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

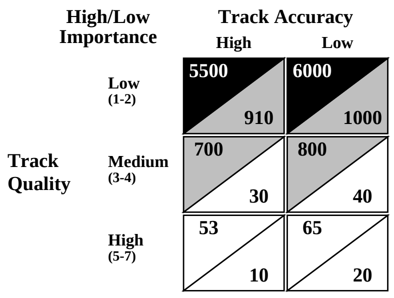

Fig. 1. Track Values as a Function of Track Quality-of-Service Metrics

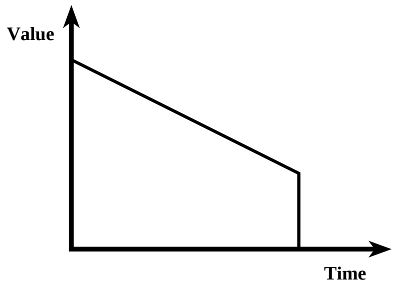

Fig. 2. Shape of Time-Value Functions for Association Computations

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.

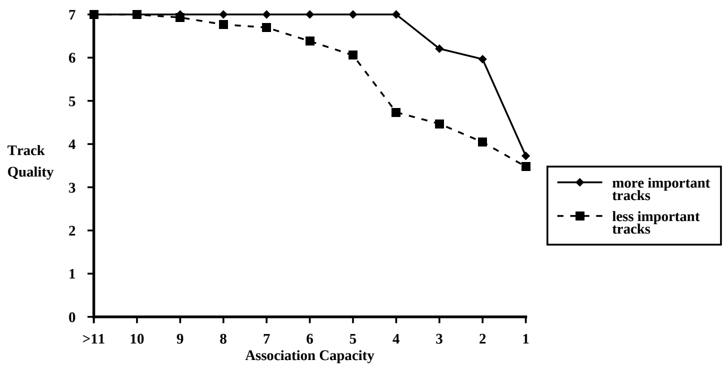

Fig. 3. Average Track Quality as a Function of Association Capacity

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)