AFRL-IF-RS-TR-2002-160
Final Technical Report
July 2002

ROBUST ARCHITECTURE OF ADAPTIVE TRACKING ADVANCED TECHNOLOGY DEMONSTRATION

The Open Group (TOG) Research Institute
MITRE Corporation

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2002-160 has been reviewed and is approved for publication.

APPROVED:

ROBERT L. KAMINSKI
Project Engineer

FOR THE DIRECTOR:

WARREN H. DEBANY, JR.
Technical Advisor
Information Grid Division

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFGA, 525 Brooks Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	JULY 2002	Final Oct 95 - Sep 98

4. TITLE AND SUBTITLE

ROBUST ARCHITECTURE OF ADAPTIVE TRACKING ADVANCED TECHNOLOGY DEMONSTRATION

5. FUNDING NUMBERS

C - F30602-93-C-0225
PE - 63728F
PR - 2530
TA - 01
WU - 56

6. AUTHOR(S)

Raymond K. Clark, Patrick M. Hurley, Arkady Kanevsky, Thomas F. Lawrence, John A. Maurer, Paul J. Wallace, Douglas Well, Thomas M. Wheeler, and Yun Zhang

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

The Open Group (TOG) Research Institute
Cambridge MA

MITRE CORP
Bedford MA

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFGA
525 Brooks Road
Rome NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2002-160

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Thomas F. Lawrence/IFGA/(315) 330-2925

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report describes the results of an Air Force Advanced Technology Demonstration (ATD) entitled Robust Architecture for Adaptive C2 Systems. The objective of this project was to demonstrate how the processing resources can be managed so that the AWACS operator would always receive the highest quality tracks possible especially during overloads to the processing subsystem or failure of a processor(s). Currently there is no way for the AWACS operator to designate tracks as being more or less important. Therefore, when anomalies occur such as overloads due to incoming sensor data, larger than the processors can handle, arbitrary changes occur on the operators console, i.e. important tracks may be lost and less important tracks remain. In this demonstration the operator can designate some tracks as being more important so that, if tracks must be lost due to overloads or loss of processing resources, the least important tracks are sacrificed. This mechanism provides a gracefully and rationally degrading capability to the operator and avoids arbitrary behavior on the part of the processing system.

14. SUBJECT TERMS

Quality of Service, Real-Time Scheduling, Adaptive Resource Management, Adaptive Tracking, Distributing Processing

15. NUMBER OF PAGES

68

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

Table of Figures

Acknowledgements:

The work contained in this report was accomplished as a joint effort among Air Force Research Labortaory – Rome Research Site, MITRE Corporation – Bedford, MA, and The Open Group (TOG) Research Institute, Cambridge, MA. MITRE support was obtained through an Air Force support contract and TOG support was obtained under Air Force contract F30602-93-C-0255.

Section 1

Project Summary

This report describes the results of an Air Force Advanced Technology Demonstration (ATD) entitled Robust Architecture for Adaptive C2 Systems. In addition to the body of the report, more detailed descriptions are contained in the Appendices. The Appendices consist of four technical papers. Three of which were presented at an international conference and one that was presented at a national conference.

1.1 Project Background

The primary objective of this project was to demonstrate how the processing resources can be managed so that the AWACS operator would always receive the highest quality tracks possible especially during overloads to the processing subsystem or failure of a processor(s). Currently there is no way for the AWACS operator to designate tracks as being more or less important. Therefore when anomalies occur such as overloads due to incoming sensor data, larger then the processors can handle, arbitrary changes occur on the operators console, i.e. important tracks may be lost and less important tracks remain. In this demonstration the operator can designate some tracks as being more important so that if tracks must be lost due to overloads or loss of processing resources the least important tracks are sacrificed. This mechanism provides a gracefully and rationally degrading capability to the operator and avoids arbitrary behavior on the part of the processing subsystem.

This report describes a United States Air Force Advanced Technology Demonstration (ATD) in which the concepts of adaptive Quality of Service (QoS) based resource management are applied to the Multi Sensor Integration (MSI) tracking component of the Airborne Warning and Control System (AWACS). The ATD, which began in January 1996 and ended in September 1998, was a joint effort among Rome Laboratory, the Open Group Research Institute, and MITRE. Rome Laboratory provided expertise in QoS (QoS Based Resource Management), the Open Group in operating systems (Real-time Micro Kernel, Best Effort Scheduler, and Distributed Threads), and MITRE in Command and Control application development (AWACS, MSI Tracking).

Real world systems are subject to constant change such as overloading, component failure, evolving operational requirements, and/or dynamic operational environments. Ideally, these systems should be able to adapt to these changes and still provide a certain degree of service to their users. The objective of this effort was to demonstrate that the principles that embody QoS can be employed to enhance real-time adaptable applications. In other words the application will continue to perform its function and avoid catastrophic failure even in the presence of anomalies. Unfortunately, most systems today either do not adapt at all or are hard coded to adapt to only a small set of anomalies.

In non-adaptive systems anomalies produce non-optimal use of the available processing resources. In such systems many important activities receive fewer resources than needed while unimportant activities waste resources by receiving more processing resources than necessary. In this demonstration processing resources are applied to the most important user relevant activities when resources become scarce. In the tracking application of this ATD the most critical tracks receive the highest QoS while less critical tracks receive lower yet acceptable QoS. The QoS of a track is established by the benefit the particular QoS metrics will bring to the operational user. With this QoS approach, resources are more effectively

and more efficiently utilized and the application can gracefully degrade based on user provided policies.

At the core of our approach is the concept of Quality of Service (QoS) [LAW 99]. This is a multi-dimensional measure that reflects how well the system is performing. The dimensions of QoS are a product's timeliness, precision, and accuracy. QoS is discussed in detail in Appendix A of this report.

The concept of system Quality of Service (QoS) is based on the attributes of timeliness, precision, and accuracy, which can be used for system specification, instrumentation, and evaluation. Timeliness measures when an event occurs or the delay time from initiation to completion of an activity. Precision is the measure of the quantity of data produced as a result of an activity. For example, in the simplest case, it is the number of bits produced. Precision is only concerned with the quantity or volume of data not whether that data is accurate. Accuracy is the measure of the amount of error in the output data. Accuracy is concerned only with errors in the bit structure of the output. This effort demonstrates two principles of QoS. The first is that QoS is defined by the combination of three attributes (timeliness (T), precision (P) and accuracy (A)). If an output is timely and precise but inaccurate, it is not correct. If an output is precise and accurate but not timely it is incorrect. If an output is timely and accurate but imprecise it is also incorrect. Only if all three attributes are sufficient is the output correct. Consequently QoS consists of a TPA set. All three attributes must be measured in order to know if the QoS is correct. The second principle of QoS is that the better behavior one gets for one attribute the worse behavior will result in one or both of the other two attributes. Consequently in an adaptive system one must consider how to make the trade off among the attributes. Again, the correctness of the output is a function of all three attributes.

To incorporate QoS into the AWACS MSI Tracking application was defined in terms of metrics for each of the QoS attributes. These metrics are used to measure how well the system is performing. These three attributes were monitored for each track. These attributes were then traded off with the objective of maintaining reasonable track QoS. The statistical metrics for timeliness, precision and accuracy for each track can be controlled to produce a desired QoS. For example, if the system is in a period of overload, processing resources could be applied to tracks that have not been updated recently (this staleness of update can be thought of as poor statistical precision). The MSI Tracking application has multiple algorithms to use each with different accuracy, resource utilization and performance characteristics. This is an example of trading accuracy to maintain track QoS.

To accomplish the above The Open Group's Best Effort real-time scheduling, tracking algorithm selection, and sensor data selection are used to adapt and control the track QoS. The processing system monitored the state of the sensor inputs and applied operator supplied policies to select the appropriate sensor fusion algorithms to optimize tracking performance and track confidence level for the current operating conditions. In addition the sensor load can be throttled, based on operator policies, in order to prevent processor overload and assure optimal use of processing resources. In this manner processing resources are applied to tracks that are of critical importance in an overload situation. Key techniques used on this project include value-based resource management and distributed threads. Value-based resource management considers the orthogonal facets of urgency (timeliness) and importance (benefit), thus eliminating the notorious complexities of mapping timeliness requirements onto priorities. A distributed thread is a locus of control that spans objects and physical

nodes, transparently and reliably, via method invocation. Value-based resource management and distributed threads are also discussed in Appendix B of this report.

In addition to adaptive QoS resource management, this demonstration used redundancy management to assure continuity of track processing should a processor fail. The MSI Tracking demonstration was distributed across several processors utilizing The Open Group's Micro-Kernel and the distributed threads mechanism. The Distributed threads mechanism, supports enforcing timeliness requirements across nodes in the distributed MSI Tracking application. If one of the nodes should fail a distributed thread break detection and thread repair mechanism will be invoked to support application recovery.

In summary the effectiveness of QoS based adaptive resource management, QoS based graceful degradation, and facilities to support dependable distributed real-time applications was demonstrated. It was found that for this application adaptivity can be value added with little change to existing application software. Based on our experience the techniques are applicable to any distributed Command and Control (C2) application. Figure 1 illustrates the hardware / software configuration used in the development of this demonstration.

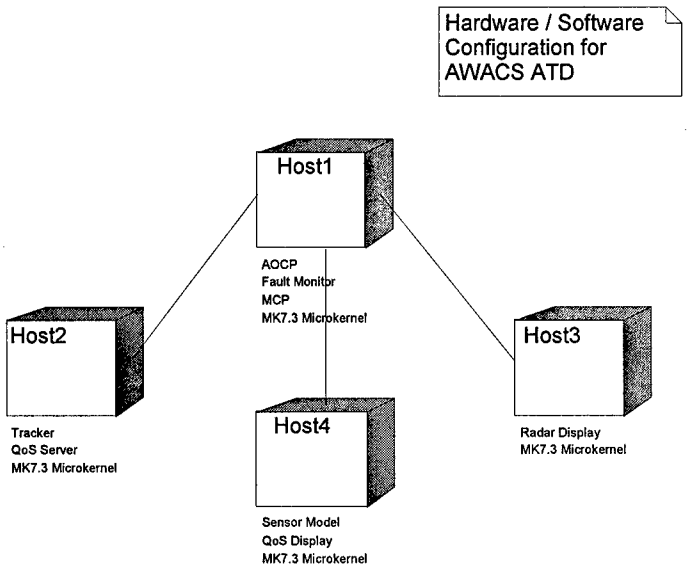

Figure 1. Hardware / Software Configuration

1.2 Results

The adaptive quality of service based tracking application was demonstrated in the AWACS Prototype Demonstration Facility (PDF) at Boeing, Seattle WA, in Sept. 1998. AWACS operators and Boeing engineers interacted with demonstration personnel as part of the ATD evaluation process.

Figure 2 shows an example of the surveillance operator display. The operator could designate important tracks on his display either by individual track or by region. The dynamic resource manager would activate the adaptive mechanism to assure that these operator designated tracks receive a higher QoS then other less important tracks.

Figure 2. Example of Surveillance Operator Display

Figure 3 shows a display of the tracking QoS metrics for both the adaptive and non-adaptive scenarios used in the demonstration.

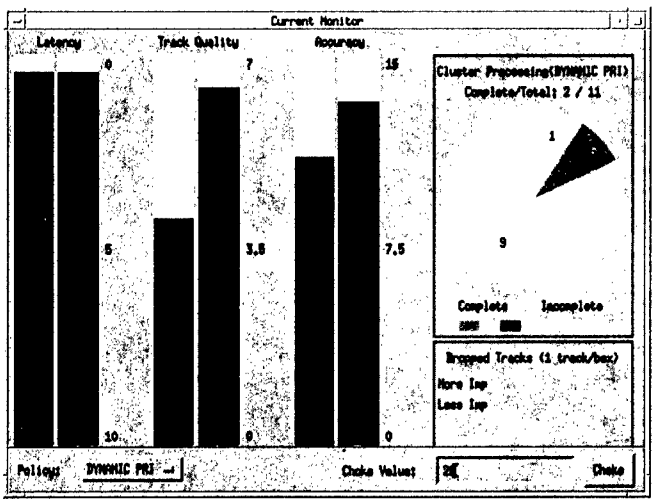

Figure 3. Results with QoS-based Management Policy

The prototype has performed extremely well during demonstrations to its target audience – AWACS operators and tracking system designers. Under "normal" (non-overload) conditions, the tracker handled all tracks as expected and delivered high QoS for all tracks. Overload conditions were simulated by artificially tightening the deadlines for the completion of association processing [LAW98]. Figure 4 shows the results of different scheduling polices (FIFO, Fixed Priority, QoS based) for the tracker algorithm during overload conditions. Track Quality is a well-known metric for AWACS tracking systems, which monitors statistical information on dropped radar scans for individual tracks. A track

is dropped if it misses three consecutive radar scans. The Quality of service based approach described in this paper uses this AWACS defined term (Track Quality) to capture the precision attribute.

When the system is constrained so that only 33% of the tracks could be updated, the prototype delivers essentially perfect QoS for the more important tracks – with an average of 7 in track quality out of a maximum of 7; while delivering a reasonable QoS for the less important tracks – an average of about 4 in track quality.

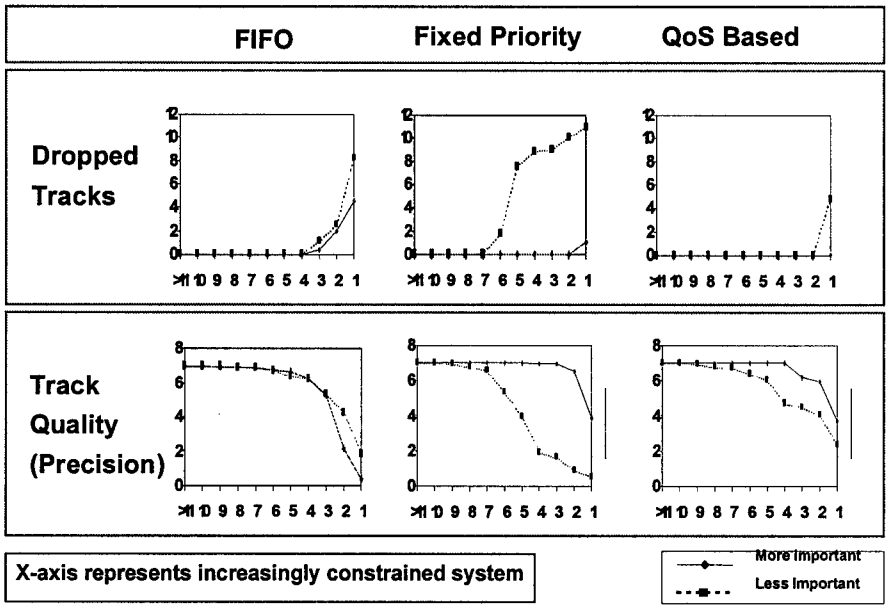

Figure 4. Results for Different Scheduling Policies

When the system is further constrained so that only 15 to 20% of the tracks can be updated, the prototype is still able to keep all of the tracks – both more important and less important tracks – alive. (In some respects, this can be likened to a system where the probability of detection of airborne objects is 15 to 20%.)

And when the system is constrained so that only 10% of the tracks can be updated, the prototype finally allows some tracks to fall to a zero in track quality – that is, the tracker does drop tracks. However, all of these tracks have been of the less important track class. No important tracks have been dropped during the demonstrations, and less important tracks were only dropped under severe resource constraints.

Finally, the demonstration has shown that this technology allows adaptation to occur not just when resources are limited, but also when new resources are added to the system. In that case, which we have demonstrated by loosening the time constraints on association algorithms, the prototype automatically resumed delivering approximately the maximum achievable QoS.

Note that these results are not easily obtainable with static-priority tracking systems. In priority-based trackers, priorities for tracks might reasonably be set according to track importance, where high importance implies high priority. In the prototype tracker, scheduling decisions are based on both importance and timeliness, and even relatively unimportant tracks can have very high application values in a surveillance mission – a

situation that would not arise with straightforward priority-based scheduling.

In the final demonstration track accuracy did not vary. Track accuracy was to change by selecting different association algorithms dynamically , however, due to resource limitations only one association algorithm was implemented.

The demonstration was very well received by the evaluators (AWACS operators and Boeing engineers). The evaluators grasp the technology being demonstrated and expressed the following comments after witnessing the demonstration. All the evaluators agreed this technology is relevant to AWACS. For example, one operator said this was absolutely the direction in which AWACS needs to go. A second operator was in agreement with this comment and said this technology would be an added value in performing his mission. He also appreciated that the system could automatically take advantage of an increase in processing capacity. The operators were especially enthusiastic toward the concept of allowing operators to identify important tracks and geographic regions; and how this operator provided guidance resulted in mission-oriented adaptive resource management. The operators want, very much, to incorporate this concept into AWACS and asked us to keep in contact with them.

Boeing Engineers were also enthusiastic. They asked the ATD team to consider teaming with Boeing on an EFX experiment which would involve combining our adaptivity strategy with a Boeing multi-sensor integration tracking subsystem.

1.3 Future Work

1.3.1 General Recommendations

We are very encouraged by the results of this Advanced Technology Demonstration. The developed prototype offers compelling evidence that the keystone technologies of multi-dimensional QoS, value-based resource management, and distributed threads are effective tools for developing robust, adaptive, distributed Command and Control (C2) systems.

We believe that this project should spawn three new activities. First, we plan to look for opportunities within the DoD to apply this adaptive technology on fielded systems. New start programs or systems that are undergoing pre-planned product improvements will be especially targeted. Second, commercial vendors should be made aware of the results of this project with the goal of having them incorporate the technology, especially value-based resource management and distributed threads in their operating systems and middleware products.

We believe further research and proof of concept developments need to be performed. This project examined building a single adaptive application and managing a single resource type – computer processors. Real-world systems typically have several applications executing concurrently. In addition, resources such as data stores and networks have to be managed. Finally, real systems have multiple users who may have conflicting QoS demands. Further research, experimentation, and technology demonstrations need to be conducted to examine how these technologies can 1) support the development of more capable C2 systems and 2) increase the affordability of future C2 systems.

1.3.2 Specific Recommendations

1.3.2.1 Objective

The objective of this future work is to demonstrate improved operational capability for C2 systems through the use of advanced QoS-driven adaptive architecture technologies.

Improvements in the operational capability of an AWACS platform and various consumers of AWACS track information, connected via a JTIDS network, would be demonstrated by the use of this adaptivity. These improvements include continuing to perform as many as possible of the most important mission critical tasks during periods of overload or resource unavailability, plus better utilization of available resources during normal operations.

This work includes key extensions to our previous work on a multi-dimensional QoS model, value-based resource management, and distributed threads [LAW 98]. The first extension is to systems containing multiple resource types and multiple customers each with different QoS requirements. The second extension is QoS-driven use of resources in addition to processors -- in particular, data stores (i.e., use of an adaptive data manager), and an adaptive JTIDS network.

The result of this effort would be the development of a C2 system of systems prototype that includes many of the C2 functions and services that exist in the real world. The prototype should offer compelling evidence of the robustness that can be architected into C2 systems through the use of our QoS-driven adaptive technologies. Demonstrable improvements in system response to changing missions and environments would be evident.

1.3.2.2 Introduction

C2 systems are subject to constant changes such as a dynamic external environment, temporary overload of internal systems, component failure, and evolving operational requirements. It is greatly advantageous for a system to adapt to these changes by reconfiguring its resources and behavior in an intelligent manner in order to provide support to the varying user/operational needs. Systems that do not employ such adaptivity often use resources in a manner where more important activities receive the same or fewer resources than far less important activities. For example, the current AWACS surveillance system updates its estimates of positions and velocities based on a first-in first-out arrival of sensor reports, i.e., tracks that are associated with the first arriving sensor data are updated before tracks that are associated with slightly later arriving sensor reports, regardless of how important the different tracks are.

Most existing C2 systems either do not adapt or have ad hoc, hard-wired mechanisms to accommodate only a small, predefined set of mode changes. These point solutions have limited ability to scale, are not based on any standard, repeatable methodology, and lack tool support for ease of analysis and implementation.

We have developed and exercised technology that directly addresses how to build and operate computer systems that can adapt to changing circumstances. Key components of our approach include the use of a multi-dimensional Quality of Service (QoS) model [LAW 99], distributed threads [JEN 90], and value-based resource management [JEN 85].

We have developed a demonstration of an adaptive AWACS surveillance tracking application that uses the above technology. This application adapts its behavior, by dynamically selecting which data to process and which algorithm to use in processing the data, based on parameters such as the desired and delivered track QoS, the available resources, and the current operational mission. The resulting improved capability clearly

demonstrates the potential benefits of the technology [LAW 98].

This demonstration was a good first step toward validating our approach, but it necessarily used several simplifying assumptions that we now propose to lift. Namely, our previous work was based on a single adaptive application that modified its behavior based on a single set of QoS requests, and controlled only a single type of resource – computer processors.

This technology now requires further development and validation in a more realistic environment where there are multiple applications and resource types (processors, data stores, networks, etc.), and where there are multiple customers which each have potentially different QoS demands. The intent of the future activity is to conduct the necessary technology development and to provide a worked example of the technology in an AWACS environment.

1.3.2.3 Prototype Description and Underlying Technologies

The prototype would be based on an AWACS platform interacting through a JTIDS network with multiple consumers of surveillance information. This environment is consistent with current and planned DoD concepts of operations where AWACS can be considered a produced of track information that is used by multiple, diverse operational elements.

The AWACS platform would contain multiple applications, including a surveillance tracker, threat evaluator, and importance evaluator. The platform would also include an adaptive data manager that controls access to the track information. All of the above applications (in this context the data manager can be considered an application) would reside on top of a distributed computing infrastructure system that incorporates distributed threads and value-based resource management technologies.

Figure 5 depicts the prototype that would be built. It includes a scenario generator that produces sensor and track information for a wide variety of military systems, an emulation of an AWACS platform, and an emulation of a JTIDS network with several nodes representing consumers of track information.

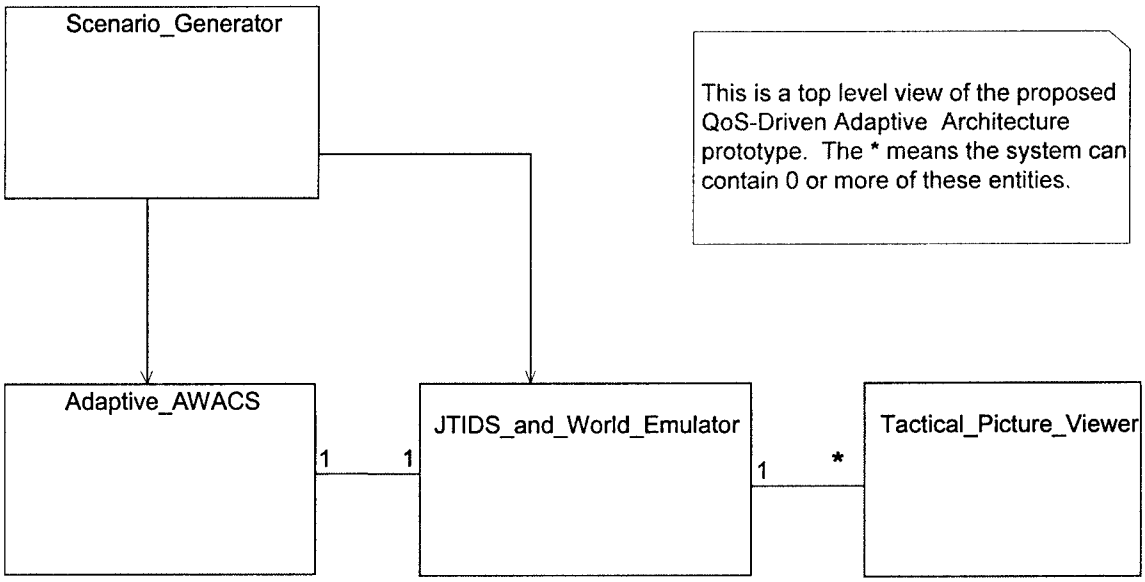

Figure 5. Static View of System Objects

Section 1.3.2.3.1 discusses the underlying technologies that we would be using and extending during this project. This includes the use of a multi-dimensional QoS model, value-based resource management, and distributed threads. A key aspect of this work would be to demonstrate the valuable contributions of this technology to having competing applications share common resources.

Section 1.3.2.3.2 discusses the critical role of data management. We would be extending the ACID properties-based traditional data management strategies to include more flexible adaptive policies that should prove more relevant for systems, such as C2 ones, that have timeliness requirements. This work would be done within the context of what we consider is a likely architecture for future C2 systems, where a data manager plays a key role as the repository of information which various C2 applications (services) use and contribute to.

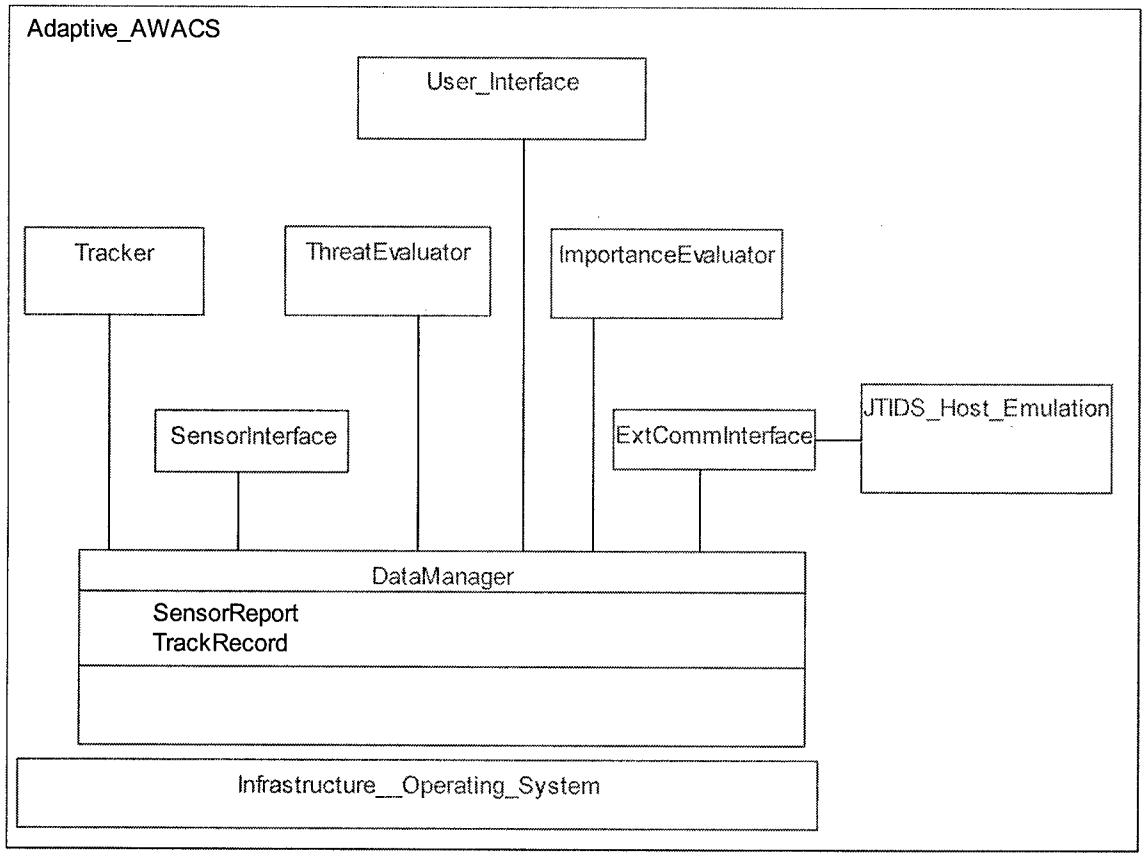

Figure 6. Static View of Adaptive AWACS Objects

Section 1.3.2.3.3 discusses future work in providing a more adaptive JTIDS networking capability. Currently, users of JTIDS information do not play a dynamic role in controlling how the network bandwidth is allocated. Thus, the system is limited in the support it can provide to changes in mission requirements. In addition to adding this role, we also plan to demonstrate how QoS can be used to intelligently broadcast information over the network though the use of adaptive queuing of this broadcast data.

2.1 Core Adaptive Technologies

Throughout our work, we would be using a multi-dimensional QoS model [LAW 98]. This model is applicable at any level of abstraction and acts as a unifying approach for system management. In this model, a data product can be defined along three dimensions: timeliness, precision, and accuracy. Timeliness measures when an event occurs or the delay from initiation to completion of an activity. Precision is a measure of the quantity of data produced as a result of an activity. Accuracy is a measure of the amount of error in the output data.

These attributes are interdependent within a single activity. Improving one generally results in degradation in one or both of the other attributes. For example, to improve the estimate of a maneuvering track's position and velocity, one could use a Kalman-filter based tracker instead of an alpha-beta based one. This results in better accuracy at the expense of increased processing time, and thus a decrease in the timeliness of when a result is available.

The AWACS platform (refer to Figure 6) would be the focus of much of our effort. This is where we would examine how to control and coordinate the multiple applications that are each competing for system resources. For example, at any given time should the processor be allocated to the tracker, which is attempting to update positional data, or to the threat evaluator, which is attempting to detect potentially threatening airborne objects?

Our approach to performing adaptive resource management in a heterogeneous application environment relies heavily on a technology called value-based resource management [JEN 85]. Time-value functions (TVF) represent application-specified profiles/desires that reflect physical time constraints. These functions describe orthogonal facets of urgency (timeliness) and importance (benefit), thus eliminating the notorious complexities of mapping timeliness requirements onto priorities. A TVF expresses the benefit of producing a result to the time at which it is produced (See Figure 7.); deadlines are a simple special case, a unit binary-valued, downward step (See Figure 8.). Time-value functions are derived by the user from the physical nature of the application environment, and may evolve dynamically over the course of a mission.

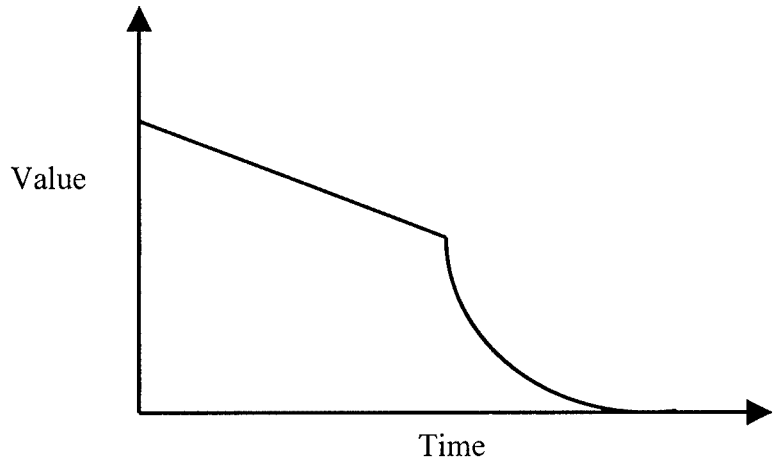

Figure 7. Time-Value Function

A Best-Effort scheduler uses the TVFs and information about resource usage expectations to create a schedule of tasks. [BE scheduling policies need to be introduced first. BE policies and TVF's are orthogonal concepts. We use both because they are so synergistic, but one wouldn't have to.] If all tasks can meet their deadlines [note that this is again specific to

Ray's algorithm, it is not generally true for BE scheduling], they are scheduled Earliest Deadline First (EDF) , which is known to be optimal for meeting deadlines on a uniprocessor. Otherwise, the scheduler uses heuristic methods to determine a scheduling order that attempts to maximize the value accrued by the overall system. Tasks that cannot meet their time receive exceptions and can execute recovery activities.

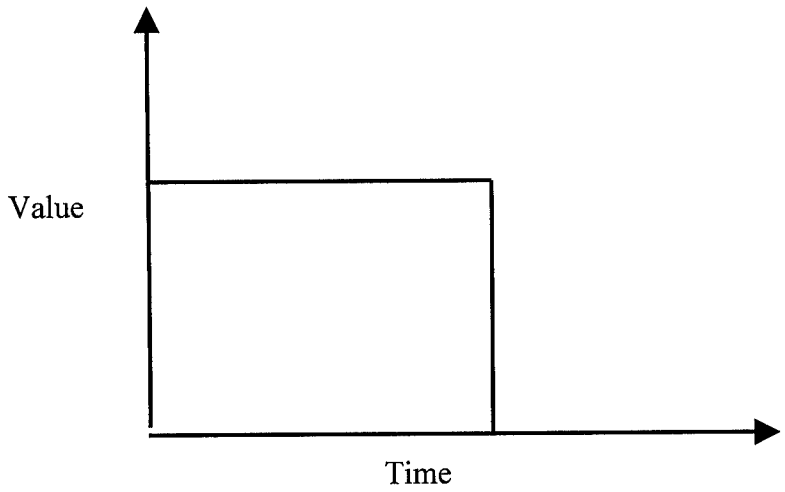

Figure 8. Deadline, a Special Case of a Time-Value Function

Value-based resource management raises a complimentary aspect of system design that is not usually addressed. Not only can workload be shed in a fine-grained manner, it can also be added in the same way. That is, not only can value-based resource management support the normal (or baseline) computations in a system, it can allocate (otherwise) "excess" resources incrementally to perform additional (optional) computations. Many systems are designed for deployment with a safety margin of excess resources. Value-based resource management could use these resources constructively to enhance the system's base capabilities. Over time, as new applications or sensors are added, consuming some of the "excess" resources, fewer and fewer extra computations will be performed. Significantly, all of this change would occur simply as a result of using value-based resource management. Also, as more resource capacity is added, it can be applied dynamically where it is most needed.

Our adaptive AWACS platform would contain several applications such as a tracker, threat evaluator, importance evaluator, and interface to external communications (JTIDS). In addition to these, the prototype will also contain an adaptive data manager. An important goal of this project is to address how to weaken traditional data management ACID properties (**A**tomic, **C**onsistent, **I**solated, and **D**urable) to allow QoS-based system adaptivity. This topic is discussed more fully in the Adaptive Data Management section of this proposal.

The applications would reside within a network of physically dispersed computers. Some applications would exist on a single computer (and compete for its resources with other applications) while other applications will be distributed across multiple computers in the sense discussed below.

Distributed threads are a key technology in our distributed processing architecture. The Alpha operating system introduced an object-oriented programming model well suited for writing large, real-time, distributed software, including the distributed thread abstraction [JEN 90]. A distributed thread, see Figure 9, is a locus of control that spans objects and

physical nodes, transparently and reliably, via method invocation.

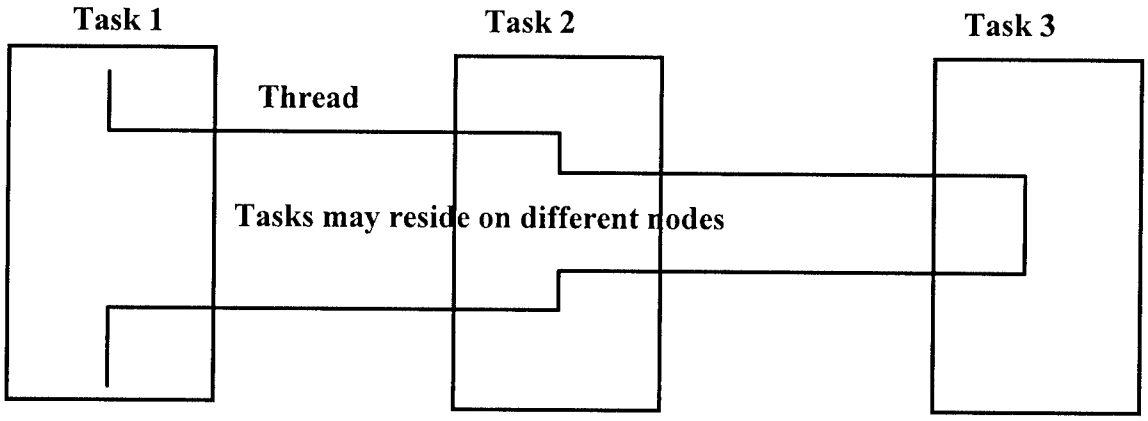

Figure 9. Distributed Thread

Distributed threads are similar in concept to local threads communicating via Remote Procedure Calls (RPC), except that a distributed thread maintains its unique system identity and its attributes (particularly its timeliness attributes) even while it executes on different nodes. The system maintains and enforces end-to-end time constraints, resource requirements and limits, and security identity throughout the execution of the thread. Thus, an application thread receives an exception upon the expiration of a time constraint regardless of whether the thread is executing in an application object, a local system object, or a remote object that happened to be invoked by a local object.

Our experience is that the users/operators play a critical role in enabling a system to adapt to changing circumstances. They provide the most effective means of identifying current mission objectives and priorities. This operational knowledge is used both during system design, when value functions and adaptation rules are developed, and during operational use when they perform situational assessments and identify important areas of interest and upcoming mission priorities.

Figure 10 depicts the currently identified human-machine interactions for the proposed prototype. The humans are separated into two categories: mission operators; and demonstration and experiment controllers. Mission operators must be able to identify their QoS demands, view delivered QoS results, identify important objects and areas, and view track information. People staging demonstrations and experiments must be able to create scenarios, initialize the system, and start/stop scenarios. We expect that the number of interactions would likely increase once we begin system analysis and design activities.

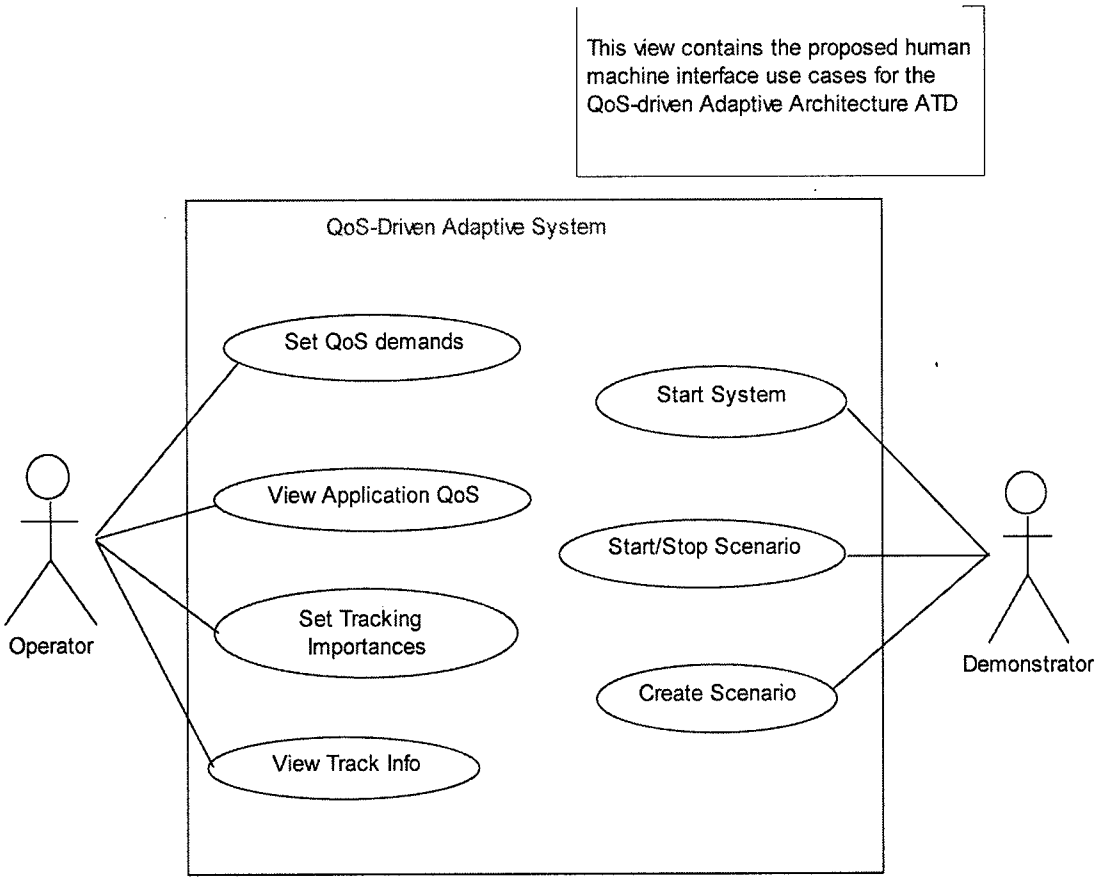

Figure 10. Human-Machine Interface Use Cases

2.2 Adaptive Data Management

We would develop data management techniques that support QoS-driven adaptivity. Most existing data management techniques are based on serializability as the correctness criterion that is enforced through approaches such as two-phase read/write locking. Timeliness is either secondary or completely ignored. Furthermore, existing data management techniques are designed without regard to operating system scheduling capabilities [in fact, database system designers deliberately create their own scheduling and other resource management facilities instead of using the OS facilities, both for portability and for performance] – in particular, time-value based QoS scheduling techniques are not employed.

Our work would extend the data management techniques in two ways. First, we would use best-effort concepts for data management transaction scheduling. This allows us not only to tradeoff scheduling of an individual database transaction, but to have a uniform representation for database transactions and application processing.

Second, we would investigate the scheduling and processing done for an individual transaction. Individual transactions can update multiple data, hence ideas of tradeoffs of timeliness, precision, accuracy can be applied across this data. The techniques that were developed for imprecise computation will be investigated [LIU 91]. Applying TPA tradeoffs hierarchically to nested transactions would provide the foundation for further development of QoS adaptive theory and benefit based scheduling.

We anticipate that the results of our work would demonstrate a definite, quantifiable,

positive impact on command and control systems, such as AWACS. These systems are currently investigating use of full data management systems to provide structured, robust access to shared, persistent data. Also, by providing support for complicated data sharing, the systems would better support future upgrades as they are needed. The system developers are finding that off-the-shelf data management systems have a sole emphasis on data accuracy, and are thus inadequate in systems that require QoS tradeoffs. For instance, an AWACS system that must deliver a track over a JTIDS data link within tight timing constraints does not want to find the data locked by the data manager due to logging taking place.

2.3 Adaptive Networking

This future work extends and applies our previous work on QoS adaptation into the realm of networks. The "application" chosen for this demonstration is JTIDS. The primary goal is to improve the ability of JTIDS to deliver the needed surveillance information to consumers based on their current (and potentially changing) mission needs. This would be done by incorporation of surveillance QoS trade-offs in queue and network capacity allocation decisions, along with feedback control from the surveillance data consumers.

JTIDS is one of DoD's primary wireless broadcast networks that support dissemination of tactical battlefield information. JTIDS consists of several host terminals for each source and each sink of information. The main JTIDS resource is bandwidth; this is divided among each of the hosts. JTIDS operates as a time division multiple access bus. The bandwidth is divided into time slots (128) per common static period. Each host executes the same algorithm that determines the host's permanent assignment of slots based on the total number of hosts in the network.

The current JTIDS has a static slot allocation, but a new approach – the time slot reallocation (TSR) algorithm [POS 96] – has been developed over the last decade, and is now being implemented for the NAVY JTIDS host terminals. The TSR algorithm allocates bandwidth dynamically based on, among other things, its knowledge of all host demand bandwidth requests. The time slot reallocation process is repeated periodically (the reallocation period is longer than the time slot period).

A significant issue, however, is that the consumers of the JTIDS broadcast network do not participate in the time slot reallocation algorithm, either in the current JTIDS network or in the upgraded TSR terminals. They are passive listeners on the broadcast network. Thus, they can not cause reallocation of slots to occur, even if they have a preference for which information they want to receive or which source of information they prefer.

The upgraded JTIDS activity created a special J0.7 message that is used by the information sources to broadcast their bandwidth demands. These demands are used by the TSR algorithm on each source platform host JTIDS terminal to determine the platform time slots. For the purposes of this proposal, we will only dynamically allocate bandwidth, above that done by TSR, for the surveillance information.

We plan to allow the consumers to use these J0.7 messages to broadcast their requests. The information sources can then adjust their bandwidth demands via the time slot allocation algorithm itself. The AWACS platform, in addition to attempting to acquire more or less of the available bandwidth, will also prioritize the information it queues for broadcast via its external communications component (see Figure 11).

The TSR algorithm only takes into account the bandwidth demands of the sources without regard to the data that they broadcast. We propose to incorporate into this bandwidth allocation algorithm the quality of service of the data that is broadcasted. The QoS

parameters that we previously defined and studied for AWACS tracks (such as importance, track latency, track quality, and track accuracy), will be extended to the track data that an AWACS JTIDS host terminal broadcasts over the net. The timeliness, accuracy and precision of the data, and the trade-offs of redundant data from multiple sources, all should be part of the JTIDS bandwidth adaptive allocation strategy to deliver the data with the best possible QoS.

Another dimension of JTIDS-related adaptivity we would examine during this project is the trade-off between throughput and jam resistance. We could potentially trade-off the amount of information that goes into the allotted slot vs. redundant information that provides jam resistance. JTIDS currently has this capability but the amount of redundancy is set-up statically and does not change based on actual jamming or user requests.

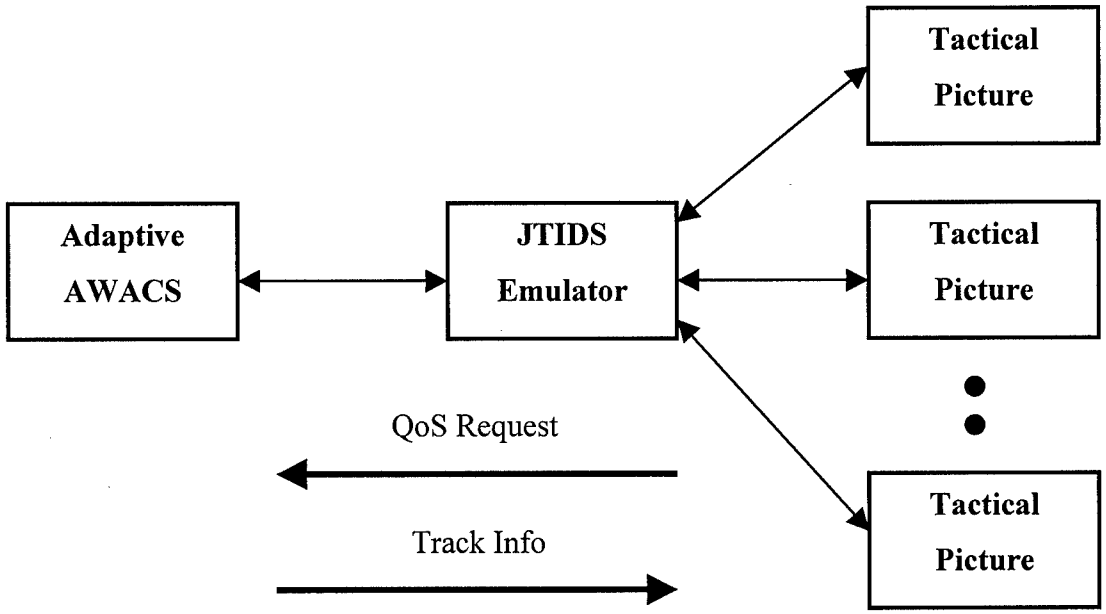

Figure 11. Adaptive JTIDS Network

We plan to take advantage of existing simulation and emulation resources for both our scenario generation and JTIDS emulation needs. The JTIDS emulation capability currently available at the Hanscom Air Force Base Modeling and Simulation Center (MASC) contains two major parts: EADSIM, which is a scenario generator, and emulates any platforms that contains JTIDS hosts; and ADSIM, which is an emulator of JTIDS itself.

EADSIM is useful because of its ability to generate analogous scenarios for multiple platforms. ADSIM simulates multiple host terminals, one for each platform emulated by EADSIM. The platforms emulated by EADSIM represent the sources of the information, and are the only hosts participating in negotiation of the bandwidth for the JTIDS network surveillance data. Multiple consumers can be connected to ADSIM, but they do not currently participate in the bandwidth allocation. Tactical picture viewers would be used by this project to demonstrate the feedback loop of surveillance consumer QoS requests, and the receipt of track information over the JTIDS network. The proposed flow of information is depicted in Figure 11.

This furture work would produce a full-scale prototype that includes QoS-driven adaptive data and network management for multiple competing applications and customers.

3.1 REFERENCES

[BEN 95] Bensley, E, et. al., "Evolvable Systems Initiative for Real-Time C3: Infrastructure Requirements", ICECCS, November 1995.

[JEN 85] Jensen, E.D., et. al., "A Time-Driven Scheduling Model for Real- Time Operating Systems," Proceedings of IEEE Real-Time Systems Symposium, 1985, pp. 112-122.

[JEN 90] Jensen, E.D., Northcutt, J.D., "Alpha: An Open Operating System for Mission-Critical Real-Time Distributed Systems—An Overview," Proceedings of the 1989 Workshop on Operating Systems for Mission-Critical Computing, ACM Press, 1990.

[LAW 98] Lawrence, T., et. al. "Quality of Service for AWACS Tracking", 4th International Command and Control Research and Technology Symposium, Stockholm Sweden, September 1998.

[LAW 99] Lawrence, T. F., "Quality of Service (QoS) a Model for Information", Proceeding of the Fourth International Workshop on Object-oriented Real-time Dependable Systems (WORDS99), Santa Barbara, CA, 27-29 January 1999.

[LIU 91] Liu, I., Lin, K., et. al. "Algorithms for Scheduling Imprecise Computation", IEEE Computer, 24(5), May 1991.

[POS 96] Post, A. E., "Time Slot Reallocation (TSR) Interfaces", MITRE Working Note WN 96B129, The MITRE Corporation, September 1996.

Appendix A

Workshop on Object-oriented Real-time Dependable Systems

This appendix contains a paper presented at the Fourth International Workshop on Object-oriented Real-time Dependable Systems. This paper describes the Quality of Service (QoS) model used in this Advanced Technology Demonstration and attempts to unify the concepts of QoS, information, objects and dependability.

Quality of Service (QoS)

A Model for Information

Thomas F. Lawrence

Air Force Research Laboratory

Rome Research Site

lawrence@rl.af.mil

ABSTRACT

A unifying quantitative information model is proposed. This model defines information in terms of the attributes of timeliness, precision, and accuracy. Each attribute can be expressed in terms of measurable quantities (metrics). Information is application dependent and is expressed as benefit (utility or value) an attribute imparts to the application. Thus information is a function of timeliness, precision, and accuracy. Statistical values of these attributes can be used to express dependability (i.e. information availability). The purpose of the model is to give quantitative expression to information and system properties, to provide a basis for characterizing system modules, and to stimulate development of analytical techniques for predicting end-to-end quality of service in complex distributed information system.

1. Quality of Service Attributes

A datum is information only if it reduces the uncertainty regarding a correct out come in a process in which it is used. A datum which has a higher probability of producing a correct out come than another datum is said to have a higher information content. This datum with a higher information content is also said to be a greater benefit, value or utility to the application. Further, a datum is characterized by its timeliness (T), its precision (P), and its accuracy (A) [Sabata1997], [Lawrence 1997].

Therefore both the effective information content delivered at an interface and correctness are a function of T,P&A. The quantity of service delivered at an interface is defined as the particular values for T,P&A measured at an interface. However, the Quality of Service (QoS) at an interface is application dependent and is defined as the benefit (B) [Jensen 1985] or utility a particular measured T,P&A set imparts to the application. Consequently zero benefit constitutes zero quality of service and zero information at that interface. Benefit exists in degrees from no benefit to maximum benefit.

Timeliness is a specification (measure) of the time (when) a particular quantity of data is to appear at an interface. The specification of timeliness at an interface includes all the effects on correct operation due to time. For example: (1) the synchronization of the output with a real world event, (2) the perishability of data, (3) human reaction time considerations, (4) time relationship between successive outputs (e.g., jitter due to inter-frame timing), (5) the synchronization of the output with outputs of other system modules.

Precision is a specification (measure) of the quantity of data produced or the degree of expressibility of the output. Precision is a syntactic issue and only measures, for example, the number of bits produced at the output not the meaning of the bits. The specification of precision at an interface includes all the effects on correct operation due to the completeness of the data. Examples of precision issues are: (1) in calculating an infinite series, each additional term becomes an increase in precision (to what extent does each additional term contribute to correct operation), (2) in a video film clip precision factors include the metrics; the number of frames per second and the number of pixels per frame. Also reducing the number of colors presented in the video reduces the number of bits per pixel and therefore the precision of the output.

Accuracy is a specification (measure) of the error in the data at the output. Accuracy is a semantic issue and is concerned with the meaning of the bit pattern of a particular output instance. Accuracy is concerned only with the error in the output due to a transformation of the input into an output. Correctness, not to be confused with accuracy, is a higher level abstraction that takes into consideration timeliness, precision, and accuracy contributions to correct operation. Each of the three attributes has instantaneous values for each output and also statistical values over multiple instances.

By developing instantaneous and statistics metrics for the T,P&A attributes at an interface in an information system the information content at that interface is quantified. At least three different types of benefit functions can be developed. The first, an instantaneous benefit function can be developed for each attribute. In this case two of the attributes are at their optimal instantaneous value and we vary the instantaneous value of the third attribute and develop a benefit function. All of these attribute benefit functions can be considered "fuzzy" sets. Where each instantaneous value of an attribute belongs to the domain of application acceptability to a certain degree (benefit). Using these instantaneous benefit functions a statistical benefit function for each attribute can be developed. The primary statistical metric is the variance from the optimal instantaneous value of each attribute. These different variance values, and their associated benefit form fuzzy statistical sets for each attribute. The variance consists of three factors: (1) the range of instantaneous attribute values over which the statistic is germane, (2) the percent of instantaneous values that fall in that range, and (3) the distribution of instantaneous values (e.g., no more than three values in sequence can fall out of the specified range).

2. Dependability

A second set of benefit functions, one benefit function for each attribute, can be produced using the statistical metric of availability. The availability of an attribute is defined as a particular attribute variance over some large time (TA). For example depending on the application, TA can be an hour, a day, or 10 years. TA is the time over which a maximum variance would have an acceptably small chance of being exceeded. The larger the TA and the smaller the variance the higher the availability of an attribute. As the variance decreases, higher levels of availability for an attribute occur thus producing a statistical "fuzzy set" for an attribute. This resulting benefit function quantifies the contribution of each attribute to the information content at an interface as a function of the variance of each attribute. The aggregate of these attribute availability metrics specifies the dependability of an object.

The third type of benefit defines the relative benefit of different objects where each object has its own T,P&A instantaneous and statistical benefit functions. That is, some objects are more important to the application that other objects and therefore have higher relative benefit and higher information content. In a gracefully degrading system [Lawrence 1998] the information content of the most important objects are maintained at a high level by minimizing the T,P&A variances while the information content of the least important objects is decreased by increasing the T,P&A variances.

3. Quality of Service Principles

Three principles become evident; (1) QoS or information content is defined by the timeliness, precision and accuracy benefit functions (fuzzy sets); (2) For a given object the better behavior in one attribute results in a worse behavior in one or both of the other attributes e.g., decreased variance in one attribute brings increased variance in one or both of the other attributes; (3) The better behavior in one attribute for one object results in a worse behavior in any other or all other attributes for any or all object(s). Principles 2 & 3 assume finite shared resources with algorithms of constant efficiency.

4. Interface Properties

Information is divided into two categories: (1) objects that should appear at an interface and (2) objects that should not appear. Objects that should appear are defined to have positive benefit and those that should not appear have negative benefit. The T,P&A availability of objects that should appear should have a variance above some minimum whereas, the T,P&A availability should be below some maximum for objects that should not appear. Objects that should appear can be considered as specifying the liveness property or functionality of the interface. If this liveness specification is satisfied "denial of service" is avoided. One can also consider the liveness specification as defining appropriate "signal" level at the interface. Objects that should not

appear can be considered as specifying the safety property or "confidentiality" of the interface. One can also consider the safety specification as defining appropriate "noise" level at the interface. Both properties (liveness & safety) must be satisfied for correct behavior of an interface. In a perfect world, in the absence of anomalies, both properties would always be satisfied. In the real world, anomalies [Cristian 1991] such as failure of components, design faults, and shared resource conflicts require the implementation of anomaly management mechanisms in order to keep the T,P&A variances in the proper range for both types of objects. The QoS metrics of the objects in the liveness set define the degree of reduction in uncertainty with respect to satisfying the liveness property. The QoS metrics of the objects in the safety set define the degree of reduction in uncertainty with respect to satisfying the safety property. The tradeoff between liveness and safety, provided by the application, ultimately defines a new "fuzzy set" that specifies the correct behavior of an interface. In this model there are two properties (liveness and safety); three attributes, (timeliness, precision, and accuracy) that define the properties; statistical availability metrics for each attribute that defines dependability; and instantaneous values for T,P&A that are used to compose the statistical metrics. This model with its quantifiable attributes becomes the basis for both describing information modules (hardware and software) in terms of the T,P&A inputs and outputs and hopefully a point of departure for quantifying the aggregate behavior of many modules connected in a parallel and/or serial manner (i.e. end to end behavior).

5. Conclusions

The conclusions drawn are:

(1) QoS or information content of an object can be known only if all three attributes are known.

(2) If any attribute is insufficient the information content is zero.

(3) Anomaly management mechanism may be described by their effects on the statistical behavior of T,P&A at some interface.

(4) The availability metric for each attribute is critical and the availability factors become parameters which can be manipulated (or specified) to provide the appropriate level of information content for objects at an interface.

(5) Information is defined by the application and user evolvement in the initial design process is necessary before any rational design process can begin.

(6) Objects can be thought of as being semi-amorphous with the degree of amorphous behavior, defined by the T,P&A "fuzzy sets".

6. References

[Jensen, 1985] Jensen, E.D., et. al., "A Time-Driven Scheduling Model for Real-Time Operating Systems," Proceedings of IEEE Real-Time Systems.

[Cristian, 1991] F. Cristian, "Understanding Fault-tolerant Distributed Systems", Communications of the ACM, Vol. 34, No. 2, February 1991.

[Sabata, 1997] B. Sabata, S. Chatterjee, M. Davis, J. Sydir, T. Lawrence, "Taxonomy of QoS Specification", Proc of the IEEE Computer Society 3rd International Workshop on Object-Oriented Real-Time, Dependable Systems (WORDS '97), Newport Beach, CA, Feb 1997.

[Lawrence, 1997] Lawrence, T.F., "QoS Perspective," Next Generation Information Environment (NGIE) Workshop, Air Force Research Laboratory, Hamilton, NY, November 1997.

[Lawrence 1998] T. Lawrence, P. Hurley, T. Wheeler, A. Kanevsky, J. Maurer, P. Wallace, D. Wells, R. Clark, "Quality of Service for AWACS Tracking," Proceedings of the Fourth International Symposium on Command & Control Research & Technology, Nasby Slott, Sweden, September 1998.

Appendix B

Workshop on Parallel and Distributed Real-Time Systems

This appendix contains a paper accepted for the 1999 International Workshop on Parallel and Distributed Real-Time Systems. This paper contains our most complete description of the adaptive tracking application and the results achieved to date with the prototype developed on this Advanced Technology Demonstration.

An Adaptive, Distributed Airborne Tracking System

("Process the Right Tracks at the Right Time")

Raymond Clark[1], E. Douglas Jensen[1], Arkady Kanevsky[1], John Maurer[1], Paul Wallace[1], Thomas Wheeler[1], Yun Zhang[1], Douglas Wells[2], Tom Lawrence[3], and Pat Hurley[3]

[1] The MITRE Corporation, Bedford, MA, USA
rkc@mitre.org
[2] The Open Group Research Institute, Woburn, MA, USA
d.wells@opengroup.org
[3] Air Force Research Laboratory/IFGA, Rome, NY 13441, USA
hurleyp@rl.af.mil

Abstract. This paper describes a United States Air Force Advanced Technology Demonstration (ATD) that applied value-based scheduling to produce an adaptive, distributed tracking component appropriate for consideration by the Airborne Warning and Control System (AWACS) program. This tracker was designed to evaluate application-specific Quality of Service (QoS) metrics to quantify its tracking services in a dynamic environment and to derive scheduling parameters directly from these QoS metrics to control tracker behavior. The prototype tracker was implemented on the MK7 operating system, which provided native value-based processor scheduling and a distributed thread programming abstraction. The prototype updates all of the tracked-object records when the system is not overloaded, and gracefully degrades when it is. The prototype has performed extremely well during demonstrations to AWACS operators and tracking system designers. Quantitative results are presented.

1 Introduction

Many currently deployed computer systems are insufficiently adaptive for the dynamic environments in which they operate. For instance, the AWACS Airborne Operational Control Program (AOCP), which includes the tracking system, has a specified maximum track-processing capacity. Since the tracker processes data in FIFO order, it fails to process later data if its processing capacity is exceeded. Since sensor reports generally come in the same order from one sweep to the next, it is likely that, under overload, sensor reports from a specific region will not be processed for several consecutive sweeps. This overload behavior can result in entire regions of the operator displays that do not get updated. This is a potentially serious problem because there is no inherent correlation between important regions of the sky and the arrival order of sensor reports.

This situation, while undesirable, is handled by skilled operators, who recognize that some data is not being processed and take remedial actions (e.g., reducing the gain of a sensor or designating portions of the sky that do not need to be processed). While these manual adaptations reduce the tracker workload, they are not ideal. Reducing sensor gain might cause smaller, threatening objects to go undetected.

Injecting more intelligence into the tracker could avoid such compromises. The US Air Force Research Laboratory at Rome (NY), The MITRE Corporation, and The Open Group Research Institute undertook a joint project to explore that approach. The project recently concluded after producing an Advanced Technology Demonstration (ATD) featuring a notional, adaptive AWACS tracker that can execute in a distributed configuration, with attendant scalability and fault tolerance benefits.

The ATD tracker "processes the right tracks at the right time" by appropriately managing the resources needed for track processing. The prototype updates all tracked-object records when it has sufficient resources, and gracefully degrades otherwise. A single underlying mechanism automatically provides this degradation, despite its manifestation as a succession of qualitative operational changes: First, more important tracks receive better service than less important tracks while all tracks continue to be maintained. Under severe, sustained, resource shortages, less important tracks are lost before more important tracks. (This is referred to as *dropping* tracks and is described more formally in Section 4.2.) Moreover, the tracker automatically delivers improved service whenever more resources become available—in essence, tracker performance gracefully degrades and gracefully improves without direct human intervention.

2 Adaptivity, Value-Based Scheduling, and Quality of Service

Military planners have observed that "you never fly the same mission twice," implying that flexibility and evolvability are important design characteristics. With that in mind, we employed a relatively straightforward approach to adaptivity for this project—decomposing an application (or a set of applications) into component computations, which are then assigned application-specific values reflecting their individual contributions to the overall mission. By scheduling computations to maximize the accrued application-specific value, the overall system can perform a given mission well.

In some cases, associating a value with a computation seems natural. For instance, in financial trading applications, the value of performing a computation might be a function of the market price and the production cost. More complex situations involve other factors, such as human safety, or deal with a non-monetary domain. The adaptive tracker described here deals with a domain that has not traditionally employed a monetary model, providing a worked example for such a domain.

This project selected a research operating system that provides a value-based scheduling policy to applications. This makes the mapping from the computation values to scheduling attributes trivial. On other operating systems, this mapping would typically translate the application-specific values into scheduling priorities, which have a much more restricted value domain. The mapping could be performed by either the application directly or by a middleware package.

While application decomposition with appropriate value assignments assists in effectively accomplishing a specified mission, application effectiveness could be further increased by employing feedback to directly drive its behavior. To do this, application-specific figures of merit (that we refer to as Quality-of-Service (QoS) metrics) are specified at design-time, and evaluated or estimated dynamically at run-time in order to monitor—and subsequently control—overall application operation. Section 4 discusses this in some depth for the AWACS ATD, where the adaptive tracker uses feedback based on QoS metrics for individual tracks to determine the allocation of processing resources for current track processing.

3 The Tracking Problem

Surveillance radar systems are an important class of real-time systems that have both civilian and military uses. These systems consist of components for sensor processing, tracking, and display. In this study we concentrated on the tracking component, which receives *sensor reports*—the output produced by the sensor-processing component—and uses them to detect objects and their movements [1]. Each object is typically represented by a *track record*, and the collection of all track records is commonly referred to as the *track file*. Sensor reports arrive at a tracker periodically, and each report describes a potential airborne object. The number of sensor reports can vary from radar sweep to radar sweep, and some sensor reports represent noise or clutter rather than planes or missiles. When new sensor reports arrive, a tracker correlates the information contained in the sensor reports with the current estimated track state to update the track records that represent the tracking system's estimate of the state of the airspace.

A typical tracker comprises *gating*, *clustering*, *data association*, and *prediction and smoothing* stages. Gating and clustering splits (*gates*) the problem data into mutually exclusive, collectively exhaustive subsets of sensor reports and track records, called *clusters*. Data association then matches the cluster's sensor reports with its track records. The final stage of a tracker—prediction and smoothing—computes the next position, velocity and other parameters for each object using its track history and the results of data association.

3.1 Tracking Software for the ATD

Advanced Technology Demonstrations are intended to transfer technology into an application setting so that it can be utilized and evaluated by the operational and acquisition communities. In order to focus on this transfer, the project used

existing tracking software and terminology, ensuring that ATD evaluators would be familiar with the overall function and structure of the tracker.

The project adapted a tracker that processed information from multiple sensors, including multiple types of radar. That tracker performed a single-threaded computation: That is, a single thread in the tracker performed all of the tracking stages described above from receiving sensor reports to updating the track file.

3.2 Adaptive Tracking

The motivating problem was the (mis)behavior of the tracker under overload, which can be intentionally caused by an enemy. Since, under overload, all of the incoming data cannot be processed, the project's goal was to allow the tracker to do a better job of selecting a subset of incoming data that could feasibly be processed by allowing scheduling decisions that had previously been made at design time to be deferred until run time. There were two major changes: processing was divided into smaller-grained units of work that could be scheduled independently and concurrently; and value-based design principles were employed to determine appropriate scheduling parameters for those individual work units.

Multithreading and Concurrency Presumably, a single-threaded adaptive tracker could be constructed by properly ordering the association computations so that those that are most critical are attempted first. While that approach addresses the overload problem at hand, it suffers from the serious limitations described below, motivating us to design a multithreaded AWACS ATD tracker that utilizes a separate thread for each association computation to be performed.

First, a single-threaded tracker could not take advantage of available distributed resources, including both multiprocessors and other processing nodes.

Secondly, ordering the association computations in the application reduces or eliminates the possibility of taking advantage of certain operating system and middleware resource management facilities. For instance, the OS employed for the AWACS ATD offered a processor-scheduling policy that accepted explicit application time constraints for computations and performed the set of computations that maximized accrued value (in application-specific terms).

Application of Value-Based Design Principles Given the multithreaded design described above, with a separate thread assigned to perform each association computation, and an underlying scheduler that attempts to maximize accrued value, the adaptive tracker design problem is reduced to a straightforward question: Can scheduling parameters be selected for tracks and clusters that reflect the value associated with their processing?

Considerable effort was devoted to answering that question for the ATD. Fortunately, at about this time, the tracking community was independently encouraged to think in terms of "selling" track information to customers—that is, operators and decision makers. That point of view helped make the notion of establishing the application-specific value of a track quite natural. In fact, the

project developed a set of QoS metrics for individual tracks. These per-track QoS metrics, described in Section 4.2, directly determine the scheduling parameters for a cluster, which, in turn, affect the future QoS metrics of each component track.

4 Value-Based Design

AWACS can perform a number of different missions: e.g., it can manage logistics such as refueling, perform air-traffic control, or carry out general surveillance. In order to maximize the depth of this initial work, we applied the principles of value-based design to a single mission. Because of its general utility and intuitive simplicity, a *surveillance* mission was chosen for the ATD.

When flying a surveillance mission, AWACS operators attempt to monitor all airborne objects in a large region. Once an object has been identified, the tracker follows its progress (i.e., "track" it). The more closely the tracker's estimate of an object's position and heading agree with reality, the better.

Moreover, once the tracker has identified a track, it should not "drop" it erroneously. A track is dropped if it is not updated for a number of input sensor cycles. While this is inevitable if no new sensor input is received for the track, the project focused on cases where sensor input is received, but is not processed. (A dropped track can be rediscovered; but this is a relatively costly operation and there is no assurance that any individual track will be reacquired.)

4.1 Adaptive Tracker Behaviors

A key step to producing an adaptive tracker was to develop a specification of its desired behavior—in particular, its behavior under overload. This is an application specification task, but was of interest to the project because it required specification of behaviors that were new to the application domain.

The specification of adaptive tracker behaviors occurred in two phases. The first phase described a high-level policy, but did not address tradeoffs that could arise when attempting to satisfy conflicting policy objectives. For example, the high-level adaptive tracker should preferentially process tracks that:

- are in danger of being dropped
- the user has identified as "important"
- have poor state (position and velocity) estimates
- are maneuvering
- potentially pose a high threat
- are moving at high speed

The second phase refined this high-level policy by addressing conflicting policy objectives. In general, project members could identify these conflicts, but needed expert assistance to determine the proper resolutions (i.e., tradeoffs).

4.2 QoS Metrics for Tracks

Policy refinement required the definition of QoS metrics. Where the high-level policy could usefully describe behaviors for more and less important tracks, or make a distinction between more and less accurate track positions, the implementation of that policy required precise specification of these terms.

There are a number of traditional metrics for tracker performance. Many of these do not address the central issues of the ATD. For example, the ability of the tracker to follow aircraft through maneuvers, while obviously a valuable capability, is primarily a function of factors such as a sensor's probability of detection, the sensor's revisit rate, and the association algorithm employed. The first two factors were out of our scope, and we have not yet implemented the dynamic selection of an association algorithm based on QoS metrics. Rather, the ATD focused on the decision of when to execute instances of known algorithms.

The project settled on three QoS metrics that could quantify track processing:

- Timeliness: the total elapsed time between the arrival of a sensor report and the update of the corresponding track file record.
- Track quality (TQ): a traditional measure of the amount of recent sensor data incorporated in the current track record. TQ is incremented or decremented after each scan and ranges between zero (lowest quality) and seven (highest quality). If TQ falls to zero, the track is dropped.
- Track accuracy: a measure of the uncertainty of the estimate of the track's position and velocity.

In addition to these QoS metrics, each track was dynamically assigned to one of two *importance classes*. A track could be deemed more important for a number of reasons, including designation by an operator, flying in an operator-designated region, or posing a significant threat to the AWACS platform.

For the sake of intellectual manageability and to simplify the task of producing the first prototype adaptive tracker, the QoS metrics and the track importance designation were coalesced into a small number of classes. As mentioned above, there were two track importance classes; in addition, there are three track quality classes and two track accuracy classes. (AWACS operators subsequently validated this level of granularity.)

4.3 Quantified Adaptations

Based on these classes, project members refined the high-level adaptive tracking policy by focusing on specific tradeoffs involving pairs of track QoS metrics or track importance. For instance, one ranking considered only the track importance and track quality of competing clusters. Several such pair-wise rankings were merged to form a total ordering of the cases (with numeric values) as a function of track quality, track accuracy, and track importance.

At that point, tracking experts helped quantify the relative values of the bins. We also examined several cluster scenarios "by hand" to assign overall values

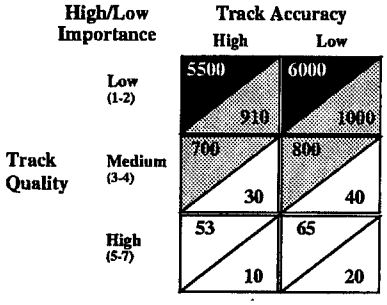

Fig. 1. Track Values as a Function of Track Quality-of-Service Metrics

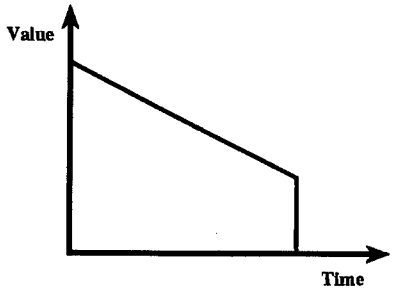

Fig. 2. Shape of Time-Value Functions for Association Computations

and perform fine tuning. Figure 1 shows the set of valuations ("bins") that were employed for our demonstrations, which are described briefly in Section 7.

Once the bins had been assigned values, the translation to scheduling parameters was straightforward. The underlying operating system—The Open Group Research Institute's MK7—provides a value-based scheduler that accepts time constraints for specific computations. Each time constraint describes the application-specific value (in this case, the bin value of the cluster) of completing the designated computation as a function of time. For association computations, all of the time constraints had a similar shape (see Figure 2), which specified that "sooner was better than later."

5 Additional Design Benefits

Although scheduling and overload behavior were important aspects of the tracker design, there were other key issues as well.

Scalability — The relatively high computational cost of association processing justifies the investigation of distributed solutions, which seem particularly feasible since each association operates on a limited amount of data (a cluster's sensor reports and track file records) that can be easily assembled during gating and clustering. Consequently, not only is the ATD tracker multithreaded, but the association algorithms have been extracted and encapsulated in a separate object class. Association object instances (called *associators*) can—and have—been run on other nodes in a distributed tracking system. Moreover, there is provision in the tracker to select an associator on a cluster-by-cluster basis. This structure lays the groundwork for a scalable distributed tracker, where additional associators can be placed on newly added nodes for immediate use by the tracker.

Fault Tolerance — The distributed structure outlined above supports a straightforward form of fault tolerance. The tracker can clearly survive the failure of individual associators—as long as at least one associator survives. When associators fail, reducing system capacity, the basic adaptive, QoS-driven nature of the system results in the selection of a reasonable subset of work to perform.

6 OS Support for the ATD Prototype Implementation

The adaptive ATD tracker builds on several features of the underlying operating system, The Open Group Research Institute's MK7 [6][7], which provides a number of standards-based facilities as well as a set of unique capabilities designed to support distributed, real-time applications.

Beyond traditional real-time support (e.g., predictable execution times, preemption, priority scheduling, instrumentation, and low interrupt latency), MK7 provides a number of advanced features. The MK7 microkernel, which has been explicitly developed to support real-time applications, contains a scheduling framework that simultaneously supports priority-based and time-based scheduling policies. A value-based processor-scheduling policy called Best-Effort scheduling [4][7], which accepts application (and system) time constraints and schedules computations according to a heuristic that attempts to maximize total accrued value, was particularly useful for the ATD. Notably, this value-based scheduling policy and the threads it schedules co-exist with and interact with other parts of the OS and application, including synchronizers, preemptions, and "priority" modifications (e.g., priority inheritances and priority depressions).

MK7 threads are ***distributed*** (or migrating) threads[2][3], which move among the processes (i.e., MK ***tasks***) of a distributed system by executing RPCs while carrying an environment that includes information like the thread's scheduling parameters, identity, and security credentials.

MK7 provides several standard interfaces—including a highly standards-compliant UNIX interface and the X Window System—that permit application programmers to reuse existing code. Both distributed threads and value-based scheduling are managed by an extended POSIX threads library. The extensions, while providing access to powerful facilities, are few in number and are generalizations of existing POSIX thread functions (e.g., through extensions of the POSIX thread cancellation interfaces)—thus simplifying the programmer's task considerably.

7 Prototype Results

The prototype performed extremely well during demonstrations to its target audience—AWACS operators and tracking system designers—most notably at the Boeing AWACS Prototype Demonstration Facility. Under "normal" (non-overload) conditions, the tracker handled all tracks as expected and delivered high QoS for all tracks. Overload conditions were simulated by artificially tightening the deadlines for the completion of association processing.

Figure 3 shows results of the tracker for a typical overload scenario. Batches of between ten and 14 sensor reports arrived at the tracker, with one-third of the tracks belonging to the more-important track class. The Association Capacity axis indicates the number of associations the tracker could usually perform in the specified processing time limit, and the Track Quality axis indicates the average TQ delivered for each track-importance class.

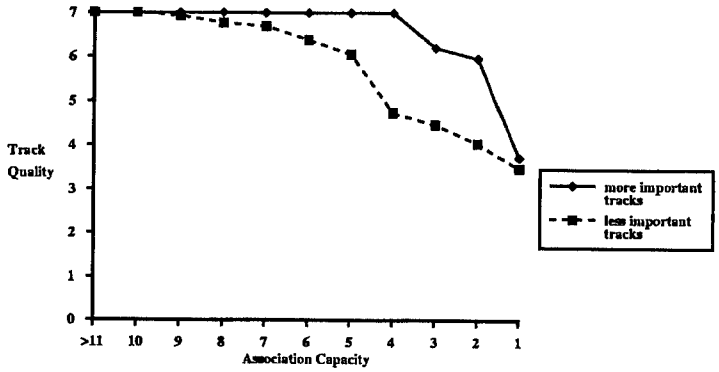

Fig. 3. Average Track Quality as a Function of Association Capacity

Figure 3 indicates that when the tracker can only process about 33% of the input, the prototype delivered essentially perfect TQ for the more important tracks, while delivering a reasonable (about 4.5) TQ for the less important tracks. (In some respects, this overload level can be likened to a system where the probability of detecting an airborne object is about 33%.) When the tracker was further constrained so that it could only process about 10% of the input, the prototype finally dropped some tracks—from the less important track class. No important tracks have been dropped during our demonstrations.

In addition, the demonstrations have shown that the tracker also adapts when new resources are added. In that case, which we demonstrate by loosening the time constraints on association processing, the prototype automatically delivers approximately the maximum achievable QoS.

Note that these results are not easily obtainable with static-priority tracking systems. In priority-based trackers, track priorities might reasonably be set according to track importance, where high importance implies high priority. In the prototype tracker, scheduling decisions are based on both importance and timeliness, and even relatively unimportant tracks can have very high application values in a surveillance mission—a situation that would not arise with straightforward priority-based scheduling.

8 Summary

The AWACS ATD project produced an adaptive, distributed tracker that was directly driven by Quality-of-Service metrics. Based on a novel design and incorporating knowledge from experts in the field, this tracker gracefully handles overloads, addressing a problem with currently deployed trackers and with trackers under development. The tracker was demonstrated to AWACS operators and tracker designers at Boeing in September 1998 and received supportive feedback, particularly regarding its behavior under overload and the operator interface.

This project and its demonstration provide another worked example in the area of value-based scheduling and further encourage our confidence in this technology. In addition, the derivation of the QoS metrics for tracks provided valuable insight into the nature and use of application-specific QoS metrics in a new application domain. Finally, the project produced a prototype tracker that can be used for further experimentation and can host future extensions, as well as be examined by tracker designers and implementers.

There are a number of open questions that should be addressed in the future. In this project, the value represented by a cluster of tracks and sensor reports was calculated by adding the values corresponding to each individual track in the cluster. While this is simple, has intuitive appeal, and produced good results in our tests, it might not be the best way to determine the value represented by a cluster. Moreover, there were several capabilities that were included in our design that have not yet been implemented—most notably the dynamic selection of an association algorithm for a cluster based on factors such as the amount of clutter, the number of maneuvering tracks, the computational cost of the algorithm, and the currently available processing resources. Also, AWACS program personnel have speculated that the overload adaptations in the ATD tracker could help manage the dramatically increased processing load associated with (next generation) multiple hypothesis trackers [5]. This seems credible and could broaden our expertise in the use of value-based scheduling in a new dimension—that is, the specification, in application-specific terms, of the benefit of evaluating each of the potentially large number of hypotheses associated with each track in the system. Finally, the tracker was designed to be able to perform different missions and to interact with other AWACS applications. Neither of these capabilities has been tested to date: The tracker has only performed a surveillance mission, and no other applications have been added. Exploring both of these areas should be fruitful, providing, among other things, an opportunity to explore hierarchical, value-based scheduling architectures.

References

1. Blackman, S.: Multiple-Target Tracking with Radar Applications. Artrech House, ISBN 0-89006-179-3 (1986)
2. Clark, R.K., Jensen, E.D., Reynolds, F.D.: An Architectural Overview of the Alpha Real-Time Distributed Kernel. Proc. of the USENIX Workshop on Micro-kernels and Other Kernel Architectures (1992) 127-146
3. Ford, B., Lepreau, J.: Evolving Mach 3.0 to a Migrating Thread Model. Proc. of the USENIX Winter 1994 Technical Conference (1994)
4. Locke, C.D.: Best-Effort Decision Making for Real-Time Scheduling. Ph.D. Thesis, Dept. of Electrical and Computer Engineering, Carnegie-Mellon University (1986)
5. Reid, D.B.: An Algorithm for Tracking Multiple Targets. IEEE Transactions on Automatic Control, Vol. AC-24 (1979) 843-854
6. Wells, D.: A Trusted, Scalable, Real-Time Operating System Environment. Dual-Use Technologies and Applications Conference Proceedings (1994) II:262-270
7. —: MK7.3 Release Notes. The Open Group Research Institute, Cambridge, MA (1997)

Appendix C

C2 Research & Technology Symposium

This appendix contains a paper presented at the 4th International Symposium on Command and Control Research & Technology. This paper contains our most complete description to date of multi-dimensional quality of service. In addition, it discusses the MK operating system, the adaptive tracking application, and the user system interface.

Quality of Service for AWACS Tracking

Thomas Lawrence and Patrick Hurley
Air Force Research Laboratory/IFGA
Rome, NY 13441
(315) 330-3624
hurleyp@rl.af.mil
Thomas Wheeler, Arkady Kanevsky, John Maurer and Paul Wallace
The MITRE Corporation
Bedford, MA
(781) 271-2589
twheeler@mitre.org
Douglas Wells and Raymond Clark
The Open Group Research Institute
Cambridge, MA
(617) 621-7366
d.wells@opengroup.org

Abstract

This paper describes a United States Air Force Advanced Technology Demonstration (ATD) in which the concepts of adaptive Quality of Service (QoS) based resource and anomaly management are applied to a real world application. The paper describes the QoS model and design philosophy. It also describes several mechanisms that use application specific QoS metrics and user specified policies to control system resources, enabling the application to adapt to a changing execution environment. These mechanisms include a real-time microkernel, a best-effort scheduler, distributed threads, tracking algorithm selection, and sensor scan data selection, which are controlled by the QoS manager to produce the required track QoS. By defining the application behavior in terms of QoS metrics, it demonstrates how QoS can be incorporated into a real world application, the Multi Sensor Integration (MSI) tracking component of the Airborne Warning and Control System (AWACS).

1.0 Introduction

Real world systems are subject to constant change such as overload, component failure, evolving operational requirements, and/or a dynamic operational environment. A System should adapt to these changes by reconfiguring its resources to provide a different though acceptable level of service to its users. Without adaptation many important activities receive fewer resources than needed while less important activities waste resources by receiving more resources than necessary.

Most existing systems either do not adapt or have ad hoc hardwired mechanisms to accommodate only a small predefined set of changes. There are no standard methodologies or

common tools to assist application developers in managing adaptation.

Our system design philosophy uses an extended and refined Quality of Service (QoS) Model, which provides a quantitative basis for efficient and effective resource management. End users define policies based on application specific QoS metrics to control system operation in order to apply resources in the best suitable manner [Lawrence, 1997] [LawrenceT, 1997].

1.1 Project Objective

The objective of this effort is to investigate the use of our QoS design model in a real world, real-time application (Airborne Warning and Control System (AWACS) Tracking). The purpose is to demonstrate a system that continues to perform its function and avoids catastrophic failure even in the presence of anomalies. Processing resources are applied to the most important activities (tracks), as dictated by the QoS metrics for each track, when processing resources become scarce. For the tracking application the most critical tracks receive the highest level of service while less critical track receive a lower level of service. In AWACS parlance the system *allocates resources to the right tracks for the current mission at the right time.*

With this QoS based approach, resources can be more effectively and efficiently utilized and the application can gracefully degrade based on user provided policies.

1.2 Project Overview

This QoS resource management concept is instantiated in the AWACS MSI application through use of a new QoS model, which asserts that the QoS for a track is a function of the three attributes of timeliness (T), precision (P) and accuracy (A). Timeliness is a measure of the time from initiation to completion of an activity. Precision is a measure of the quantity of data. Accuracy is a measure of the error within the data (see section 2.2). Consequently the QoS of a track can be measured by its TPA metrics and controlled by manipulating the statistical metrics for each attribute. Based on these metrics, the system applies the user defined policies to select the tracking algorithm, filter data based on the sensor, prioritize the tracks and employ value-based scheduling to apportion more resources to processing the more important tracks.

Timeliness and Precision (of a track) are manipulated by the QoS Manager at the interface to the real-time scheduler. A Best-Effort real-time scheduler [Locke, 1986] preferentially allocates system resources based on the resulting value to the mission as specified by the user through the QoS Manager. Important tracks receive priority in acquiring additional resources in order to maintain the best Timeliness and Precision possible--given the resources available and user provided constraints. Less important tracks achieve a lower QoS when resources become scarce.

The statistical metric for the precision of a track is manipulated by selecting out some of the sensor scan data. For instance if the precision of a track is too high, thus wasting processing resources,

Adaptive Anomaly Management Concept

| Liveness Property ----------→

Anomaly Mgmt Strategies	Optimistic	Pessimistic	Ultra-pessimistic
Mechanisms	Backward or Forward Recovery (Detection)	Masking Avoidance	Reduce Timeliness, Precision and/or Accuracy

| Increasingly Pessimistic Environment ----------→

Figure 1

some of the radar scan data is dropped. This selection is done by the QoS manager through the interface to the real time scheduler. Sensor data selection is based on relevance to the current mission and characteristics of the sensor. The system monitors the sensor data and applies operator-supplied policies to filter this data. In this way, sensor load can be throttled to control processor load and to assure optimal use of processing resources.

The accuracy of a track is manipulated by the QoS Manager through selection of a particular tracking algorithm. The MSI Tracking application has multiple algorithms available, each with different accuracy, and resource utilization. Switching among these algorithms permits accuracy manipulation.

Through use of these mechanisms, the QoS Manager can trade off Timeliness, Precision and Accuracy of a particular track or group of tracks.

2.0 Adaptive QoS Based Resource and Anomaly Management

2.1 Adaptive Anomaly Management

Static approaches to anomaly management are generally not adequate because the application and its environment are dynamic. A static approach provides a point solution of limited utility in a dynamic world. Consequently adaptive anomaly management under the banner of adaptive end-to-end Quality of Service (QoS), has become a new domain of research [Sabata, 1997].

Two things can change at the application level: 1) the mission or mode of the application, and 2) the environment in which the application exists [Hiltunen, 1996]. If the mission changes the Quality of Service specifications at the user interfaces will change. If the environment changes the threats to which the system is subjected to will change. That is the types of anomalies and/or their rate of occurrence can change. Either or both of these changes will require the adaptive Quality of Service based information system to change its anomaly management mechanisms / strategies.

This adaptivity gives the information systems flexibility and a wider range of utility, thus, permitting it to adapt to a dynamic world (See Figure 1). With a static approach either the optimal output is produced or nothing is produced. Quality of Service attributes and their

quantitative metrics become the objective function for controlling the adaptive information system. Different anomaly management mechanisms handle different types of anomalies, provide different QoS, and require different kinds and different quantities of resources. Consequently choosing a particular anomaly management mechanism has a direct effect on the resulting QoS provided by the information system.

2.2 The Quality of Service (QoS) Model

The Quality of Service model provides a recursive model applicable at any level of abstraction and encompasses concepts of real-time, dependability, and security. The concept of system Quality of Service (QoS) is based on the attributes of timeliness, precision, and accuracy, which can be used for system specification, instrumentation, and evaluation.

Our system design uses application-specific QoS metrics and user-defined policies to manage system resources to process the most appropriate data available by the most appropriate algorithm(s) at the right time. We have identified three principles that hold true for processing in a resource-constrained environment.

The first principle is that QoS is a function of timeliness, precision and accuracy (TPA).

The second principle is that TPA attributes are interdependent within a single activity. Improving the behavior for one attribute of a particular activity results in a degradation of one or both of the other two attributes.

The third principle extends the second principle to a system level. Improving the behavior of one attribute for a given activity results in a degradation of one or more attribute(s) in the system (considering all activities).

Timeliness measures when an event occurs or the delay time from initiation to completion of an activity. This information system activity could be processing, communication or data storage related.

Precision is the measure of the quantity of data produced as a result of an activity. For example, in the simplest case, it is the number of bits produced. Precision is only concerned with the quantity or volume of data not whether that data is accurate. One common statistical measure of precision is throughput (the number of bits produced by N successive events associated with an activity).

Accuracy is the measure of the amount of error or noise in the output data. Accuracy is concerned only with errors in the bit structure of the output.

However, the correctness of the output is a function of all three attributes (See figure 2). If an output is timely and precise but inaccurate, it is not correct. If an output is precise and accurate but not timely it is incorrect. If an output is timely and accurate but imprecise it is also incorrect. Only if all three attributes are sufficient is the output correct. All three attributes must be measured in order to know if the QoS is correct.

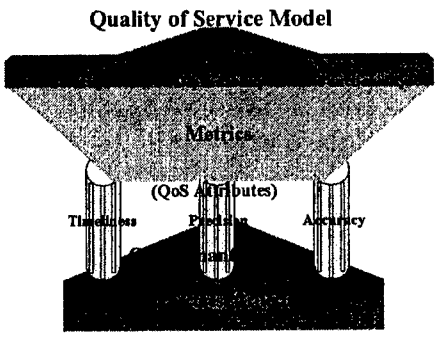

Figure 2

Each attribute has an instantaneous value and a statistical value. Each value belongs to the domain of application acceptability (benefit) to a different degree, thus fuzziness. The statistical specification of each attribute is extremely important since, in an imperfect world, we cannot specify perfection. Therefore, how much badness in an attribute the application can accept (i.e., can experience and still maintain useful operation) must be specified. The statistical specification for each attribute consists of two metrics a collective metric such as percent, average or mean, and a distribution of values for each attribute over some time or number of events. For instance the timeliness specification may require that 90% of all events must occur within 5 minutes of 2pm. However, no three of these daily events, in succession shall be outside that range. The statistical metrics are important since they are the basis for rational anomaly management.

Dependability in QoS terms is defined as the minimum acceptable statistical metric for each attribute. Therefore availability is not one number for an entire system but three metrics for each user interface to the system. If timeliness, precision, and accuracy are above the minimum statistical behavior that interface is dependable. In this project a track is dependable if it satisfies its statistical metric for each attribute [Cristian, 1991].

Selection of a particular anomaly management mechanism will influence the timeliness, the precision and/or accuracy provided at the end user interface. For example, a schedule can be formed to optimize timeliness of transactions in a database management system by interleaving them [Bernstein, 1987]. This interleaving can result in inaccurate results due to conflicting reads and writes. The schedule can be modified to provide degraded timeliness but acceptable accuracy. Reducing the number or size of the transactions (precision manipulation) could restore timeliness. This example illustrates the effect of anomaly management on QoS and the tradeoff among the three QoS attributes.

2.3 Identifying QoS Metrics

Using QoS-based resource management for a specific application requires the identification and selection of application-specific metrics. Extracting these metrics is a relatively new discipline. In some cases, application fields with a long history will already have some basic QoS metrics defined. For instance, in the AWACS Tracker, "Track Quality" is a well known metric for tracking systems that captures the Precision attribute at the level of abstraction of an individual track.

While it is necessary to identify a set of QoS metrics that can evaluate the

operation of the self-adaptive system, it is not sufficient. It must also be reasonable to compute the QoS metrics--or an acceptable approximation--dynamically on-line. Again, using a tracking example: an external observer with an "eye of God" view could evaluate the error in any given assessment of an individual track's position. Such information is not available to the Tracker, though. Nonetheless, the Tracker can employ estimation theory to account for sensor and association errors and yield an estimate of the uncertainty in the position of any given track. This accuracy statistic is used as a QoS metric to control the Tracker.

Now the statistical metrics for timeliness, precision and accuracy for each track can be monitored and controlled to produce a desired track QoS. For example, if the system is in a period of overload, processing resources could be applied to trade off track accuracy for track precision by selecting the precision attribute (i.e., by updating "stale" tracks) over the accuracy attribute (i.e., choosing a less accurate tracking algorithm).

Finally, QoS metrics for different levels of abstraction are related, but the relationships involve application-specific knowledge. We hypothesize that the relationships and the derivation of information needed to control lower system levels and to inform higher system levels can be captured and localized in a relatively small portion of an application.

2.4 Anomalies

An anomaly is an incorrect state, which can result in degraded or inappropriate Quality of Service at some interface(s) in the information system. Anomalies for both hardware and software can be placed in three general categories or classes. An initial anomaly (Class 1) appears during operation where none had existed before--i.e., a failure of a previously working component. A design anomaly (Class 2) occurs due to an incorrect specification or the incorrect implementation of a correct specification. A shared resource anomaly (Class 3) occurs when independent activities in the information system compete for the same resource(s) [VanTilborg, 1991]. Strategies for handling these anomaly categories in run-time are either avoidance strategies or fault-tolerance strategies [Anderson, 1981].

3.0 Mechanisms Used to Control QoS

Resolution of anomalies in the tracking system has been included in the basic application design process. The application builds upon mechanisms in the underlying operating environment to address anomalies and maintain the highest level of service possible under the circumstances.

Some initial (class 1) anomalies are managed by the application, others are delegated to the underlying OS. Sensor failures, for example, are managed by the application. Some targets are detected by multiple sensors. The tracking application integrates the results of multiple sensor types and will continue to provide (limited) information about the target.

The task of detecting and identifying component failures is delegated to the operating environment—under rules specified by the application. The tracking application has been restructured and distributed into multiple object components running on multiple processing nodes. Each of the objects communicates with the others via Remote Procedure Calls (RPC) using the distributed threads provided by the MK 7 operating system (see below). When a processing node fails or when an individual object fails, the operating system detects the failure in a timely manner and completes the RPC with an error indication. The application can repeat the object invocation, either immediately or on the next sensor input, thus an initial anomaly (class 1) can cause a shared resource anomaly (class 3).

Design (class 2) anomalies are addressed via the use of multiple tracking algorithms. Tracking algorithms are selected dynamically each time that an individual sensor report is processed, typically based on localized conditions (such as clutter, maneuvering aircraft). Alternatively, if a tracking algorithm has recently failed multiple times for the particular track, an alternate algorithm can be selected.

As a general principle, when class1 and/or class 2 anomalies occur, the tracking application tends to cause shared resource anomalies (class 3). Such contention for shared resources can be prioritized and scheduled using the value-based scheduling method described later in this paper. It does not matter whether the resulting overload is caused by component failure or by the presence of a higher workload then originally anticipated. The system continues to process the most critical, most valuable subset of tasks even in situations with severely constrained resources.

3.1 The MK 7 Real-Time Operating System

MK 7 [Wells, 1994] is a modular operating system comprising several components, including the MK microkernel, an operating-system personality server, and various stand-alone servers. The standard OS personality is derived from OSF/1, a commercial quality version of UNIX that complies with SVID, POSIX, XPG3, XPG4, and numerous other standards. MK 7 has been developed by the Research Institute (RI) of The Open Group (TOG) to provide performance competitive with commercial systems while supporting reliable, distributed real-time applications.

The base set of real-time facilities in the MK microkernel provides real-time capabilities within a single node by addressing low latency applications as might be found in simulation and factory automation environments. Utilizing a fully preemptible microkernel, interrupt latencies are bounded and typically less than 200 usec (on a 100-MHz Pentium PC). A scheduling framework included in the microkernel supports both priority-based algorithms, such as POSIX FIFO, and deadline-based algorithms, such as Earliest Deadline First. Concurrent threads are implemented in the microkernel and a POSIX threads interface simplifies the development of multi-threaded

applications. Application developers can use the ETAP instrumentation package to gain visibility into the detailed operation of the operating system.

Other facilities enable distributed real-time applications. CORDS (Communication Objects for Real-Time Dependable Systems) [Travostino, 1994] is a networking framework and a set of objects that operate within that framework to provide a set of services that enable high levels of reliability and availability within a distributed real-time application. The Path capability allows explicit management of system resources so that non-real-time applications do not interfere with real-time applications on the same machines. GIPC (Group IPC) [Travostino, 1996] [TravostinoL, 1998] provides a real-time group communication model in an environment that detects and recovers from node failures in less than 200 msec.

3.2 QoS Feedback and Value-Based Scheduling

Many of the real-time capabilities in MK 7 build upon concepts developed in the Alpha [Clark, 1992] operating system and incorporated into MK 7 as part of the Alpha/Mach Integration project [Goldstein, 1993]. One of Alpha's primary design philosophies was the concept of value-based scheduling, whereby an overall system would manage both computer and non-computer resources to maximize benefit for application(s).

This model of computation embraces the concept that systems are designed and built to provide specified services, and that the execution of significant portions of system code can be characterized as representing value (or benefit) to the "user" of the system. (Note that, particularly in the case of embedded systems, not all users are people.) It is then possible, and in fact, desirable to specify metrics to characterize those values.

Using this value, which might vary depending upon time and other conditions, for each computation to be performed, along with a characterization of the required resources needed to produce the result, it is possible to investigate optimal methods for allocating resources to applications. The Alpha system managed resources according to Best-Effort scheduling [Jensen, 1985], which determined a heuristic, near-optimal schedule for a resource based on available information about the computation.

3.3 Best-Effort Scheduling

Alpha introduced a new paradigm for real-time computing in which it managed system resources according to time-value functions (TVF) [Jensen, 1985], which represent application-specified profiles/desires that reflect physical time constraints. These functions describe orthogonal facets of urgency (timeliness) and importance (benefit), thus eliminating the notorious complexities of mapping time into priorities. A TVF expresses a range of acceptable times for delivering each output and relates the value (benefit) of producing that output to the time at which it is produced; deadlines are a simple special case, a binary-value, downward step. Time-value functions are derived by the user from the physical nature of the

application environment, and may evolve dynamically over the course of a mission (see figure 3).

In the case of this project, for example, the TVF for a track report from the Doppler radar is approximately a square wave. The value of processing such a report immediately reaches its maximum value and then drops off slightly, reflecting the increasing staleness of the report. After a time of one rotation period, the value rapidly drops to zero, reflecting the higher value of the new report which will likely have been received. The values of processing each track are determined according to the importance of the associated target. Vulnerable or threatening targets will have higher values, as will maneuvering or low-quality targets.

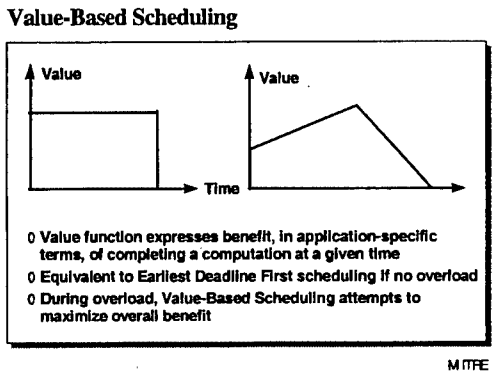

Figure 3

A Best-Effort scheduler uses the TVFs and information about potential resource usage to create a schedule of tasks. If there are no resource conflicts, all tasks will execute based on an Earliest Deadline First (EDF) ordering. Otherwise, the scheduler uses heuristic methods to determine a scheduling order that attempts to maximize the value accrued by the overall system. Tasks that cannot meet their time constraints receive exceptions and can execute recovery activities.

The MK 7 Best-Effort CPU scheduler is derived from the basic Locke algorithm [Locke, 1986]. It has been significantly enhanced to allow nested time constraints, to operate in a symmetric multiprocessing environment, and to coexist with priority-driven scheduling algorithms, such as those defined by POSIX. In addition, the MK 7 Best-Effort scheduler has been enhanced so that the application developer sees a familiar, POSIX-like interface with the addition of simple calls to define the beginning and end of time constraints.

One of the benefits of value-based resource management is the additional flexibility that is possible for application behavior. For instance, robust systems are expected to continue to operate effectively even in situations where available resources can change dynamically -- or, equivalently, where significant additional workload can arrive dynamically. Often, systems are built to operate in small number of discrete "operating modes" corresponding to the relative workload for the currently available resources. Such systems, then, have a few regions of operation; and the transition from one region to another is characterized by a significant degradation in service for some, and possibly all, application functions and users.

Value-based resource management techniques offer the possibility of more graceful, more dynamic degradation in service. At any given time, much smaller units of the workload can be shed,

offering users degradation that is more nearly continuous than it is discrete. (Of course, value-based techniques can certainly support more discrete degradation if that is desired for some reason. To do so, the computations to be performed can be grouped together into a small number of collections, where each collection corresponds to a large-grained functional increment associated with an operational mode).

A complementary aspect of system design is relevant when using value-based resource management. Not only can workload be shed in a fine-grained manner, it can also be added in the same way. That is, not only can value-based resource management support the "normal" (or "baseline") computations in a system, it can allocate (otherwise) "excess" resources incrementally to perform additional (optional) computations. Many systems are designed for deployment with a safety margin of excess resources. Value-based resource management could use these resources constructively to enhance the system's base capabilities. Over time, as new applications or sensors are added, consuming some of the "excess" resources, fewer and fewer "extra" computations will be performed. Significantly, all of this change will occur simply as a result of using value-based resource management.

3.4 Distributed Threads

The Alpha system also introduced distributed threads [Jensen, 1990], an object-oriented programming model well suited for writing large, real-time, distributed software. Distributed threads are loci of control that span objects and physical nodes, transparently and reliably, via operation invocation. Distributed threads are similar in concept to local threads communicating via Remote Procedure Calls (RPC), except that a distributed thread maintains its unique system identity even on different nodes. The system maintains and enforces end-to-end time constraints, resource requirements and limits, and security identity throughout the execution of the thread. Thus, an application thread receives an exception immediately upon the expiration of a time constraint regardless of whether the thread is executing in an application object, a local system object, or a remote object that happened to be invoked by a local object.

The distributed thread model naturally captures the processing of tracks in the AWACS system. Identifying clusters--that is, determining groups of nearby sensor reports and tracks--requires rapid access to all tracks and is done on a specific, single node in the current project. Association, which pairs individual sensor reports with specific tracks, is much more computationally expensive and requires access to information about only a small subset of the track information; it is performed on any of a number of arbitrary computation nodes. A distributed thread originates on the clustering node, carries the sensor reports and track information about a single cluster to a computation node, and finally delivers the updated track information to the node maintaining the track file (which may or may not be the clustering node).

The computation required to cluster information is performed by a single

thread. Time constraints control the progress of the cluster processing. An overall time constraint controls the thread from its origination to its update of the track file. Nested time constraints may impose shorter deadlines for partial operations, such as association or database updates. If a time constraint expires or if a node fails, the thread is immediately notified, wherever it is currently executing. In the case of a node failure, the application can optionally continue the computation on an alternate node. In either case, there is no need to waste additional computing resources on behalf of overall results that cannot be successfully completed.

Distributed threads in MK 7 are provided within a POSIX-like thread model. Basic MK 7 POSIX threads--that is, threads that are based on the POSIX standards but do not exercise distributed thread features--are implemented by a run-time library that offers the POSIX API by adding a thin veneer on top of the MK microkernel's native thread model.

That same run-time library has been extended so that a thread executing an MK 7 RPC (and hence migrating to a new POSIX process by means of a distributed thread) can continue to use POSIX services in the server process, just as it had in the client process. This extension is largely transparent when an RPC is invoked because most of the setup needed to provide such a distributed POSIX thread is done by coordinating the state maintained by POSIX run-time library instances and by the MK 7 application programmer when the server process is created.

When creating a server process, the programmer creates a number of "empty" POSIX threads to service incoming RPCs. The number of empty threads is a function of the desired concurrency and resource utilization levels that the designer anticipates. A distributed POSIX thread is governed by the resource limits associated with the underlying distributed thread; but in most respects it behaves as a normal POSIX thread: it has a POSIX thread identifier and can fully utilize POSIX resources, such as the file system and synchronizers.

In a POSIX system without the MK distributed thread extensions, an application developer would manage a remote service by creating a new POSIX thread, immediately performing a RPC receive, and then dispatching based upon the requested service. While the mechanics of this approach to the use of RPC are similar to that provided by MK, there is much less coordination of the resultant computation. For instance, time constraints are not propagated by the RPC and time-constraint exceptions are not propagated.

4 USING QoS FOR AWACS TRACKING

The prototype being developed to demonstrate the above QoS model is based on the AWACS Command and Control System [Kanevsky, 1998]. AWACS can be thought of as being in the business of "selling" track information. Thus the quality of this information is essential. The adaptivity built into our prototype is based on optimizing this information.

Our prototype includes a sensor model

that produces data for two different sensor types: pulse doppler radar and identification friend or foe (IFF). We then use a fairly traditional tracking system that has gating and clustering, association, and smoothing functions. In order to support system adaptivity, we have two different association algorithms that can be dynamically chosen during runtime. The nearest neighbor-mahanalobis algorithm generally provides accurate results and consumes only a moderate amount of CPU resources. The joint probabilistic data association (JPDA) algorithm tends to provide more accurate results in high clutter environments but can require significantly more CPU resources, especially when the input data set is large (See Figure 4).

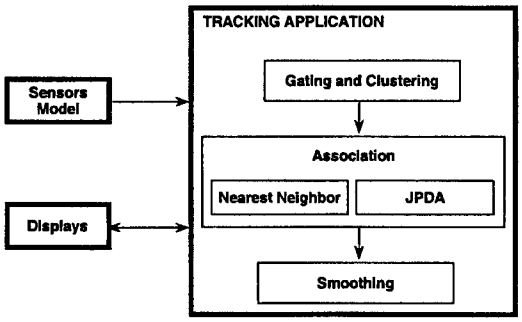

Figure 4

4.1. Define QoS metric for AWACS tracking Application

The approach used to create an adaptive surveillance system was to first define what the QoS timeliness, precision, and accuracy attributes mean in the context of a tracking application. We then identified behaviors that an adaptive surveillance system should exhibit. Time-value scheduling functions were parameterized based on the QoS attributes values. These time-value functions are used by the operating system scheduler during periods of overload in determining which threads should be run.

We found it easy to identify existing tracking attributes that fit the QoS attributes of timeliness, precision, and accuracy. Timeliness is defined as the difference in time between when a track's position and speed estimate has been updated in the central track file and the time the relevant sensor report arrived at the system boundary. This can be thought of as the latency.

Precision is defined as equal to a track's "track quality" (TQ). TQ has long been used in surveillance systems and is a measure of how stale the track information is. TQ is increased if the information has been updated during a specified time interval and it is decremented if it has not. For accuracy, we are using the trace of the covariance matrix of the Kalman Filter coefficients that are used in updating the track state. This is used because the larger the trace, the higher the uncertainty there is in the system's estimate of the track information. It should be noted that accuracy has to be estimated in a surveillance system since the exact position and speed of a track is not known.

4.2 Importance

We reached the conclusion that system QoS is not the total or average QoS of the individual tracks. Rather, some tracks should have a higher QoS than others. The reason is that Command and Control systems do not exist without a context; they are used as part of an

operational mission. We believe system QoS must capture the benefit a TPA set for a particular track will derive to the user, relative to TPA's of other tracks.

To better reflect the system's support of a mission, we introduced the concept of importance. Importance can be either manually or automatically assigned and implies that the object that is designated as important has special significance. In our prototype, importance can be assigned to geographic areas, tracks, and sensors. We chose to use two levels of importance (the default is less important) although an operational system may decide to use more levels in order to increase the fidelity of the adaptive behavior.

While we envision importance to typically be assigned by operators, we have made provisions for the computer system to temporarily automatically designate individual tracks as being important. This would occur when predefined criteria are met. For example, if a track appears to be threatening to the platform or if the track is declaring an emergency.

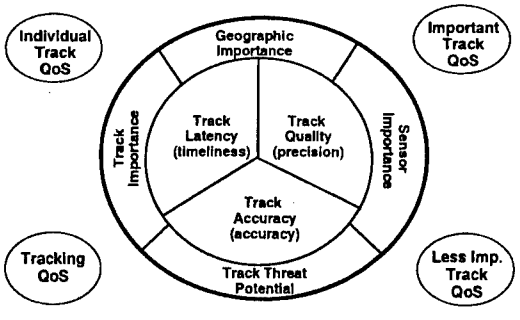

Figure 5. QoS Attributes

Figure 5 identifies the QoS attributes for our prototype. At the core of the large circle are the timeliness, precision, and accuracy attributes for individual tracks. The outer "importance" ring of this circle shows different criteria for declaring a track as being important. Finally, the four small ellipses indicate different levels at which QoS can be aggregated. QoS exists for individual tracks, all the important tracks, all the less important tracks, and all the tracks.

With the addition of importance, we found we had sufficient metrics in which to base our adaptivity decisions. The tracking system is designed so that there are separate operating system threads used for each cluster of sensor reports and tracks. This allows the application designer to rely heavily on the operating system scheduler in order to perform fine-grained adaptations. As mentioned previously, the MK scheduler uses a best effort algorithm where if it believes deadlines can not be met, it recursively discards the work request with the lowest value density until a schedulable set of requests can be reached.

Thus when constructing the time-value functions, it is important that they cause the system to behave in a manner that best meets the user's goals. For control of the prototype, we have the goal of allocating resources to the right tracks for this mission at this time. Behaviors that the application should exhibit include: a preference for not "dropping" tracks; a preference for processing user identified important tracks; a preference for updating tracks with poor state estimates; a preference for processing maneuvering tracks; and a preference for processing high threat tracks.

Clearly, there can be scenarios where not all of these behaviors can be

achieved simultaneously. For example, there may not be sufficient resources to process a track that is in danger of being dropped and a user identified "important" track that has an excellent state estimate. The time-value functions are the primary tool that is used to resolve these conflicts.

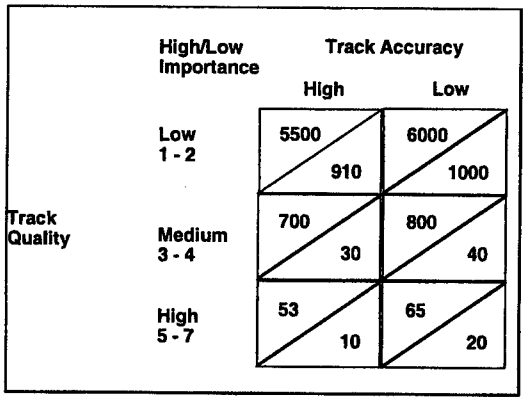

Figure 6 Initial Heights of Time-Value Functions

We decided to use simple time-value function shapes (straight, down-sloping lines). The initial heights (where height indicates the relative value of completing this work unit) are shown in Figure 6. This Figure shows a table of values based on track QoS and importance. The table shows, for example, that a track that is important and has low track quality and accuracy will have a time-value function with an initial value of 6000. A track with low importance and high quality and accuracy will have an initial value of 10. Close inspection of the table will reveal that the system strongly favors processing important tracks over less important tracks. Also, track quality is a dominating factor over accuracy for tracks with low track quality; but as the quality improves, accuracy begins to play a stronger role in the value calculation.

4.3 AWACS User / Operator Interface

The user system interface for our prototype differs slightly from that of a normal surveillance system. While we continue to use a display that shows track symbology and sensor reports overlayed on top of a map, we also have added two windows that capture delivered QoS metrics. Figure 7 shows a window that displays aggregate, delivered QoS performance.

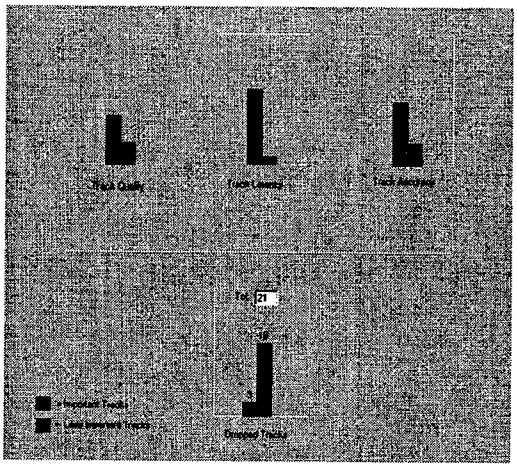

Figure 7 System QoS Display

Average timeliness, precision, and accuracy values are presented for all important tracks and for all less important tracks. We chose not to display the average for all tracks, which would be a combination of the important and less important numbers, because we believe that the total number would not be as useful to the operators for their decision making. In addition to the QoS display, we also augmented a tabular display so that an operator can view the instantaneous QoS attributes (latency, TQ, accuracy) for individual tracks.

4.4 Results

Most of our analysis to-date of the performance of our adaptive system has been through the use of simulations. This is because the integration of the adaptive tracking application and the MK microkernel is ongoing. We have seen promising early results with the real prototype, namely that important tracks are receiving better QoS than less important tracks when the system is under overload conditions, but we are not far enough along in the integration to report concrete results.

The simulations strongly indicate that the adaptive strategy results in significantly better performance (with performance being measured with respect to how well the prototype can satisfy mission goals given constrained resources). For example, given a scenario where there were only enough resources to update the estimates of 25% of the tracks in any one sensor sweep period, the adaptive system was able to keep all of the tracks alive. As a point of comparison, if the system had been controlled using a traditional strategy, a FIFO queue of tracks would have been worked on where the position in the queue would likely be a result of the geographic location of the track. This would have resulted in the first 25% of the tracks receiving excellent service while the remainder got little or none - regardless of how important they are in meeting the mission objectives.

5.0 Conclusion and Future Directions

In summary we demonstrate the effectiveness of QoS based adaptive resource management, QoS based graceful degradation, and facilities to support dependable distributed real-time applications. We have found that for this application adaptivity can be value added with little change to existing application software. Based on our experience the techniques are applicable to any distributed application.

This project has demonstrated that the QoS model provides a quantitative basis for the effective and efficient use of the processing resource. Future activities would need to demonstrate that the QoS model can be used in the same manner for managing processing, communication, and data storage resources in support of a distributed application.

6.0 Project Participants

A United States Air Force Advanced Technology Demonstration (ATD) in which the concepts of adaptive Quality of Service (QoS) based resource and anomaly management are applied to a real world application. The ATD, which began in January 1996 and is expected to end in September 1998, is a joint effort among the Air Force Research Laboratory, The Open Group Research Institute, and The MITRE Corporation. The Air Force Research Laboratory provides expertise in QoS (QoS Based Resource / Anomaly Management), The Open Group in operating system mechanisms (Real-time Microkernel, Best-Effort Scheduler, and Distributed Threads), and MITRE in Command and Control application development (AWACS, MSI Tracking).

7.0 Trademarks

OSF, OSF/1, and Open Software

Foundation are trademarks and UNIX is a registered trademark of The Open Group.

8.0 References

[Anderson, 1981] T. Anderson and P.A. Lee, "Fault Tolerance Principles & Practices", Prentice Hall 1981.

[Bernstein, 1987] P. A. Bernstein, V. Hadzilacos, N. Goodman, "Concurrency Control and Recovery in Database Systems", Addison-Wesley, 1987.

[Clark 1992] R.K. Clark, R.K, Jensen, E.D., Reynolds, F.D., "An Architectural Overview of the Alpha Real-Time Distributed Kernel," Proceedings of the USENIX Workshop on Micro-kernels and Other Kernel Architectures, pp. 127-146, Seattle, WA, April,1992.

[Cristian, 1991] F. Cristian, "Understanding Fault-tolerant Distributed Systems", Communications of the ACM, Vol. 34, No. 2, February 1991.

[Goldstein, 1993] I. Goldstein, and D. Wells, "Alpha/Mach Integration Study," OSF Research Institute Operating Systems Collected Papers, Vol. 2, OSF, Cambridge, MA, October, 1993.

[Hiltunen, 1996] M. A. Hiltunen and R. D. Schlichting, "Adaptive distributed and fault-tolerant systems", Computer Systems Science and Engineering, 1996 CRL Publishing Ltd.

[Jensen, 1985] Jensen, E.D., et. al., "A Time-Driven Scheduling Model for Real- Time Operating Systems," Proceedings of IEEE Real-Time Systems Symposium, 1985, pp. 112-122.

[Jensen, 1990] Jensen, E.D., Northcutt, J.D., "Alpha: An Open Operating System for Mission-Critical Real-Time Distributed Systems—An Overview," Proceedings of the 1989 Workshop on Operating Systems for Mission-Critical Computing, ACM Press, 1990.

[Kanevsky, 1998] Kanevsky, A., et. al., "Design of a Quality of Service (QoS) Driven Adaptive C2 Prototype", MITRE Technical Report 98B002, January 1998.

[Lawrence, 1997] Lawrence, T.F., "The Quality of Service Model and High Assurance", IEEE, High Assurance Systems Workshop, Bethesda, MD, August 1997.

[LawrenceT, 1997] Lawrence, T.F., "QoS Perspective," Next Generation Information Environment (NGIE) Workshop, Air Force Research Laboratory, Hamilton, NY, November 1997.

[Locke, 1986] Locke, C.D., "Best-Effort Decision Making for Real-Time Scheduling," Ph.D. Thesis, Department of Electrical and Computer Engineering, Carnegie-Mellon University, 1986.

[Sabata, 1997] B. Sabata, S. Chatterjee, M. Davis, J. Sydir, T. Lawrence, "Taxonomy of QOS Specification", Proc of the IEEE Computer Society 3rd International Workshop on Object-Oriented Real-Time, Dependable Systems (WORDS '97), Newport Beach, CA, Feb 1997.

[Travosinto, 1994] Travostino, F., Reynolds, F.D., "An O-O

Communication Subsystem for Real-time Distributed Mach," Proceedings of the 1994 IEEE Workshop on Object-Oriented Real-Time Dependable Systems, Irvine, CA, October, 1994.

[Travostino, 1996] Travostino, F., Menze, E., Reynolds, F., "Paths: Programming with System Resources in Support of Real-time Distributed Applications", Proceedings of the 2nd IEEE Workshop on Object-Oriented Real-Time Dependable Systems, pp. 36-45, February 1996.

[Travostino, 1998] F. Travostino, L. Feeney, P. Bernadat, F. Reynolds, "Building Middleware for Real-Time Dependable Distributed Services," First IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC), Kyoto, Japan, 1998.

[TravostinoL, 1998] F.Travostino, L. Feeney, P. Bernadat, and F. Reynolds, "Building Middleware for Real-Time Dependable Distributed Services," Proceedings of the First IEEE International Symposium on Object-Oriented Real-time Distributed Computing (ISORC), Kyoto, Japan, April 20-22, 1998.

[VanTilborg, 1991] A. M. VanTilborg, G. M. Koob editors, "Foundations of Real-Time Computing: Scheduling and Resource Management, Kluwer Academic Publishers, 1991.

[Wells, 1994] D. Wells, "A Trusted, Scalable, Real-Time Operating System Environment," 1994 Dual-Use Technologies and Applications Conference Proceedings, pp. II-262/270, Utica, NY, May, 1994.

Appendix D

Real-Time Systems Symposium

This appendix contains a paper presented at the 1998 IEEE Real-Time Systems Symposium's work-in-progress session. The paper contains a concise overview of the ATD project including a description of the underlying technologies, the adaptive application, and results.

Application of QoS-Driven Adaptive Computing

T. Wheeler, A. Kanevsky, J. Maurer and P. Wallace
The MITRE Corporation
Bedford, MA
twheeler@mitre.org

D. Wells, R. Clark and Y. Zhang
The Open Group Research Institute
Cambridge, MA
d.wells@opengroup.org

T. Lawrence and P. Hurley
Air Force Research Laboratory/IFGA
Rome, NY 13441
hurleyp@rl.af.mil

Abstract

This paper describes a prototype being developed that provides a worked example of a distributed, real-time system that utilizes application level Quality of Service (QoS) metrics to adapt to various, changing environmental conditions. The underlying system infrastructure is able to interpret the application level QoS expressions in order to maximize the use of available resources. Underlying technologies include a multi-dimensional QoS model, distributed (or migrating) threads, and value-based resource management.

1 Introduction

Real-time military command and control (C2) systems are subject to constant changes such as a dynamic external environment, temporary overload of internal systems, component failure, and evolving operational requirements. It is greatly advantageous for a system to adapt to these changes by reconfiguring its resources and behavior in an intelligent manner to provide support to the varying user/operational needs. Systems that do not employ such adaptivity often use resources in a manner where more important activities receive the same or fewer resources than far less important activities. For example, the current AWACS airborne surveillance system updates its estimates of positions and velocities based on a first-in first-out arrival of sensor reports— i.e., tracks that are associated with the first arriving sensor data are updated before tracks that are associated with slightly later arriving sensor reports, regardless of how important the different tracks are.

Most existing C2 systems either do not adapt or have ad hoc, hard-wired mechanisms to accommodate only a small, predefined set of mode changes. These point solutions have limited ability to scale, are not based on any methodology, and lack tool support for ease of analysis and implementation.

We are developing and exercising technology that directly addresses how to build and operate computer systems that can adapt to changing circumstances. Key components of our approach include the use of a multi-dimensional QoS model [LAW 98], distributed threads [CLA 92], and value-based resource management [JEN 85]. The operating system being used is the Open Group Research Institute's MK 7 [WEL 94].

To exercise this technology, we are developing a prototype of an adaptive AWACS surveillance tracking application. This application adapts its behavior, by dynamically selecting which data to process and which algorithm to use in processing the data, based on parameters such as the desired and delivered track QoS, the available resources, and the current operational mission.

Section 2 of this paper briefly discusses the underlying technology being used. Section 3 describes the adaptive application while Section 4 discusses results we have achieved to-date. Section 5 discusses conclusions and future work.

2 Underlying Technologies

Throughout our work, we use **a multi-dimensional QoS model** [LAW 98]. This is applicable at any level of abstraction and acts as a unifying approach for system management. In this model, a data product can be defined along three dimensions: timeliness, precision, and accuracy. Timeliness measures when an event occurs or the delay from initiation to completion

of an activity. Precision is a measure of the quantity of data produced as a result of an activity. Accuracy is a measure of the amount of error or noise in the output data.

These attributes are interdependent within a single activity. Improving one generally results in degradation in one or both of the other attributes. For example, to improve the estimate of a maneuvering track's position and velocity, one could use a Kalman filter based tracker instead of an alpha-beta based one. This results in better accuracy at the expense of increased processing time, and thus a decrease in the timeliness of when a result is available.

Our approach to performing adaptive resource management relies heavily on a technology called *value-based resource management* [JEN 85]. Time-value functions (TVF) represent application-specified profiles/desires that reflect physical time constraints. These functions describe orthogonal facets of urgency (timeliness) and importance (benefit), thus eliminating the notorious complexities of mapping timeliness requirements onto priorities. A TVF expresses the benefit of producing a result to the time at which it is produced (See Figure 1); deadlines are a simple special case, a unit binary-valued, downward step (See Figure 2). Time-value functions are derived by the user from the physical nature of the application environment, and may evolve dynamically over the course of a mission.

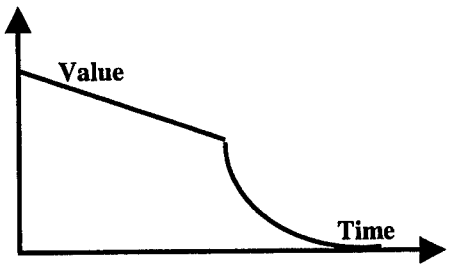

Figure 1. Time-Value Function

A Best-Effort scheduler [LOC 86] uses the TVFs and information about resource usage expectations to create a schedule of tasks. If all tasks can meet their deadlines, they are scheduled Earliest Deadline First (EDF), which is known to be optimal for meeting such deadlines on a uniprocessor. Otherwise, the scheduler uses heuristic methods to determine a scheduling order that attempts to maximize the value accrued by the overall system. Tasks that cannot meet their time constraints receive exceptions and can execute recovery activities.

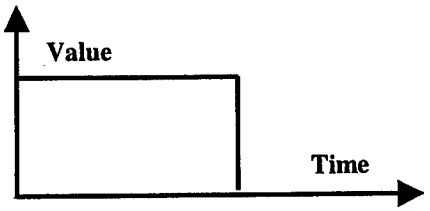

Figure 2. Deadline, a Special Case of a Time-Value Function

Value-based resource management raises a complimentary aspect of system design that is not usually addressed. Not only can workload be shed in a fine-grained manner, it can also be added in the same way. That is, not only can value-based resource management support the normal (or baseline) computations in a system, it can allocate (otherwise) "excess" resources incrementally to perform additional (optional) computations.

Distributed threads are a key technology in our distributed processing architecture. The Alpha operating system introduced an object-oriented programming model well suited for writing large, real-time, distributed software, including the distributed thread abstraction [JEN 90]. A distributed thread, see Figure 3, is a locus of control that spans objects and physical nodes, transparently and reliably, via method invocation.

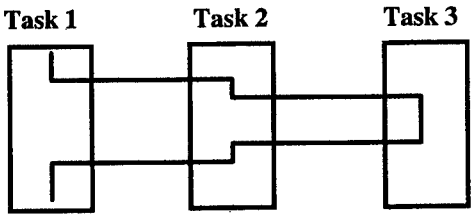

Figure 3. Distributed Thread

Distributed threads are similar in concept to local threads communicating via Remote Procedure Calls (RPC), except that a distributed thread maintains its unique system identity and its attributes (particularly its timeliness attributes) even while it executes on different nodes. The system maintains and enforces end-to-end time constraints, resource requirements and limits, and security identity throughout the execution of the thread. Thus, an application thread receives an

exception upon the expiration of a time constraint regardless of whether the thread is executing in an application object, a local system object, or a remote object that happened to be invoked by a local object.

3 Adaptive Application

The prototype being developed is based on the AWACS Command and Control System [KAN 98]. AWACS can be thought of as being in the business of "selling" track information. Thus the quality of this information is essential. The adaptivity built into our prototype is based on optimizing this information.

Our prototype includes a sensor model that produces data for two different sensor types: pulse doppler radar, and identification friend or foe (IFF). We then use a fairly traditional tracking system that has gating and clustering, association, and smoothing functions. To support system adaptivity, we will have two different association algorithms that can be dynamically chosen during run-time. The nearest neighbor-mahanalobis algorithm generally provides accurate results and consumes only a moderate amount of CPU resources. The joint probabilistic data association (JPDA) algorithm tends to provide more accurate results in high clutter environments but can require significantly more CPU resources, especially when the input data set is large.

We found it easy to identify existing tracking attributes that fit the QoS attributes of timeliness, precision, and accuracy. Timeliness is defined as the difference in time between when a track's position and speed estimate has been updated in the central track file and the time the relevant sensor report arrived at the system boundary. This can be thought of as the latency.

Precision is defined as equal to a track's "track quality" (TQ). TQ has long been used in surveillance systems and is a measure of how stale the track information is. TQ is increased if the information has been updated during a specified time interval, and it is decreased if it has not. For accuracy, we are using the trace of the covariance matrix of the Kalman Filter coefficients that are used in updating the track state. This is used because the larger the trace, the higher the uncertainty there is in the system's estimate of the track information. It should be noted that accuracy has to be estimated in a surveillance system since the exact position and speed of a track is not known.

To better reflect the system's support of a mission, we introduced the concept of importance. Importance can be either manually or automatically assigned and implies that the object that is designated as important has special significance. In our prototype, importance can be assigned to geographic areas, tracks, and sensors. We chose to use two levels of importance (the default is less important) although an operational system may decide to use more levels in order to increase the fidelity of the adaptive behavior.

4 Current Results

Our evaluation to date is based on a system configuration where the surveillance application resides on a single computer node. The sensor probability of detection is set at 1.0 so that the results will best reflect the adaptivity decisions rather than the ability of the sensors (pulse doppler and IFF) to provide inputs. In addition, measurements so far have only been made for a single association algorithm, nearest neighbor – mahanalobis. All results are based on an environmental load of 19 airborne objects (tracks).

We use a "choke" control to reduce or increase the amount of time available to complete processing. This is an effective method for emulating conditions such as an overload that could be caused by an increase in the number of objects being tracked or perhaps a loss of equipment.

Our key adaptivity criteria for this application are to deliver the highest QoS for important tracks but to attempt to not "drop" any tracks – regardless of their importance level. The definition of drop is when the precision attribute for a track, which is an integer between 0 and 7, becomes 0.

During any 3 second interval, there are typically 10 to 14 (of the 19) tracks that are available to be updated. This means that a sensor has received data on these tracks during the preceding interval.

Several different "operating points" have been

examined. When there is sufficient time for all processing to complete, we see that all tracks, regardless of their importance, have the maximum possible delivered QoS, i.e., a precision of 7.

When the choke is set so that only 4 tracks can be updated (of the 10 to 14 available for updating), the prototype was able to maintain maximum precision for the important tracks while the less important tracks had lower values, typically around 4. Even though the less important tracks had significantly lower QoS, none of them were dropped. This clearly indicates the system is being effective in its dynamic selection of which tracks should receive system resources.

When the choke was applied so that only 2 tracks could be updated in an interval, which represents a severe overload, the precision of important tracks dropped to around 5 while the precision of less important tracks dropped to around 3.5. No tracks were dropped.

The system finally had to drop tracks when the choke was set so only a single track could be updated in a three second interval. Nevertheless, none of the important tracks were dropped.

5 Summary and Future Directions

This work clearly indicates the above technologies can significantly improve the ability of a system to adapt to a changing environment. Further, QoS can be expressed in application level terms that the infrastructure can then use when managing available resources.

Future work includes investigating how to scale this technology to handle enterprise level processing requirements. In addition, we will be investigating how to employ this technology in systems that use commercial operating systems. Finally, we plan to develop a more demanding prototype that includes multiple competing applications along with the adaptive management of multiple resource types (we currently only manage the CPU).

6 References

[CLA 92] R.K. Clark, R.K, Jensen, E.D., Reynolds, F.D., "An Architectural Overview of the Alpha Real-Time Distributed Kernel," Proceedings of the USENIX Workshop on Micro-kernels and Other Kernel Architectures, pp. 127-146, Seattle, WA, April 1992.

[JEN 85] Jensen, E.D., et. al., "A Time-Driven Scheduling Model for Real- Time Operating Systems," Proceedings of IEEE Real-Time Systems Symposium, 1985, pp. 112-122.

[JEN 90] Jensen, E.D., Northcutt, J.D., "Alpha: An Open Operating System for Mission-Critical Real-Time Distributed Systems—An Overview," Proceedings of the 1989 Workshop on Operating Systems for Mission-Critical Computing, ACM Press, 1990.

[KAN 98] Kanevsky, A., et. al., "Design of a Quality of Service (QoS) Driven Adaptive C2 Prototype", MITRE Technical Report 98B002, January 1998.

[LAW 98] Lawrence, T.F., et. al., "Quality of Service for AWACS Tracking", 4th International C2 Symposium, Stockholm, September 1998.

[LOC 86] Locke, C.D., "Best-Effort Decision Making for Real-Time Scheduling," Ph.D. Thesis, Department of Electrical and Computer Engineering, Carnegie-Mellon University, 1986.

[WEL 94] D. Wells, "A Trusted, Scalable, Real-Time Operating System Environment," 1994 Dual-Use Technologies and Applications Conference Proceedings, pp. II-262/270, Utica, NY, May 1994.

*U.S. GOVERNMENT PRINTING OFFICE: 2002/710-038/10264