The Control, Coordination, and Organizational Design of a Distributed Sensor Network*

Roger Mailler†, Bryan Horling‡, Victor Lesser§, and Régis Vincent¶

December 12, 2003

Abstract

In this paper we describe our solution to a real-time distributed tracking problem. The system works not by finding an optimal solution, but through a satisficing search for an allocation that is "good enough" to meet the specified resource requirements, which can then be revised over time if needed. The agents in the environment are first organized by partitioning them into sectors, reducing the level of potential interaction between agents. Within each sector, agents dynamically adopt various roles depending on the current state of the environment. These roles form goals which then are instantiated as task structures for use by the SRTA control architecture. These elements exist to support resource allocation, which is directly effected through the use of the SPAM mediation protocol. The agent problem solving component first discovers and generates commitments for sensors to use for gathering data, then determines if conflicts

*Effort sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory Air Force Materiel Command, USAF, under agreements number F30602-99-2-0525 and DOD DABT63-99-1-0004. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. This material is also based upon work supported by the National Science Foundation under Grant No. IIS-9812755. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), Air Force Research Laboratory or the U.S. Government.

†R. Mailler is with the University of Massachusetts.

‡B. Horling is with the University of Massachusetts.

§V. Lesser is with the University of Massachusetts.

¶R. Vincent is with SRI International.

exist with that allocation, finally using arbitration and relaxation strategies to resolve such conflicts. We have empirically tested and evaluated these techniques in both the Radsim simulation environment and using a hardware-based system.

1 Introduction

Recently, distributed sensors networks have begun to receive a great deal of attention from researchers in a variety of disciplines such as electrical engineering, computer networking [1, 2], and multi-agent systems [3]. Classically, sensors and their associated networks have received the most attention from engineers and researchers studying the properties and fabrication of the sensors themselves. These sensors have most commonly been controlled and monitored using high-powered, expensive centralized facilities which rely on dedicated underlying networks to transmit control information and receive raw or partial processed data from the sensors. In addition, most existing sensor networks are special purpose and entirely independent of one another when they could be use to fulfill multiple purposes. For example, weather monitoring and air traffic control radar systems have classically been isolated from one another even though they essentially use the same type of sensors.

With the increasing availability of a range of network technologies, researchers have begun to investigate the idea of using low-cost, off-the-shelf computers to monitor and control large numbers of shared sensors in a distributed way. There are a number of benefits to migrating control and monitoring away from a small number of centralized facilities. Among these are fault tolerance, localization of computation and communication within the networks for scalability, and rapid adaptability of shared sensing resources to changing environmental conditions.

For example, envision a system in which a number of chemical sensors are deployed throughout a region. Each of these sensors may have some limited ability to sense chemical signatures from the environment. From day to day, these sensors receive tasks of varying importance from scientists performing environmental studies such as monitoring the effects of greenhouse gases, acid rain, heavy metals, etc. At the same time, in the background, these same sensors monitor for more insidious chemical signatures, such as chemical weapons.

Now imagine that one of these sensors detects a poisonous gas. Immediately, it would autonomously switch its focus and processing power to specifically monitor the concentrations of that specific gas. It could begin to coordinate with neighboring chemical and weather tracking sensors to verify the detection, hone in on a specific chemical by using sensors with other capabilities, as well as monitor and predict the chemical's spread. The computer controlling the sensor could enlist the assistance of other computers which could then disseminate the information to the proper authorities, begin the evacuation of effected areas, etc.

In this article, we describe a distributed sensor network for discovering and tracking targets that has a number of characteristics in common with the example chemical detection sensor network described above. In particular, each of the sensors within our system has multiple, competing goals of various importance, high-level goals can only be accomplished by coordinating the actions of several sensors, and the control needs to be both distributed and autonomous in order to operate in real-time.

Our approach to managing this network consists of three major architectural and behavioral contributions. First, each sensor platform is controlled by a single agent which exists as part of a larger, heterogeneous organizational structure. This structure helps bound and focus the computation necessary to solve the distributed tracking problem by associating individual agents with one or more roles within that organization, taking responsibility for different parts of the overall goal. Second, individual agents are sophisticated, autonomous problem solvers. Each incorporates a domain independent soft real-time control architecture (SRTA) which is used to model and control the activities of the agent, and a domain specific problem solving component which reasons about and reacts to the surrounding environment. Finally, a negotiation mechanism and protocol (SPAM) is employed to allocate sensor resources and resolve conflicts. Agents in the organization responsible for tracking use this protocol to ensure sufficient quantity and qualities of data are obtained for all targets to be tracked, if at all possible.

Each of these technologies plays an important role solving the sensor-to-target allocation problem. The SPAM protocol [4] lies at the heart of this process, enabling agents to request sensors, and then dynamically detect and resolve conflicts when they arise by using distributed negotiation. Where SPAM resolves conflict between agents, the SRTA architecture [5] resolves conflicts that

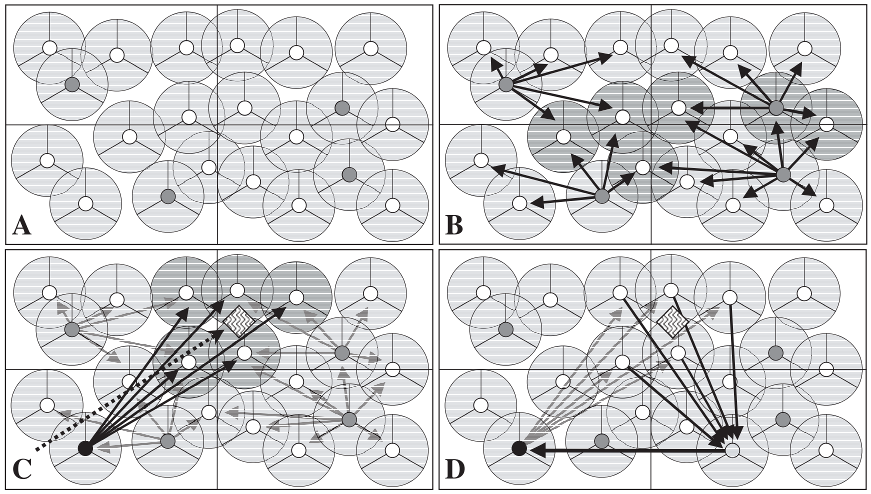

Figure 1: High-level architecture. A: sectorization of the environment, B: distribution of the scan schedule, C: communicating over tracking measurements, and D: fusion of tracking data.

exist within an agent. By modeling tasks and commitments and using several scheduling techniques to manage local activities, SRTA allows allocations to accrue quality even when there are unresolved conflicts. The organizational design both acts to limit the distance over which information must be propagated and concentrates the information needed for the various organizational roles to operate efficiently. This has the important effect of reducing communication effort and facilitate the allocation process. All of these technologies must operate in real-time to be effective in this distributed sensor network environment.

In the remainder of this article, we will give an overview of the sensor network used in this work, describe its organization, the agent control mechanisms, and the resource allocation protocol. Section 6 discusses results and experiences from implementing the system both in simulation and on actual hardware. The article will conclude with some final thoughts and discussion of future work.

2 Overview

The sensor platforms used in our network each incorporate three Doppler radar-based sensor heads (Zemany and Gaughan in [3]) that have a range of approx-

Figure 2: The sensor platforms have three Doppler radar sensor heads and are controlled by a BASIC stamp micro-controller.

imately 20 feet and a 120 degree viewable arc (see figure 2). The platform is controlled by a BASIC stamp micro-controller which may only collect data from one of these sensor heads at any time. Each measurement takes about 400 milliseconds and returns an average amplitude (relative distance) or frequency shift (relative velocity) over the collection period. Sensor platforms can communicate with one another over an 8-channel, radio-frequency (RF) system. But, due to the close proximity of the receiver and transmitter within the platforms, they are unable to simultaneously transmit and receive messages regardless of channel assignment. In addition, because this is RF, multiple agents cannot transmit on a single channel at the same time without causing interference. Each of the platforms is connected to a computer which is capable of locally hosting one or more processes.

Several key items should be extracted from this architectural style description of the sensor platforms. The first is that no single platform can effectively track a target by itself. Tracking, in this sense, should be interpreted as being able to estimate a target's position with some degree of certainty and collecting these estimates over time to model and predict the targets future position. Because the sensors are only capable of returning relative distance, single platforms cannot localize a target to region smaller than a 120 degree arc at some distance based on a noisy measurement. So, in order to improve its estimate it must enlist the aid of other sensor platforms to triangulate the target's position. If these additional measurement occur in a temporally coordinated fashion the quality of the triangulation improves, particularly when the target is moving.

The need for explicit coordination prevents us from using methods like those presented in [6] which only required one sensor to track a target and migrated the task through the field as the target moved.

The second key feature of these platforms is that their communication is unreliable, has limited bandwidth, and limited range. The effect of this is that information and control must be distributed in order to be reactive to environmental changes. For instance, let's say that we were to employ a single agent to manage all of the coordination needed to track the several targets. In order for this agent to assign sensors to each of the targets it would need to know, at very least, the targets' locations. Since the targets are moving, these locations would need to be updated frequently enough for the agent to continue allocating the correct resources to track. In addition, every time a change is needed to the allocation, this agent would need to communicate with each of the affected sensor platforms. It should be clear that in a limited communications environment, this is likely to be infeasible for anything but a small number of targets. Distributing the work across a number of agents prevents bottle necks by parallelizing the communications and control.

A high-level view of the solution described in this article can be seen in Figure 1. Each sensor is controlled by a single agent, and the organizational design divides these sensor agents into location-based sectors. Each sector has a *sector manager*, a role in the organization which has several responsibilities associated with information flow and activity within the sector. Among these responsibilities is the dissemination of a scan schedule to each of the sensors in its sector, which specifies the rate and frequency which should be used to scan for new targets. An example schedule might be to have the sensors on the edge of the sector scanning frequently for incoming targets and the sensors in the interior of the sector lying dormant until a target is detected. The scan schedule is used by each sensor to create a description of the scanning task, which is in turn used by the SRTA architecture to schedule local activities.

When a new target is detected, the sector manager selects a *track manager*, a different organizational role responsible for tracking that target as it moves through the environment. This entails estimating future location and heading, gathering available sensor information, requesting and coordinating over the sensors, and fusing the data they produce. After determining which sensors can see its target, the track manager requests commitments from the them.

Upon receipt of such a commitment, like 'repeatedly take measurements using head 1', sensors takes on a *data collection* role. Like the scan schedule, these commitments form the basis of task descriptions used by SRTA to schedule local activities. If conflicting commitments are received by a sensor, which implies that the agent has been asked to perform multiple concurrent data collection roles, SRTA will attempt to satisfy all requests as best possible. This provides a window of marginal quality where SPAM can detect the conflict, and then negotiate with the competing agent to find an equitable long-term solution. As the data is gathered, it is transmitted back to the track manager for incorporation into the target's track.

The process of collecting measurements, fusing the data into position estimates, modeling the targets predicted path based on these estimates, and allocating new sensor resources continues until the target is lost or moves outside the sensor field. In either case, the track manager revokes all commitments from the sensors and informs the sector manager that it is no longer tracking its target. This allows the sensors to be freed up to potentially re-acquire the target, and allows the sector manager to determine that the detection must be assigned to a new track manager.

In the next section, we will discuss how the environment is partitioned into sectors and how roles are assigned to the agents within them.

3 Organizational Design

The notion of "organizational design" is used in many different fields, and generally refers to how entities in a society act and relate with one another. This is true of multi-agent systems, where the organizational design of a system can include a description of what types of agents exist in the environment, what roles they take on, and how they interact with one another. The objectives of a particular design will depend on the desired solution characteristics, so for different problems one might specify organizations which aim toward scalability, reliability, speed, or efficiency, among other things.

The organizational design used in this solution primarily attempts to address the scalability problem by concentrating knowledge within the agents that are most likely to need it and by imposing limits on how far certain classes of information propagate. As will be seen below, this is done at the expense of

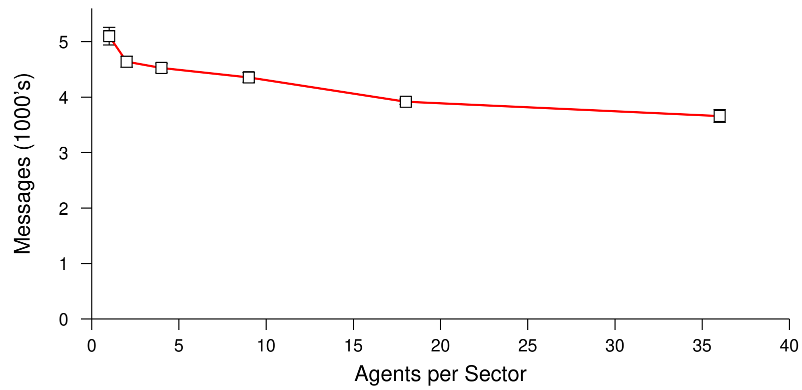

Figure 3: Total number of messages for 36 agents tracking 4 targets given various sector sizes.

reaction speed, because by limiting the scope any single agent has, one necessarily increases the required overhead when the agent's task moves outside that scope.

The environment itself is organizationally partitioned into a series of sectors, each a non-overlapping, rectangular portion of the available area, shown in Figure 1A. The purpose of this division, as will be shown below, is to exploit locality and limit the interactions needed between sensors. This is an important element of our attempt to make the solution scalable; these same pressures lead to the similar approachs taken in [7, 8]. In figure 1A, sensors are represented as divided circles, where each 120 degree arc represents a direction the node can sense in. As agents come online, they must first determine which sectors they can affect. Because the environment itself is bounded, this can be trivially done by providing each agent the height and width of the sectors. The agents can then use this information, along with their known position and sensor radius, to determine which sectors they are capable of scanning in. We use this technique to dynamically adapt the agent population for scanning and tracking activities to better partition and focus the flow of information.

Within a given sector, agents may work concurrently on one or more of several high level goals: managing a sector, tracking a target, producing sensor data, and processing sensor data. The organizational hierarchy is abstractly represented in Figure 5. The organizational leader of each sector is a single sec-

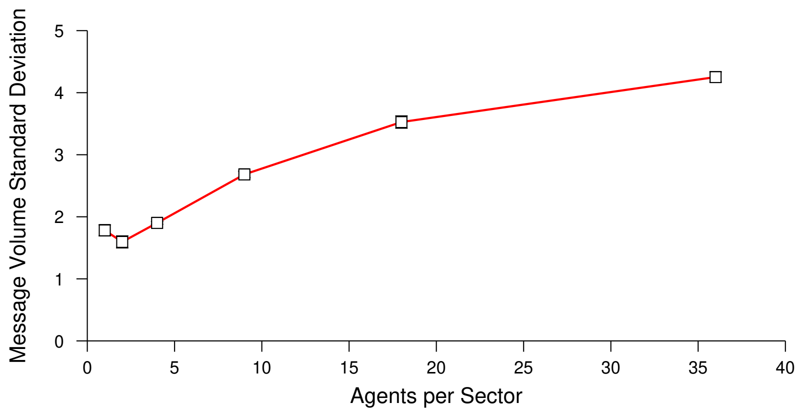

Figure 4: Communication load disparity among agents within each sector.

tor manager, which serves as the locus of activity for that sector. This manager generates and distribute plans (to the sensor data producers) needed to scan for new targets, stores and provides local sensor information as part of a directory service, and assigns track managers. The sector managers act as hubs within a nearly-decomposable hierarchical organization, by directly specifying scanning activities, and then selecting agents to oversee tracking activities. They also concentrate nonlocal information, facilitating the transfer of that knowledge to interested parties. Individual track managers initially obtain their information from their originating sector manager, but will also interact directly, though less frequently, with other sector and track managers, and thus do not follow a fixed chain of command or operate solely within their parent sector as one might see in a fully-decomposable organization. Track managers will also form commitments with one or more agents to gather sensor data, but this relationship is on a voluntary basis, and that gathering agent's behavior is ultimately determined locally. Because much of the information being communicated is contained within sectors, the size and shape of the sector has a tangible effect on the system's performance. If the sector is too large, and contains many sensors, then the communication channel used by the sector manager may become saturated. If the sector is too small, then track managers may spend excessive effort sending and receiving information to different sector managers as its target moves through the environment.

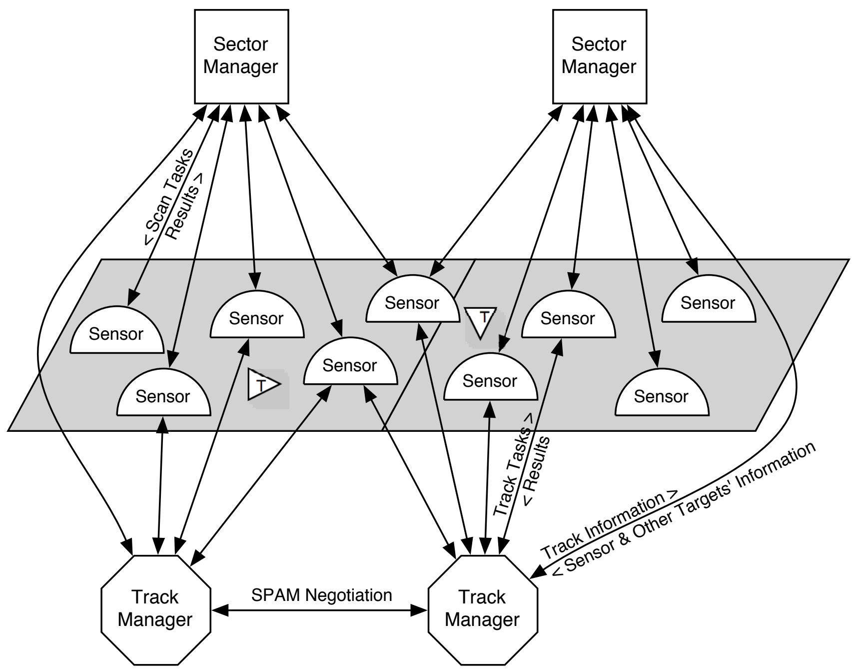

Figure 5: Overview of the agent's organizational hierarchy, with some information flows represented.

To study the effects of sector size on communication cost, we conducted an empirical study. In these experiments, a group of 36 sensors were organized into between 1 and 36 equal-sized sectors. Four targets were introduced into the environment and the number of messages were measured for a fixed duration simulation. These simulations were conducted using a perfect communications model (i.e. channel congestion is not accounted for). As you can see from Figure 3, sector size has a fairly strong effect on the number of messages being sent. As you increase the number of agents within a sector, the number of messages goes down. What is noteworthy about this effect is that it diminishes rapidly as you increase the sector size. This suggests that there is a point at which the benefits of increasing the sector size are outweighed by the cost of centralizing too much information and the resulting communication within the sector manager. Figure 4 verifies this intuition. This figure shows that as you put more agents within a single sector, the standard deviation of the message volume increases. The increase is caused by having a greater portion of the organization load concentrated within fewer agents.

We have found empirically that a reasonable sector would contain 8 sensors, but would still function adequately with as many as 10 or as few as 5. The physical dimensions of such a sector depend on the density of the sensors, and in different environments one would need to take into account sensor range, communication medium characteristics and maximum target speed. Further information on partitioning agent populations, including a more sophisticated technique which utilizes heterogeneous regions, can be found in [9].

To see how the organization works in practice, consider a scenario starting with agents determining what sectors they can affect, and which agents are serving as the managers for those sectors. Ideally, the sector managerial duty would be delegated and discovered dynamically at runtime, but due to the lack of a true broadcast capability in the RF communication medium, we statically define and disburse this information a priori[1]. In Figure 1, these sector managers are represented with shaded inner circles. Once an agent recognizes its manager(s), it sends each a description of its capabilities. This includes such things as the position, orientation, and range of the agent's sensor. The manager then has the

[1] A limited broadcast capability does exist, which can reach all sensors listening on a single channel. It is not possible in this architecture to broadcast a single message to agents which are using different channels.

task of using this data to organize the scanning schedule for its sector. The goal of the scan schedule is to use the sensors available to it to perform inexpensive, fast sensor sweeps of the area, in an effort to discover new targets. The manager formulates a schedule indicating where and when each sensor should scan, and communicates with each agent over their respective responsibilities in that schedule (see Figure 1B). The manager does not strictly assign these tasks - the agents have autonomy to locally decide what action gets performed when. This is important because sensors can potentially scan in multiple sectors, thus there is the possibility that an agent may receive multiple, conflicting requests for commitments from different sector managers. The agent's autonomy and associated local controller permit the agent itself to be responsible for detecting and resolving these conflicts. If one receives conflicting requests for commitments, it can elect to delay or decommit as needed. Shaded sensors in the previous figure show agents receiving multiple scan schedule commitments.

Once the scan is in progress, individual sensors report any positive detections to the sector manager which assigned them the scanning task, which can then spawn a new track manager as shown in Figure 6. Internally, the sector manager maintains a list of all local agents that currently perform the role of track manager, and location estimates for the targets they are tracking. These location estimates are used to determine the likelihood of the positive detection being a new target, or one already being tracked. If the target is new, the manager uses a range of criteria to select one of the agents in its sector to be the track manager for that target. Not all potential track managers are equally qualified, and an uniformed choice can lead to very poor tracking behavior if the agent is overloaded or shares communication bandwidth with other garrulous agents. Therefore, in making this selection, the manager considers the agents' estimated load, communication channel assignment, geographic location and activity history. Ideally, it will select an agent which has minimal channel overlap, is not currently tracking a target, but has tracked one previously. This will minimize the potential for communication collisions, which occur if two agents on the same channel attempt to send data at the same time, but maximize the potential amount of cached organizational data the agent can be reuse. As we have seen previously, this notion of limited communication is an important motivating factor and recurring theme in this architecture which contributes to the organizational structure, role selection, protocol design and the frequency and verbosity of communication actions.

The assigned track manager (shown in figure 1C with a blackened inner circle) is responsible for organizing the tracking of its assigned target. To do this, it first discovers sensors capable of detecting the target, and then communicates with members of that group to gather the necessary data. Discovery is done using the directory service provided by the sector managers. One or more queries are made asking for sensors which can scan in the area the target is predicted to occupy. The track manager must then determine when the scans should be performed, considering such things as the desired track fidelity and time needed to perform the measurement, and coordinate with the discovered agents to disseminate this goal (see Figure 1C). As with scanning, conflicts can arise between the new task and existing commitments at the sensor, which the agent must resolve locally.

The data gathered from individual sensors is collected by an agent responsible for fusing the measurements into a location estimate and extending the computed track (see Figure 1D). In a general sense, this data fusion agent could be any agent in the population able to communicate efficiently with both the data sources and the ultimate destination of the tracking data. However, the data fusion process for this application (see Vargas et al. in [3]) is fairly lightweight, and thus does not benefit from distribution for load balancing purposes. In addition, transferring the fused data results introduces an unnecessary delay while it is being communicated to the track manager. For this reason, in this work, the data fusion and track manager roles are always performed by the same agent.

When raw measurement data is received by the track manager, it verifies the quality and association of the measurement. Quality can be measured by several means and in this work, the signal-to-noise ratio is used. If a measurement is returned with a significantly high signal strength, it is considered for potential fusion. The association of a measurement really involves two different things: temporal and target association.

Temporal association attempts to match measurements taken from various sensors based on the time the measurements were taken. As previously mentioned, the accuracy of triangulating a moving target is in part dependent on the relative temporal coordination of the measurements. If a track manager were to fuse measurements that were taken over a wide range of time, the resulting

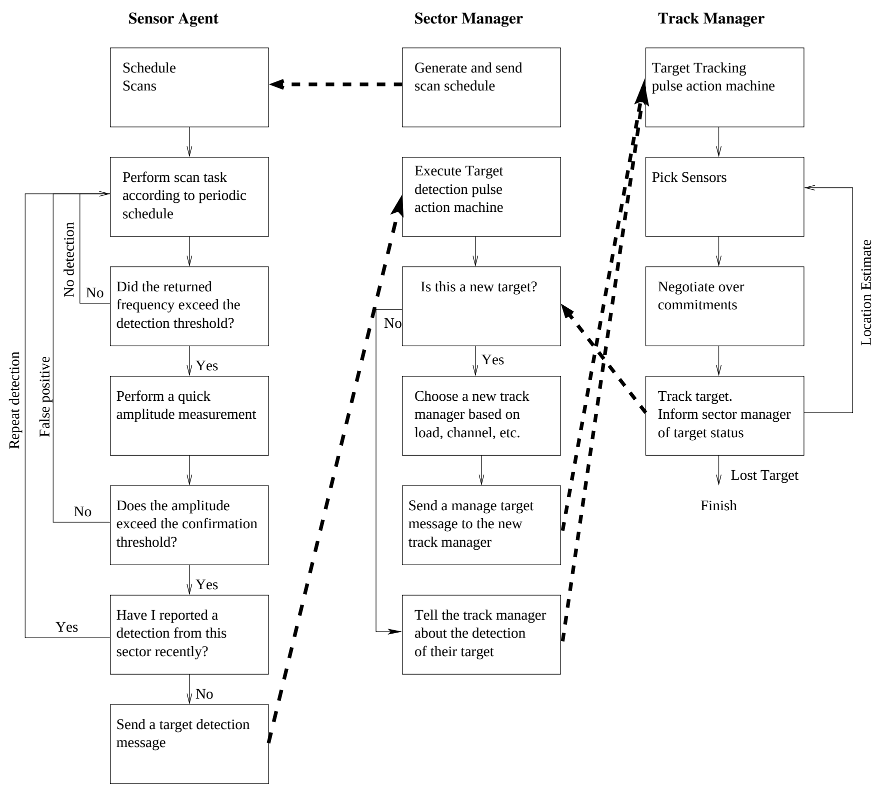

Figure 6: An abstraction of the messages and reasoning used for target detection by sensor agents, sector and track managers.

estimation could be quite poor. There are several reasons why a measurement may be returned that cannot be temporally associated with measurements from other sensors. One reason is delays and loss introduced by the communications. For example, if the commitment request from the track manager never gets to the sensor, no measurement will be taken. Another reason is unresolved resource contention within the individual sensors. If, for example, one of the sensors decides to delay the start of a measurement in order to deconflict its internal schedule, it will be harder to match the resulting measurement with data from other sensors.

Within our system, temporal association maintained by queuing measurements for a finite period and matching them with one another based on their actual measurement time. We make the assumption that the computers controlling the sensors are time synchronized using a Network Time Protocol (NTP) like mechanism.

Target association, as the name implies, tries to match measurements with targets. Consider the case of two targets, T1 and T2, that are following one another through the sensor field. Now imagine that T1 moves out of the viewable area of one of the sensor heads being used to measure it and T2 moves into its this head's view. If a measurement is taken by that head before the T1's track manager changes its allocation, the resulting amplitude will be of the wrong target. When this measurement is fused with the track of T1, it will appear to be closer to T2 than before. Over time, if this pattern continues, the targets will be indistinguishable from one another. In fact, T1 could end up being ignored and both track managers could try to track T2.

To prevent this from happening, track managers check to ensure that each measurement has a higher likelihood of belong to their target than others that may be near to it. To do this, we exploit the view of the sector manager. Because sector managers are periodically given estimated target locations for all of the targets within their sector, it is easy for them to determine when two or more track managers are likely to have a target association problem. When the sector manager detects this possibility, it informs the appropriate track managers by sending them the locations of the target that are close to them. This allows the track managers to discard any measurement that was likely to have been collected by a sensor viewing another target. If, for some reason, two track managers do fuse their target into one, the sector manager can also recognize

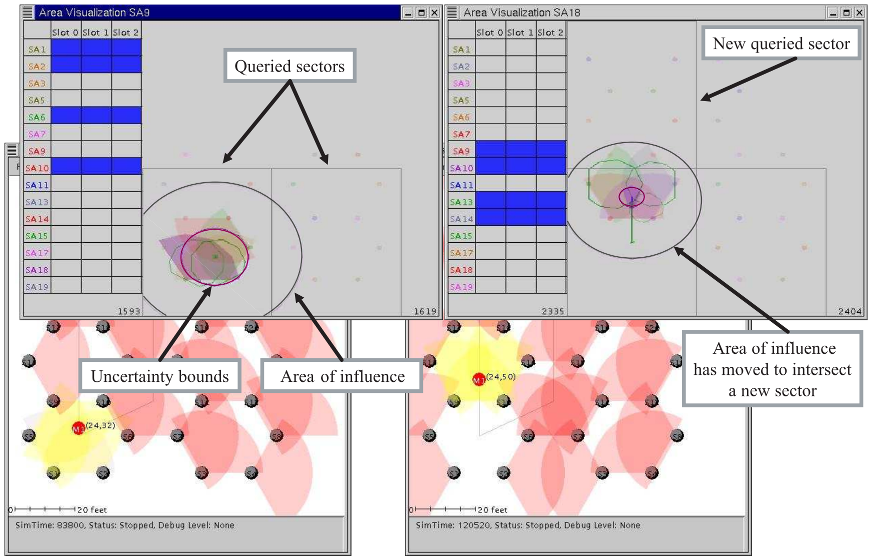

Figure 7: As the target moves to another sector, the track manager queries the manager of the new sector.

this fact and inform one of the managers to stop tracking. This alleviates the unnecessary resource contention created by having two track managers fight over the same resources to track the same target.

If the data values returned are of high enough quality, and the agent determines those measurements were taken from the correct target, then they are used to triangulate what the position of the target was at that time. This data point is then added to the track, which itself is used as a predictive tool when determining where the target is likely to be in the future. At this point the track manager must again decide which agents are needed and where they should scan, and the sequence of activities is repeated.

Partitioning the environment reduces the amount of information and processing that agents must perform for several different tasks. For example, generating a coherent scan schedule for a group of sensors is simplified by only taking into account a tractable number of them. Similarly, when a new target is detected as a result of a scan, that information can be sent to only the appropriate sector manager, which can determine directly if it is a new or existing target based on local information. Sectors also facilitate gathering data about the sensors themselves, as track managers need only perform a single query to the appropriate sector manager to discover all the sensors available in within that region (see figure 7). In fact, the partitioning makes nearly every aspect of this solution scalable to arbitrary numbers, with the exception of the tracking allocation, which has its own solution to this problem, as shown later. As a side effect, partitioning does reduce the system's reactivity, because an extra step may be required to fetch information that is not available locally. We cope with this problem wherever possible by caching such data to avoid redundant queries, and by assigning new roles whenever possible to agents which have served that same role in the past, to take advantage of that cached data.

Although not required in the scenarios presented in this article, it is interesting to note the applicability of this organization to situations where agents have an additional limitation or attenuation of communication capability based on the geographic distance separating the participants. In this case, this partitioned organization could serve as the basis of an ad-hoc network, where messages are routed from one sector to the next, using the organizational structure as a guide, until they reach their destination. This further emphasizes the notion that "local" communication is more efficient, and the locality of information should be exploited by the organization to take advantage of it.

In this section we have shown how the organization plays a critical role in ensuring that information flowing within the sensor network is managed to both minimize the delay and expensive of communication and to maximize its availability for effective decision making. The next section describes some of the mechanisms that go into forming the agent level control of the sensors.

4 Agent Architecture

4.1 Soft Real-Time Control

The Soft Real-Time Control Architecture (SRTA), the agent control engine used by this solution, provides several key features to the agents within our sensor network:

1. The ability to quickly generate plans and schedules for goals that are appropriate for the available resources and applicable constraints, such as deadlines and earliest start times.

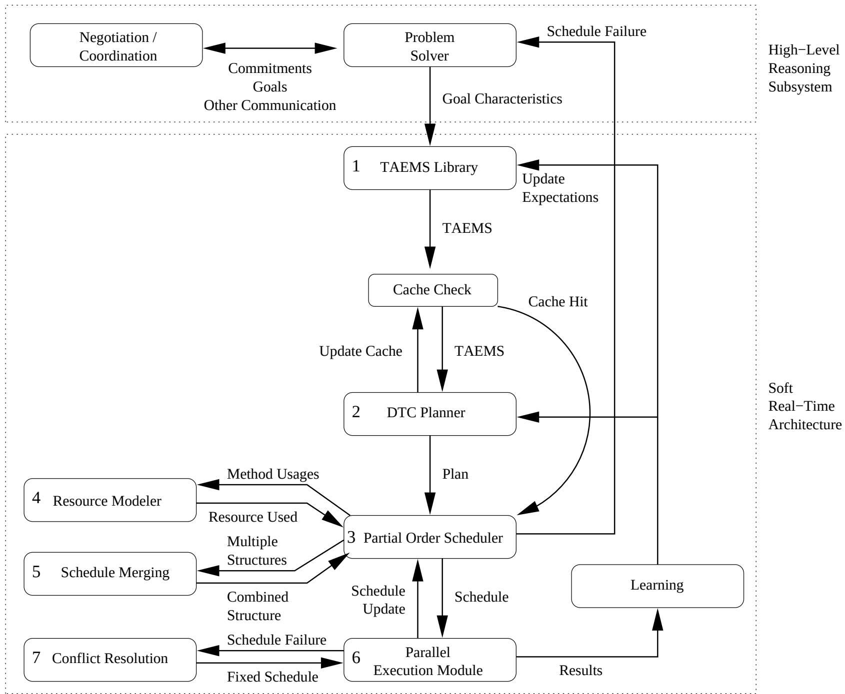

Figure 8: The soft real-time control architecture.

2. The ability to merge new goals with existing ones, and multiplex their solution schedules.

3. The ability to use explicit representations of uncertainty and efficiently handle deviations in expected plan behavior that arise out of variations in resource usage patterns and unexpected action characteristics.

Abstractly, SRTA (shown in Figure 8) operates as a single functional unit within an agent, which itself is running on a conventional (i.e. not real-time) operating system. The SRTA controller is designed to be used in a layered architecture, occupying a position below the high-level reasoning component in an agent. In this role, it accepts new goals, report the results of the activities used to satisfy those goals, and also serve as a knowledge source about the potential ability to schedule future activities by answering what-if style queries.

The components that comprise SRTA assume a majority of the responsibility needed to satisfy goals, which allows the high-level reasoning system to focus on goal selection, determining goal objectives and other potentially domain-dependent issues. For example, agents may elect to coordinate using abstractions of their activities or resource allocations which are then locally translated into a precise schedule [4]. SRTA can then use these schedules to both enforce the semantics of the commitments which were generated, and automatically attempt to resolve conflicts that were not addressed through coordination.

In the rest of this section, TÆMS, the modeling language used to describe goals to SRTA, is explained. Following that, we will show how the components of the SRTA architecture work together to provide the agents with the ability to handle multiple, concurrently executing goals. Lastly, we will describe a specialized component, called the Periodic Task Controller (PTC). The PTC provides SPAM with an abstract view of the current resource schedule which it uses as a basis for mediation.

4.2 TÆMS

TÆMS, the Task Analysis, Environmental Modeling and Simulation language, is used to quantitatively describe the alternative ways a goal can be achieved [10]. A TÆMS task structure is essentially an annotated task decomposition tree. The highest level nodes in the tree, called task groups, represent goals that an agent may try to achieve. For example, the goal of the structure shown in Figure 9 is `Setup-Hardware`. Below a task group there will be a set of tasks and methods which describe how that task group may be performed, including sequencing information over subtasks, data flow relationships and mandatory versus optional tasks. Tasks represent sub-goals, which can be further decomposed in the same manner. `Setup-Hardware`, for instance, can be performed by completing `Startup`, `Init`, and `Obtain-Background-Noise`.

Methods, on the other hand, are terminal, and represent the primitive actions an agent can perform. Methods are quantitatively described, in terms of their expected quality, cost and duration. `Activate-Sector_0`, then, would be described with its expected duration and quality, allowing the scheduling and planning processes to reason about the effects of selecting this method for execution. The quality accumulation function (QAF) below a task describes how the quality of its subtasks is combined to calculate the task's quality. For example, the `q_min` QAF below `Init` specifies that the quality of `Init` will be the minimum quality of its subtasks - so all the subtasks must be successfully performed for the `Init` task to succeed. Interactions between methods, tasks,

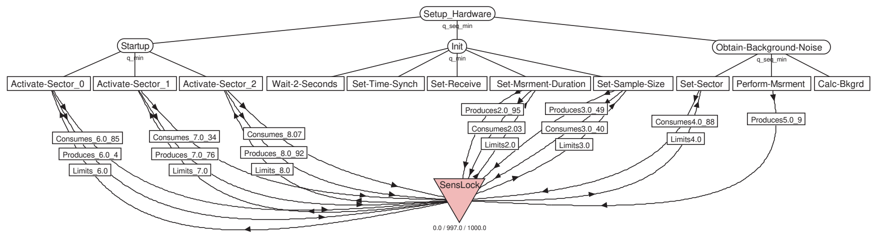

Figure 9: An abbreviated view of the sensor initialization TÆMS task structure.

and affected resources are also quantitatively described. The curved lines in Figure 9 represent resource interactions, describing, for instance, the produces and consumes effects method `Set-Sample-Size` has on the resource `SensLock`, and how the level of `SensLock` can limit the performance of the method.

TÆMS structures are used by our agents to describe how particular goals may be achieved. Rather than hard coding, for instance, the task of initializing the sensor, we encode the various steps in a TÆMS structure similar to that shown in Figure 9. This simplifies the process of evaluating the alternative pathways by allowing the designer to work at a higher level of abstraction, rather than be distracted by how it can be implemented in code. More importantly, it also provides a complete, quantitative view that can be reasoned about by planning, scheduling and execution processes. A given task structure begins its existence when it is created, read in from a library (Figure 8-1), or dynamically instantiated from a template at runtime. Planning elements are involved both in the generation of the structure, and then in the selection of the most appropriate sequence of methods from that structure which should be performed to achieve the goal given the currently available resources. This sequence is then used by a scheduling process to determine the correct order of execution, with respect to such things as precedence constraints and resource usage. Finally, this schedule will be used by an execution process to perform the specified actions, the results of which are written back to the original task structure.

The schedules produced by individual TÆMS structures are the building blocks for an agent's overall schedule of execution. A valid schedule completely describing an agent's activities will allow it to correctly reason about and act upon the deadlines and constraints that it will encounter, for example a resource restriction. Typically, however, schedules are only used to describe lower-level activity. In this domain, this encompasses sensor initialization, scanning and tracking activity, data fusion and the like.

4.3 Scheduling

In the SRTA architecture, we have attempted to make the scheduling and planning process incremental and compartmentalized. New goals can be added piecemeal to the execution schedule, without the need to re-plan all the agent's activities, and exceptions can be typically handled through changes to only a small subset of the schedule. Figure 8 shows the organization of SRTA. In this architecture, goals can arrive at any time, in response to environmental change, local planning, or because of requests from another agents. The goal is used by the problem solving component to generate a TÆMS task structure, which quantitatively describes the alternative ways that goal may be achieved. The TÆMS structure can be generated in a variety of ways; in our case we use a TÆMS "template" library, which we use to dynamically instantiate and characterize structures to meet current conditions. Other options include generating the structure directly in code, or making use of an approximate base structure and then employing learning techniques to refine it over time.

SRTA uses the Design-To-Criteria component [11] to generate linear plans solving the goal described in the TÆMS structure (Figure 8-2). It employs a battery of techniques to efficiently discover and reason about the various activity schedules which can address that goal. The ability to make trade-offs while respecting commitments is particularly important, as DTC attempts to select the quantitatively "best" plan which meets the specified requirements. DTC uses criteria such as potential deadlines, minimum quality, external commitments, and soft and hard action interrelationships to select an appropriate sequence of activities.

The resulting plan is used to build a partially ordered schedule, which uses structural details of the TÆMS structure to determine precedence constraints and search for actions which can be performed in parallel (Figure 8-3). The partial order scheduler (POS) also provides the ability to quickly shift methods' execution order at any point in time instead of performing costly re-planning [5]. In a real-time environment, schedule adjustments are more frequent; by not imposing unnecessary ordering constraints on our agent's schedule the agent has a better chance of achieving the time, cost and quality criteria of its goal. A pair of specialized components are used to assist the POS during this final scheduling phase. The first, a resource modeling component, is used to ensure that resource constraints are respected (Figure 8-4). A schedule merging module then allows the partial order scheduler to incorporate the actions derived from the new goal with existing schedules (Figure 8-5).

Our notion of "parallel" in this architecture includes activities which run concurrently in parallel, as in a multiple processor environment, and those which run virtually in parallel, as in a time-slicing, multi-processing operating system. If we view the sensor as a specialized, separate processor, our task structures contain both types of methods. For instance, a sensor measurement action can take place concurrently with actions on the primary processor. Unifying these notions simplifies the scheduling process, and can be represented appropriately using TÆMS.

Once the schedule has been created, an execution module is responsible for initiating the various actions in the schedule (Figure 8-6). It also keeps track of execution performance and the state of actions' preconditions, potentially re-invoking the partial order scheduler when failed expectations require it. Using the ordering constraints described in the schedule, the execution component can directly determine which methods can be run concurrently. By overlapping their execution, we reduce the total execution time, which effectively increases the agents overall work capacity. The gain in execution time, and resulting flexibility, is used to address resource availability, in addition to improving the likelihood the scheduler can accommodate real-time changes without breaking deadline constraints.

If this is unsuccessful, a conflict resolution module is used to reason about mutually-exclusive tasks and commitments, determining the best way to handle conflicts (Figure 8-7). Repairs can be accomplished in a variety of ways, for instance, by relaxing constraints such as the goal completion criteria or delaying its deadline, completing a substitute goal with different characteristics, or decommiting from a lower priority goal or the goal causing the failure.

The execution characteristics of the SRTA architecture as a whole depend largely on the frequency and complexity of the goals it is asked to plan and schedule. On average, we observe cycle times of between 50 and 100 milliseconds on 400 MHz x86-based systems, although this can jump to a half-second or more if a particularly complex situation arises. A cycle represents a single pass of the SRTA engine analyzing the current goals and executing methods. Because the system runs on a conventional operating system (Linux in this case), competing external processes may add an additional level of performance uncertainty.

4.4 Periodic Tasks

In addition to the general purpose scheduling outlined above, the agents also incorporate a more specialized periodic scheduling mechanism. In order to reduce communication, commitments between agents can be expressed as tasks which are to be performed periodically and indefinitely until notified otherwise. To reduce computation, these periodic tasks are arranged and scheduled according to a slot-based scheme, where a given task will be assigned to run in one or more slots in a repeating cycle. For example, in this architecture a cycle consists of three slots, each of which has a length of one second, for a total cycle time of three seconds. Then, elements of a commitment might specify that a scan or track measurement should be performed indefinitely on sensor head 1 in slot 2. A subsequent message would indicate when to stop performing the task. Using this technique, a pair of messages can cause a large amount of data to be gathered, and the required local scheduling overhead is reduced by abstracting the continuous timeline into discrete independent portions.

Like any scheduling process, conflicts can arise when two tasks are to be performed in the same slot. Different resolution techniques are used to cope with this situation. Individual tasks have priorities associated with them - for example, tracking tasks are more important than scanning tasks, and as mentioned previously, some tracking tasks may be more important than others. These priorities can be used to give preference to certain tasks. If a task is preempted, it may either be suspended for the lifetime of the higher priority task, or shifted to a free time slot if one is available. Alternately, if two tasks have the same priority, the scheduler will attempt to divide the slot between them, so that the first will run during one cycle, the second during the subsequent cycle, then the first, and so on. A third technique, not used in this domain, could also attempt to perform both tasks in the same slot if sufficient time, or a shorter duration alternative, exists to do so.

In this section, we described the SRTA architecture and showed how it is used within our system to provide the agents with autonomous reasoning capabilities. These capabilities are exploited to both reduce communications, by allowing agents to transmit control information in an abstract manner, and to provide

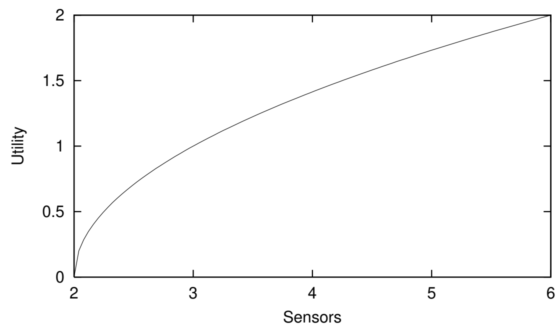

Figure 10: Utility of taking a single, coordinated measurement from a set of sensors.

tolerance to rapid environmental changes, by allowing agent to re-plan and reschedule reactively based on local information. The next section of this article describes the SPAM resource allocation protocol which provides this system with the ability to handle resource conflicts by elevating the decision making to the track managers.

5 Resource Allocation

5.1 Tracking as Resource Allocation

Modeling the target tracking domain as a resource allocation problem is fairly straightforward. Each of the targets in the environment can be considered a task, which is assigned to a track manager. The sensors are the resources and the job of the track managers is to obtain enough sensing time from the correct sensors to track their targets.

There are a number of characteristics about this particular resource allocation problem that make it challenging. The first is that, although this problem can be solved crudely using constraint satisfaction techniques, it lends itself most naturally to being solved using some type of optimization. Optimization is a good fit for two reasons. The first is that the value of a sensor platform to a track manager is directly correlated with the distance and relative angle of its most appropriate sensor head to the manager's target. Increasing the angle or distance decreases the accuracy and the resulting value of a measurement. In addition, increasing the number of sensors involved in taking coordinated measurements improves the accuracy of the resulting location estimation (see figure 10). The goal of the track managers, therefore, is not simply to come up with a conflict free resource assignment, but to derive a solution that *maximizes* the number and the value of the sensors used to track the targets.

The second major difficulty is the need to temporally coordinate the actions of the sensors. Coordinating the resources in this way adds another dimension to the problem and increases its difficulty considerably. The biggest problem with temporal coordination is that time is a continuous value. There are essentially an infinite number of possible combinations to consider when trying to deconflict and coordinate the sensor schedules. As mentioned in the previous section, the method used in this system to reduce the number of possible combinations is to use a finite planning horizon on a slot based schedule. By using temporally coordinated slots, the track managers lose the ability to perform fine-grained scheduling, but the overall complexity of the resource allocation is decreased. So, in fact, the track managers not only have to maximize the number and value of the sensors, but have to maximize the amount of time those sensors are dedicated to taking measurements for their target.

If we say that M_s^i is the set of good sensors measurements (can see the target) leading to the positional estimate in a single slot s for a task i, and Util(M_s^i) is defined as the utility function in Figure 10 then the utility function for that task during a specific period is:

$$U_i(a_i) = \sum_{s=1}^{k} Util(M_s^i)$$

Fortunately, the need for coordination actually allow us to consider a much smaller subset of the possible allocations for a given task. In fact, track managers within our system use a simplified set of *objective levels* defined by their utility functions to assign resources to their targets. Each objective level is expressed as a cross product $D_m \times D_s$ denoting the number for sensors, desired for a number of slots in the planning horizon. For example, a track manager may wish to have three sensors for two slots, which is denoted 3×2. Although the number of slots in a period is variable, for this domain, we typically set it to match the number of sensor heads on each platform, which is three. In order to prevent certain targets from being ignored in order to improve the quality of another targets estimate, track managers are penalized for not triangulating their target during a full period.

The third characteristic of interest is the dynamic nature of the problem. As the targets move through the environment, moving in and out of the range of the sensors, the underlying resource allocation problem changes in structure. This drives the need to constantly monitor, re-evaluate, and reallocate the sensors used to measure the positions of the targets. It also means that any method used to allocate the resources has to be responsive to changes that occur in the middle of the allocation process.

The last major challenge is that the allocation technique needs to be resource aware. As mentioned in Section 2, the communications infrastructure is RF-based, which in this case makes it is very slow and unreliable. These properties make it essential to not only limit the amount of communications, but to be aware of and adapt to changes in the overall ability to communicate. For example, if it is taking a long time to get messages to a particular track manager, it probably makes sense to avoid creating conflicts with it. Also, if a particular allocation is only useful for a very short period of time, it may not make sense to engage in a complex reallocation process when a simpler one may meet the basic need to track.

5.2 The SPAM Protocol

Scalable, Periodic, Anytime Mediation (SPAM) algorithm is built around the principle of "good enough, fast enough". As such, the protocol is actually divided into two major stages. Stage 1 of the protocol uses local information to derive and bind temporary solutions that are seldom free of conflict and are often based on inaccurate, incomplete information. Stage 2 of the protocol solves conflicts and distributes resources by initiating a *cooperative mediation* session. During a mediation session, one of the track managers takes on the role of mediator. As the mediator, it gathers information from track managers that are in conflict, computes and recommends possible solutions to the problem, and then announces a final solution.

The SPAM protocol is activated under two conditions. The first condition is occurs when the resources needed to track a target change due to a target's movement. These types of changes often alter the structure of the underlying optimization problem. As such, whenever they occur, managers instantiate the first stage of the protocol to quickly modify their current solution or to create an initial solution in the case of a new target assignment. Once stage 1 completes, a manager can choose to begin a mediation session if they determine that the benefits outweigh the costs.

The second case occurs when a manager detects a conflict within one of the resources its using. This type of conflict is caused when another manager is unable to mediate a session, is able to mediate but due to latency has not yet begun it, or is unaware that the resource is already being used. In each of these cases, the manager detecting the conflict has the option to immediately activate stage 2 and mediate a session to repair it. A distributed locking mechanism prevents more than one manager from concurrently mediating if latency was the cause of the conflict detection.

5.3 Stage 1

Stage 1 of SPAM serves three primary functions. The first function is to attempt to find a solution within the context of the information that the protocol has when it starts up. Like the Asynchronous Weak Commitment (AWC) protocol [12], each of the agents tries to find an assignment that is consistent with its potentially incomplete or inconsistent *agent_view*. However, because this protocol attempts to maximize the social utility, each of the agents tries to maximize their local utility without causing new constraint violations. If this can be done, then no further mediation is necessary, and the protocol terminates at the end of stage 1.

We should mention that a trade-off exists between communication overhead and utility, due to the initial selections of the objective level in stage 1. If each of the managers chooses to use every available resource (sensors able to see their target), the possibility for contention over resources greatly increases in the environment, thereby causing the execution of stage 2 to occur more frequently. However, if the agents decide to start with at a lower objective level (and correspondingly less utility), the social utility may suffer unnecessarily.

Stage 1 has what we refer to as a concession rate. The concession rate defines what percentage of the local solution quality a track manager is willing to concede to find a violation-free solution in an attempt to avoid a potentially expensive stage 2 mediation session. So, as the manager's utility drops, the amount they are willing to concede drops as well. This causes managers to mediate (Stage 2) more frequently in critically constrained tracking environments.

The second function of stage 1 is to ensure some utility is obtained while waiting for stage 2 to complete. Since these temporarily applied solutions are only applied when a completely conflict-free assignment is not possible, unresolved conflict are left to the individual sensor agents to handle. As mentioned in the previous section, sensor agents can use one of a number of techniques, including slot boundary shifting, less expensive measurement types, or task rotation, in order to resolve such conflicts. To the track manager, whether or not they get a measurement from a conflicted sensors is probabilistically random.

Temporarily applied solutions do not use the concession rate. In fact, because of environmental changes and the probabilistic nature of getting measurements from conflicted sensors, managers always use their maximum possible objective level (within the bounds of the number of sensors that can see the target). The reason for this is rather subtle, but important. Let's say that a new resource were added to the possible resources that could be used by manager T1. Let's also say that another manager, T2, who has more than enough available resources to itself, were using that entire resource. If T1 starts a mediation at that lower level, it can never obtain its highest level as a result of the session, even though a solution exists where T2 just gives up the entire conflicted resource.

The third purpose of stage 1 relates to the anytime characteristics of the protocol. Because a solution is always derived and applied during stage 1, managers don't necessarily have to enter stage 2. They can stop the process at the end of stage 1 and accept the results that they have achieved. This is often done if a target's movement causes the resource needs to change faster than the expected time it would take to complete stage 2. The expected time to complete stage 2 is computed based on both previous experience and the current estimated channel loads for the track managers that would be in the mediation session..

5.4 Stage 2

Stage 2 is the heart of the SPAM protocol (See Figure 11). Stage 2 attempts to resolve all of the local conflicts a track manager has by elevating the problem to the track managers that are using the desired resources. To do this, the originating track manager takes the role of the mediator (note that multiple sessions can occur in parallel in the environment). As the mediator, it becomes responsible for gathering all of the information needed to generate alternative

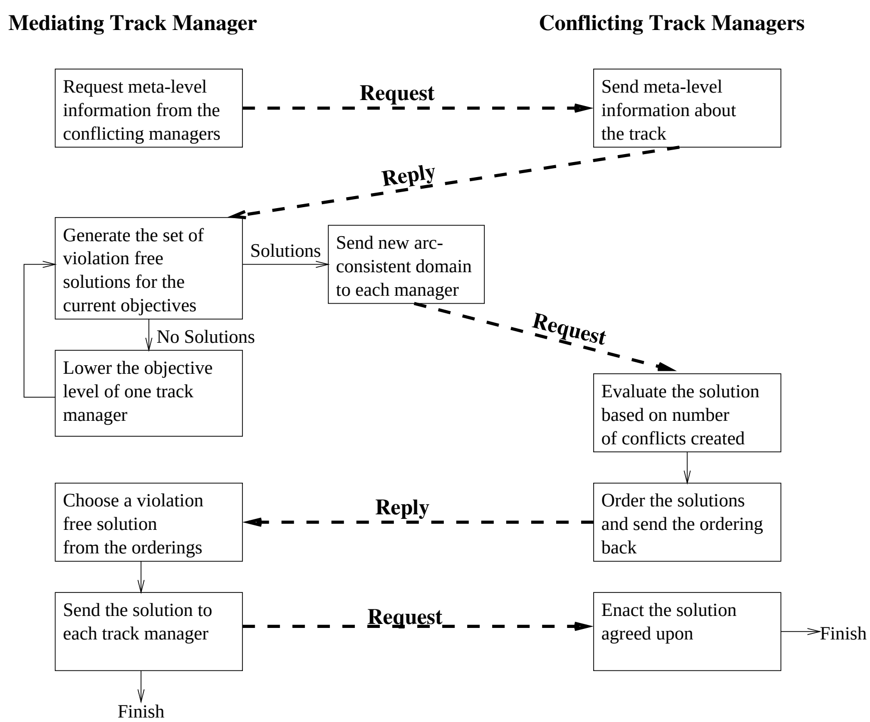

Figure 11: Stage 2 of the SPAM protocol.

solutions, creating solutions which may involve changes to the objective levels of the managers involved, and finally choosing a solution to apply to the problem. These solutions are generated without full global information and may lead to newly introduced non-local conflict. If this occurs, other track managers can begin sessions which propagates the conflict even further.

The best way to explain the operation of stage 2 is through an example. Consider Figure 12, which depicts a commonly encountered form of contention. Here, track manager T2 has just been assigned a target. The target is located between two existing targets that are being tracked by track managers T1 and T3. This creates contention for sensors S3, S4, S5, and S6.

Following the protocol for the example in Figure 12, track manager T2, as the originator of the conflict, takes on the role of mediator. It begins the mediation session by requesting information from each of the track managers involved in the resource conflict. Upon receiving the request, each track managers replies with their current objective level, the number of sensors which can see their target, the names of the sensors that are in direct conflict with the mediator, and any additional conflicts that the manager has. To continue our example, T2 sends a request for information to T1 and T3. T1 and T3 both return that they have 4 sensors that can track their targets, the list of sensors that are in direct conflict (i.e $T1(S_3, S_4)$, $T3(S_5, S_6)$) their objective level (4×3 for both of them) and that they have no additional conflicts outside of the immediate one being considered.

As seen in Figure 11, T2 enters a loop that attempts to generate solutions followed by lowering the objective level of one of the track managers if none exist. A heuristic method, designed to balance the resources, is used to choose the track manager to lower. Namely, the track manager is chosen that has both the highest objective level and is unable to support it without using resources from the set of sensors being mediated over. Whenever two or more managers have the same highest objective level, we choose to lower the objective level of the manager with the least amount of external conflict. By doing this, it is our belief that track managers with more external conflict will maintain higher objective levels, which provides them with leverage in resolving subsequent conflicts that may be occur as a result of propagation.

You should note that although this has similarities to the techniques used in Partial Constraint Satisfaction (PCSP)s, this differs in that the problem changes

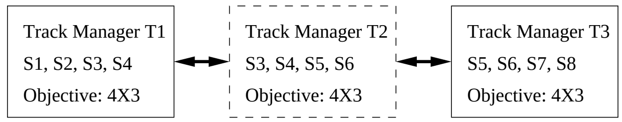

Figure 12: Example of a common contention for resources. Track manager T2 has just been assigned a target and contention is created for sensors S3, S4, S5 and S6.

as the objective levels are changed. PCSP techniques, such as [13] choose to satisfy some subset of the constraints, this technique changes the constraints themselves until the problem is satisfiable.

The solution generation loop is terminated under one of two conditions. First, if given the current objective levels for each of the track mangers, the mediator is able to generate at least one satisfying assignment, the session enters the solution evaluation phase. Second, the session ends if the mediator cannot generate a satisfying assignment and it cannot drop the objective level of one of the track managers in the session. Under these conditions, the session is terminated and the mediator lowers its own objective level to the lowest possible level, conceding that it cannot find a satisfying assignment, and binds a solution which minimizes the number of conflicts.

Continuing our example, T2 first lowers the objective level of T1 (choosing T1 at random because they all have equal external conflict). No satisfying assignments are possible under the new set of objective levels, so the loop continues. It continues, in fact, until each of the track managers has an objective level of 3×2 at which time T2 is able generate a set of 216 satisfying assignments to the problem. More information on the solution generation process can be found in [4].

During solution evaluation, the mediator proposes the set of solutions to each track managers by sending them a list of sensor assignments that occur in at least one solution. Because each of these assignments is part of a solution, each the lists have the important property that they are arc-consistent with the lists being transmitted to the other managers in the session. In addition, the mediating track manager is guaranteed to have a conflict-free sensor assignment whenever it successfully concludes a session.

Upon receiving their list, each of track managers rates the assignments based on their local *agent_view* and their internal utility functions. Continuing our example, T2 sends a list of assignments to T1, a list of assignments to itself, and a list of assignments to T3. In our system, track managers rank the assignments based a utility function that includes the amount of conflict that will be introduced by taking the assignment and on the desirability of the sensors. This is similar to the min-conflict heuristic [14] and is an integral part of the hill-climbing nature of the algorithm.

Once the mediator has the ratings from the track managers, it chooses a particular solution to apply to the problem. This is done using a dynamic priority method based on the number of constraints each of the managers has external to the mediation, a form of meta-level information. The basic notion is similar to the priority order changes in AWC [12]; try to find the task which is most heavily constrained and elevate it in the orders. Our impression is that this helps stem the propagation because it leaves the most constrained tasks with the best choices. This allows those managers to maintain violation-free solutions if they exist in the alternatives presented to them.

In our example, T2 collects the ordering from T1, T2, and T3. T3 is given first choice. By its ordering, it ranked alternative 0 the highest. This restricts the choice for T2 to alternatives 0, 1, 2, and 3. T2 ranked 0 highest from this set of alternatives, restricting T1's choice to its 0th, 1st, and 2nd alternatives. It turns out that T1 likes its 0th solution the best so the final solution is composed of T3's alternative 0, T2's alternative 0, and T1's alternative 0.

The last phase of the protocol is the solution implementation phase. Here, the mediator simply informs each of the track managers of its final choice. Each of the track managers then implements the final solution. At this point, each of the track managers is free to propagate and mediate if it chooses. Figure 13 shows the configuration of the sensors before and after T2 completes stage 2.

This section described the Scalable, Periodic, Anytime Mediation (SPAM) protocol. This protocol uses cooperative mediation to solve a dynamic, distributed optimization problem. The next section, presents results from experimentation that was done in simulation and later verified with hardware.

	Slot 1	Slot 2	Slot 3
S1	T1	T1	T1
S2	T1	T1	T1
S3	T1/T2	T1/T2	T1/T2
S4	T1/T2	T1/T2	T1/T2
S5	T3/T2	T3/T2	T3/T2
S6	T3/T2	T3/T2	T3/T2
S7	T3	T3	T3
S8	T3	T3	T3

	Slot 1	Slot 2	Slot 3
S1		T1	T1
S2		T1	T1
S3	T2		T1
S4	T2	T1	T2
S5	T2		T2
S6	T3	T3	T2
S7	T3	T3	
S8	T3	T3	

Figure 13: A solution derived by SPAM to the problem in Figure 12. The table on the left is before track manager T2 mediates over T1 and T3. The table on the left is the result of stage 2.

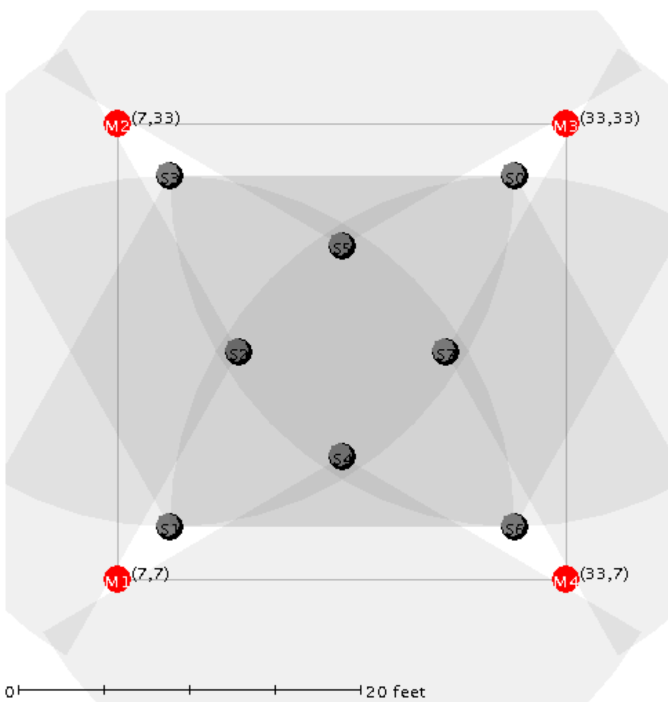

Figure 14: Radsim environment with 8 sensor nodes (S1-S8) and 4 targets (M1-M4).

6 Results

6.1 Resource Allocation

A distributed sensor network environment lends itself to many different types of metrics. The most obvious is the accuracy of the tracking process, as determining accurate positions for the various targets is arguably the point of the system. In a conventional, unified system this would be the case, however, because the tracking component of this solution was a black box developed by a third party, we instead focus our attention on the number and relative coordination of the measurements being taken for each of the targets.

To test the effectiveness of SRTA and SPAM, we used a simulation tool, called Radsim (see Lawton in [3]), that very closely models the target tracking environment and sensor hardware. Radsim simulates the behavior of one or more targets that have defined paths through the environment. Agents connect to Radsim and use a standard API to send control signals to their respective sensor platforms. Radsim, in turn, uses a model of the platforms' behavior to return, in the case of taking a measurement, a hypothetical value for amplitude and frequency based on the current locations of the targets.

We conducted 400 experimental runs, each 3 minutes long, using the environment in Figure 14 consisting of 8 sensors and between 1 and 4 mobile targets. The paths of the targets and the layout of the sensors were chosen to maximize contention with minimal possibility of target ambiguity, thus creating a need for proper sensor allocation to effectively track. We analyzed three resource management techniques, comparing them based on their average periodic utility (see section 5.1) during the course of the scenario.

The first and simplest allocation process, single commitment, specifies that each sensor can work on only one commitment at a time. New commitments accepted by a sensor override any existing ones. In the second set of experiments, agents use SRTA to work on multiple commitments, but only local conflict resolution strategies are employed. In this case, the periodic task controller is responsible for managing conflicted commitments as best it can. The third strategy uses the SPAM + SRTA allocation style which represents the full solution presented earlier in this article. Commitments are generated, and existing initial conflicts are resolved locally using SRTA. SPAM then attempts to resolve these conflicts through more intelligent allocation of the sensor resources.

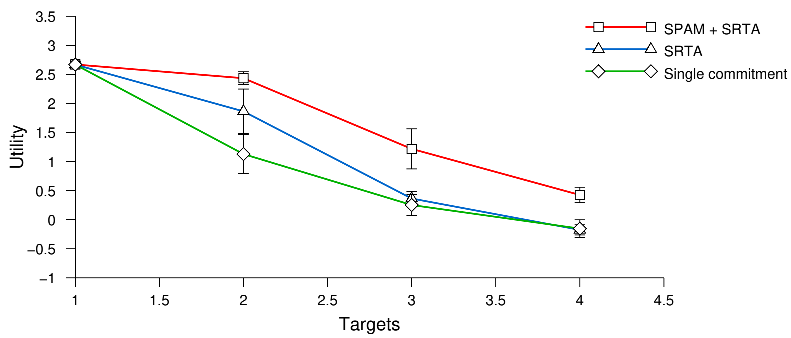

Figure 15: Average utility for a period for three allocation techniques under various conditions.

As seen in Figure 15, when there is only one target, all three of the methods do equally well, because there is no contention for the sensors. The score of about 2.6 stems from the fact that the target, over the course of its movement, enters regions where sensor coverage is limited to just 2 sensors. During these periods, the score becomes -1, lowering the overall average for the track.

As the number of targets increase, resource contention becomes more significantand the differences between the methods becomes clear. For two targets, the single commitment strategy essentially ignores one of the targets whenever contention for the sensors occurs. The SRTA-only strategy has better performance than the single commitment strategy because, although it does not produce a conflict free solution, the local agents are able to occasionally get coordinated measurements. Because this happens essentially as a random event, however, this strategy results in a relatively high standard deviation among scenarios. The SPAM + SRTA strategy clearly does better on average and also has a very small standard deviation. This indicates that in most instances, SPAM is able to resolve the conflict and maintain coordination between the sensors.

As the number of targets increases above two, the differences become more apparent. By four targets (which is highly over constrained), the SRTA only method does no better than single commitment. This is caused by the effects of random coordination of measurements from the sensors causing one or more of the targets to be ignored for most of the scenario. SPAM + SRTA also degrades but the average stays very close to 1 indicating that most of the time, all of the

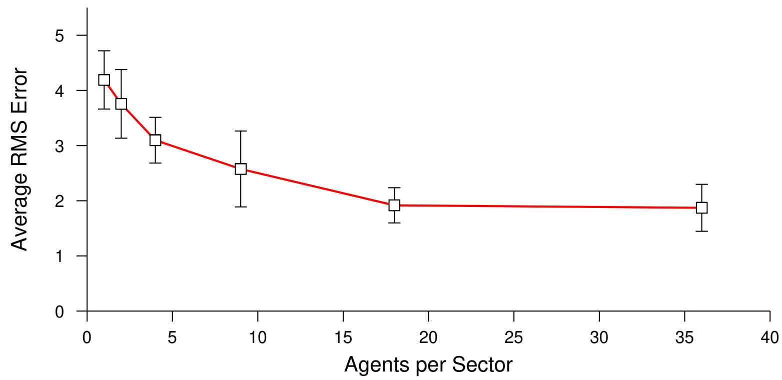

Figure 16: The effects of sector size on tracking quality with ideal communications.

targets are being tracked to some degree.

6.2 Effects of Sector Size

We also conducted a number tests to understand the effect sector sizes on tracking accuracy. In these experiments, a group of 36 sensors were organized into between 1 and 36 equal-sized sectors. Four targets were introduced into the environment and the average tracking accuracy was measured for fixed duration of time. These tests were conducted using ideal communication conditions, i.e. no message loss or congestion effects. As you can see in Figure 16, as you increase the number of agents within a single sector, the RMS error and its variance goes down. As can been seen in Figure 17 this effect can be partially attributed to an increase in the number of sensor measurements being received by the track managers and partially caused by the a reduced delay in receiving sensor information from the sector manager as the target moves through the environment. Again, these results do not take into account the effects of message congestion. If these tests had been conducted using the RF-based system, the effects of centralizing too much information and processing would have become apparent as the number of agents per sector increases. As seen in figure 4, his can cause excessive load in particular agents.

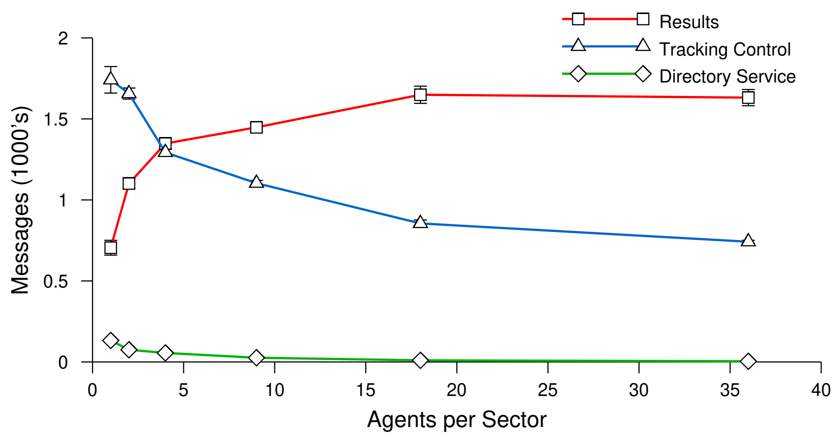

Figure 17: The effects of sector size on various messages being communicated.

Figure 18: Researchers work on a demonstration involving 36 sensors and 3 mobile targets.

6.3 Hardware Evaluation

A fair amount of testing was also done during several hardware-based evaluations (see Figure 18). In these evaluations, sensors were placed in an environment which had a fixed number of model trains, acting as the targets, moving through it. The goal of the evaluation was to track the trains with the highest possible quality while minimizing the amount of communication being used. The results of these tests were mixed. Although our system proved to be very robust, it had difficulty maintaining cohesive tracks for long periods of time. There are a number of factors which caused this to occur. At the lowest level, the raw sensor data proved to be quite poor due to the effects of multi-pathing, poor calibration, and equipment failure. These low-level difficulties caused false target detections and large amounts of uncertainty in estimating the targets' locations and velocities. Compounding the problem is the closed loop nature of target tracking. Since location estimates are used to select sensors, poor estimates lead to improper selection which potentially reduces the quality of the next estimate due to unnecessary increases in resource contention and target association problems. In the end, the track managers lost their targets which were later re-detected and tracked again. Post-processing of the data collected during these experiments showed the agents performed the correct actions given the projections they were provided by the underlying tracking components. This supports the claim that the results derived in simulation are represenative of the behavior we would expect with components that provided more reliable data.

7 Conclusions

In this paper we have described our solution to a real-time distributed tracking problem. The agents in the environment are first organized by partitioning them into sectors, reducing the level of potential interaction between agents. Within each sector, agents dynamically specialize to address scanning, tracking, or other goals, which are instantiated as task structures for use by the SRTA control architecture. These elements exist to support resource allocation, which is directly effected through the use of the SPAM negotiation protocol. The agent problem solving component first discovers and generates commitments for sensors to use for gathering data, then determines if conflicts exist with that allocation, finally using arbitration and relaxation strategies to resolve such conflicts. We have empirically tested and evaluated these techniques in both the Radsim simulation environment and using a hardware-based system.

Despite the fact that many of the details of our solution were designed for the distributed sensor net problem, much of the higher-level architecture is quite general, and applicable to different problems. SRTA, for instance, uses the domain-independent TÆMSlanguage as its basis, which can and has been used successfully in a variety of domains. The SPAM negotiation protocol can be used to solve new distributed, interdependent resource allocation problems by implementing a suitable objective function. SPAM's technique of allowing conflicts to exist and be resolved by local control concurrent with a more complete allocation search can be used in nearly any environment where the participants are tolerant of such uncertainty. Our organizational structure as a whole is quite specific, but individual aspects such as partitioning, task migration and local control are general and applicable to a variety of different distributed architectures.

Our solution as described covers many different aspects of the distributed sensor interpretation problem including uncertainty, target association, and resource allocation. Despite this, there remain parts of the domain which we have not yet explored that seem to offer additional possibility for intellectual contribution. For example, while the sensors in this environment had several different modes of operation with different execution characteristics, in practice it rarely if ever made sense to use anything but the cheapest, fastest measurement type. In addition, the sensor population as a whole was discriminated only by location and orientation, and not by differing capabilities or qualities. If the sensors were more heterogeneous, or if they offered a range of useful modes of operation, it would offer the opportunity for a richer reasoning process. Agents would need to determine not only which sensors to use, but which modes they should operate in, and which functionalities should be exploited, and to trade off these choices against the additional costs they would likely incur. Related to sensor heterogeneity, the ability discriminate among targets also presents a new dimension in which to reason. If targets were identifiable, and correlated with either known characteristics or expected routes, this information would allow agents the possibility of a more effective tracking procedure by exploiting such knowledge. As mentioned earlier, the notion of distance-attenuated communication could also create an interesting environment, requiring agents to more directly reason about the consequences of long-distance agent relationships.

Acknowledgment

We would like to acknowledge the following people for their important contributions to this research. Jiaying Shen and Kyle Rawlins for implementing the Periodic Task Controller (PTC), Tom Wagner for supporting our scheduling efforts, Tim Middelkoop for evaluating hardware experiments, Raphen Becker for implementing track visualization tools, and Kenneth Parker, Alex Meng and Harry Hogenkamp for their time and expertise preparing for hardware demonstrations. We would also like to thank Stephen Fitzpatrick for his photo of the hardware evalulation.

References

[1] C.-Y. Wan, A. Campbell, and L. Krishnamurthy, “Psfq: A reliable transport protocol for wireless sensor networks,” in *Proceedings of the First Workshop on Sensor Networks and Application (WSNA)*, Atlanta, GA, September 2002.

[2] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, “A two-tier data dissemination model for large-scale wirelesssensor networks,” in *Proceedings of ACM MobiCOM*, Los Angeles, CA, 2002.

[3] V. R. Lesser, C. L. Ortiz, and M. Tambe, Eds., *Distributed Sensor Networks: A Multiagent Perspective.* Kluwer Academic Publishers, 2003.

[4] R. Mailler, V. Lesser, and B. Horling, “Cooperative Negotiation for Soft Real-Time Distributed Resource Allocation,” in *Proceedings of Second International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2003).* Melbourne, AUS: ACM Press, July 2003, pp. 576–583. [Online]. Available: http://mas.cs.umass.edu/paper/241

[5] B. Horling, V. Lesser, R. Vincent, and T. Wagner, “The Soft Real-Time Agent Control Architecture,” *Proceedings of the AAAI/KDD/UAI-2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems*, July 2002. [Online]. Available: http://mas.cs.umass.edu/paper/221

[6] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration for tracking applications," *IEEE Signal Processing Magazine*, 2002.

[7] D. Li, K.Wong, Y. Hu, and A. Sayeed, "Detection, classification, tracking of targets in micro-sensor networks," *IEEE Signal Processing Magazine*, pp. 17–29, March 2002.

[8] R.Brooks, P. Ramanathan, , and A. Sayeed, "Distributed target classification and tracking in sensor networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1163–1171, 2003.

[9] M. Sims, C. Goldman, and V. Lesser, "Self-Organization through Bottom-up Coalition Formation," in *Proceedings of Second International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2003)*, vol. AAMAS03. Melbourne, AUS: ACM Press, July 2003, pp. 867–874. [Online]. Available: http://mas.cs.umass.edu/paper/238

[10] K. S. Decker and V. R. Lesser, "Quantitative modeling of complex environments," *International Journal of Intelligent Systems in Accounting, Finance, and Management*, vol. 2, no. 4, pp. 215–234, Dec. 1993, special issue on "Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior".

[11] T. Wagner, A. Garvey, and V. Lesser, "Criteria-Directed Heuristic Task Scheduling," *International Journal of Approximate Reasoning, Special Issue on Scheduling*, vol. 19, no. 1-2, pp. 91–118, 1998, a version also available as UMASS CS TR-97-59.

[12] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara, "The distributed constraint satisfaction problem: Formalization and algorithms," *Knowledge and Data Engineering*, vol. 10, no. 5, pp. 673–685, 1998.

[13] E. C. Freuder and R. J. Wallace, "Partial constraint satisfaction," *Artificial Intelligence*, vol. 58, no. 1–3, pp. 21–70, 1992.

[14] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird, "Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems," *Artificial Intelligence*, vol. 58, no. 1-3, pp. 161–205, 1992.

A Figure Captions

Figure 1: High-level architecture. A: sectorization of the environment, B: distribution of the scan schedule, C: communicating over tracking measurements, and D: fusion of tracking data.

Figure 2: The sensor platforms have three Doppler radar sensor heads and are controlled by a BASIC stamp micro-controller.

Figure 3: Total number of messages for 36 agents tracking 4 targets given various sector sizes.

Figure 4: Communication load disparity among agents within each sector.

Figure 5: Overview of the agent's organizational hierarchy, with some information flows represented.

Figure 6: An abstraction of the messages and reasoning used for target detection by sensor agents, sector and track managers.

Figure 7: As the target moves to another sector, the track manager queries the manager of the new sector.

Figure 8: The soft real-time control architecture.

Figure 9: An abbreviated view of the sensor initialization TÆMS task structure.

Figure 10: Utility of taking a single, coordinated measurement from a set of sensors.

Figure 11: Stage 2 of the SPAM protocol.

Figure 12: Example of a common contention for resources. Track manager T2 has just been assigned a target and contention is created for sensors S3, S4, S5 and S6.

Figure 13: A solution derived by SPAM to the problem in Figure 12. The table on the left is before track manager T2 mediates over T1 and T3. The table on the left is the result of stage 2.

Figure 14: Radsim environment with 8 sensor nodes (S1-S8) and 4 targets (M1-M4).

Figure 15: Average utility for a period for three allocation techniques under various conditions.

Figure 16: The effects of sector size on tracking quality with ideal communications.

Figure 17: The effects of sector size on various messages being communicated.

Figure 18: Researchers work on a demonstration involving 36 sensors and 3 mobile targets.

B Notes

B.1 Manuscript received.....

B.2 Author affiliations

- Roger Mailler, Department of Computer Science, University of Massachusetts, 140 Governors Drive, Amherst, MA 01003.

- Bryan Horling, Department of Computer Science, University of Massachusetts, 140 Governors Drive, Amherst, MA 01003.

- Victor Lesser, Department of Computer Science, University of Massachusetts, 140 Governors Drive, Amherst, MA 01003.

- Régis Vincent, SRI International, Room EJ274, 333 Ravenswood Avenue, Menlo Park, CA 94025-3493.

B.3 Sponsorship

Effort sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory Air Force Materiel Command, USAF, under agreements number F30602-99-2-0525 and DOD DABT63-99-1-0004. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. This material is also based upon work supported by the National Science Foundation under Grant No. IIS-9812755. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), Air Force Research Laboratory or the U.S. Government.

B.4 Footnotes

Footnote 1: A limited broadcast capability does exist, which can reach all sensors listening on a single channel. It is not possible in this architecture to broadcast a single message to agents which are using different channels.