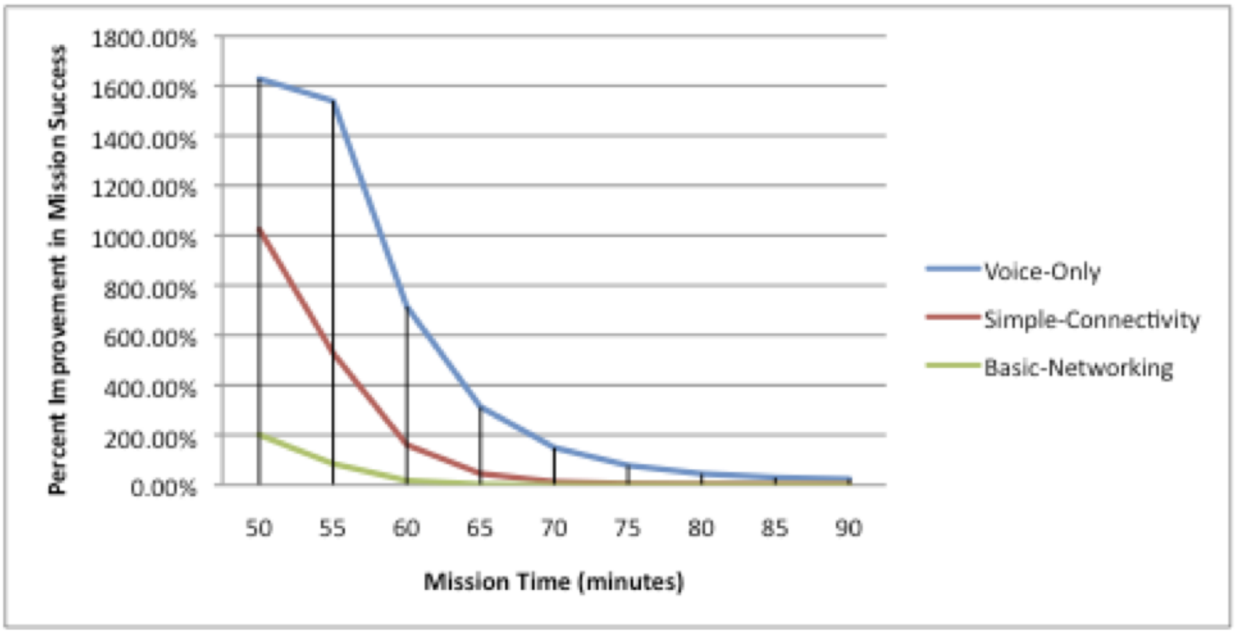


Figure 5.1: Sensitivity of Improvement Relative to Time

dleware shows a 25% improvement in probability of mission success. Comparisons to Simple Connectivity and Basic Networking are not as significant with almost equal probabilities of mission success occurring at 75 minutes and 65 minutes, respectively. A reasonable conclusion is that Flexile Middleware can provide a significant advantage when success depends on meeting aggressive times.

The base case contains a major significant event impacting probability of mission success, and that is contamination from a radiation cloud. Variations in the probability of contamination may be due to inaccuracies in the sensing equipment or swirling winds that make predicting the movement of the cloud very difficult. For this sensitivity analysis, probability of contamination is allowed to range from 50% less probable to 100% more probable in 25% increments. These changes were incorporated into the mathematical model to produce the results shown in Figure 5.2, and the values for this graph are shown in Table 5.7.

Figure 5.2 shows that there is a linear change in percentage of improvement comparing Flexile Middleware to the Voice-Only case. This is not surprising because, there is a much higher probability that the Voice-Only case will not make an early decision compared to the other three. Improvements of Flexile Middleware compared to the other two cases are also

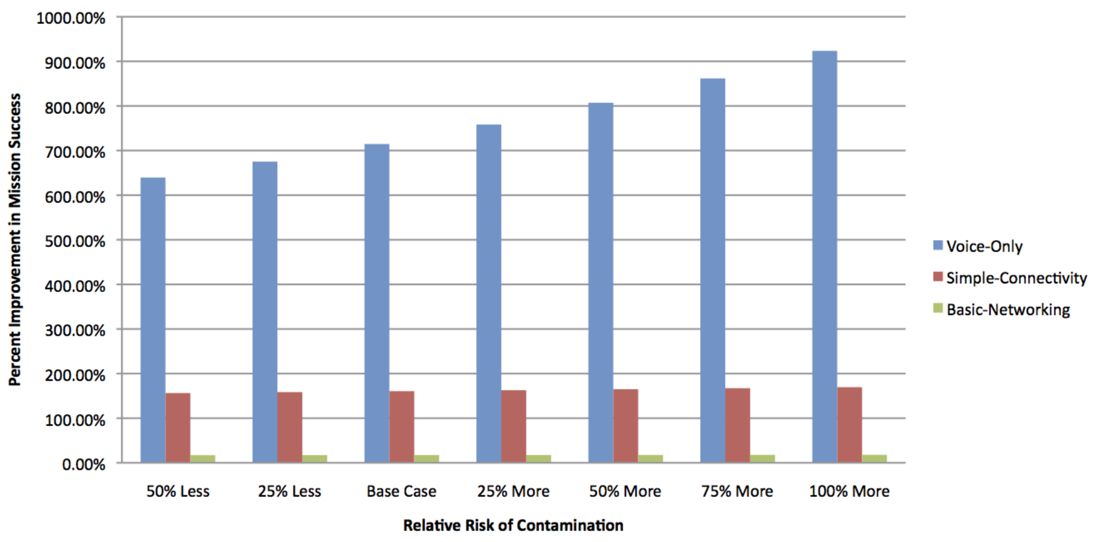


Figure 5.2: Sensitivity of Improvement Relative to Probability of Contamination

Contamination Risk	Voice-Only	Simple-Connectivity	Basic-Networking
50% Less	639.47%	156.33%	17.03%
25% Less	675.20%	158.43%	17.17%
Base Case	714.62%	160.57%	17.30%
25% More	758.33%	162.75%	17.44%
50% More	807.06%	164.97%	17.57%
75% More	861.74%	167.22%	17.71%
100% More	923.52%	169.53%	17.84%

Table 5.7: Contamination Sensitivity Values

shown, but not very significant.

5.7 Discussion of Metric Results

This metric analysis was based on one mission with two decision points, and one major event that could cause the mission to be a failure. Better situational awareness is directly proportional to better decision making. In fact, in the general case, it is reasonable to assume that more decisions means less effective mission execution. A generalization of the model presented in the previous sections suggests that more decisions increases the mission time in comparison to making better, earlier decisions. Of course, it is possible to contrive a scenario that initially appears to contain a good early decision based on timely, mission information, but it leads to a less desirable end result compared to not making the decision at all. Simply stated, these contrived cases represent nothing more than bad luck, but the contention is that bad luck is more probable when good information is either ignored or unavailable to make a mission-critical decision. In these cases, more decisions later in the mission will need to be made to either address the bad luck or the consequences of not using the information.

Another observation about the metric results is that more complex missions will value decision-making based on real-time situational awareness information compared to other less complex missions. In more complex missions, there may be too much information potentially leading to cognitive overload that may prevent good decisions early in the mission. The features of Flexile Middleware are well-suited to address this challenge with the expectation that if more complex missions were analyzed using this methodology, there would be an even larger improvement of Flexile Middleware over other methods of communication and networking.

In summary, though the scenario only has two decision points and one mission-ending event, it does represent the general case of critical missions intended for the application of Flexile Middleware. It is a medical evacuation operation with potential contamination from a radiation cloud, but it could also have been a military assault operation with potential failure from an enemy ambush, or a maritime patrol operation with potential failure from straying into unallowable regions. In all these cases and more, the common theme is the need to accomplish the mission within a predetermined amount of time at risk of failure due to known or unknown events, and mission-relevant information is available. The representative scenario is a conservative case, and as discussed, the benefits of advanced networking with

Flexile Middleware would exceed the predictions presented in this chapter for similar or more complex critical missions.

6

Operational Environment and Requirements

Flexile Middleware will increase the administrative overhead of a network. As such, the application environment needs characteristics that will make this added overhead justifiable. Flexile Middleware is dynamically adaptive to the operational environment both in functional behavior and elasticity to available network processing resources and constraints. Critical missions have the opportunity to exploit these capabilities for improved mission effectiveness.

Certain characteristics are expected of networks designed to operate in critical missions: (a) Ad hoc network formation; (b) Interoperability among a wide variety of node capabilities; (c) Real-time or near real-time responsiveness; (d) High bandwidth data dissemination (e.g. video); (e) Quality of service including information assurance; and (f) Execution of network applications that may have an impact on human life or other effect deemed critical by human society. Using networks for operations associated with military, humanitarian, and civil/police missions have these characteristics. DuBois and Perry introduced the concept of Flexile Middleware and discussed the potential for its operational utility[51]. Tactical networks depend on the affordable integration of communication systems, net-ready applications, middleware, and processing for practical use. A key point is the use of open standards due to expected changes in component networking technologies. Adaptive middleware can be used to improve operational performance in tactical networks, especially those involving mobile communications among air and ground systems for the purpose of achieving some shared operational goal among networked members of a team[52].

Presented in the four sections that follow are descriptions of example missions that will highlight expected performance of the network, modeling requirements, net-ready applications, and processing architectures. The mission examples will be used as a basis for the construction of models to rate the performance of Flexile Middleware compared to other forms of static and adaptive middleware.

6.1 Net-Ready Requirements

Both commercial and military domains have requirements for net-ready applications. Commercial requirements are principally derived from the business associated with cellular-based data services. Since this thesis focuses on critical applications, there is no reason to discuss market forces associated with mobile data services as a principal source of commercial net-ready requirements.

Figure 6.1 illustrates an outline for addressing networking requirements in military applications. The requirements development process is a part of the acquisition process and includes major milestones. An Information Support Plan (ISP) is required and it contains details on how to tailor the other requirements documents. The program requirements will state compliance with the Net-Ready Key Performance Parameter (NR-KPP)[53], which is refined by the ISP. The figure shows the major elements of the NR-KPP, which defines the test requirements to show compliance with the NR-KPP. Information Assurance is an important requirement, and the government has other contributing directives and instructions to explain how networked security requirements will be addressed. Following these requirements leads to additional test requirements. Ultimately, the program needs to obtain a Joint Interoperability Test Certificate (JITC)[53] following a successful test program to be approved for networked military operations.

6.1.1 The Acquisition Process

Addressing military net-ready requirements starts with the acquisition process known as Joint Capabilities Integration and Development System (JCIDS)[54]. JCIDS specifies that acquisition from concept through full operating capability will be accomplished through a series of milestones from A to C leading to Initial Operating Capability (IOC), and then operations and support. The military recognizes the value of exploiting networks, so to

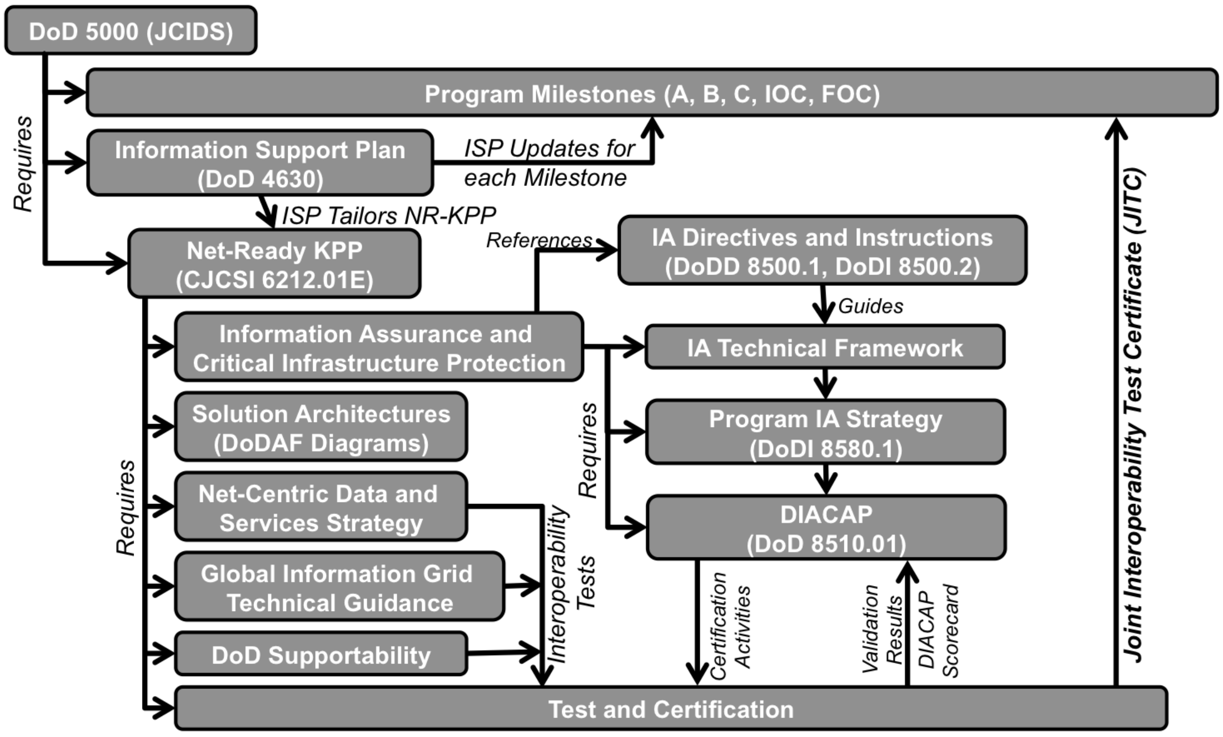

DoD 5000 (JCIDS)
Program Milestones (A, B, C, IOC, FOC)
Requires
Information Support Plan (DoD 4630)
ISP Updates for each Milestone
ISP Tailors NR-KPP
Net-Ready KPP (CJCSI 6212.01E)
References
IA Directives and Instructions (DoDD 8500.1, DoDI 8500.2)
Guides
Information Assurance and Critical Infrastructure Protection
IA Technical Framework
Solution Architectures (DoDAF Diagrams)
Requires
Program IA Strategy (DoDI 8580.1)
Net-Centric Data and Services Strategy
DIACAP (DoD 8510.01)
Requires
Global Information Grid Technical Guidance
Interoperability Tests
Certification Activities
Validation Results
DIACAP Scorecard
DoD Supportability
Joint Interoperability Test Certificate (JITC)
Test and Certification

Figure 6.1: Net-Ready Requirements Flow Diagram

complement JCIDS, it mandates the creation of an Information Support Plan[55] (ISP) which evolves as a system matures according to the JCIDS process. The goal of this process is to ensure that when the military develops a weapon system, it can interoperate with other systems using networked communications. Most of the details emphasize the digital data aspects of this interoperability, but there are sections of the ISP where conventional, non-networked radios must be mentioned.

6.1.2 Net-Ready Key Performance Parameter (NR-KPP)

JCIDS and the ISP describe the process for a specific military system to be networked, but the details on how that networking will be accomplished is contained in the Net-Ready Key Performance Parameter, which is specified in Chairman of the Joint Chiefs of Instruction 6212.01E[53]. All major DoD programs need to comply with the NR-KPP, which contains five key elements and a test and certification process for compliance. The five elements are:

1. Solution Architectures
2. Net-Centric Data and Services Strategy
3. Global Information Grid Technical Guidance
4. Information Assurance and Critical Infrastructure Protection
5. DoD Supportability

The ISP describes how the NR-KPP should be tailored for the specific weapon system to be produced.

6.1.2.1 Solution Architectures

The DoD Architecture Framework, or DoDAF[56] describes the content of solution architectures, and the NR-KPP specifies which architecture products are required for each stage of the acquisition process. The solution architectures are grouped as a series of views (all, operational, system, and technical). Most of the architecture products are prepared using either the Unified Modeling Language$^{\text{TM}}$ (UML$^{\text{TM}}$)[57, 58] or the System Modeling Language$^{\text{TM}}$ (SysML$^{\text{TM}}$)[59]. Fortunately, many commercial tools exist to help developers produce solution architectures[60, 61, 62, 63, 64].

6.1.2.2 Net-Centric Data and Services Strategy

This key element of the NR-KPP specifies that a service-oriented, inter-networked information infrastructure must be used. It also specifies compliance with the DoD Net-Centric Data Strategy[65] and the DoD Net-Centric Services Strategy[66]. The Defense Industry Standards Agency (DISA) maintains tools to enable network data exchanges, such as a metadata registry, service registry, and enterprise catalog. By having a centralized source for this type of information, the DoD is planning to ensure that all weapons systems use the same conventions for sharing interoperable information.

6.1.2.3 Global Information Grid Technical Guidance

The DoD uses the term Global Information Grid (GIG) to describe broad use of its networking capabilities for military purposes[67]. The NR-KPP contains a key section which provides technical guidance for interacting with the GIG. To conform with this element of the NR-KPP, weapon system program managers need to fill out questionnaires about the operational use of their system, which will be used to create GIG Enterprise Service Profiles (GESPs). GESPs include items such as interoperability requirements, a technical implementation profile (including quality of service requirements), the compliance testing method, and data descriptions for shared information. The weapon system program manager is free to choose the method for sharing information, but that selection comes from those listed in the online Defense Industry Standards Registry (known as DISROnline) with details specified in other architecture products.

6.1.2.4 Information Assurance and Critical Information Protection

This element of the NR-KPP was mandated by Congress through the Clinger-Cohen Act. It specifies compliance with DoD Instruction 8580.1 (Information Assurance in the Defense Acquisition System)[68] and the Defense Acquisition Guidebook[69]. In addition to showing compliance with these instructions, this element of the NR-KPP is where the weapon system program manager describes the details on encryption and other information assurance requirements, as well as details on how the program will follow the Defense Information Assurance Certification and Accreditation Process (DIACAP[70]) process to obtain an Information Assurance (IA) certification.

6.1.2.5 DoD Supportability

This element of the NR-KPP addresses requirements for compliance with the Electromagnetic Environmental Effects and Spectrum Supportability Policy. It also specifies the use of the Joint Tactical Radio System[71] for all communications from 2 MHz to 2 Ghz unless waived according to approved procedures. This element is also where the weapon system program manager details the approach for "anti-spoofing" and how non-networked data link information is made available to the network. This element requires a program to perform a bandwidth analysis and show the impact on interoperability of the weapon system into the GIG.

6.2 Modeling Requirements

Since the motivation and goals are focused on improving the probability of mission success, the best way to measure the value of Flexile Middleware is to simulate its performance in a model and measure the probability of mission success with and without the features associated with Flexile Middleware. Generally, these modeling requirements are as follows:

1. The model needs to be able to simulate military, humanitarian, and civil/police missions.
2. Actors in the simulation need autonomous capabilities limited to constraints defined in the scripts for the scenarios, i.e. decision points to determine courses of action.
3. The modeling system needs to have the ability to adequately represent operational environments with various terrain options including urban overlays and settings.
4. The model needs to be aware of node connectivity as it is impacted by breaks in line-of-sight. Examples impacting line-of-sight include natural obstructions, adversary communications jamming, antenna implementation range, and inability to obtain access to priority satellite communication channels.
5. The model needs to be able to represent selected system parameters, such as data rate with compressed video streaming.
6. The model needs to track assumptions and collect metric data.

7. The model needs the ability to execute rules dynamically during the simulation to represent Flexile Middleware behavior.

A wide variety of communication systems is expected for military, humanitarian, and civil/police missions. Such missions will normally involve aircraft, ground vehicles, individual first responders or soldiers, and possibly unmanned systems. Each of these systems has different communication systems operating on different frequencies with different waveforms and transmission ranges. All of these characteristics need to be modeled. To show the utility of Flexile Middleware, the approach is to use a tool to measure operational performance. The expectation is that operational performance is improved with adaptive middleware that can adjust to the mission context.

AGIs Satellite Toolkit (STK®) was selected as the modeling tool for this research and analysis. Specific modules used for the studies include: (a) Terrain and Cityscape Modeling; (b) Constraints for Visibility; (c) Communication Link Analysis; (d) Imagery and Map Data; (e) Sensor Modeling; (f) Maneuver Modeling; (g) Aircraft and UAV Modeling; and (8) System Performance Analysis. Though originally designed for satellite studies, the STK® product has evolved into something much more generally useful with the features to address all of the modeling requirements needed for this research.

One area identified for future research is to analyze the network performance of Flexile Middleware (e.g. latency, packet loss, etc.). The Common Open Research Emulator (CORE)[72] is a tool that allows emulation of entire networks. CORE consists of a user interface for easily drawing topologies and can be used to model mobile ad hoc networks. This tool can be used to collect metrics such as bandwidth, latency, packet losses, and restart time after recovering from lost connections.

6.3 Critical Mission Examples

This section describes three different critical mission areas: military, humanitarian, and civil/police.

Military strategists are giving greater attention to irregular warfare operations . A key characteristic of these types of operations is the organization of a force package designed to address a specific adversarial situation. Many of these situations are small operations in urban or mountainous environments. The following mission characteristics are to be expected:

- No satellite connectivity
- Objective changes during the mission
- Heterogeneous communication systems in use
- Strategic assets are not always available
- Line-of-sight may be complicated by terrain and structures
- Small Unmanned Air Vehicles (UAVs) expected

A network can be used to improve the performance of military missions with these characteristics. An ad hoc network allows tactical subnets to form where those subnets have a common mission objective. Satellite connections are expensive and sometimes the group assigned a mission does not have the priority to access available satellite bandwidth. Small UAVs with long endurance (e.g. InSitu ScanEagle® [73]) can be positioned to ensure all members of a tactical subnet group are able to communicate with each other.

The combat team commander will often assign a varied mix of assets to address any given military threat. Since these assets will most probably have different communication systems, the tactical subnet will have different bandwidths between different assets. In any instant of time, there is a weighted graph of network topology, where the weights are proportional to the bandwidth. The mobile nature of missions will make this weighted topology dynamic. Since the network will be expected to run applications to support mission objectives, knowledge of this changing topology is important, especially since some of the applications will require bandwidths to support video streaming. The military desire to use real-time video for improved mission performance is growing[74]. To be successful from a mission perspective, actions derived from network information need to be processed within the reaction time of the adversary.

In summary, networks for military missions are highly dynamic (varying topology and bandwidth), have heterogeneous communication systems, need to run applications with different processing requirements that are tailored to mission objectives, and have real-time performance considerations. It is conceivable that every application can be designed to incorporate a knowledge representation that takes these factors into consideration, but the desire to leverage commercial network applications, principally for cost reasons, makes it more practical to include this functionality in the middleware. Not only would this middleware include the services needed by the applications, but it would also contain the functionality to use mission objectives and network characteristics towards the goal of maximizing the probability of mission success.

Humanitarian missions are similar to military missions but with some adjustments. They are similar with limited access to wideband satellite connections, a dynamic network topology with varying bandwidths, and opportunities to use UAVs to support network creation. They are different in that there are no human adversaries trying to perform a competing mission objective. Also, depending on the nature of the mission, there may or may not be ground-based communication systems that need to be connected. These could be cell phones attempting to be used by survivors or radio systems commonly used by first responders. By the same logic as in military missions, humanitarian missions would need the same advance middleware functions to use a network to maximize the probability of a successful operation.

A major riot situation in an urban environment is a good example of a civil/police mission where a network can be used to improve mission success. From a networking perspective, there are complications introduced by disruptions to line-of-sight (the so-called urban canyon), multiple agencies needing to communicate with each other, and the need to bring in military force if the situation escalates. Civil/police missions have characteristics similar to both military and humanitarian missions, and therefore can use a network with the discussed middleware functions to maximize the probability of mission success. The operational environment is composed of different radios that connect to form a system of systems dedicated to accomplishing a well-defined mission and purpose. These radios may be upgraded as technologies to support more complex and bandwidth-intensive applications move into production.

6.3.1 Evaluation Scenarios

This subsection provides details on two critical mission scenarios that can be used to evaluate and measure the operational utility of networking, middleware, adaptive middleware, and Flexile Middleware. The basic approach is to calculate operational metrics based on conducting this scenario with the following assumptions: (1) Only voice communications; (2) Simple point-to-point connectivity; (3) Basic networked communications; and (4) Advanced networked communications using an implementation of Flexile Middleware. Additional background information on the scenarios follow, and the results of analyzing one of these scenarios was presented in Chapter 5. The humanitarian mission was chosen for metric evaluation because it is less demanding in terms of situational awareness, criticality of real-time communications, and higher risk of mission-ending events.

6.3.1.1 Humanitarian Mission with Radiological Complications

One critical scenario is a humanitarian mission with radiological complications. Analytically, the purpose of this mission is to emphasize how networking can improve communications for better command control, and then how Flexile Middleware can be used to improve safety, survivability, and efficiency of the operation. The STK® product will be used to measure operational performance, which provides results applicable to networks with adaptive middleware.

Obviously motivated by the devastating tsunami to hit Japan, this scenario starts with a UAV sent to investigate a damaged nuclear power plant. This would be a long endurance UAV with a relatively small payload for communications and processing. Since there is no pilot, services to support a pilot-vehicle interface are not necessary. This particular UAV is equipped with a radiation detection sensor. From a middleware perspective, active services would be associated with flying search patterns and monitoring sensors. Next, the UAV detects high levels of radiation. This detection would trigger a rule in the middleware that would reconfigure active services to process changes in the flight plans towards the goal of assessing the size, direction, and heading of the radiation cloud. Data is sent to human ground controllers who identify areas that need to be evacuated, and dispatch helicopters to perform the evacuations. More UAVs are sent to both monitor the radiation cloud and monitor the evacuations. Some of these UAVs would be under control of the manned helicopters performing the evacuations. Evacuation teams would be equipped with handheld devices. Helicopters would have situational awareness displays to show video from UAVs and for basic command and control functions. Line-of-sight communications would be required because the catastrophe knocked out the usual communications infrastructure. Some UAVs are positioned to act as relays. These agents have many different functions with different processing capabilities and needs for different middleware services. The scenario also includes a change in weather, with critical relevance connected to a change in wind direction. This environmental dynamic changes the network topology significantly, which in turn changes the mission roles of the agents. This description shows how a critical mission can be accomplished with ad hoc networking and Flexile Middleware. Experiments using this scenario will be used to both show the operational value and to establish more detailed functionality to embed in Flexile Middleware.

6.3.1.2 Mountainous Search And Rescue

Another critical scenario is mountainous search and rescue, which due to terrain, would have more complications introduced by breaks in line-of-sight communications. The same analysis approach for the Humanitarian Mission will be applied here as well. This scenario is similar to the Humanitarian scenario, but is set in mountainous terrain. It further emphasizes communications among multiple agents collaborating to perform a search. Middleware services to support applications that are aware of real-time search patterns would be a higher priority with this scenario.

A mountainous search and rescue operation would have additional complications with lower visibility from clouds and other weather conditions. If it is in high altitudes, awareness of atmospheric pressure is very important for rotorcraft operations. Another rotorcraft concern is rotor blade icing. All of these conditions could lead to mission-ending events. It is therefore reasonable to assume that whatever advantages can be offered by networking (or advanced networking with Flexile Middleware) for a humanitarian mission is even more advantageous for mountainous search and rescue.

Both scenarios emphasize attributes common to most any critical mission. They are:

- Communications are mainly line-of-sight
- Connectivity needs to be ad hoc
- Real-time performance is directly connected to probability of success
- Networks and applications need to be robust enough to handle major unforeseen changes in the operational environment (expect the unexpected)
- There is a wide range of processing capabilities for networked devices

- Automated decisions made with incomplete or uncertain information can be made, but need to have high confidence or human review and approval

To estimate the expected utility offered by Flexile Middleware, a critical mission was defined with a relatively small number of decisions and risks. However, it is still necessary to define the mission to be critical. Accordingly, a humanitarian mission with two decision points and one risk of a mission-ending event was selected. The measure of success is making the rescue within a short period of time while avoiding a mission-ending event. Most other critical missions would have more decision points and more risk of mission-ending events, so the advantages offered by networking with Flexile Middleware would be greater than the humanitarian mission chosen for this analysis.

6.4 Future Wideband Communications

Starting in 2010, the telecommunications industry began a serious transition from 3G to 4G networks as evidenced by ubiquitous advertisements on this subject. 3G networks ushered in data services at rates targeted to 3 Mbps with actual data rates closer to 1 Mbps. 4G networks are supposed to achieve data rates anywhere from 100 Mbps to 1 Gbps, basically providing the ability to handle high quality video streaming. The underlying network has a definite impact on middleware requirements, especially those with features to address quality of service, like Flexile Middleware.

Critical operations are witnessing the same transition to wider band forms of communication. Military experiments, like JEFX-10[75, 76], are demonstrating the operational value of networked wideband communications. Military requirements [77, 78, 79, 80, 81, 82] state the desire to provide the capability for Beyond-Line-Of-Sight (BLOS) communications with enough bandwidth to support high quality video. Wideband Satellite Communications (SATCOM) are an approach for ubiquitous coverage[83], but its implementation has challenges for critical operations, especially those that need rotorcraft[84]. To address these challenges, DuBois, et al.[5], suggest an approach involving more use of networking. The premise of this approach is that it is possible to make small changes to standard concepts of operations (CONOPS) as an effective alternative to wideband SATCOM. If Beyond-Line-Of-Sight communications is operationally necessary, then a mature, lower bandwidth solution is acceptable given the relative rarity of its expected use. In this case, users would notice degradation in video quality, but it would happen so infrequently that it would be a reasonable trade. UAVs will play a critical role in the new CONOPS with the assumption of a moderate degree of networking technology in the operational environment.

6.5 Net-Ready Applications

The principal purpose of net-ready applications is improved situational awareness. Applications result from the direct implementation of user desires for networking capabilities to support operational requirements. Situational awareness information is extremely valuable to those performing critical operations. From a military perspective, asymmetric and irregular warfare are more common, causing new challenges for interoperability. Networked communications between assets collaborating to accomplish a mission can decisively improve performance. Demand for information to support modern operations continues to increase. Persistent surveillance is a driving requirement, and this data must be made available to those who need it, and when they need it.

Ad hoc connectivity is required for interoperability among a comparatively smaller number of units often uniquely tailored to address an adversary or situation. In this environment, there is a need to directly control collection assets including a strong desire for real-time video. Once connections are established, assured connectivity needs to be maintained. Additional sensors alone are not the answer, nor are point-to-point data links. Increasing the number of sensors will saturate the available bandwidth and cause confusion in the receivers of this information as the same areas are sampled multiple times. Point-to-point data links only support the two entities that are connected and impede integration and fusion with other networked information.

When networked capabilities are considered for military, humanitarian, and civil/police missions, several net-ready applications would be most desirable:

- **Video Dissemination:** For critical missions, it is important to maintain a high level of vigilance that is best provided by video sources (visual or infrared). This data needs to be available to all participants in the mission. Broadcasting video across a network impacts bandwidth. Compression can be used, but for some situations only lossless compression techniques would be allowable.

- **Networked Weather:** Weather can often be a hindrance to mission success, and in military situations, it can also sometimes be a source of exploitation to improve the probability of success. Some systems can have their own weather radars, but many will not. Regardless of system capabilities, weather can be a major contributor to success or failure of critical missions, and needs to be available to participants when they need it.
- **Tactical White-Boarding with Chat:** Situations in critical missions can change quickly and without warning. All participants need to adapt to those changes. A tactical white-board provides the ability to take a picture of an area of operation, and then mark it up with notes, symbols, and drawings. This information is immediately viewable by all participants on the network, and it represents a depiction of how the situation has changed, and how the team expects to react to that change. This application is fairly common with networked systems and includes chat capabilities as well.
- **Common Operating Picture:** With any networked system, there is the risk of information overload. As such, there also needs to be an application to view the most important network information. The common operating picture fuses multiple sources of networked information, and will often contain user-selectable filters for visualization.

These are a few net-ready applications expected for use in critical missions. Other mission details create needs for additional applications. Net-ready applications have an impact on processing throughput and memory availability, and are expected to call middleware services to share information across the network.

Operational mission requirements determine the net-ready applications. Physical factors can scope the number and types of applications that can be installed on the network processor. The net-ready applications determine which middleware services are needed. The middleware also has a direct impact on processing throughput and memory, because it is installed on the network processor with the applications. The diversity of communication systems and available network processing resources are driving the need for adaptive and flexible middleware. Systems collaborating to accomplish a mission need to prioritize their applications and be able to adapt their network functionality to support changing mission roles.

6.6 Network Processing Architectures and Systems

The Joint Tactical Radio System (JTRS)[71] was selected by the U.S. Department of Defense to be the radio of choice for the future. It has ad hoc networking features and the bandwidth to support critical operations. By analogy, each JTRS is essentially a mobile hot spot and can form a network based on geographic proximity[85]. Other IP-based network radios can be used to create operational networks suitable for critical missions (e.g. SeaLancet^{TM}).

The radio is only one element of a networked system. Other components include processor/router, middleware, and application software. Boeing developed an Advanced Tactical Network System (ATNS)[52], which employs open architecture principles addressing all four of these major components of networked systems. The Boeing ATNS was installed on a V-22 tiltrotor aircraft. It has been tested with multiple radios, processors, routers, and middleware with minimal changes to the underlying executive software. It has also been tested with numerous applications, both developed in house and from third party sources.

ATNS is representative of emerging network architectures for tactical platforms. Middleware functionality needs to contain processing to support a network infrastructure composed of systems like ATNS, but also for other more disadvantaged network connections used by others collaborating to accomplish a mission. As explained earlier in this section, critical missions are often accomplished in an ad hoc manner employing dynamic changes to initial plans. Due to time constraints, infrastructure limitations, and other details associated with these missions, there is no opportunity to study the information requirements and design the optimal network to address those requirements, and then deploy it. This ad hoc, heterogeneous environment demands adaptive middleware for any network that can be formed to connect collaborating agents. It is obvious that advantages can be gained by providing the ability for that middleware to configure itself to fit on host processors regardless of capabilities, to reconfigure dynamically based on changing roles during the mission, and to comprise the best set of service functionality to achieve the goals and objectives of the mission.

7

Summary

In summary, middleware is important in the design of networked systems, and with creative approaches, new forms of middleware can be used to improve the operational performance of networked systems. As network connections become more ubiquitous, opportunities for collaboration among heterogeneous nodes to accomplish a common goal have started to emerge and will continue for the foreseeable future. Heterogeneous networks will contain powerful and disadvantaged nodes operating together. Middleware that is aware of operational goals with knowledge of node-level processing and bandwidth resources is one way to improve operational performance of collaborative systems. This thesis presented Flexile Middleware as such an approach to middleware that flexes based on operational goals and node-level resources. It is a form of adaptive middleware, but deserves a class of its own due to the many ways in which it can flex, and the different ways it can be implemented yet still comply with the characteristics defined for Flexile Middleware.

Missions that would value more intelligent middleware processing are critical in nature. These missions may have challenging operational goals or may lack important information if done without this capability. This thesis presented a close examination of critical missions and how Flexile Middleware may be used to improve operational performance in comparison to other approaches. Adding functionality to middleware introduces more processing latency. For most critical missions, there is a real-time processing consideration associated with operational success.

There is a need to show that the added overhead for Flexile Middleware processing is justified. It would appear that this is the case, as was shown by modeling and analysis of a representative critical mission, which compared Flexile Middleware to other forms of networked communications. Flexile Middleware was used to represent advanced networking, and was modeled by mapping its characteristics against information expectations. Those expectations were used for the design of an experiment that would model a representative mission in a tool, so that many different ways to accomplish the mission can be tried. Mission time and probability of a mission-ending event were used to calculate probability of mission success, and by using the simulations, it was possible to build a statistical model that could be used to compare the four types of networked communications. The representative mission contained two decision points, and one mission-ending event. A sensitivity analysis showed that when a mission had challenging time constraints, Flexile Middleware could provide significant improvement in operational performance. It also showed that Flexile Middleware could definitely improve operational performance with most any assignment of probabilities of encountering a mission-ending event. Since the mission that was modeled was relatively simple (two decision points and one mission-ending event), the value of Flexile Middleware would be greater for more complex missions provided there are tight time constraints.

A chapter in this thesis was dedicated to describe how Flexile Middleware works. It contains data flow diagrams, sequence diagrams, and flow charts to illustrate processing steps. At the heart of Flexile Middleware is a processing module referred to as the "Reconfigurator". This particular module provides the flex to Flexile Middleware, and is described in more detail compared to the other modules. This chapter suggests a creative implementation of Flexile Middleware that exploits multi-core architectures to reduce the time for the middleware to flex and change states.

Several areas of future research relative to Flexile Middleware are worth noting. For example, setting parameters associated with monitoring and how the Reconfigurator works are very important to its implementation. These parameters can be estimated analytically, but would probably require detail modeling of implementations to settle on the right values. Another topic would be to study the relationship between ad hoc routing algorithms and Flexile Middleware. In particular, it may be the case that if more intelligent ad hoc routing algorithms are employed then less capabilities in Flexile Middleware would be required to achieve the same level of improvement in operational performance, or vice versa. In practice, it may be the case that certain applications need to be present all the time. Services and applications to implement Flexile Middleware functionality need to be omnipresent, but for practical reasons it maybe desirable to expand the set of core applications and services to reduce the processing overhead. The thesis contains a suggestion to exploit the capability of a multi-core processing architecture to minimize the time for reconfiguration. However, Flexile Middleware would be expected to run on processors that may not have this type of architecture. Accordingly, it would be desirable to measure the processing overhead assuming reconfiguration that does not use a multi-core architecture. Finally, from an implementation perspective, most developers would settle on the use of data models and application program interfaces for an additional layer of abstraction between the middleware and the applications. Understanding the impact of this added level of software abstraction to Flexile Middleware is important.

8

Trademarks

The following trademarked items are mentioned in this thesis:

- CoreDX™ is a registered trademark of Twin Oaks Computing, Inc.
- DDS™, OMG®, SysML™, System Modeling Language™. UML™, and Unified Modeling Language™ are registered trademarks of the Object Management Group
- Integrity OS® is a registered trademark of Green Hills Software, Inc.
- LynxOS® is a registered trademark of LynuxWorks
- Microsoft® is a registered trademark of Microsoft, Inc.
- OpenMP® is a registered trademark of the OpenMP Architecture Review Board
- OpenSplice® is a registered trademark of PrismTech, Inc.
- Rackspace® is a registered trademark of Rackspace US, Inc.
- ScanEagle® is a registered trademark of Insitu, a wholly owned subsidiary of The Boeing Company
- SeaLancet ™ is a registered trademark of Harris Corporation
- STK® is a registered trademark of AGI, Inc.

With certain trademarked items (e.g. DDS) that are mentioned frequently, and in agreement with common practice, the notation will be used once at the beginning of the chapter.

Appendix A

DDS Middleware Trade Study

This section contains selections from a middleware trade study that was developed as part of a National Rotorcraft Technology Center (NRTC) project, titled Net-Ready Applications to Improve Rotorcraft Safety and Survivability. The author of this thesis in his capacity as a Boeing employee is the principal investigator on this project, and the detailed analysis was performed by Gustavo Baptista and Markus Endler of Pontifícia Universidade Católica - Rio de Janeiro under subcontract to Boeing.

The trade study is structured as follows: First, the main aspects of existing DDS implementations are presented and compared. Then, an analysis of the architectures of the two most well-established implementations is presented, with their respective implications on performance. The set of known existing COTS implementations of DDS^{TM} to be investigated in the trade study document is the following:

- OpenSplice® DDS (PrismTech)[20]
- RTI DDS (Real Time Innovations)[19]
- CoreDX^{TM} (TwinOaks Computing)[21]
- MilSoft DDS Middleware (MilSoft)[86]
- InterCOM DDS (Gallium)[87]

A.0.1 Comparison of Attributes

Attributes of evaluation include:

- **DDS Specification Compliance:** This includes compliance with five DDS profiles: (1) Data Centric Publish/Subscribe (DCPS) Minimum Profile – Implements the publish/subscribe communication paradigm for high performance information dissemination. With this profile, publishers and subscribers can communicate asynchronously and anonymously by sending and receiving data through Topics. A Topic contains strongly typed definitions of data to be transmitted atomically, i.e. a data structure definition in OMG® IDL which can be used to generate typed Writers and Readers for specific programming languages, a set of Quality of Service (QoS) settings and a unique name. The DCPS profile also includes the QoS framework that allows the middleware to match requested and offered Quality of Service parameters between Readers and Writers respectively. (2) DCPS Ownership Profile – This profile allows replication of information publishers, allowing each publisher to have an associated value which represent a strength, so that only the 'highest-strength' information will be made available to interested parties. (3) DCPS Content Subscription Profile – Implements content-filters, allowing applications to express content-based subscriptions. It also provides ways of specifying projection-views and aggregation of data, and dynamic queries for subscribed topics using a subset of the SQL language. (4) DCPS Persistence Profile – Implements durability, which offers transparent and fault-tolerant availability of non-volatile data, allowing late-joining of subscribers. (5) Data Local Reconstruction Layer (DLRL) Profile – Extends the previous four data-centric DCPS profiles with an object-oriented view on a set of related topics thus providing typical Object Oriented features such as navigation, inheritance and use of value-types, performing the mapping between the object oriented model to the underlying Topic relational model automatically. It also contains an object caching mechanism which is automatically synchronized with the distributed global system state.

- **Technical Maturity, Real-World applications and Industry Acceptance:** A mature, field proven system is required, already deployed in a number of mission critical scenarios, widely accepted in the industry and with previous known deployment success.

- **Quality of Service Control:** Effective Quality of Service (QoS) control mechanisms are needed in order to achieve reliability and availability in the target rotorcraft Mobile Ad-hoc Network (MANET) scenarios. The set of main QoS settings in the DDS specification need to be provided by the implementations. Specifically, QoS configurations for fault tolerant mechanisms (e.g. handling and recovery of network losses) are essential and must be available for definition and testing. The Terminal Control Protocol (TCP) provides one level of QoS, but additional capabilities are needed which depend on how DDS handles publish-subscribe. One example of this would be data persistence, which is a parameter that can be set to inform the middleware that a data item is still relevant for a certain period of time even if it may not have been published as quickly as expected.

- **Network Scheduling Control:** For effective real-time communication, explicit control of network resources is necessary to ensure predictable, scalable, and dependable behavior. Thus, mechanisms for such control are expected in the system.

- **Communication Protocols and Bandwidth:** The Real Time Publish/Subscribe (RTPS) protocol was specifically developed to support the unique requirements of data-distribution systems. It is widely adopted as a standard wire protocol that allows DDS implementations from multiple vendors to interoperate, and it is specifically targeted to address the requirements and to take advantage of the main features of the DDS specification (e.g. it takes advantage of DDS QoS settings to optimize its behavior). It is designed to be able to run over connectionless transport protocols. Thus, the implementation of RTPS and support for lower-level protocols such as UDP/IP, using different communication techniques such as unicast, broadcast and multicast are investigated. Support for these different communication techniques are an important aspect to be considered, since in future stages of the project specific networking and MANETs infrastructures will have to provide support for them, and will require use of specific techniques, or even the use of different techniques depending on the scenario.

- **Runtime Size and Memory Footprint:** Memory footprint and runtime size need to be investigated in order to map system non functional requirements and to evaluate how deployment would take place in real world platforms, such as onboard computers.

- **Real-Time Performance, Throughput, and Scalability:** The scope of this study was not to execute new tests with the products for benchmarking. Intra-nodal scalability is also important, allowing different applications in the same node to access the DDS domain in a scalable and efficient manner (e.g. with low intra-nodal latency). The impact on bandwidth is a very important aspect, considering the restrictions of the radio-based MANET scenarios. For instance, the amount of traffic of meta-data exchanged by the system should be evaluated.

- **Supported Interfaces like Web Services, Sockets, Databases:** Exposing the DDS system through other standard interfaces facilitate interfacing with other types of systems. For example, a system which requires access to data being shared into the DDS Domain could use a Service-Oriented Architecture Protocol (SOAP) interface to retrieve such information, or a database which can be synchronized with this shared data. Therefore, the interfaces provided by each of the products are investigated.

- **Security:** Considering that not every deployment scenario will have secure networks for data transmission, supported security mechanisms must be analyzed. The goal is to evaluate how well the candidate systems could operate in unsecured networked scenarios, or to provide additional security features in an existing secure network. For example, in an unsecured network, data encryption would be crucial for confidentiality. On the other hand, even in a secure network, access control (possibly role-based) would ensure that only authorized actors would have appropriate access rights to the shared data, providing any desired level of confidentiality. Data authentication and integrity are also requirements to be considered.

- **Platform Support and Language Bindings:** Supported platforms and language bindings are presented since the middleware should be used for application development in Windows and Unix environments. Also, bindings for programming languages available to be used for application development are listed.

- **Cost:** The licensing model of different versions of each product and respective associated costs need to be compared in order to evaluate the best cost/benefit relation for choosing the middleware to be used in this project. This analysis must take in to account costs of productivity tools, development tools and run-time licensing. The cost of each product will then be related to the other aspects in the trade study, in order to consider the benefits associated with the cost.

- **Productivity:** The complexity of application development with the solution provided, complexity of the programming languages supported and tools to improve productivity, such as code generators, specialized Integrated Development Environments (IDEs) and modeling tools are important aspects to be considered. Such tools improve the quality of the resulting product, reduce time-to-market, and provide better maintainability. Also, debugging, testing and monitoring tools are crucial.

Table A.1: OpenSplice® DDS Evaluation

Aspect	Assessment	Description
1 – OMG® DDS Standard Specification Compliance	High	• DDS API 1.2 • Minimum Profile • Durability Profile • Ownership Profile • Partial Content Subscription Profile (ContentFilteredTopic and QueryCondition) • Data Local Reconstruction Layer (DLRL)

Continued on next page

Table A.1 – continued from previous page

Aspect	Assessment	Description
2 – Technical Maturity and Real World Applications	High	• PrismTech participated on the OMG® DDS specification definition. • Initially developed as SPLICE-DDS by Thales Naval Netherlands (TNL) • Considered a field proven middleware • Information backbone of TNLs TACTICOS CMS currently deployed in 15 navies around the world (OpenSplice® DDS is the 2nd generation COTS evolution of this product). • Deployed in many different mission- and business-critical systems including: Automated Trading, Air Traffic Control and Management, Combat Management Systems, Naval Systems, SCADA. Recently, was selected to be used by the European Flight Data Processor.

Continued on next page

Table A.1 – continued from previous page

Aspect	Assessment	Description
3 – Industry Adoption	High	• Wide set of customers in different fields, such as Defense and Aerospace, Financial Services, Transportation, SCADA and Utilities, Telecom and others. • Customers in the Defense and Aerospace Industry: THALES, Raytheon, Yaltes, Nexter, General Dynamics, Northrop Grumman, Boeing, BAE Systems, NASA, Lockheed Martin, EADS, Aselsan, Rockwell Collins, US Department of Defense, US Navy, Embraer, Australian Department of Defense, Scientific Research Corporation.
4 – QoS Control	High	Complete set of QoS Policies from DDS specification.

Continued on next page

Table A.1 – continued from previous page

Aspect	Assessment	Description
5 – Network Scheduling Control	High	• Networking architecture scheduling mechanisms allow full control over networking resources • Scheduler uses QoS settings information to schedule data transmission properly • Allows the definition of different communication channels which have associated priorities (priority lanes), as well as networking resources in terms of bandwidth (traffic shaping) and urgency levels.
6 – Communication Protocols	Complete	• RTPS 2.1 • UDP/IP • Broadcast • Multicast
7 – Bandwidth Overhead	Unknown	The traffic shaping mechanism can be used to limit bandwidth utilization. Data about overhead on bandwidth not found/available.

Continued on next page

Table A.1 – continued from previous page

Aspect	Assessment	Description
8 – Run-Time Size and Memory Footprint	Small and Scalable	• Shared-memory utilization without applications is about 1 Mbyte (for built-in-topics, related administration and internal topics). • Libraries that provide the API's to applications are shared-objects i.e. are in memory only once regardless of the number of applications. • The size of the shared-libraries are between 1.5 (for C-language) and 2.5 (for C++/Java) Mbytes. • Regardless of the number of applications, the 'payload' of topics (i.e. not the instance-administration) is present in shared-memory only once regardless of the number of applications. • Optimizations allow for many applications (existing combat-management-system use-cases run hundreds of applications). • Example with 200 applications with 2 Topics with simple data types consumed 1.5 Mbyte of shared-memory. • Example with 1000 applications with 2 Topics with simple data types consumed 21.8 Mbyte of shared-memory.

Continued on next page

Table A.1 – continued from previous page

Aspect	Assessment	Description
9 – Architecture Features and Optimizations	Wide Set	• Multi-core ready and shared-memory based architecture for minimizing intra-nodal latency, as well as maximizing nodal scalability. • The communication by shared memory can be done between applications as well between services on a single node. • Pluggable Service Architecture. • High performance and scalable marshalling, implemented to maximize processor cache usage. • The networking service implements marshalling for all types. Marshalling only happens once, instead of being performed multiple times by applications which subscribe to the same data.
10 – Performance, Throughput and Scalability	High	TBD

Continued on next page

Table A.1 – continued from previous page

Aspect	Assessment	Description
11 – Supported Interfaces Such as Web Services, Sockets, and Databases	Wide Set	• SOAP-Connector: Allows to expose the (remote) DDS system to a SOAP-aware application such as the OpenSplice® Tuner thus allowing remote monitoring and control of any DDS target-system equipped with the SOAP-connector. • DBMA-Connector: Enables transparent two-way exchanges between the real-time DDS data space and any ODBC v3.0 compliant database.
12 – Security	Available	• Data Authentication • Integrity • Confidentiality • Access Control
13 – Platform Support	Wide Set	Linux, Windows, AIX, and Solaris, along with the leading RTOSes, such as, VxWorks, Integrity OS®.
14 – Programming Language Bindings	Wide Set	C, C++, Java, and C#
15 – Licensing/Cost	Medium	TBD

Continued on next page

Table A.1 – continued from previous page

Aspect	Assessment	Description
16 – Productivity	High	• Information modeling (topic definitions in IDL, code-generation for topic QoS) • Code-generation for application frameworks and DDS entities such as publishers/writers, subscribers/readers • Information partitioning, network-configuration and durability configuration resulting in XML-based configuration data. • 100 percent Java based • Aids design, implementation, test and maintenance of OpenSplice® based distributed systems.

The table below provides evaluation details on the RTI DDS product.

Table A.2: RTI DDS Evaluation

Aspect	Assessment	Description
1 – OMG® DDS Standard Specification Compliance	High	• DDS API 1.2 • Minimum Profile • Durability Profile • Ownership Profile • Partial Content Subscription Profile (ContentFilteredTopic and QueryCondition) • Durability Profile implemented by separate service
2 – Technical Maturity and Real World Applications	High	• RTI participated on the OMG® DDS specification definition. • According to RTI, it has more than 300,000 copies licensed for use in over 400 unique designs. • Proven in successful U.S. Department of Defense missions, qualified for the highest Technology Readiness Level, TRL 9.

Continued on next page

Table A.2 – continued from previous page

Aspect	Assessment	Description
3 – Industry Adoption	High	• Wide set of customers in different areas such as: Aerospace and Defense, Communications, Control Systems, SCADA and Instrumentation, Energy Systems, Financial Services, Simulation, Transportation, Unmanned Vehicles • List of customers in Defense: Advanced Fusion Technologies, BAE Systems, Boeing, Dot21, Lockheed Martin, Northrop Grumman, QinetiQ, Raytheon, Saab, Samsung Thales, Sperry Marine (Northrop Grumman), Tactical Communications Group, U.S. Air Force, U.S. Navy, Ultra Electronics
4 – QoS Control	High	Complete set of QoS Policies from DDS specification.
5 – Network Scheduling Control	Unknown	Mechanisms for network scheduling control and priority channels not found.
6 – Communication Protocols	Average – No Broadcast Support	• RTPS 2.1 • UDP/IP, IPv4 and IPv6 • Broadcast • Multicast • Concurrent use of different communication protocols
7 – Bandwidth Overhead	Unknown	Data not found
		Continued on next page

Table A.2 – continued from previous page

Aspect	Assessment	Description
8 – Run-Time Size and Memory Footprint	Small and Scalable	• Fully deterministic memory utilization with no dynamic allocation required after system initialization • Small footprint – with an extremely small-footprint version (as low as 130 KB available for high assurance and severely resource limited systems
9 – Architecture Features and Optimizations	Wide Set	• Shared memory for intra-node communication • Pluggable interface for custom transports, ability for applications to concurrently use multiple different transports.

Continued on next page

Table A.2 – continued from previous page

Aspect	Assessment	Description
10 – Performance, Throughput and Scalability	High	• Up to 80,000 messages or 950 megabits of data per second • Latency as low as 30 microseconds over Gigabit Ethernet • Application-to-application throughput as high as millions of messages per second with no inherent limit on overall system-wide capacity – aggregate throughput of hundreds of millions of messages per second can be achieved
11 – Supported Interfaces Such as Web Services, Sockets, and Databases	Wide Set	JMS, WSDL/SOAP, SQL Lightweight CORBA Component Model, Sockets, File, Custom via adapter interface, Relational databases, Microsoft® Excel, Complex Event Processing engines, Visualization platforms, Application Servers and ESBs
12 – Security	Wide Set	Utilizes many different techniques and partner security companies for providing security in different levels, from local networking security provided by operating systems policies (tailored by company partners), to a WAN communication security service called RTI Secure WAN Transport.
		Continued on next page

Table A.2 – continued from previous page

Aspect	Assessment	Description
13 – Platform Support	Wide Set	• Integrity OS® • Linux, SELinux and Embedded Linux • LynxOS® and LynxOS-SE • Mac OS X • QNX • Unix AIX and Solaris • VxWorks ,VxWorks 653 and VxWorks MILS • Windows and Windows CE/Mobile
14 – Programming Language Bindings	Wide Set	ANSI C, C++, C# (.NET), Java, Ada
15 – Licensing/Cost	High	Available in three standard editions (Basic, Professional, and Elite), and one safety-critical edition for high assurance applications

Continued on next page

Table A.2 – continued from previous page

Aspect	Assessment	Description
16 – Productivity	High	Rich Run-Time Tools and Services to accelerate debugging and testing while easing management of deployed systems: • **Analyzer** – a system-level visualization and debugging tool that discovers all DDS objects on a network, organizes them, and shows their properties such as Quality of Service (QoS) settings. • **RTI Scope** – a data visualization and logging tool that captures DDS data traffic • **Monitor** – Accelerates testing and optimization while easing management of deployed systems. Provides comprehensive insight into real-time performance and system health. • **Spreadsheet Add-in** – Allows use of Microsoft® Excel for real-time data visualization, analysis and injection. • **Recording Service** – Logs high-speed real-time data for future analysis. Replays recorded data for testing and simulation.

Similar analyses were preformed against CoreDX^{TM}, MilSOFT DDS Middleware, and InterCOM DDS (Gallium). The following table summarizes the comparative evaluations of all five DDS products.

Table A.3: DDS Evaluation Summary

Aspect	OpenSplice®	RTI	CoreDX™	MilSoft	Gallium
1 – OMG® DDS Standard Specification Compliance	High	High	Low	Average	Unknown
2 – Technical Maturity and Real World Applications	High	High	Unknown	High	High
3 – Industry Adoption	High	High	Unknown	Unknown	Unknown
4 – QoS Control	High	High	Incomplete	Unknown	Unknown
5 – Network Scheduling Control	High	Unknown	Unknown	Unknown	Unknown
6 – Communication Protocols	Complete	Average	Average	Unknown	Unknown
7 – Bandwidth Overhead	Unknown	Unknown	Unknown	Unknown	Unknown
8 – Run-Time Size and Memory Footprint	Small and Scalable	Small and Scalable	Small and Scalable	Unknown	Unknown
9 – Architecture Features and Optimizations	Wide Set	Wide Set	Unknown	Unknown	Unknown
10 – Performance, Throughput and Scalability	High	High	High	Unknown	Unknown
11 – Supported Interfaces Such as Web Services, Sockets, and Databases	Wide Set	Wide Set	Low	Unknown	Unknown
12 – Security	Available	Wide Set	Low	Unknown	Unknown
13 – Platform Support	Wide Set	Wide Set	Wide Set	Unknown	Good
14 – Programming Language Bindings	Wide Set	Wide Set	Wide Set	Good	Good
15 – Licensing/Cost	Medium	High	Unknown	Unknown	unknown
16 – Productivity	High	High	High	High	Low

A.0.2 Architectural and Performance Analysis

As shown in the previous section, OpenSplice® and RTI are the leading DDS products. The Open Architecture Benchmark[88] provides benchmarks for evaluating the performance of DDS-based middleware. Before presenting performance results, it is necessary to discuss the architectural differences between OpenSplice® and RTI.

A.0.2.1 OpenSplice® DDS Architecture

OpenSplice® DDS[20] uses a separate Data-Centric Publish-Subscribe (DCPS) daemon process for each nodes network interface, which is responsible for communicating with daemons running on other nodes. This federated architecture is designed to allow a level of decoupling of local communication endpoints from the networking service. This approach results in advantages including simplification of configuration, advanced networking scheduling mechanisms and intra-nodal scalability.

The daemon processes store common configurations and parameters shared by all local communication endpoints associated with a network interface, decoupling the applications (which run in separate user processes) from DCPS configuration and communication-related details. Using this separate daemon process to mediate access to the network simplifies application configuration, and allows the definition of policies for groups of participants associated with the same network interface. Changing the networking configurations does not affect application code or processing[20].

The networking service allows the definition of data communication channels. Each channel handles communication and QoS for all the local participants requiring particular properties. The definition of network channels helps enforce message priority even on non-priority-preserving transports, such as UDP/IP. These “priority lanes” allow prioritizing data for every single node to ensure that the more important data always preempts less important data. Predictive behavior of data even on worst case scenarios is then possible, avoiding priority inversion problem[89]. OpenSplice® DDS networking service functionality is summarized in Figure A.1.

A.0.2.2 RTI DDS Architecture

RTI DDS uses a decentralized architecture, which places communication and configuration related capabilities into the same user process as the application itself. These capabilities execute in separate threads (rather than in a separate daemon process) that the DCPS middleware library uses to handle communication and QoS (put ref here). The advantage of a decentralized architecture is that each application is self-contained, without needing a separate daemon. As a result, latency and jitter are reduced, and there is one less configuration and failure point. However, some disadvantages of using this approach are:

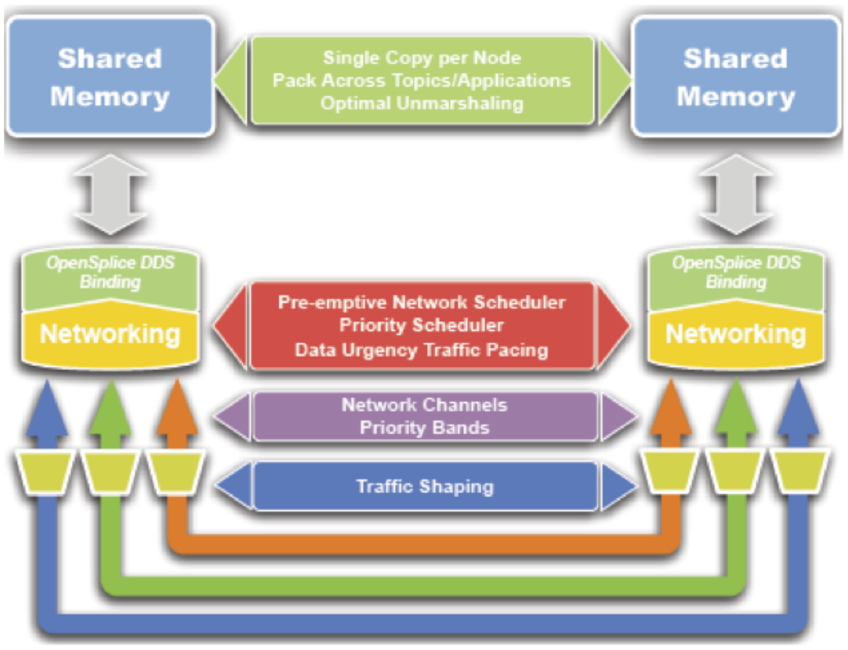


Figure A.1: OpenSplice® DDS Networking Service Functionality

- Specific configuration details, such as multicast address, port number, reliability model, and parameters associated with different transports, must be defined at the application level, which is tedious and error-prone.

- Contrasted with the daemon-based architecture, it elevates the number of networking endpoints, possibly impacting the scale on global discovery times. It also elevates the amount of required data transmissions, since data is replicated in each user process.

- Application misbehavior (under/over responsiveness) can impact other nodes. For example, a non-responsive low-priority process can trigger system-wide retransmissions, while an over-responsive high-priority process can overload system-wide network and processing resources. As a result, every application is a potential failure source.

- This architecture also makes it hard to buffer data sent between multiple DDS applications on a node, and thus does not provide intra nodal scalability benefits.

A.0.2.3 Performance Evaluations

Xiong[90] developed the DDS Benchmarking Environment (DBE), which is an open-source framework for automated DDS testing. Parameters for evaluation are latency, jitter, and throughput. Latency is defined as the roundtrip time between the sending of a message and reception of an acknowledgment from the subscriber. It is calculated as the average value of 10,000 round trip measurements. Jitter is the standard deviation of latency. Throughput is defined as the total number of bytes received per unit time. The only available comparison of RTI and OpenSplice® DDS uses multicast, since RTI doesn't support broadcast.

Xiong, et al [90] conducted tests of performance latency between DDS-based middleware and conventional publish/subscribe middleware. These tests were performed on a single computer to avoid clock synchronization issues, and to insulate the middleware architectures from the network. Two scenarios were tested, one with messages/events carrying a simple data type, and the other with a more complex and nested data type definition. The results confirm that DDS performance is superior to conventional pub/sub middleware. RTI presents the best performance for the transmission of simple data types and complex data types with smaller payloads. However, since these tests were performed on a single machine, OpenSplice® DDS optimizations for intra-nodal scalability start to show improvement on the transmission of complex data types above 512 bytes.

The low latency of RTI is due to its mature implementation, and to its decentralized architecture, which does not require communication to pass through a separate daemon process. In contrast, although OpenSplice® DDS also has a very mature implementation, its federated architecture involves extra hops through the publishers and subscribers daemon processes, which explain why its latency is higher than RTIs.

Xiong, et al. also conducted comparison tests of multicast performance between RTI and OpenSplice®. The results indicate that RTI outperforms OpenSplice® DDS for smaller payload sizes. As the size of the payload increases, however, OpenSplice® DDS performs better. It appears that the difference in the results comes from the different architectures. Since all subscriber nodes are running on the same machine, again the OpenSplice® DDS architecture provides optimizations for intra-nodal scalability.

Analyses are based on a very limited set of scenarios. They were performed several years ago, and newer versions of both products have been released. However, the presented benchmarks show how DDS performs better than standard publish-subscribe mechanisms, and it shows a baseline performance comparison between two mature DDS products. The results and studies indicate that the differences in performance between these two main DDS implementations is not significantly large, and that performance is not the decisive factor when deciding specifically between these two solutions.

Since MANET is assumed for critical applications, the most important aspect is the capacity of the middleware to comply with the best network communication technique, such as unicast, multicast or broadcast. MANET protocols and communication layers have to be taken into account when deciding which communication method to use. For instance, existing studies such as in [91] indicate that using broadcast for communication over MANETs is more efficient than using multicast. Others, like [92], show that the best use of broadcast or multicast over MANETs is only achieved when taking into account particular scenarios of different node densities and degree of mobility, and due to the dynamics of the network, even one strategy could be preferable over the other at different times and in different localized regions.

Between the products evaluated, in addition to a complete set of interesting features, the only one that provides support both for multicast and broadcast is OpenSplice® DDS, which makes this product the most flexible choice. Its network scheduling mechanisms provide a for predictive system behavior in complex and constrained networking scenarios. In addition, its traffic shaping features will allow appropriate use of the constrained bandwidth of radio-based MANETs to be used by assets participating in critical missions. Finally, the optimizations of this product for high intra-nodal scalability is suitable to the critical missions, in which many applications running on the same node must take advantage of this net-centricity with low impact on bandwidth and onboard computing resources.

References

[1] B. Li, W. Jeon, W. Kalter, K. Nahrstedt, and J. Seo. **Adaptive Middleware Architecture for a Distributed Omni-Directional Visual Tracking System**. In *Proceedings of SPIE/ACM MMCN 2000*, pages 101–112, 2000. 1, 9, 22

[2] O. Othman and D. Schmidt. **Optimized Distributed System Performance via Adaptive Middleware Load Balancing**. Technical report, Department of Electrical and Computer Engineering, University of California, Irvine, 2001. 1, 23

[3] X. Yu, K. Niyogi, S. Mehrotra, and N. Venkatasubramanian. **Adaptive Middleware for Distributed Sensor Environments**. *IEEE Distributed Systems Online*, 2003. 1, 22

[4] R. Couto A. da Rocha, M. Endler, and T. de Siqueira. **Middleware for Ubiquitous Context-Awareness**. In *MPAC '08 Proceedings of the 6th International Workshop on Middleware for Pervasive and Ad-hoc Computing*, 2008. 1, 20

[5] T. DuBois, W. Blanton, J. Clemens, P. Meyers, and A. Patrick. **Interoperability with Unmanned Air Vehicles as an Operationally Effective Alternative to Wideband SATCOM on Rotorcraft**. In *Proceedings of the American Helicopter Society Specialists Meeting on Unmanned Air Vehicles and Network Centric Operations*, January 2011. 3, 69

[6] C. Britton and P. Bye. *IT Architectures and Middleware: Strategies for Building Large, Integrated Systems*. Addison-Wesley Professional, second edition, 2004. Chapter 3: Middleware: A History of Objects, Components, and the Web. 9, 12

[7] J. Loyall, P. Pal, K. Rohloff, and M. Gillen. **Issues in Context-Aware and Adaptive Middleware for Wireless, Mobile Networked Systems**. Technical report, BBN Technologies, Cambridge, MA, 2009. 9, 11

[8] OBJECT MANAGEMENT GROUP. **Data Distribution Service for Real-time Systems**. Technical report, OMG, January 2007. 10, 12, 13

[9] P. NAUR AND B. RANDELL. **Software Engineering**. In *NATO Software Engineering Conference Proceedings*, page 14, January 1969. 10

[10] D. KRIEGER AND R. M. ADLER. **The emergence of distributed component platforms**. *Computer*, **31**(3):43–53, 1998. 11

[11] R. MATEOSIAN. **COM and DCOM – Microsoft's vision for distributed objects [Book Reviews]**. *IEEE Micro*, **18**(2):9–10, 1998. 11

[12] A. WATSON. **OMG (Object Management Group) architecture and CORBA (common object request broker architecture) specification**. In *Proc. IEE Colloquium Distributed Object Management*, 1994. 12

[13] W. EMMERICH. **An overview of OMG/CORBA**. In *Proc. IEE Colloquium Distributed Objects - Technology and Application (Digest No: 1997/332)*, 1997. 12

[14] MINQI ZHOU, RONG ZHANG, DADAN ZENG, AND WEINING QIAN. **Services in the Cloud Computing era: A survey**. In *Proc. 4th Int. Universal Communication Symp. (IUCS)*, pages 40–46, 2010. 12

[15] D. VASSILOPOULOS, T. PILIOURA, AND A. TSALGATIDOU. **Distributed technologies CORBA, Enterprise JavaBeans, Web services: a comparative presentation**. In *Proc. 14th Euromicro Int. Conf. Parallel, Distributed, and Network-Based Processing PDP 2006*, 2006. 12

[16] OBJECT MANAGEMENT GROUP. **Common Object Request Broker Architecture (CORBA) Specification, Version 3.1**. Technical report, Object Management Group, 2007. 12, 23

[17] D. G. SCHMIDT AND F. KUHNS. **An overview of the Real-Time CORBA specification**. *Computer*, **33**(6):56–63, 2000. 12

[18] UAS CONTROL SEGMENT. **The Data Distribution Service – Reducing Cost Through Agile Integration**. Technical report, Department of Defense, 2011. UNCLASSIFIED. 12

[19] RTI. **RTI Data Distribution Service**. `http://www.rti.com`, January 2011. 12, 77

[20] PRISMTECH. **OpenSplice DDS**. `http://www.prismtech.com`, January 2011. 13, 77, 96

[21] TwinOaks Computing. **CoreDX DDS Data Distribution Service Middleware**. `http://www.twinoakscomputing.com/coredx`, 2011. 13, 77

[22] IBM. **Implementing Vendor-Independent JMS Solutions**. `http://www.ibm.com/developerworks/java/library/j-jmsvendor/`, 2011. 13

[23] A. Corsaro. **Advanced DDS Tutorial**. Technical report, PrismTech, 2008. 13

[24] G. Pardo-Castellote. **Introduction to DDS**. In *OMG Real-Time Workshop*. RTI, Inc., 2008. x, 13, 14

[25] G. Hunt. **DDS - Advanced Tutorial Using QoS to Solve Real-World Problems**. In *OMG Real-Time & Embedded Workshop*. RTI, Inc., July 2006. Conference Presentation. 13

[26] P. Mell and T. Grance. **The NIST Definition of Cloud Computing (Draft), NIST Special Publication 800-145 (Draft)**. Technical report, National Institute of Standards and Technologies, 2011. 15

[27] D. Plummer, T. Bittman, T. Austin, D. Clearley, and D. Smith. **Cloud Computing: Defining and Describing an Emergent Phenomenon**. Technical report, Gartner Group, 2008. 15

[28] **Force.com Developer Documentation**. `http://wiki.developerforce.com/page/Documentation-Development_on_Force.com`. 15

[29] **Google App Engine**. `http://code.google.com/appengine/`. 15

[30] **IBM Smart Cloud**. `http://www.ibm.com/cloud-computing/us/en/index.html`. 15

[31] Microsoft. **Windows Azure**. `http://www.microsoft.com/windowsazure`. 15

[32] **Rackspace Cloud**. `http://www.rackspace.com/cloud/`. 15, 16

[33] **Amazon EC2 Getting Started Guide**. `http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide`. 16

[34] S. Sadjadi and J. McKinley. **A Survey of Adaptive Middleware**. Technical report, Department of Computer Science, Michigan State University, 2003. x, 16, 17

[35] D. Schmidt and T. Suda. **An Object-Oriented Framework for Dynamically Conguring Extensible Distributed Systems**. *BCS/IEE Distributed Systems Engineering Journal*, 1995. 16

[36] J. Lukkien. **Middleware: A Survey**, 2009. 17

[37] M. Uland. **System of Systems Common Operating Environment (SOSCOE) Support to Net Centricity**. `www.sei.cmu.edu/library/assets/uland.pdf`, March 2007. 18

[38] P. Schoen. **System of Systems Common Operating Environment**. `http://www.boeing.com/ids/soscoe/about.htm`, August 2005. Approved for Public Release, Distribution Unlimited, TACOM. 18

[39] V. Sacramento, M. Endler, H. Rubinsztejn, L. Lima, K. Gonçalves, F. Nascimento, and G. Bueno. **MoCA: A Middleware for Developing Collaborative Applications for Mobile Users**. *IEEE Distributed Systems Online*, **5**(10), 2004. Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro. 22

[40] V. Sacramento, M. Endler, H. Rubinsztejn, S. Lima, and G. Bueno. **An Architecture Supporting the Development of Collaborative Applications for Mobile Users**. Technical report, Departamento de Informática, PUC-Rio, R. Marquês de São Vicente, 2004. 22

[41] R. Balan, M. Ebling, P. Castro, and A. Misra. **Matrix: Adaptive Middleware for Distributed Multi-Player Games**. *Middleware 2005, G. Alonso, editor, LNCS 3970*, pages 392–405, 2005. 23

[42] O. Othman, J. Balasubramanian, and D. Schmidt. **Performance Evaluation of an Adaptive Middleware Load Balancing and Monitoring Service**. In *Proceedings of the 24th IEEE International Conference on Distributed Computing Systems*, pages 135–146, 2004. 23

[43] Editors of the American Heritage Dictionaries. *The American Heritage Dictionary of the English Language*. Houghton Mifflin Company, fourth edition, 2000. 24

[44] OpenMP Architecture Review Board. **OpenMP Application Program Interface, Version 3.1**. Technical report, OpenMP Organization, 2011. 37

[45] M. Endsley. **Toward a Theory of Situation Awareness in Dynamic Systems: Situation Awareness**. *Human Factors*, **37**:32–64, 1995. 38, 39

[46] M. Endsley, R. Sollenberger, and E. Stein. **Situation awareness: A comparison of measures**. In *Proceedings of the Human Performance, Situation Awareness and Automation: User-Centered Design for the New Millenium*, Savannah, GA, 2000. 38

[47] C. Hoffman, Y. Kim, R. Winkler, J. Walrath, and P. Emmerman. **Visualization for Situation Awareness**. In *Proceedings of the 1998 Workshop on New Paradigms in Information Visualization and Manipulation*, pages 36–40. ACM, 1998. 39

[48] E. Feibush, N. Gagvani, and D. Williams. **Visualization for Situational Awareness**. *IEEE Computer Graphics and Applications*, **1**:38–45, 2000. 39

[49] D. Overby, J. Wall, and J. Keyser. **Interactive Analysis of Situational Awareness Metrics**. In *SPIE - IS&T Electronic Imaging*, **8294**, 2012. 39

[50] T. DuBois, W. Blanton, F. Reetz III, M. Endler, W. Kinahan, G. Baptista, and R. Johnson. **Open Networking Technologies for the Integration of Net-Ready Applications on Rotorcraft**. In *Proceedings of the American Helicopter Society Conference*, May 2012. 40

[51] T. DuBois and R. Perry. **Flexile Middleware: Responding to Operational Demands of Critical Network-Based Applications**. In *Information Technologies for the Next Generation (ITNG) Conference Proceedings*, April 2010. 58

[52] T. DuBois. **Networking Challenges for Tactical Aviation Systems**. In *Presentation to Airborne Networks Conference*, November 2009. 58, 72

[53] Chairman of the Joint Chiefs of Staff. **CJCSI 6212.01E Net-Ready Key Performance Parameter**. Technical report, Department of Defense, 2000. 59, 61

[54] Chairman of the Joint Chiefs of Staff. **CJCSI 3170.01 Joint Capabilities Integratation and Development System (JCIDS)**. Technical report, Department of Defense, 2000. 59

[55] DoD Chief Information Officer. **Interoperability and Supportability of Information Technology (IT) and National Security Systems (NSS) (DoDD 4630.05)**. Technical report, Department of Defense, 2007. 61

[56] Defense Industry Standards Agency. **DoD Architecture Framework Version 2.0**. Technical report, Department of Defense, 2009. 61

[57] Object Management Group. **Unified Modeling Language (UML), Infrastructure, V2.1.2**. Technical report, Object Management Group, 2007. 61

[58] Object Management Group. **Unified Modeling Language (UML), Superstructure, V2.1.2**. Technical report, Object Management Group, 2007. 61

[59] Object Management Group. **Systems Modeling Language (SysML)**. Technical report, Object Management Group, 2010. 61

[60] IBM. **IBM Rational Software**. `http://www-01.ibm.com/software/rational/`, 2011. 61

[61] Sparx Systems. **Enterprise Architect Software**. `http://www.sparxsystems.com/`, 2011. 61

[62] No Magic Inc. **MagicDraw UML Software**. `http://www.magicdraw.com/`, 2011. 61

[63] MetaStorm Inc. **MetaStorm ProVision Software**. `http://www.metastorm.com/products/provision_ea.asp`, 2011. 61

[64] Atego Inc. **Artisan UML Software**. `http://www.atego.com/`, 2011. 61

[65] DoD Chief Information Officer. **Department of Defense Net-Centric Data Strategy**. Technical report, Office of the Secretary of Defense, May 2003. 62

[66] DoD Chief Information Officer. **Department of Defense Net-Centric Services Strategy**. Technical report, Office of the Secretary of Defense, May 2007. 62

[67] DoD Chief Information Officer. **DoD Global Information Grid Architectural Vision**. Technical report, Office of the Secretary of Defense, June 2007. 62

[68] DoD Chief Information Officer. **Information Assurance (IA) in the Defense Acquisition System**. Technical report, Office of the Secretary Defense, July 2004. 62

[69] Defense Acquisition University. **DoD Acquisition Guidebook**. Technical report, Office of the Secretary of Defense, 2010. 62

[70] DoD Chief Information Officer. **DoD Information Assurance Certification and Accreditation Process**. Technical report, Office of the Secretary of Defense, 2007. 62

[71] A. Feikert. **The Joint Tactical Radio System (JTRS) and the Army's Future Combat System (FCS): Issues for Congress**. Technical report, U.S. Congress, November 2005. CRS Report for Congress, Order Code RL33161. 63, 72

[72] J. Ahrenholz, C. Danilov, T. Henderson, and J.H. Kim. **CORE: A real-time network emulator**. In *Proceedings of IEEE MILCOM Conference*, 2008. 64

[73] InSitu. **ScanEagle Web Page and Data Sheet**. `http://www.insitu.com/scaneagle/`. 65

[74] T. DuBois. **Personal discussion between Thomas A. DuBois and Brigadier General Walters (USMC)**. Meeting at New River Marine Corps Air Station, New River, NC, September 2010. 65

[75] United States Air Force (USAF) Global Cyberspace Integration Center (GCIC). **JEFX 10-03 Initiatives: Irregular Warfare**. `http://www.gcic.af.mil/shared/media/document/AFD-100625-053.pdf`, April 2010. 69

[76] Linda Maines. **Joint Expeditionary Force Experiment 10-3**. *Inside the Air Force*, April 2010. 69

[77] R. Gates. *National Defense Strategy*. United States Government Printing Office, June 2008. 69

[78] U.S. Joint Forces Command. *Joint Tactics, Techniques, and Procedures for Joint Intelligence Preparation of the Battlespace (Joint Publication 2-01.3)*. United States Government Printing Office, May 2000. 69

[79] U.S. Army. *Decisive Force: The Army In Theater Operations (Field Manual 100-7)*. United States Government Printing Office, May 1995. 69

[80] U.S. Navy. *Naval Doctrine Publication 1 – Naval Warfare*. United States Government Printing Office, March 1994. 69

[81] U.S. Marine Corps. **MCDP Command and Control – PCN 142 000001 00**. Technical report, U.S. Marine Corps, October 1996. 69

[82] U.S. Air Force. *Air Force Transformation: The Edge*. United States Government Printing Office, 2005. 69

[83] U.S. Air Force. **Wideband Global SATCOM Satellite Fact Sheet**. `http://www.afspc.af.mil/library/factsheets/factsheet_print.asp?fsID=5582&page=1`, August 2010. 69

[84] D. Wilcoxson, B. Sleight, J. O'Neill, and D. Chester. **Helicopter Ku-band SATCOM On-the-Move**. In *Proceedings Military Communications (MILCOM) Conference 2006*. IEEE, 2006. 69

[85] Joint Program Executive Office (JPEO) Joint Tactical Radio System (JTRS). **Software Communications Architecture Specification**. Technical report, JTRS Standards Certification Authority, November 2010. Version: Next Draft; Approved for public release; Distribution is unlimited. 72

[86] MilSoft. **MilSoft DDS Middleware**. `http://dds.milsoft.com.tr/en/dds-home.php`, 2011. 77

[87] Gallium Inc. **InterCOM DDS**. `http://www.gallium.com/solutions/solutions-intercom.htm`, 2011. 77

[88] B. McCormick and L.Madden. **Open Architecture Publish-Subscribe Benchmarking**. `www.omg.org/news/meetings/workshops/RT_2005/03-3_McCormick-Madden.pdf`, 2005. OMG Real-Time Embedded System Work Shop. 95

[89] A. Corsaro. **10 Reasons to Choose OpenSplice DDS as your Messaging Middleware**. Technical report, PrismTech, Inc., 2011. 96

[90] M. Xiong, J. Parsons, J. Edmondson, H. Nguyen, and D. Schmidt. **Evaluating Technologies for Tactical Information Management in Net-Centric Systems**. In *Proceedings of the Defense Transformation and Net-Centric Systems Conference*, 2007. 98

[91] T. Kunz. **Multicast versus Broadcast in a MANET**. *Ad-Hoc, Mobile, and Wireless Networks*, **1**:630, 2004. 99

[92] L. K. Law, S.V. Krishnamurthy, and M. Faloutsos. **Understanding and Exploiting the Trade-Offs between Broadcasting and Multicasting in Mobile Ad Hoc Networks**. *IEEE Transactions on Mobile Computing*, **1**:264–279, 2007. 99