GLOBAL SENSOR WEB COORDINATION AND CONTROL USING MULTI-AGENT SYSTEMS

By

John S. Kinnebrew

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

May, 2010

Nashville, Tennessee

Approved:

Professor Gautam Biswas

Professor Julie A. Adams

Professor Lawrence W. Dowdy

Professor Douglas C. Schmidt

Professor Quan Wen

To my parents for their unconditional love and guidance

and

To Jeanna for her unending support and patience

ACKNOWLEDGMENTS

There are many people to whom I am grateful for guidance, support, and encouragement during my graduate education at Vanderbilt University. Without their help, this dissertation would not have been possible.

First, I would like to thank Professor Gautam Biswas, my advisor, for the opportunity to work with him at the Institute for Software Integrated Systems at Vanderbilt University. During my graduate education, Gautam has been a continuous source of encouragement, guidance, support, and inspiration, for which I will always be grateful. Next, I would like to thank my dissertation committee: Professor Douglas C. Schmidt for all his guidance and collaboration, Professor Lawrence W. Dowdy for all of the stimulating discussions and words of encouragement, Professor Julie A. Adams for the numerous suggestions and help in furthering my research, and Professor Quan Wen for providing a different and enlightening perspective on my research.

I would also like to thank the other individuals and organizations who have supported my research: Dipa Suri and Adam S. Howell at Lockheed Martin Advanced Technology Center, Palo Alto, CA, for all their work in collaborating with us on the MACRO project and for being great hosts during our stays in Palo Alto; Matt Heavner and the SEAMONSTER team at the University of Alaska Southeast for providing much-needed domain information and collaboration, as well as for their excellent hospitality during our visits to Juneau, AK; and the NASA Advanced Information Systems Technology (AIST) Program for their support of this research under grants NNA04AA69C and NNX06AG97G.

Over the past five years, I have also benefited from discussions and friendship with many other students at Vanderbilt University. I would like to thank Daniel L. C. Mack for the fruitful and enjoyable discussions and his collaboration on SA-POP. I would also like to thank Benjamin Podgursky for his contributions to SA-POP. I am thankful to Nishanth Shankaran and William R. Otte for their help, collaboration, and instruction on the

middleware, and for being wonderful traveling companions on our many trips to Palo Alto and Juneau. I would also like to thank the other members of the DOC group at Vanderbilt University for their work on the middleware and their willingness to provide help and support.

Last, but certainly not least, I would like to thank my family: my parents, George and Cynthia, for their love, support, and guidance throughout my life and my wife, Jeanna, for all her love, encouragement, and support every day. Without them none of this would have been possible.

John S. Kinnebrew

Vanderbilt University

31 March 2010

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The availability of a variety of sensors around the globe and the ability to quickly make this data available in remote locations give today's scientists an unprecedented advantage in studying and predicting weather, natural disasters, and climate change. Unfortunately this wealth of sensor data is often difficult to locate, access, analyze, and integrate with other data to gain more precise and accurate understanding of earth science issues. Additionally, new sensors are being added at a prodigious rate, further confounding attempts to identify and use the most appropriate sensors to accurately answer particular earth science questions. Technology has advanced to the point that these sensors are also increasingly configurable in terms of measurements, data rate, and sometimes location (*e.g.*, for remote sensing and mobile platforms). Naturally, selecting and coordinating an appropriate subset of these heterogeneous and distributed sensors for large-scale tasks (*e.g.*, studying/predicting natural disasters and climate change) is a complex task.

For example, sensors must first be located and may have to be reconfigured and recalibrated to collect the needed data. Moreover, changing local conditions (*e.g.*, increased cloud cover reducing available power from solar panels) and conflicting goals (*e.g.*, maintaining power reserves versus collecting data at a high rate) require local system adaptation for efficient use of available resources. Effective, timely adaptation requires intelligent reasoning for trade-offs among goals and reconfiguration in response to rapidly evolving situations. However, even intelligent reasoning with a local view of conditions, goals, and tasks is insufficient when there are dependencies between distributed resources for achieving a complex task like natural disaster prediction/tracking.

Coordination among these distributed sensor and computational resources is required for efficient and effective execution of complex tasks. For these reasons and more, NASA's

Earth Science Vision calls for the design and implementation of a global sensor web to research and resolve a variety of Earth science issues [53]. Figure 1 illustrates a global sensor web as envisioned by NASA [78] that is composed of constituent sensor networks, possibly incorporated into smaller sensor webs, designed for a variety of environments and science missions. Further, the sensor networks comprising the global sensor web would employ a wide variety of hardware and software platforms with on-board information processing [53]. The global sensor web vision includes orchestrating real-time collaborative operations among these heterogeneous platforms and computing facilities [53].

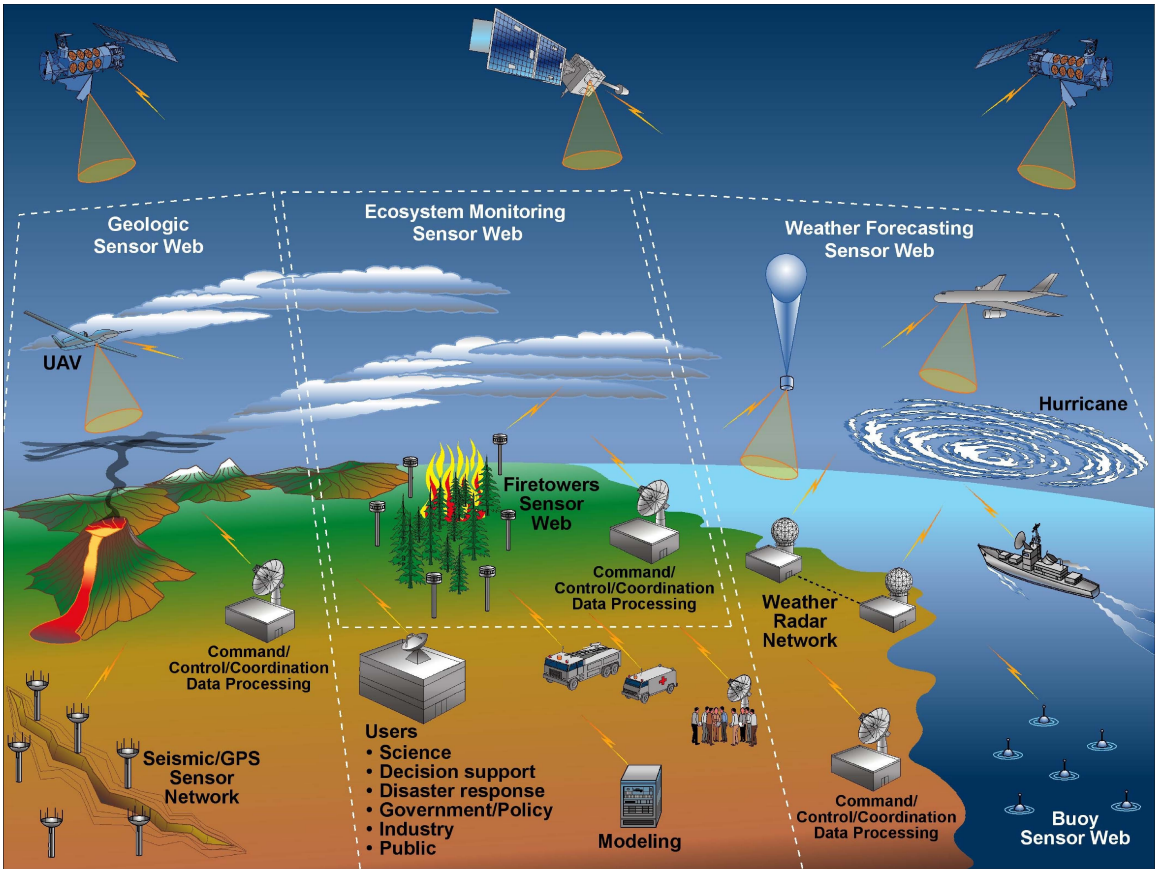

Figure 1: A global sensor web [78]

Effective, collaborative operation of a large, diverse system like a global sensor web presents many challenges across a wide array of fields. In this dissertation research, we will

address the autonomous coordination and distributed control needs of a global sensor web through the design of a multi-agent framework, coordination mechanisms, and autonomous adaptation mechanisms. The proposed framework spans the scope of a sensor web from high-level coordination among distributed (*i.e.*, geographically dispersed) sensor networks to intelligent control of individual sensor platforms in the field.

Major requirements of a multi-agent framework for autonomous coordination and distributed control of a global sensor web are:

- definition of autonomous agent roles/responsibilities, organization, and supporting infrastructure;
- efficient, fair allocation of sensor web resources to address many tasks simultaneously;
- global, task-oriented planning, scheduling, and distributed coordination for sensor and computing resources;
- local, decision-theoretic planning and scheduling for adaptability in a dynamic, uncertain environment;
- system integration, including interoperability of task/resource representations across sensor network domains and a coordination protocol between the global and local representations of task plans and schedules.

I.1 Sensor Webs and Multi-Agent Systems

The vast array of heterogeneous sensors and platforms making up a global sensor web presents a complex problem for effective control and cooperative operation in a highly distributed environment. One aspect of this problem is that the sensor web's resources (*e.g.*, sensors, servers, bandwidth) are not owned or controlled by any single entity. Various institutions, governments, and corporations will have the final say on how their resources

are deployed and used. Even assuming these entities are willing to participate in a collaborative sensor web, they will certainly have internal goals and tasks that can sometimes conflict with requests for use of their resources from other sensor web participants or users.

Further, a global sensor web will have many independent, heterogeneous "users" (*e.g.*, weather modeling and prediction systems, disaster recognition and management systems, scientists, educators, political groups, and members of the public interested in tracking natural phenomena) requesting access to, and control of, the sensor platforms to support their research and analysis activities. The finite resources available in a sensor web ensure that, at times, the goals of these users will compete for control of those resources. A multi-agent system (MAS) provides a natural approach to handling the complexity of distributed resource allocation and task coordination in such a large-scale system composed of heterogeneous, independent entities.

A multi-agent system is composed of independent, intelligent agents, which, in the case of a sensor web, can represent the interests of the participants and the users. These agents interact through communication of local beliefs and negotiation to achieve their goals. Properly designed protocols for agent communication and negotiation facilitate effective global behavior in the pursuit of system objectives. Further, individual agents employ intelligent reasoning capabilities necessary for semi-autonomous behavior that accurately represents the interests of their owner(s). In a sensor web MAS, these reasoning capabilities allow sensor networks to adapt the use of local resources to achieve current goals, under changing system and environmental conditions. Intelligent agent technology can allow the system to respond quickly to changing local conditions, as well as external user requests, with minimal human input (*e.g.*, high-level guidance in the form of local sensor network goals, tasks, and constraints).

In addition to supporting the independence and adaptability needs of a global sensor web, a MAS allows the complex problem of sensor web control to be addressed in a distributed fashion. Coordination/negotiation protocols defining inter-agent communications,

coupled with the planning and decision-making activities of individual agents, are, in effect, a complex distributed algorithm for sensor web control. Solving the control problem in such a distributed fashion can make use of relatively small amounts of computational power available to many, distributed agents rather than requiring the addition of a massive amount of computational power for a single, centralized solution to the problem. Together, the heterogeneous make-up, independent control of resources, need for dynamic semi-autonomous adaptation, and availability of distributed computational power in a sensor web make a multi-agent system a natural and promising solution for global sensor web coordination and control.

I.2 Sensor Web MAS Requirements

Although a multi-agent system provides a powerful solution framework for sensor web coordination and control, the design of its components, protocols, and functionality is a large and complex undertaking. To illustrate the major coordination and control requirements of a sensor web MAS, consider a weather modeling application designed to predict and track hurricanes. In its nominal configuration, *i.e.*, in the absence of extreme weather conditions likely to result in hurricanes, the application requires relatively low-resolution, aggregated data from a large number of sensors distributed around the world. For simplicity, we will assume that this data is always being collected and available during the normal operation of the sensor web.

Analysis of the data over a period of time may indicate that the thunderstorm activity in a particular area is likely to evolve into a hurricane in the near future. At this point, the application must identify additional resource types and/or sensor configurations that will lead to richer data collection for more accurate analysis and prediction. Further, these resources must be tasked and configured to provide the requested information. Therefore, control of the relevant resources must be granted to the hurricane application rather than other applications/users whose tasks are of lower priority.

Reconfiguration, data collection, and pertinent information extraction by the applicable sensor resources may take different amounts of time and could be performed in multiple ways to enable detection and accurate tracking of the evolving hurricane. However, effective use of these individual resources for the overall hurricane tracking task is not independent of one another (*e.g.*, the hurricane application may require synchronized, or otherwise compatible, data rates from two sets of sensors and require that data from both be provided for analysis within specified time deadlines). Therefore, to achieve the desired analysis in a timely fashion with distributed, independently-controlled resources requires coordination among the representatives of those resources to plan and schedule related tasks. Having identified, tasked, and configured an appropriate set of sensor resources, the application can receive and process the additional information from the sensor web to produce more accurate predictions and tracking of relevant phenomena.

This scenario illustrates three required capabilities of the sensor web MAS for the hurricane prediction/tracking application: (1) task/resource allocation, (2) global, coordinated planning/scheduling, and (3) local, dynamic planning/scheduling. Figure 2 illustrates the major coordination and control steps in the hurricane tracking scenario with the related capabilities and general MAS issues. First, the sensor web must allocate sensor resources to the hurricane application that are capable of producing the necessary data. This is the *task/resource allocation* capability, whose steps include identifying applicable sensor resources and negotiating the use of an appropriate subset of those resources. Second, the agents representing allocated resources must coordinate their planned activity in order to achieve the overall hurricane tracking task on an acceptable schedule. This is the *global, coordinated planning/scheduling* capability that consists of: (1) high-level planning and scheduling of subtasks by individual agents, and (2) coordination of plans/schedules to ensure that individual agent plans obey inter-agent constraints imposed by the overall hurricane task. Finally, the servers and individual sensor platforms in a sensor network must

create a detailed plan and schedule for execution of their local subtasks, replanning and rescheduling as necessary. This is the *local, dynamic planning/scheduling* capability.

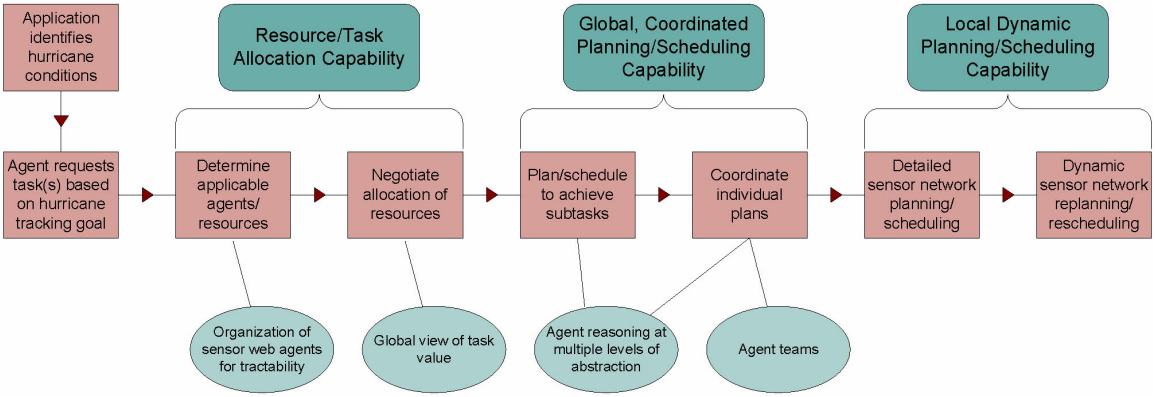

Figure 2: Key steps and capabilities for the hurricane tracking scenario

Although this scenario and Figure 2 present the major coordination and control steps as a simplified task flow, they are actually closely intertwined, and often do not follow such a linear progression. For example, in order to allocate sensor web resources, some planning and scheduling is required to determine the resources a given task may require. The overlap and interactions among the described capabilities suggest two more major requirements for a global sensor web MAS. First, to consistently and efficiently provide these capabilities requires the design of an appropriate *multi-agent architecture*, including agent roles, organization, and supporting infrastructure. Second, the interactions among coordination protocols and translations between different representational formats requires significant *system integration* work to weld these capabilities into a functioning sensor web system.

The global-level coordination and control requirements (*i.e.*, task allocation and coordinated planning/scheduling) are complex, and they raise a number of issues in the design of an appropriate multi-agent architecture. As illustrated in Figure 2, the resource/task

allocation capability involves two steps: (1) matching the hurricane tracking task to an appropriate set of agents capable of providing required sensor data and processing/analysis in a timely manner, and (2) allocating agent (sensor network) resources to that task. Given the global scope of the sensor web, there will likely be other users of the system that are also requesting, or already allocated, resources from the set of agents applicable to the hurricane task. While some resources may be simultaneously allocated to multiple users (*e.g.*, providing a series of measurements from a particular sensor at the same rate), others may be mutually exclusive (*e.g.*, using satellite-based sensors to monitor one location versus another or limited power requiring the use of one sensor versus another).

This aspect of the allocation problem also illustrates two architecture issues that must be resolved in the design of a sensor web MAS. Broadcasting the hurricane tracking task to all agents is infeasible at the global scale of the sensor web. Therefore, a streamlined organization of the agents for solving this problem is an important aspect of the system design. For example, broker agents may be used to mediate allocation of sensor web resources and reduce the number of required messages between agents. Also, in order to commit to fulfilling all or part of the requested task, the agents must reason about their existing goals and commitments in light of current conditions and the newly requested hurricane task. Providing the appropriate global information to agents for comparison of potential tasks is a non-trivial part of ensuring an efficient and effective solution to the allocation problem.

After an initial allocation of resources by a set of agents to the high-level hurricane task, those agents must, in effect, produce a distributed plan and schedule for achieving the task that takes into account available local resources, as well as dependencies among their subtasks. In fact, at least some of the initial planning and scheduling will have to be carried out in determining a valid allocation. For example, an agent should reasonably determine whether it can achieve a task/subtask before committing to it.

The global planning, scheduling, and coordination requirement also highlights two other important aspects of a sensor web MAS. Because the agents will, in effect, act as

a group, or team, to fulfill the broad hurricane task, their organization, individual roles, and coordination is again an important issue for efficient and effective use of the system. In this case, there is the added complication that their organization and roles will necessarily be dynamic as different, probably overlapping, sets of agents will be working on different tasks over time. Further, in order to efficiently coordinate plans and schedules, agents must communicate and reason over multiple levels of abstraction. For example, to successfully achieve a high-level goal, an agent must be able to produce a detailed, scheduled plan of action. However, it must also be able to efficiently coordinate with other agents, which will likely require exchanging information largely at a higher level of abstraction.

I.3 Research Challenges

The scope of a global sensor web requires an open, efficient, scalable system to allocate, plan/schedule, and coordinate the operation of its many heterogeneous, independent components in a dynamic, effective, and equitable manner. This work focuses on a number of specific research challenges for providing these coordination and control capabilities.

Section I.2 illustrated the need for fair and efficient allocation of complex tasks in the global sensor web. The global task/resource allocation requirement presents three significant challenges (detailed in Section III.2):

- definition of a metric for sensor web task/resource allocations that combines fairness and efficiency;
- design of an allocation mechanism through which system designers can influence allocations for fair and efficient use of resources, while limiting infrastructure computational overhead; and
- efficient, effective subcontracting for complex, hierarchically-decomposable tasks.

In addition to task allocation, a global sensor web requires autonomous planning and

scheduling capabilities. In Section I.2 we described the need for coordinated planning/-scheduling at the global level of interaction across sensor networks. Section IV.1 describes the MACRO solution to this global planning/scheduling requirement. Further, we identified the local, dynamic planning/scheduling requirement for adaptive operation of individual sensor networks. This requirement presents three significant challenges (detailed in Section IV.3):

- the design and application of an efficient heuristic to guide the planning process in the search for valid, high expected utility plans;
- the efficient integration of an appropriate scheduling mechanism with the planning process; and
- representing and producing plans that include both execution of traditional, individual actions and assembly/deployment of applications comprising configured software components.

For effective coordination and control of a global sensor web, the task allocation and planning/scheduling capabilities must work together in a complete, coherent system. Integration of these sensor web capabilities presents three significant challenges (detailed in Section V.2):

- definition of a flexible, extensible agent internal architecture that can effectively coordinate its allocation, planning/scheduling, plan/schedule coordination, and task execution activities;
- translation between task representation formats, including aggregation of domain information across independently-designed sensor networks; and
- efficient coordination and translation between the different planning and scheduling mechanisms/representations at the global and local levels.

I.4 Research Approach and Contributions

The research conducted for this dissertation has resulted in the design and implementation of agents and coordination mechanisms in the Multi-agent Architecture for Coordinated Responsive Observations (MACRO), which resolves the challenges identified in Section I.3. This work has produced a number of research contributions in the fields of multi-agent systems and autonomous planning and scheduling.

MACRO provides a metric for task allocation combining fairness (*i.e.*, user satisfaction) and efficiency (*i.e.*, system value) considerations that does not require parameter tweaking for different system configurations (*e.g.*, system load and relative importance of user agents). MACRO also provides a distributed mechanism for the fair and efficient allocation of hierarchically-decomposable tasks. Specifically, we enhance a two-phase contract net protocol with efficient subcontracting and broker agents for mediating negotiations and evaluating user tasks. Experimental results in Section III.6.1 verify the efficiency of MACRO's subcontracting for hierarchically-decomposable tasks. Further, experimental results in Section III.6.2 verify MACRO performance in producing fair and efficient task allocations. In general, the novel, MACRO task allocation metric and mechanism are suitable to large multi-agent systems with heterogeneous users, hierarchically-decomposable tasks requiring multiple agents' resources, and limited infrastructure computational capabilities.

At the local sensor network level of MACRO, the Spreading Activation Partial Order Planner (SA-POP) provides a decision-theoretic planning and scheduling service for local sensor network agents operating on shared computational resources. Section IV.6 presents experimental results verifying SA-POP's ability to produce plans with near-optimal expected utility under tight scheduling constraints. SA-POP allows MACRO agents operating on shared computational resources to achieve goals in a dynamic, uncertain environment with limited resources. In general, the SA-POP planning and scheduling service is suitable for autonomous action and adaptation of system applications by agents operating in a dynamic, uncertain environment with limited, shared resources.

Further, MACRO resolves system integration challenges for a proof-of-concept implementation integrating the MACRO task allocation and planning/scheduling capabilities. By defining appropriate meta-data for sensor network domain information, MACRO allows aggregation of potential tasks across sensor networks and translation between user and provider task representations. Further, MACRO implements context-sensitive coordination between the different forms and representations of planning/scheduling employed at the global and local levels. This coordination mechanism is applicable to agents in a hierarchical relationship where the top level employs hierarchical task decomposition and (re)scheduling while the bottom level employs decision-theoretic, first-principles (re)planning and (re)scheduling for adaptation to local system conditions while obeying top-level goals and constraints. Experimental results presented in Section V.7 illustrate some of the communication and computation overhead benefits of this coordination. Finally, we illustrate the integration of MACRO's sensor web coordination and control capabilities in a case study with three simulated sensor networks presented in Section V.6.

I.5 Dissertation Organization

The remainder of this dissertation is organized as follows: Chapter II gives an overview of the Multi-agent Architecture for Coordinated Responsive Observations (MACRO), defining agent roles, organization, and services on a middleware infrastructure; Chapter III details MACRO's sensor web metric for fair and efficient task allocation, brokered task auctions, and efficient subcontracting in an enhanced contract net protocol; Chapter IV provides details of the autonomous planning and scheduling in MACRO, including the SA-POP decision-theoretic planning and scheduling service and MACRO's use of a global, distributed planning and scheduling representation and mechanism; Chapter V describes the integration of MACRO coordination and control capabilities, including an extensible agent internal architecture, aggregation of domain information across independently-designed sensor networks, translation between task representation formats, and efficient

coordination between the different planning and scheduling mechanisms/representations at the global and local levels; and Chapter VI provides a summary of the presented work on MACRO, lessons learned, and future research directions.

CHAPTER II

THE MULTI-AGENT ARCHITECTURE FOR COORDINATED RESPONSIVE OBSERVATIONS

As illustrated in Section I.1, a multi-agent system is a powerful solution for sensor web coordination and control. However, its design is a large and complex undertaking. The definition of agent roles, communication, and implementation standards provides the necessary basis for the design and implementation of effective coordination mechanisms and adaptive agent capabilities in any multi-agent system. Naturally, as those specific coordination mechanisms and capabilities have been designed, implemented, and refined, they have also driven revisions to the agent architecture presented in this chapter.

A significant contribution of this research has been the definition and implementation of the Multi-agent Architecture for Coordinated Responsive Observations (MACRO). Ultimately, MACRO provides a powerful computational infrastructure for enabling the deployment and adaptive operation of large, distributed systems, such as a sensor web, that require both high-level coordination of complex tasks across agents and local, dynamic adaptation for effective use of limited resources in dynamic, uncertain environments. This chapter presents the design of the MACRO framework, including agent roles, agent organization, agent services, the underlying system infrastructure, and related work relevant to these aspects and more generally to the coordination mechanisms discussed in subsequent chapters.

II.1 Related Research

Effectively managing global sensor web activities to achieve a diverse set of goals requires a significant degree of coordination among the agents representing individual sensor networks, as well as between these agents and the agents representing external users. In

fact, coordination is generally considered the primary aspect of a multi-agent system distinguishing it from a simple collection of independent agents. Nwana et al. [80] identified five, overlapping, reasons for coordination among agents. All of these points are applicable to global sensor web control in at least some degree:

- *Preventing anarchy or chaos* – Anarchy is an inherent danger in most multi-agent systems because the independent agents are not constrained by any centralized control. In a global sensor web, an agent responsible for a single sensor network is knowledgeable about its current state and goals. However, without coordination to provide information about other agents' situations, intents, and potentially conflicting goals, independent agent action may produce undesirable and ineffective system behavior. In particular, tasks requiring the resources of multiple sensor networks may not be correctly achieved without coordination.

- *Efficiency* – When agents are operating with independent resources and goals, they can generally achieve their goals with no coordination. However, in some applications they can coordinate their activities to achieve their independent goals more efficiently. For example, if data analysis is shifted from an agent with highly utilized computing resources to an under-utilized one at one point, and then the reverse occurs at a later point, both agents are able to achieve their goals more quickly. Further, given limited resources, more efficient use of those resources will allow more tasks to be completed even when it is not possible to complete all tasks.

- *Meeting global constraints* – A successful multi-agent *system* is designed such that the interactions of its agents, as a whole, solve some large, complicated problem(s). Generally there are global constraints and/or metrics for system success, which require coordination among the agents to meet those constraints or optimize for system metrics. Clearly such metrics exist for a sensor web (*e.g.*, fair-share access to

resources and timely completion of tasks), requiring agents to coordinate their activities.

- *Distributed expertise, resources, or information* – A global sensor web, by definition, is made up of many distributed resources, including the sensors and computational power. Reconfiguration of multiple component sensor networks to achieve a user's task may be possible in a large number of ways acceptable to the user, but only a few of those may be acceptable to the relevant sensor networks because of their individual goals and constraints. Without coordination among sensor web agents, individual users would have to indirectly impose coordination, possibly attempting many configurations before finding one acceptable to all relevant sensor networks, in order to effectively use sensor web resources.

- *Dependencies between agents' actions* – Although agents operate independently, their actions may affect the ability of other agents to effectively achieve their goals. For example, two sensor networks may be producing data that a user intends to combine and analyze. If one of the sensor networks is forced to change the rate or time window of data collection due to unexpected events, this may present difficulties for the user's analysis, unless it coordinates with the other sensor network to also change its rate/window. In general, there may be a variety of data quality and synchronization dependencies between subtasks achieved by various sensor networks.

Although preventing anarchy and system efficiency are always important, the latter three of these goals are more specifically applicable to a global sensor web MAS and have been achieved in other MASs in a variety of ways. Nwana et al. [80] categorize the methods for achieving coordination into four, again overlapping, groups: organizational structure, contracting, negotiation, and multi-agent planning. Each have their particular benefits and potential drawbacks, and each are applicable to some of the challenges faced by a global sensor web MAS.

II.1.1 Types of Coordination

A well-defined, comprehensive organizational structure is one of the easiest ways of achieving a basic level of coordination in a multi-agent system. Individual agents have specific roles and responsibilities that determine their relationship and appropriate communication with other agents in the system. Many organizational structures include some degree of power imbalance among agents (*i.e.*, some agents have at least partial authority over other agents) [54]. Such dominance relationships significantly simplify coordination reducing the number of "peers" that must negotiate or otherwise reach mutually agreeable decisions. Other, more peer-oriented, relationships also reduce the complexity of coordination by limiting the number of other agents an individual agent must interact with based on its particular role in the organization [63]. The global scope of a sensor web requires such limitations because if, instead, each of the hundreds (potentially thousands or more) of agents needed to interact with a significant fraction of the others, message and processing time overhead would be too great for effective system operation. On the other hand, the more effectively an organizational structure limits interactions among agents, particularly hierarchical organizations or others with many dominance relationships, the less flexible the agents are in their interactions. Consequently, extremely rigid and limiting organizational structures can preclude many of the benefits of distributed systems, such as concurrency, robustness, and minimal bottlenecks [54].

Another category of coordination techniques is contracting. Contracting generally involves the creation of temporary contracts between agents for the execution of tasks or use of resources. In a sense, a contract prescribes a temporary, well-defined relationship between two or more agents. It provides a powerful method of organizing agents with greater flexibility than static organizational roles. The price of this flexibility is that more complex communication among potentially contracting agents is required to produce the contracts, and additional computing infrastructure may be needed to manage the contracting process. Contracting is often done through some form of auction (*e.g.*, [56, 65, 86, 104]), which can

provide an efficient method of allocating tasks or resources. However, there are numerous flavors of auctions differing in dimensions including how the winner(s) are chosen, the price paid by the winner(s), and whether items may be bundled [117]. Therefore, choosing the appropriate form of auction by considering the characteristics of the items and bidders, as well as which provides the best incentives for achieving global metrics, is a significant challenge that must be addressed based on the particular system to which it is applied. Relevant contracting and auction techniques are covered in more detail as they relate to resource allocation in Section III.1.3.

A related, and much broader, category of coordination techniques is negotiation (*e.g.*, [41, 57, 80]. Similar to, and often encompassing, the economics-inspired auctions for agent contracts, negotiation techniques are often based on economic and game theories, which can provide important information on the potential strategies for different negotiation scenarios and their equilibria and efficiency. Generally, negotiation techniques are more competitive and require less inherently benevolent/cooperative agents than strict organizational structures and some contracting schemes. Allowing for more self-interested agents makes these negotiation techniques particularly useful for open systems in which the benevolence or adherence to cooperative strategies of agents cannot be assumed. On the other hand, negotiation usually requires repeated interactions between agents to reach agreement or even to determine that there is no mutually acceptable agreement. Because of this, open-ended negotiation is likely to be an impractical technique for the primary coordination mechanisms in a global sensor web. However, negotiation techniques can be applied within the confines of other coordination mechanisms. For example, organizational relationships between agents can allow for some degree of negotiation and contract auctions could allow negotiation over contract details rather than simply acceptance or rejection of proposed contracts. A subset of negotiation techniques applicable to resource allocation and contract nets is discussed further in Section III.1.

Finally, the fourth category of coordination techniques identified by Nwana et al. is

multi-agent planning [80]. These techniques perform coordination in terms of a plan distributed among multiple agents (*e.g.*, [29, 30, 32, 110]. They are primarily differentiated from other coordination techniques in that the goal of coordination is the production of a valid, consistent plan distributed among the agents. Interactions among these agents often include communicating details of their local plans and reaching agreement on modifications to their plans to resolve conflicts. The scope of a global sensor web implies that no single agent will have a global view of the system, so significant communications may be necessary to provide agents with sufficient knowledge of others plans. Further, coordination of agents through multi-agent planning is likely to require significant computation and repeated negotiation to resolve conflicts. Despite these costs, multi-agent planning still provides a useful coordination mechanism when the primary goal of coordination is to determine dependencies among agents' planned tasks and resolve related conflicts. Multi-agent planning techniques applicable to a sensor web are discussed in detail in Section IV.1.

II.1.2 Scalability of Coordination Techniques

The variety of available coordination techniques provides a great deal of flexibility in designing multi-agent systems but also requires careful consideration of their advantages and disadvantages in choosing an appropriate coordination scheme for a particular system. Durfee [34] suggests three dimensions in which multi-agent coordination mechanisms are stressed: agent population properties, task/environment properties, and solution properties.

Agent population properties include agent heterogeneity, agent complexity, quantity of agents/agent-interactions. The agent population in a sensor web could take many different forms so we will specifically consider the MACRO agents. Agent heterogeneity is certainly significant at the MACRO mission-level because Mission agents have control of a wide array of different sensor and computing resources, with a similarly wide spectrum of goals. However, the physical distribution inherent in this heterogeneity actually reduces the coordination stress because resources are generally discrete/uniquely-owned and Mission agent

actions rarely affect the outcomes of one another. Agent complexity is also significant in MACRO because the agents have relatively powerful reasoning capabilities and a high degree of autonomy. The relationships imposed by the MACRO organizational structure help to alleviate this stress by obviating the need for most agents to build complex internal representations of each other to interact effectively. Finally, considering the extent of the sensor web from field operations in sensor networks up to the global mission level, the total quantity of agents in a sensor web will be very large. Therefore, the MACRO organization significantly limits the potential number of other agents any agent must interact with at a given time. For example, the hierarchical and federated aspects of the MACRO organization limit the number of peers each agent must interact with, particularly at the resource level. However, at the mission level, any given agent may have to interact with any other agent, depending on the situation, to ensure allocation constraints and plan coordination are satisfied. Generally contract auctions and other economic coordination schemes scale well to large number of agents in terms of generating effective solutions, but limiting communications to applicable/interested agents can be a significant problem. Fortunately, brokers can help reduce the number of mission-level agent interactions for resource allocation, and dynamic goal-oriented teams can significantly reduce interactions for distributed planning.

Task and environment properties are another important dimension of stress in sensor web coordination. Durfee [34] divides this dimension into the degree of interaction, distributivity, and dynamics of tasks/environment. The fact that most actions in the sensor web involve sensing and processing data, rather than affecting their environment, results in a relatively low degree of interaction among actions not related to the same task. Similarly, distributivity does not place a significant stress on sensor web agent coordination, because the physical distribution and heterogeneity of sensor web resources limits which tasks are applicable to which agents. The dynamics of the environment, on the other hand, are a significant stress in a sensor web. Local environmental changes require autonomous adaptation by resource-level agents, which in turn can necessitate modification of mission-level

agent plans and commitments. The coordination stress of such a dynamic environment can be minimized by the use of summary information and abstraction in defining tasks/-subtasks and goals/subgoals at different levels of the system. For example, if a Mission agent directs a resource-level agent to achieve a subgoal rather than execute a specific set of actions to achieve that subgoal, when conditions change, the resource-level agent may be able to achieve the subgoal with a different set of actions without compromising the Mission agent's high-level plan.

The third dimension of coordination stress identified by Durfee is the system solution properties [34]. This dimension includes the quality, robustness, and overhead limitation of system-wide solutions/operation. Sensor webs provide a moderate stress in terms of coordination quality because efficient use of resources and coordination to achieve user goals is important but does not have to be optimal for effective sensor web operation. For example, a multi-agent plan to achieve a particular user goal must meet provided constraints such as data quality and time for completion, but by limiting the number of tasks they accept, the relevant Mission agents may not need to determine the optimal plan for task achievement in order to meet the constraints. Further, while more efficient resource allocation is always preferable, determining the optimal allocation at any given time is not necessary or even reasonable because the dynamically changing user goals and environmental conditions necessitate continuous revisions to agent plans and resource allocation. On the other hand, overhead limitations are a major stress in a sensor web. For example, resource-level agents may have limited bandwidth for communication and may be operating in a soft real-time environment requiring minimal communication and time overhead for effective operation. At the mission level, time overhead to achieve an allocation of resources and multi-agent plans must be kept low because of the dynamic nature of user goals and environmental conditions (*e.g.*, if it takes too long to allocate resources, the time deadlines on user requests may have passed and the set of current requests may have changed significantly).

The scale and scope of a global sensor web produce significant difficulties in achieving

effective multi-agent coordination. As such, the combination of multiple strategies to address the inter-twined requirements of task allocation, global distributed planning/scheduling, and local dynamic planning/scheduling is a logical approach identified in Section I.2. To choose or design appropriate coordination mechanisms requires consideration of how they scale in the face of the characteristics of a global sensor web, particularly:

- quantity of agents,
- dynamic environment,
- time/message overhead.

Through careful consideration of these factors and the interactions among applicable coordination mechanisms, we can create an efficient, scalable system for sensor web coordination and control.

II.2 Unresolved Challenges

Knowledge of existing paradigms and major factors in agent organization and coordination, such as those discussed in Section II.1, provides a solid basis for design of a multi-agent system, such as MACRO. However, it does not provide a ready-made solution to the definition of an effective agent architecture for a complex, distributed system like a global sensor web. In particular, the scope of a global sensor web encompasses the need for both coordination of large-scale, complex tasks across sensor networks and dynamic adaptation for effective use of local resources in dynamic, uncertain environments.

Therefore, one challenge in designing the MACRO framework is defining a set of agent roles and relationships that enable efficient resolution of large-scale coordination problems (*i.e.*, allocation of complex tasks and distributed planning and scheduling for those tasks), as well as local adaptation problems (*i.e.*, autonomous operation with limited resources to achieve local goals in a dynamic, uncertain environment). The MACRO agent roles and organizational structure that resolve this challenge are presented in Section II.3.

Another challenge for the MACRO framework is the need for agent services and a flexible infrastructure to enable efficient and effective implementation of the MACRO agents and sensor web tasks. In particular, MACRO agents operating on shared, local sensor network resources require capabilities for autonomous planning/scheduling and adaptive resource management. Further, to effectively implement and operate MACRO agents and services in a distributed, heterogeneous computing environment (*e.g.*, a variety of hardware and operating system platforms interacting over wireless and wired networks) requires a software layer (*i.e.*, middleware) to automate system configuration, management, and networking tasks. The MACRO agent services and middleware infrastructure that resolve this challenge are presented in Section II.4.

II.3 MACRO Agent Roles and Organization

As illustrated in Section I.1, multi-agent systems provide a promising solution for global sensor web control. As part of a joint project with the Lockheed Martin Advanced Technology Center, we have developed the Multi-agent Architecture for Coordinated, Responsive Observations (MACRO) (*e.g.*, [106, 107]). MACRO is designed to provide a flexible agent architecture and infrastructure for enabling the deployment and operation of a sensor web at both the local sensor net "resource level" and the global sensor web "mission level." Further, its infrastructure is designed to be amenable for use with other potential sensor web software and agents. For example, all agents communicate using messages based on a subset of the FIPA Agent Communication Language [44]. MACRO also employs interoperable sensor and data description standards in the content language of agent messages. Specifically, the OGC SensorML [9] provides a standardized content language for describing sensor capabilities and processing of sensor data, and the OGC Observations and Measurements standard [25] provides a standard format for encoding sensor data. This also allows MACRO to support interoperability with other external tools and systems employing these standards.

In MACRO, there are multiple types of agents, whose roles and responsibilities are partially defined by the organizational structure of the system. As illustrated in Figure 3 there is a broad, two-level hierarchy of control. The top (*mission*) level is comprised of *User agents*, *Mission agents*, and *Broker agents*.

II.3.1 MACRO Mission-Level Agents

The MACRO mission level is comprised of *User agents*, *Mission agents*, and *Broker agents*. User agents provide the high-level tasks/goals to be achieved by the system. The User agents are interfaces to mission scientists or wrappers for earth science applications and legacy systems, such as weather modeling and simulation. The Mission agents achieve these tasks with the resources under their control. Specifically, each Mission agent represents and controls a *sensor net*. This hierarchy is a natural result of the sensor web's structure in which there are many low-level sensor platforms with limited computational resources that are supported and directed by computing facilities with far greater resources. Moreover, this hierarchical structure can take advantage of the efficiency gained through task decomposition and separation of concerns [54]. The disadvantage of a hierarchical structure is the potential for brittleness and bottlenecks [54]. In MACRO, these disadvantages are mitigated by the fact that the top level of the hierarchy includes a large number of Mission, Broker, and User agents rather than a single agent, and this group can be dynamically expanded as additional sensor nets and users are added to the global sensor web.

The Mission agents take on overlapping roles in the allocation and planning aspects of the sensor web, and their interactions with each other and the User agents ultimately provide the global resource allocation and distributed planning/scheduling necessary for coherent operation of the sensor web. Finally, Broker agents are responsible for providing matchmaker services between User agent requests and applicable Mission agents. They also mediate negotiations between User agents and Mission agents. As opposed to most User and Mission agents, Broker agents are implemented by the MACRO system designers,

providing them with a way to directly influence overall system performance. A global sensor web could easily contain thousands of these various mission-level agents, or more. However, given the processing power required for each mission-level agent, we will limit the scope of most experiments and discussion in this work to hundreds of mission-level agents rather than thousands.

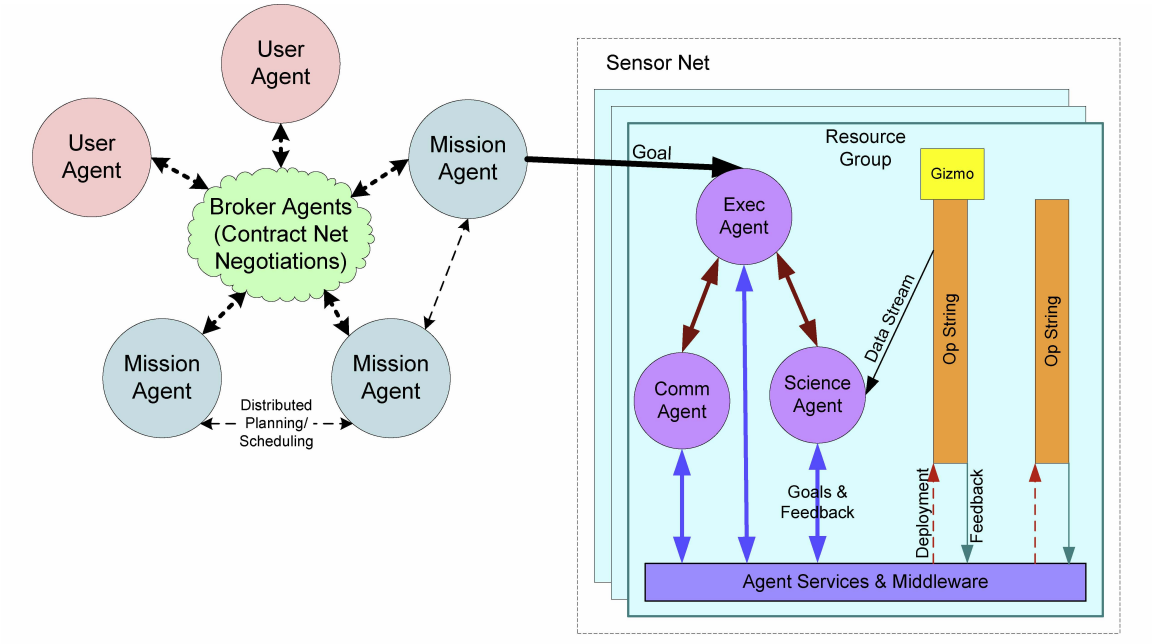

Figure 3: MACRO system architecture

II.3.2 MACRO Resource-Level Agents

The lower (*resource*) level of the MACRO agent hierarchy consists of the agents located on the servers and field hardware making up individual sensor networks. Each sensor net is controlled by a single Mission agent and is composed of individual *resource groups*. A resource group is a natural grouping of computational resources that are directly connected to device nodes (*i.e.* sensors and actuators). For example, a spacecraft may have multiple processors with a range of different device nodes, but it can logically be viewed as a single

set of shared resources. Similarly a cluster of *in situ* ground sensors connected by a bus to one or more processing nodes would be a single resource group. The key characteristic of a resource group is that it is a set of *shared* computational resources. Within the resource group, the resource-level agents are organized by the federation paradigm, with the Exec agent acting as the intermediary between the Mission agent and specialized resource-level agents.

Resource-level agents are defined by the necessary roles and responsibilities of specific resource groups. The particular set of agents used in a resource group depends on the physical system itself. For example, a spacecraft requires guidance and navigation, while an *in situ* ground cluster does not. The central agent, which exists in any resource group and may be the only agent for limited field hardware, is the *Exec agent* [107]. The Exec agent oversees achievement of the goals provided by the sensor network's Mission agent. It also arbitrates conflicting resource requests between other agents in its resource group and is responsible for monitoring overall resource group health (*e.g.*, fault detection). Other resource group agents may include one or more *Science agent(s)* responsible for decision-making associated with achieving particular science objectives and other, specialized, domain-dependent agents (*e.g.*, a *GNC agent for guidance, navigation, and control* on a spacecraft) [107].

Resource-level agents are defined by the roles appropriate to a specific sensor network. Initial work on MACRO focused on providing autonomy for a constellation of multiprocessor spacecraft [107], such as the NASA Earth Science Enterprise's Magnetospheric Multiscale Mission (MMS) [26]. In this scenario, spacecraft sensing and computational functionality is broken into four agent roles illustrated in Figure 4. The Exec agent in overall control of the spacecraft and directs other agent activities by providing appropriate subgoals. The Science agent performs sensor configuration and on-board data analysis, making dynamic changes to sensor operation and data processing based on current conditions and science subgoals. The spacecraft also has a Communications agent to handle

communication with other spacecraft and the ground station. Finally, a guidance, navigation, and control (GNC) agent maintains the spacecraft in the appropriate orientation to the other spacecraft in the constellation. All of these agents share the computational resources of the spacecraft, which drove the initial development of the shared planning and resource management services for the resource level.

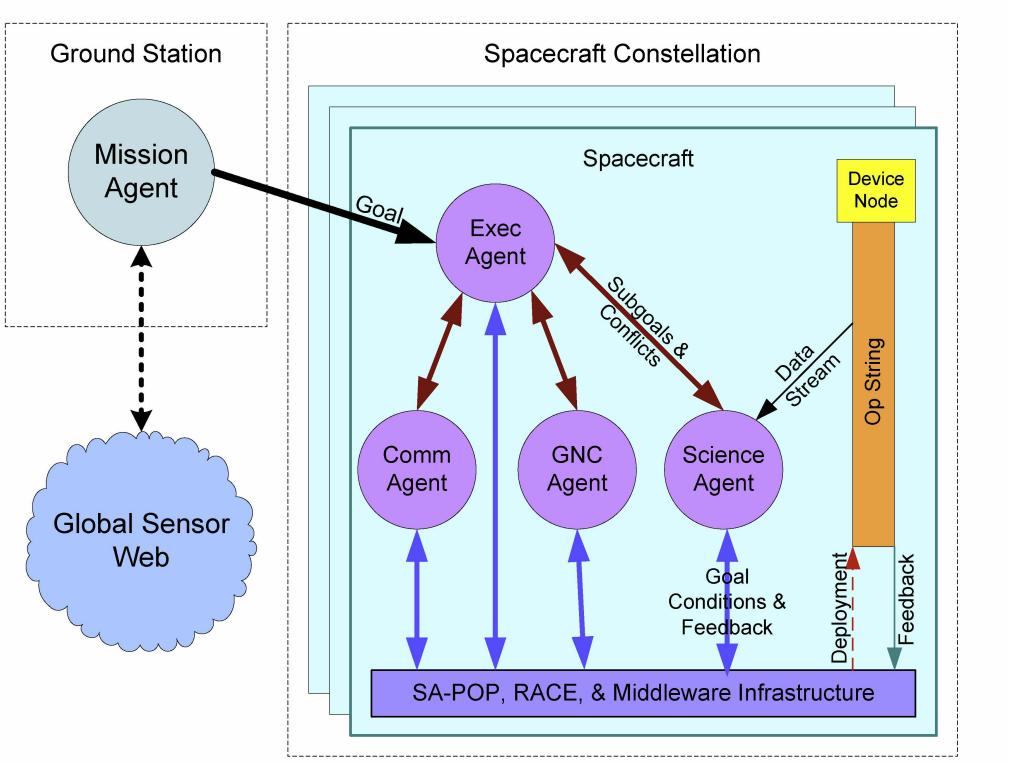

Figure 4: MACRO configuration for MMS

More generally, other sensor networks, such as those for weather monitoring, can be divided into resource groups of shared computational resources, each with an Exec agent, and likely one or more Science and other agents, with roles defined by the needs of the particular resource group. This initial work on the MACRO resource-level agents resulted in the design and implementation of middleware-based agents and the SA-POP and RACE services they employ, which are summarized in Sections II.4.1 and II.4.2, respectively.

Testing of the implementation and deployment of these resource-level agents and services in an MMS scenario provided results used to refine the interaction of SA-POP and RACE, as well as the modeling and development process for MACRO resource-level agents. These results, described in papers [C-9] and [C-11], illustrate the effectiveness of model-driven design, middleware, and services for planning and resource management to allow rapid design and implementation of autonomous agents for distributed, real-time embedded systems.

II.4 MACRO Agent Services and Infrastructure

In MACRO, multiple resource-level agents may share a set of computational resources to perform their particular functions. To efficiently employ and control a set of shared computational resources, they use services for decision-theoretic planning/scheduling (*i.e.*, SA-POP) and processor resource allocation/management (*i.e.*, RACE).

II.4.1 Planning Service for Resource Group Agents

Typically, deliberative agents achieve their goals by creating and executing plans. Most agent architectures include planning as an integral part of an individual agent's reasoning mechanisms. However, this picture is complicated in MACRO by the need for scheduling and awareness of resource utilization on *shared* computational resources by some resource-level agents. The Spreading Activation Partial Order Planner (SA-POP) [59] is designed to provide planning (and scheduling) as a service to resource-level agents, rather than the traditional inclusion of planning as an integral part of an individual agent's reasoning mechanisms. By providing efficient planning and scheduling as a shared service, the agents do not require detailed knowledge of the plans for other agents operating on the same set of shared computational resources. Rather, the agents are free to employ meta-level reasoning and communicate only high-level details for inter-agent coordination, while SA-POP, with

a system-wide awareness of agents' plans, is employed to efficiently meet their planning and scheduling needs.

II.4.2 Dynamic Resource Management Service

In addition to a planning service, MACRO resource-level agents use a service to provide dynamic resource management, freeing them from considering low-level resource allocation and control decisions. In large systems, the sheer number of software component sequences often poses a combinatorial deployment problem, *i.e.*, mapping components to computing nodes [76]. Moreover, the dynamic nature of the operations require runtime management and modification of deployments [50]. At the level of individual platforms (*e.g.*, individual satellites and ground-based sensor installations with a small set of servers) these problems necessitate a system with the ability to make resource allocation and control decisions at runtime. In MACRO, this service is provided by the Resource Allocation and Control Engine (RACE), which performs autonomous resource (re)allocation of components and (re)configuration of their settings, such that data gathering and analysis quality of service (QoS) requirements are met [101].

The Resource Allocation and Control Engine was designed and implemented as part of Nishanth Shankaran's dissertation research [100]. RACE provides a range of resource allocation and control algorithms that can use middleware deployment and configuration mechanisms to allocate resources to operational strings and control system performance after operational strings have been deployed [101]. In particular, it uses *Resource Monitors* and *Application QoS Monitors*, to track system resource utilization and application QoS respectively. The broad architecture of RACE and its interplay with SA-POP is illustrated in Figure 5.

RACE's algorithms determine how to (re)deploy an application specified by operational strings and ensure desired QoS requirements are met, while maintaining resource

utilization within desired bounds at all times. The allocation algorithms determine the initial component deployment by determining the best mapping of these components to the appropriate target nodes based on the availability of system resources. For example, an allocation algorithm could apportion both CPU and memory resources of a set of processors to components in such a way that avoids saturating these resources. Likewise, RACE's control algorithms adapt the execution of an operational strings' components at runtime in response to changing environments and variations in resource availability and/or demand. For example, a control algorithm could (1) modify an application's current operating mode, (2) dynamically update component implementations, and/or (3) redeploy all or part of an operational string's components to other target nodes to meet end-to-end QoS requirements.

RACE uses mechanisms provided by the underlying middleware to perform the allocation and control decisions made by its algorithms. For example, RACE uses standard mechanisms defined by the Lightweight CORBA Component Model (CCM) [82] to (1) (re)deploy and (re)configure application components, (2) transition application components from idle states to operational states and monitor the performance of the system, and (3) modify components and/or operational strings to realize the adaptation decisions of control algorithms.

The integration of RACE with SA-POP as services for MACRO resource-level agents has provided a powerful infrastructure that includes planning and scheduling for actions and application composition, multi-capacity resource allocation, system monitoring, and adaptive resource management. The results presented in paper [C-6] illustrate the effectiveness of using a suite of resource allocation algorithms in RACE when multiple resource constraints (*e.g.*, processing power and memory) must be considered. Experiments conducted for a system employing RACE and SA-POP together showed that they provided effective, autonomous adaptation to both large and small fluctuations in resource availability as well as dynamic changes in the environment. These results are presented in papers [J-1], [C-7], and [W-1] and illustrate the flexibility and autonomous performance benefits of integrating

SA-POP and RACE as part of an agent infrastructure. Further details on RACE and its integration with SA-POP are available in paper [J-1].

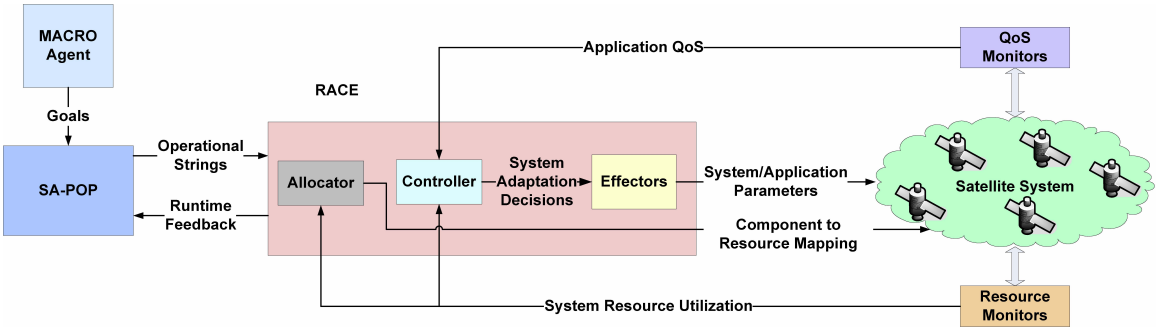

Figure 5: RACE architecture

II.4.3 Middleware Infrastructure

Many of the platforms comprising a global sensor web will be distributed, real-time embedded (DRE) systems, such as field sensors, spacecraft and airborne systems. Modern DRE systems implement task sequences, such as data processing, using *component middleware* [52], which automates remoting, lifecycle management, system resource management, deployment, and configuration. To perform efficiently in a distributed, heterogeneous computing environment, MACRO agents rely on the underlying component middleware. The CORBA Component Model provides a way to logically bundle interfaces into service families and specify the configuration and deployment of objects as complete applications. This results in flexible, scalable implementations for distributed systems that are easier to adapt and maintain.

To perform efficiently in a distributed, heterogeneous computing environment, all agents in MACRO, as well as resource-level agent services, are implemented using state-of-the-art component middleware, implemented as an open source project led by the Distributed Object Computing (DOC) group under Dr. Douglas Schmidt's direction at the Institute for

Software Integrated Systems, Vanderbilt University. Specifically, MACRO is built upon the Component Integrated ACE ORB (CIAO) and the Deployment and Configuration Engine (DAnCE), which are open source implementations of the OMG's Lightweight CORBA Component Model (CCM) [82] and Deployment and Configuration (D&C) [81] specifications. CIAO and DAnCE are built atop The ACE ORB (TAO). TAO is a highly configurable, open-source, real-time CORBA Object Request Broker (ORB) that implements key design patterns to meet the demanding quality-of-service (QoS) requirements of distributed systems, including servers as well as real-time embedded systems. CIAO extends TAO by abstracting key QoS concerns (such as priority models, thread-to-connection bindings, and timing properties) into elements that can be configured declaratively via metadata (such as standards for specifying, implementing, packaging, assembling, and deploying components). Defining and configuring QoS properties as metadata disentangles code for controlling these concerns unrelated to functionality from code that implements the application logic, thus making MACRO development more flexible and productive. DAnCE extends CIAO by allowing application deployers to specify how implemented components should be packaged, assembled, and customized into reusable subsystems and services. More information, including relevant publications, on DOC group middleware is available from `http://www.dre.vanderbilt.edu/`.

II.5 Summary

This chapter defined the Multi-agent Architecture for Coordinated Responsive Observations (MACRO) and provided an overview of its agents' roles, relationships, and services, as well as the middleware infrastructure on which they are built. MACRO uses QoS-enabled component middleware to help automate many system configuration and management tasks for sensor web agents and applications. Atop the middleware infrastructure, MACRO's dynamic resource management service, RACE, provides efficient allocation and control of computational resources, while MACRO agents employ a decision-theoretic

planning and scheduling service, SA-POP, to autonomously adapt system functionality to changing science objectives and environmental conditions. The specialized roles and two-level hierarchy (*i.e.*, a mission level spanning the sensor web's constituent sensor networks and a resource level for adaptive operation of local sensor network resources) of MACRO agents enables tractable solutions to the coordination and control problems facing a system with the scope of a global sensor web, identified in Section I.3. Ultimately, MACRO provides a powerful computational infrastructure for enabling the deployment and adaptive operation of large, distributed systems that require both high-level coordination of complex tasks across agents and local, dynamic adaptation for effective use of limited resources in dynamic, uncertain environments.

CHAPTER III

TASK ALLOCATION

In large-scale, distributed, multi-agent systems (MAS) with limited resources and heterogeneous users, task allocation is a necessary system capability. Further, effective task allocation requires the optimization of allocations for an appropriate allocation metric. As discussed in Chapter I, NASA's Earth Science Vision calls for the development of a global sensor web that provides coordinated access to constituent sensor network resources for research and resolution of Earth science issues [53]. An effective task allocation mechanism for a global sensor web must account for both allocation efficiency (*i.e.*, value or importance to the sensor web as a whole) and fairness (*i.e.*, individual user satisfaction). This task allocation mechanism must select and allocate an appropriate subset of heterogeneous, distributed sensors and computational resources for user tasks that often require collaboration among multiple constituent sensor networks. Further, the global scale and distributed nature of the sensor web require that most computation be performed by users and constituent sensor networks, rather than providing a large computational infrastructure in addition to user and sensor network resources.

As illustrated in Figure 1 in Section I.1, a sensor web is made up of many independent sensor networks. One difficulty in task allocation for a sensor web is that available resources (*e.g.*, sensors, servers, bandwidth) are not owned or controlled by any single entity. Various institutions, governments, and corporations will have the final say on how their resources are deployed and used. Further, a global sensor web will have many independent, heterogeneous "users" (*e.g.*, weather modeling and prediction systems, disaster recognition and management systems, and scientists) requesting access to, and control of, the sensor platforms to support their research and analysis activities.

In a global sensor web MAS, user tasks and plans for their achievement have a high

degree of complexity and may span multiple sensor networks. Hierarchical analysis helps deal with this complexity, both for problem/task representation by domain experts and for coordinated planning and scheduling among multiple Mission agents. Section IV.1 describes how MACRO Mission agents employ a modified implementation of the *Task Analysis, Environment Modeling, and Simulation* (TÆMS) [55] language, which provides a hierarchical task network representation for multi-agent planning and scheduling. Further, Section V.4 details MACRO's augmentation of TÆMS hierarchically-decomposable tasks with the OGC SensorML [9] representation of sensors and data processing. This provides standardized descriptions of task/subtask requirements, effects, and classification across sensor networks.

To maintain tractability at the scale of a global sensor web, MACRO decouples the problems of task allocation and planning/scheduling for task achievement. Mission agents plan and schedule achievement of allocated tasks/subtasks represented in the augmented TÆMS representation. However, User agents are not required to have explicit knowledge of how tasks can be achieved in the system. Instead they announce tasks in the standardized SensorML format more appropriate to their needs and knowledge. Because MACRO augments TÆMS tasks with SensorML metadata, broker agents can translate between User agents' SensorML task announcements and the the augmented TÆMS representation employed by Mission agents. The details of the translation between these representations is presented in Section V.4.

In this chapter we present the MACRO agent negotiation mechanism for fair, high-utility allocation of these complex, hierarchically-decomposable tasks to the sensor web resources. Sensor web users require resources in the sensor web to accomplish their tasks. Because different sensor web users may be of different importance/priority in the sensor web, we assume that an organizing body, such as NASA, assigns a guideline percentage *share* of the sensor web resources to each User (and Mission) agent. Generally, the task allocation problem can be viewed as that of determining the best selection of high-level

user tasks, which are divided into subtasks assigned to sensor web agents with resources capable of fulfilling those subtasks. The "best" selection of tasks is defined by the combined considerations of system value/importance and user fairness.

III.1 Related Research

Task allocation has been dealt with in MASs through a variety of techniques suitable to a range of applications (*e.g.*, [33, 49, 102]). Many of the techniques for "task allocation" are applicable primarily to cases in which the capabilities of many agents are the same or at least largely overlapping. In the sensor web, while computing resources, and to some extent software resources, may be interchangeable/overlapping, sensor networks also have disparate and sometimes unique capabilities defined by the particular types of sensors in their network and their geographical location. Thus task allocation techniques designed for agents with heterogeneous, substitutable and non-substitutable, capabilities and resources may be more useful for the sensor web allocation problem.

A related area of research to "task allocation" is that of "resource allocation," in which resources are allocated to agents so they can complete tasks (*e.g.*, [18, 72, 77]). In fact, "task allocation" can sometimes be viewed as an inverse form of resource allocation, where subtasks are treated as resources with a cost rather than a utility, which are then allocated to agents "interested" in those subtasks (*i.e.*, the ones capable of performing them) [18]. However, this view of task allocation may introduce significant complications when there are causal and other scheduling dependencies between subtasks, because resource allocation algorithms are not designed to handle such constraints among resources [18]. Regardless of whether the problem is viewed as "resource allocation" or "task allocation," there are dependencies between tasks/subtasks that are not accounted for in most allocation mechanisms. This implies that the *task/resource allocation* capability is intertwined with the *global, coordinated planning/scheduling* capability, which is discussed further in Chapter IV.

III.1.1 Allocation Problem Characteristics

In order to determine applicable task/resource allocation algorithms for sensor web allocation, one must consider the characteristics of the problem to be solved. First, it must be clear which "resources" of the sensor web must be allocated. These are primarily configurations of sensors and software capable of data collection, distribution, and processing. Other "resources" (*e.g.*, power and computational resources) in the sensor web are more appropriately handled as constraints on the availability of the aforementioned resources because their usage is not the primary concern of the users requesting tasks. For example, the hurricane tracking application from Section I.1 must acquire data from appropriately configured sensors and have it processed in a particular fashion, but it is agnostic to the particular use of power, CPU, memory, and bandwidth by the sensor networks accomplishing its task.

The characteristics of the allocable sensor web resources are an important factor in the choice/design of resource and task allocation algorithms. Chevaleyre et al. [18] suggest some basic characteristics of resources that determine the applicability of most resource allocation algorithms: static vs. dynamic, continuous vs. discrete, divisibility, single-unit vs. multi-unit, and shareability. Because configured sensors and software are neither consumable (*i.e.*, they can be used more than once without replenishment/replacement) nor perishable (*i.e.*, they do not expire or become unusable over a relatively short period of time like food products), they are static (reusable) resources for the purposes of allocation. Additionally, sensor web resources can be considered discrete, because data products are produced by sensors with particular configurations and software capable of particular forms of data processing. The static and discrete characteristics of sensor web resources significantly reduce the complexity of potential allocation algorithms.

Further, although divisibility is more a characteristic of how resources are treated during allocation, the discreteness of sensor web resources implies that they can be considered

indivisible for allocation purposes. Also, because most are unique (defined by sensor capabilities and location) or only useful in the given amount (computational resources for a particular piece of software), a simple, single-unit representation is most appropriate. Although there is significantly more overlap in data processing software available on multiple servers, there is still generally no need for multiple units of those equivalent resources in sensor web tasks.

Finally, shareability of resources is an important, complicating factor in the sensor web allocation problem. While many resource allocation algorithms are designed solely for non-shareable resources, these are not particularly appropriate for sensor webs because in many cases the production of data from a sensor can be shared among the tasks of many users. Of course, allocation would not be an issue if resources were always shareable, and there are also many cases where tasks imply mutually exclusive use of a sensor web resource (*e.g.*, a satellite and its imaging sensors that can only be aimed at a single location at a time or field equipment that only has enough power reserves to operate one set of sensors at a time). This is further complicated by the fact that the shareability of a resource may actually be determined by the choice of allocation. Consider a sensor configured to a particular data rate for a user task. This resource is not shareable, without modification, with a task requiring a higher data rate, but is (probably) shareable with tasks requiring the same, or lesser, data rate.

III.1.2 Allocation Solution Characteristics

In addition to characteristics of the allocation problem, characteristics of, and metrics for, the desired solution are an important consideration in the design of an allocation technique. Criteria for assessing allocations in economics and MASs, referred to as "social welfare" metrics, often fall into the two, somewhat overlapping, categories of "efficiency" and "fairness" [13, 28]. Efficiency metrics are primarily concerned with maximizing the benefit to (agent or human) society as whole, whereas fairness metrics generally attempt

to improve the lot of individuals, particularly those that are worst off, in comparison to the rest of society. Chevaleyre et al. [19] suggest some common types of social welfare metrics applicable to MASs:

- *Pareto efficient* allocations (*e.g.*, [38]) are ones for which no other allocation can be found in which no agents would be worse off and some would be better off. Pareto efficiency would be a desirable characteristic for sensor web allocations, although one that may not be easily computed or provable given the potential variety of agents in a global sensor web.

- *Utilitarian* allocations (*e.g.*, [38]) maximize the sum of individual utilities for members of the society. A related utilitarian measure is the Nash product, which is the product of individual utilities. Assessing these metrics requires that individuals can assign a utility to their potential allocations of resources, rather than just expressing a preference order over them. When this is possible, utilitarianism provides a useful, although very strong (with strict maximization), metric for social efficiency of allocation.

- *Egalitarian* allocations (*e.g.*, [39]) maximize the utility of the individual that is worst off in the society. The leximin metric extends egalitarianism in the cases where multiple allocations have the same utility for the least satisfied individual(s) by comparing the utilities of the next least satisfied individual(s) across allocations, and so on. Egalitarian metrics could provide a useful measure of fairness in sensor web resource allocation, although the emphasis on the least satisfied individual(s) can be limiting.

- *Envy-free* allocations (*e.g.*, [37]) are ones in which no individual would prefer the resources allocated to any other individual. The task-based nature of sensor web allocations limit the usefulness of this metric. Since the set of resources an agent desires are the ones necessary for the achievement of a particular task, direct comparison of resources across agents is rarely relevant.

Because one goal of a sensor web is to provide synergistic benefits to its member sensor networks as well as other users, allocation efficiency (*i.e.*, system utility) is an important consideration. However, fairness is also important because repeated allocations that do not respect some measure of fairness will likely be unacceptable to the human operators/owners of sensor networks, as well as other human users, who receive the least resources. De Jong et al. make the point that MASs that do not incorporate additional, human notions of fairness risk producing unexpected behavior and not achieving intended goals, particularly when some agents are intended to represent the interests of individual humans or organizations [28]. In particular, inequity aversion (the tendency to avoid outcomes that are not equitable among participants) suggests that egalitarianism may be too narrowly focused on the least satisfied individual to provide an appropriate fairness metric. Rather an extension that considers not just the least satisfied individuals, but rather the range or other disparity measures in utility across the society, may be more useful.

De Jong et al. also suggest that priority awareness (the inclusion of additional information about the importance of priority of participants in judging the equity of a solution) is an important enhancement to inequity aversion [28]. This is particularly important in the case of sensor web allocations because some members will be contributing more resources than others and some users (*e.g.*, a hurricane tracking application) will have priority over others (*e.g.*, a web interface to members of the public interested in earth science). Therefore, an appropriate fairness metric for sensor web allocations would be an extended egalitarian metric incorporating inequity aversion rather than simply considering the least satisfied individual, which accounts for priority awareness. Such a metric is discussed further in Section III.3.

Overall, it is clear that the appropriate metric(s) for sensor web allocations should include both efficiency and fairness considerations. Unfortunately, strong fairness and strong efficiency criteria often can not be simultaneously accommodated [11, 12]. This necessitates some form of tradeoff between fairness and efficiency metrics in optimizing sensor

web allocations. Thus allocation algorithms that do not consider fairness and efficiency criteria simultaneously are ill-suited to the sensor web allocation problem. Our solution to the combination of fairness and efficiency criteria is presented in Section III.3.

A significant amount of research has been done on algorithms for combinatorial auctions [46, 85, 89], including many for task/resource allocation and scheduling (*e.g.*, [27, 56]). While these algorithms have been successful in optimizing allocations for efficiency metrics, they do not consider fairness in attempting to find an allocation. Further, combinatorial auctions generally require a central party to determine allocations, making them poorly suited to a distributed system on the scale of a global sensor web. Although work has been done on distributed task allocation/scheduling algorithms that incorporate combinatorial auctions (*e.g.*, [56, 65]), the available algorithms still require a significant amount of centralized computation not amenable to distribution across the participating agents.

Others have considered task allocation as the problem of coalition formation (*e.g.*, [68, 96, 99, 102, 112]). From this perspective, allocation of tasks is achieved by determining an optimal or near-optimal set of coalitions each of which includes the resources necessary to achieve a particular task. Most solutions to the coalition formation problem involve forms of set covering techniques or market/auction techniques. However, set covering and similar techniques rely on centralized computation of coalitions or on negotiation among provider agents (*i.e.*, Mission agents in the MACRO framework) without allowing requesting agents (*i.e.*, User agents in the MACRO framework) to choose among potential coalitions based on task achievement criteria (*e.g.*, time to completion, data rate, and other quality of service characteristics in a sensor web). While market/auction techniques (*e.g.*, [70, 111]) for coalition formation might be extended to allow user preferences on task achievement, they require requesting agents to have knowledge of the resources required to achieve their task and often also require centralized computation, such as solution of combinatorial auctions. Therefore, other auction techniques without these requirements may provide a task allocation solution more appropriate to a global sensor web.

III.1.3 Contract Nets for Allocation

A widely implemented MAS solution to resource/task allocation that uses computationally simpler, single auctions is the Contract Net Protocol (CNP) [104]. The CNP is a well-studied [86, 121] multi-agent negotiation mechanism for distributed task/resource allocation, where agents have different, or even changing, resources, capabilities, and performance. In particular, the CNP and its derivatives can allow efficient allocation of resources without restricting the criteria individual agents can apply to determining their preference for tasks or bids to complete tasks. This flexibility is particularly important in designing a system for a tradeoff between efficiency and fairness metrics. Further, allowing arbitrary preference criteria is vital when the individual agents must represent, and may be designed by, many different parties with varying internal goals and constraints, such as in a global sensor web.

A number of variations and extensions to the classic CNP have been proposed to increase performance and handle additional problems in a range of applications (*e.g.*, [1, 84, 97, 109, 113]). One difficulty in using the classic CNP for sensor web task allocation is that subtasks are interdependent. Sandholm recognized the problem of dependencies between tasks and suggested grouping of tasks for bidding to handle dependencies [95]. While this solution may be insufficient for a global sensor web, explicit recognition of subtask dependencies by decomposition of a high-level user task can allow other planning/scheduling algorithms to be used in conjunction with the contracting mechanism of a CNP.

A related approach is to organize agents interested in dependent subtasks into teams. For example, Sims et al. apply bottom-up formation of teams based on marginal utilities to achieve efficient resource coverage in bidding on interdependent tasks/subtasks [103]. However, this particular approach assumes agents are singularly motivated to improve global utility and that many resources are equivalent for a given task. Still the concept of organizing agents into teams for bidding on tasks could be combined with planning/scheduling among team participants to provide a viable solution in a sensor web contract net.

Another significant problem in application of the CNP is the determination of when contractors should bid [98]. One solution is to allow bidding on multiple tasks requiring the same resource, plus a conflict resolution stage [84]. Another, more popular solution that alleviates the problem of when to bid, is to allow decommitment from contracts [97, 109, 113]. However, decommitment penalties may be necessary to prevent instability problems in such systems [40]. In a sensor web, decommitment penalties, such as the leveled commitments introduced in [97], may not be possible because the sensor network Mission agents do not directly receive any monetary or other benefit from contracted tasks. Another solution is to separate initial bidding from commitment [1, 62, 118].

Aknine et al. propose an extended (prebidding plus definitive bidding phases) CNP negotiation protocol [1]. This extension is primarily useful when task announcements come in rounds or negotiations for many announcements tend to overlap in time. Experimental results show that the multi-phase CNP negotiation allows efficient allocation of resources with less time overhead than a classic CNP implementation when many announcements are made by managers over relatively short periods of time [1]. A related benefit is that decommitments from contracts tend to be unnecessary due to the parallel negotiating and more efficient allocation.

Finally, a significant problem in using contract nets for sensor web allocation is how to effectively handle agents with independent goals and constraints that are not purely cooperative and interested in maximizing for system-wide social welfare metrics. Sandholm proposed bid pricing based on marginal cost and bid selection based entirely on these costs, allowing effective use of the CNP with competitive agents as well as cooperative ones [95]. Although such restrictions are impossible when agent design and implementation, are not under the control of the system designer, introducing some form of price for bids may be a useful tool. When bids can be priced by trusted broker agents, under the control of the system designers, they can influence bid choices in the system. The use of third-party agents/infrastructure has been suggested in other extensions of the CNP. For

example, MAGNET enhances a leveled-commitment contracting process with a general market infrastructure acting as third-party intermediaries [24].

III.1.4 Brokers for Contract Nets

Beyond infrastructure/services to facilitate communication and market interactions, some multi-agent systems have used full-fledged agents as matchmakers, brokers, or mediators, collectively referred to as "middle agents." The term "matchmaker" usually applies to an agent who simply helps locate service providers for service requesters. Matchmaking is clearly an important function of a middle agent for mission-level coordination in sensor webs and has been successfully applied in contract nets (*e.g.*, [74]). However, using middle agents only for matchmaking may require User agents with more capabilities to consider alternative providers and include additional communication overhead [31]. Beyond matchmakers, more capable middle agents have been suggested to provide "market-maker services, domain expertise, trust, visibility, assurance, and certification" [61].

Some dimensions for classifying middle agents include how much information is provided to the middle agent (*e.g.*, simply capabilities and requests, or also parameters and preferences) and whether the middle agent is an intermediary in communication/negotiation between the other agents [119]. In a sensor web contract net, "brokers," who also act as an intermediary between the other agents, provide a solution that puts greater control in the hands of the system designers. For example, a broker can effect optimization of the system (*e.g.*, in terms of load-balancing) while maintaining privacy [31].

There is no doubt that the centralization inherent in a single broker presents a communication bottleneck and a single point of failure [31], which would be unacceptable in a sensor web. Scalable approaches have been suggested for distributed matchmaking without the centralization of middle agents (*e.g.*, [83]), but they do not provide the other benefits of a trusted third-party or an effective way to influence negotiations in the system. Instead, a dynamic team of brokers could provide most of the benefits of a single broker. Although

multiple brokers requires additional message traffic to synchronize and aggregate information among them, it avoids the centralization of a single broker. For example, Kumar et al. suggest persistent teams of brokers sharing agent registration information with dynamic spawning of additional brokers to address fault tolerance and recovery concerns in brokered systems [64].

A team of broker agents to mediate contract net negotiations could alleviate many of the difficulties in using a contract net for sensor web resource allocation. The brokers could provide efficient matchmaking/locater services, minimizing communication overhead. The use of brokers also provides possibilities for affecting contract net negotiations, such as assigning priority/utility values to task announcements forwarded to Mission agents or by controlling a virtual currency and taxing bids forwarded to User agents. Even though User agents and Mission agents may be implemented by parties other than the system designers, employing these techniques through broker agents will allow the system designers to influence the contract net negotiations, and potentially optimize the resource allocation for efficiency and fairness metrics. Further, Zhang and Zhang suggest that matchmaking algorithms be extended to consider the track record of agents in providing advertised capabilities [125]. Similarly, in a sensor web, brokers could maintain records of agents who fail to meet contracts or violate other system conventions.

Overall, a multi-phase contract net protocol provides an effective, flexible allocation solution for a sensor web MAS. However, existing research does not resolve all of the difficulties in employing a contract net protocol for allocation of complex tasks in a large, open MAS. Employing brokers for matchmaking reduces communication overhead in a contract net, but does not limit it sufficiently for efficient consideration of all possible solutions to the combinatorial problem of contracting subtasks that may be performed by multiple agents. Further, an open MAS, in which announcers and contractors are independently designed, also requires an infrastructure that can influence allocations, optimizing for fair and efficient use of resources. Finally, in order to accurately evaluate allocation techniques,

we must also define a metric that combines fairness and efficiency appropriate to sensor web allocations. Section III.2 illustrates these challenges in more detail, and Sections III.3, III.4, and III.5 describe the proposed approach to resolve them in MACRO.

III.2 Unresolved Challenges

As discussed in Section III.1.2, the definition of preferred allocations for the sensor web must allow for a trade-off between fairness and efficiency. Fairness depends on a comparison of individual agents' satisfaction with their allocated tasks, while efficiency depends on the overall system definition of utility for the allocated tasks. Even if the utility value assigned to tasks is based on the assigned share of the sensor web resources of a User agent, considering only the efficiency of an allocation (total system utility), could result in the monopolization of system resources by the users with greater shares at the expense of the users with smaller shares. If, on the other hand, only fairness is considered in allocations, the overall system utility could suffer significantly. Thus, the first challenge in achieving appropriate allocations is defining a metric for sensor web allocations that combines fairness and efficiency. The MACRO sensor web task allocation metric and its variations are presented in Section III.3.

One of the major advantages of using the contract net protocol for allocation is that it provides flexibility in defining the utility or preference structure that User agents employ for task bids, which depends on the needs of the people or application they represent. However, this flexibility also presents a challenge in designing an allocation architecture that performs well under a given metric. The standard contract net protocol does not provide a method to influence allocations other than by the definition of task/bid creation and preferences in announcers and bidders. This is insufficient for MACRO because User and Mission agents are implemented independently of the MACRO infrastructure. Therefore, another challenge in achieving appropriate allocations is providing a framework in which

system designers can influence the task auctions to yield allocations that provide fair and efficient use of resources, while limiting infrastructure computational overhead. Section III.4 discusses how our implementation of MACRO addresses this challenge and Section III.6.2 provides experimental verification of the MACRO solution approach.

Another difficulty in applying the contract net protocol in a sensor web MAS is its emphasis on two-party contracts. In a sensor web, a User agent may require the resources of multiple Mission agents to achieve a high-level task. Therefore, MACRO uses the concept of subcontracting, allowing resources of multiple Mission agents to be assigned to a single task through a primary contract and additional subcontracts between Mission agents. However, with multiple required resource types and multiple possible decompositions for a task, there may be many Mission agents that could play the role of the primary contractor, each with many possibilities for subcontractors. If each potential primary contractor announces subtasks and receives subtask bids before bidding on the task, a great deal of communication and computation must be performed before each Mission agent can bid on the task. The space of possible task allocations, including task decompositions and subtask allocations, is large and complex, making consideration of all possible contractor-subcontractor sets computationally infeasible in a global sensor web. On the other hand, if each Mission agent bids on the task before soliciting subcontract bids, the uncertainty in the resulting primary contract bids would significantly diminish a User agent's ability to make an informed choice. Therefore, efficiently and effectively performing subcontracting with decomposable tasks is another challenge, which is addressed in Section III.5 and experimentally verified in Section III.6.1.

III.3 Sensor Web Metric

Comparing potential allocations and measuring performance of an allocation mechanism in a sensor web presents the challenge of defining an appropriate allocation metric, as indicated in Section III.2. In the sensor web, allocation of resources to announced tasks

is an ongoing process and the "allocation" can be defined as the set of tasks that Mission agents have committed resources to achieve over a given window of operation. Comparing potential allocations and measuring performance of a sensor web allocation mechanism requires an appropriate task allocation utility metric. In particular, a sensor web task allocation utility must account for two, sometimes competing, concerns: 1) *efficiency*: value of the allocated tasks to the system as a whole, and 2) *fairness*: satisfaction of individual system users.

Many multi-agent allocation metrics and mechanisms have been devised based on using a single-dimensional utility measure [49]. However these metrics are generally employed by multi-agent systems in which agents' self-interest extends only as far as necessary to result in the preferred system behavior. When some agents represent real, self-interested parties, as in the sensor web, fairness becomes an important consideration. Rather than relying solely on allocation efficiency to determine overall utility, we propose a sensor web task allocation metric that incorporates both efficiency and fairness.

To provide a metric combining these two considerations, we first define two separate metrics for evaluating the efficiency and fairness of a given allocation. To measure the efficiency of a sensor web allocation, we use a utility-based approach, in which the ideal would be to determine a measure of the importance of each allocated task to the system and sum them to provide an efficiency measure for the allocation. In a system where each User agent assigns the same utility to each task and is of equal importance in the sensor web, the allocation efficiency could simply be measured as the sum of the utility accrued by each User agent for all allocated tasks. However, to aggregate utility values in a sensor web, there are two complicating factors that must be considered: 1) *agent importance* and 2) *task importance*.

User agents will be of varying importance in the sensor web as a whole, so the utility accrued to an individual User agent must be weighted by the User agent's importance to accurately represent its overall value. We assume an organizing body for the sensor web

can assign an appropriate *share* of the sensor web, as a percentage of the whole, to each User agent.

Definition s_a: the percentage share of the sensor web assigned to agent a, indicating relative importance of agent a in the system.

Further, User agents often assign different utilities to the same task (*e.g.*, a storm front data gathering and analysis task of a weather simulation application may be of high utility to that application but of no utility to a glacial watershed research application). Therefore, the utility of tasks from different User agents must be normalized to allow comparison or aggregation across agents. Over a sufficiently long period of time, a reasonable method for normalizing utility across User agents is to scale the utility of the tasks such that the sum of utilities for all desired tasks of an agent equals a pre-defined value, such as 100.

Definition $\hat{u}_{a,\acute{t}}$: the normalized utility of agent a's task $\acute{t}$ equal to $100 \bullet \frac{u_{a,\acute{t}}}{\sum_{t=0}^{D_a} u_{a,t}}$, where D_a is the number of tasks desired by agent a and $u_{a,t}$ is the utility to agent a of task t.

We propose to measure the efficiency of a sensor web allocation as the sum of normalized utility values for allocated tasks, where each task utility value is also weighted by the announcing agent's share of the sensor web. This accounts for both task and agent importance in the efficiency metric, defined as:

$$\sum_{a=0}^{A} \sum_{t=0}^{T_a} \hat{u}_{a,t} \bullet s_a$$

where A is the number of agents, T_a is the number of allocated tasks for agent a, and $\hat{u}_{a,t}$ is the normalized utility of agent a's allocated task t

Many fairness metrics rely on the utility of an allocation to each individual in order to rank allocations based on the least satisfied individual(s). Because User agents assess task utility differently and MACRO assumes a "share" approach to sensor web resource assignment, we prefer a fairness metric that compares allocated resources rather than accrued utility. Similarly, MACRO assumes that each Mission agent's resources can be defined as

a percentage of the total sensor web resources. In addition, each Mission agent can provide an estimate of the percentage of its resources, over time, used to achieve an allocated task.

Given the available task resource usage information, we can determine the total resources used in an allocation and the quantity of resources each agent should have received based on its share. For each agent receiving less than its share of resources, we calculate the deviation, by percentage, between the resources it should have received and the resources it actually received.

Definition d_a: agent a's percentage deviation below fair share of resources, which is 0 if $\acute{r}_a \geq r_a$ (where $\acute{r}_a$ is the quantity of resources actually allocated to agent a, and r_a is a's assigned share of the total allocated resources), and otherwise is equal to $1 - \frac{\acute{r}_a}{r_a}$.

We scale the fair share deviations from the range of $[0\%, 100\%]$ to the range of $[0, \frac{1}{A}]$, where A is the number of agents in the system, such that the total of all agents' scaled deviations never exceeds 1. Any agent receiving its share or more of resources will have a deviation of 0, so if all agents were completely satisfied the sum of fairness deviations would be 0. At the other extreme, if all agents were completely unsatisfied the sum of fairness deviations would be 1.

We suggest that an appropriate sensor web fairness metric should account for all unsatisfied individuals, instead of just the least satisfied individual. Therefore, we aggregate individual agent fairness deviations by summing and produce the allocation fairness metric value by subtracting this sum from 1. This provides a measure of user satisfaction that accounts for all individuals in the system. Further, this value has a range of $[0, 1]$, such that it can be used as weight on the measure of allocation efficiency. We define the fairness metric as:

$$1 - \sum_{a=0}^{A} \frac{d_a}{A}$$

One method of combining an efficiency and fairness metric would be to weight each individually and sum them to produce the metric value. Such a scheme has the advantage of allowing different tradeoffs between efficiency and fairness by using custom weights on

the component metrics. However, different system loads (*i.e.* number of tasks in the system) can result in vastly different efficiency values, and, therefore, require careful tweaking of the weight parameters for each load condition. Instead, we employ the fairness metric, with a value in the range $[0,1]$, as a weight on the efficiency metric. Naturally, in allocations where all agents are satisfied, the fairness metric will have a value of 1, so allocations are compared purely by efficiency. When all agents are completely unsatisfied, both the efficiency and fairness metrics will have values of 0 because no tasks have been allocated. In general, the combined metric accounts for agent importance, task importance, and priority-aware agent satisfaction by using the overall system value of the allocation weighted by the level of agent satisfaction. This combination provides a consistent measure that adjusts automatically to system load, as well as number and relative importance of user agents. The overall sensor web task allocation metric is provided by multiplying the efficiency metric value by the fairness metric value:

$$\left(\sum_{a=0}^{A}\sum_{t=0}^{T_a}\hat{u}_{a,t}\bullet s_a\right)\bullet\left(1-\sum_{a=0}^{A}\frac{d_a}{A}\right)$$

III.4 Allocation Mechanism and Optimization

Given a sensor web task allocation utility metric, MACRO mission-level agents must efficiently employ the preferences defined by such a metric in selecting the subset of announced tasks to allocate to sensor web resources. As described in Section II.3.1, MACRO User agents request tasks to be completed with sensor web resources. The User agents negotiate with Mission agents, who are capable of completing some or all of the announced task. Ultimately Mission agents will perform the tasks/subtasks allocated to them during the negotiation with User agents, which is mediated by the broker agents. Designing a flexible, low-overhead task allocation mechanism that can optimize allocations based on the sensor web metric presents a significant challenge, as identified in Section III.2.

III.4.1 MACRO Task Auctions

Problem. Heterogeneous User agents may have different preferences for task achievement characteristics (*e.g.*, completion time, sensor data resolution, and data compression level). The independence of individual sensor networks requires that Mission agents individually determine whether they can complete announced tasks, given internal goals and constraints, and that they estimate expected task achievement characteristics. Further, the vast majority of the computational power available to the sensor web is likely to reside with the constituent networks and users, rather than including a significant amount of dedicated infrastructure hardware. Therefore, MACRO requires a distributed (*i.e.*, primarily relying on User agent and Mission agent computation and communication, with minimal computation by broker agents), flexible (*i.e.*, allowing arbitrary user preferences on task achievement) task allocation mechanism.

Solution → Single task auctions. To provide distributed, flexible task allocation, MACRO employs single task auctions mediated by broker agents. This minimizes the additional computational power necessary for system infrastructure (*i.e.*, the broker agents in MACRO), as compared to more complicated allocation schemes (*e.g.*, solving combinatorial auctions). Further, single task auctions allow User agents to apply arbitrary preference criteria to the selection of received bids. Since MACRO task allocation is based on single auctions to negotiate a contract for performance of a task, it can be considered an extended form of the Contract Net Protocol (CNP) [104].

The first step in allocating high-level tasks is to determine which agents are capable of executing part or all of the task. In MACRO, broker agents are employed to provide this matchmaker service and the related translation of SensorML task announcements to possible sets of TÆMS subtasks. The directed announcement of tasks to agents capable of some part of the task significantly reduces the communication overhead compared to the traditional CNP, in which each task announcement is sent to every agent. MACRO employs two types of broker agents to mediate task allocation. Although a single type of

broker agent is theoretically sufficient to perform all necessary services, MACRO divides broker agents into two tiers based on their specific roles and responsibilities to simplify system deployment and dynamic modification.

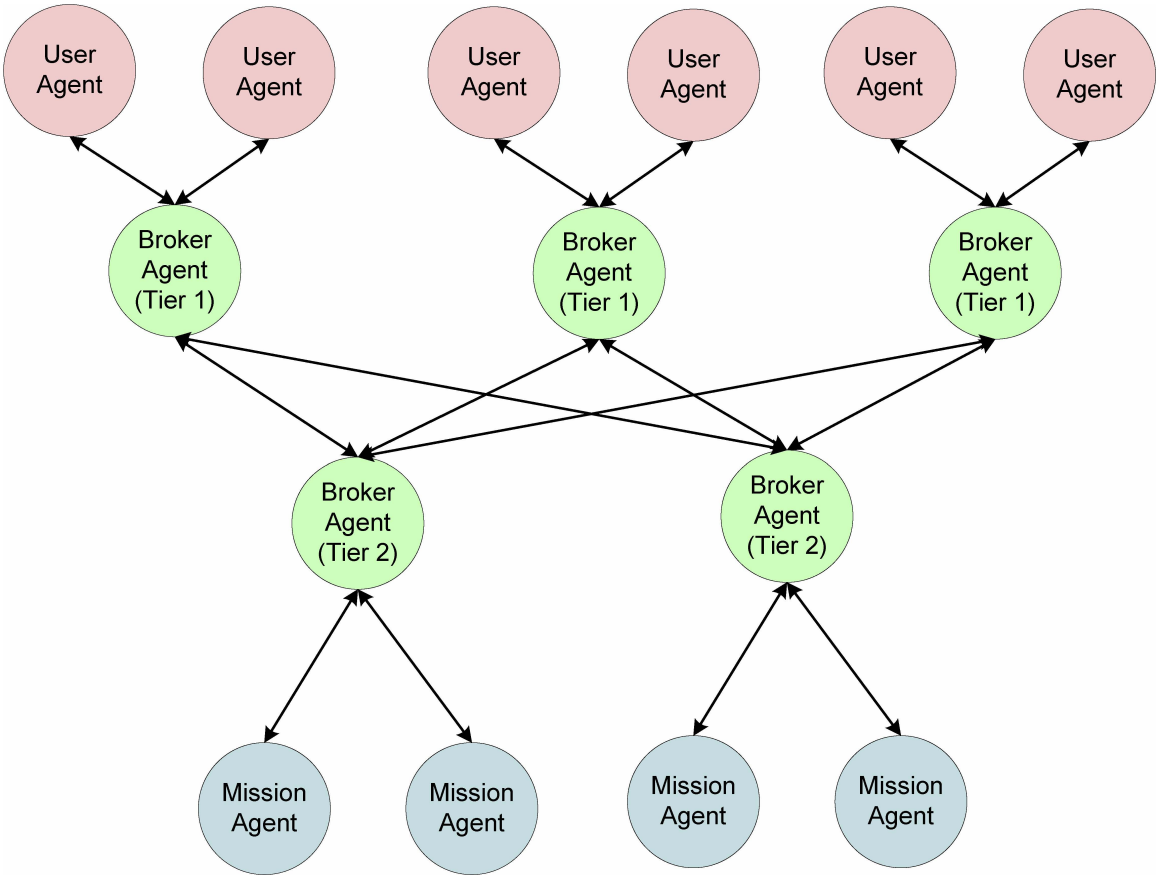

Figure 6: MACRO contract net

One responsibility of MACRO brokers is to provide an efficient matchmaking/locater service (*i.e.*, determining agents capable of performing all or part of an announced task and forwarding messages appropriately). Tier 2 Broker agents cluster Mission agents by geographic region and maintain a directory of sensor and computational capabilities for the Mission agents in their region. The requirements of an announced task are used by Tier 1 Broker agents to forward task announcements to appropriate Tier 2 Broker agents, who then relay the announcement to applicable Mission agents. Tier 1 Broker agents are also

responsible for assigning system utility values to task announcements taking into account User agent priority and task importance, as well as past system performance. Even though User agents and Mission agents may be implemented by parties other than the system designers, broker valuation of tasks allows the system to influence the contract net negotiations, and optimize task allocation for efficiency and fairness with the sensor web task allocation metric. Other roles and responsibilities of MACRO Broker agents are discussed in Section V.4

In a sensor web, a user task may require the resources of multiple Mission agents. Therefore, MACRO task allocation also includes subcontracting, such that a task may be achieved by multiple Mission agents through a primary contract and additional subcontracts. To reduce redundant communication and computation in the generation of subcontract bids, MACRO separates initial bidding from final bidding, extending the approach in the two-phase CNP proposed by Aknine et al. [1]. MACRO subcontracting for complex tasks is detailed in Section III.5.

III.4.2 MACRO Task Valuation

Problem. In addition to efficiently enabling flexible task auctions, MACRO allocation must also effectively handle resource contention. When multiple tasks require mutually exclusive control of the same resource during the same period of time, the system must ultimately determine which task will be allocated. While User agents can rank task bids based on a variety of task achievement characteristics, Mission agents are providing sensor network resources as a requirement of participation in the global sensor web and have little reason to prefer one task over another. Therefore, additional information is required by the Mission agents to make an informed choice among competing task announcements. Moreover, to achieve fair and efficient task allocations, the system must have a mechanism for influencing which task will be allocated.

Solution → Broker task valuation. Since broker agents provide the MACRO agent

infrastructure, and are the only MACRO agents entirely designed and implemented by the system designers, they perform the important role of influencing task auctions to optimize for fairness and efficiency. As mediators of all MACRO task auctions, they are in a position to provide additional information to Mission agents regarding announced tasks and thereby influence their preference between competing tasks. Specifically, the broker agents evaluate each task to determine an approximation of its marginal fairness-weighted utility based on past allocation performance.

In order to determine past allocation performance, Tier 1 broker agents track all current and completed task contracts for their assigned User agents. They maintain information on system performance, including: 1) utility to the User agent of each announced task, 2) completed tasks and their resource usage, 3) current task contracts and their expected resource usage. Periodically, each Tier 1 Broker agent broadcasts updates to its fellow Tier 1 Broker agents, allowing each agent to maintain a system-wide view of allocation performance. Although there is a lag time inherent in this update scheme, the brokers evaluate system-wide performance over a large moving window to minimize the impact of any lag times.

Tier 1 Brokers use the sensor web task allocation metric described in Section III.3 to evaluate past system performance within the given time window. Since the metric relies on summation of utility and resource usage values, revising the system performance value as new information arrives requires only relatively simple, constant time, additions to the running totals. Similarly, removal of tasks falling outside the moving window requires only the inverse computation (*i.e.*, subtraction of the relevant values from the running totals).

When a new task is announced by a User agent, its assigned Tier 1 Broker assigns the task a value corresponding to the approximated, marginal utility for allocation of the task based on the sensor web task allocation metric. This value is an approximation of the task's marginal utility because an individual broker's knowledge of the current allocation performance is incomplete due to the lag in receiving information from its peers. However,

we expect a reasonably large time window should prevent rapid changes in the fairness weight and corresponding over- or under-compensation. Given the broker's most up-to-date information on system-wide allocation performance, the task's marginal utility is calculated as the difference between the metric value (fairness-weighted system utility) if the task were allocated and the existing metric value.

III.5 Efficient Subcontracting

Sensor web users may often request allocation of tasks requiring resources from multiple independent sensor networks, each represented by an independent agent. Further, when the overall task is broken down into subtasks, an individual subtask may require resources that could be provided by multiple agents (*e.g.*, when there is overlap in sensor or data processing capability between multiple sensor networks in the sensor web). Consequently, there may be many combinations of agents capable of executing the overall task, as represented by the possible subtask allocations, which will be of varying utility to the requesting agent. This presents the challenge, identified in Section III.2, of designing an efficient mechanism for achieving a high utility allocation of subtasks among applicable agents.

As discussed in Section III.4, MACRO allocates tasks using auctions based on the Contract Net Protocol (CNP) [104] enhanced with brokers to accommodate both fairness and system utility considerations. In particular, the CNP and its derivatives can allow effective allocation of tasks/resources without restricting the criteria individual agents can apply to determine their preference for tasks or bids. This flexibility is especially important when individual agents are designed by many, heterogeneous parties with varying internal goals and constraints, as in a sensor web.

A primary issue in applying the contract net protocol to a sensor web MAS is its emphasis on two-party contracts. In MACRO, a User agent may require the resources of multiple Mission agents to achieve its high-level task. Therefore, MACRO uses subcontracting, allowing resources of multiple Mission agents to be assigned to a task through

a primary contract and additional subcontracts between Mission agents. However, with multiple possible decompositions of a task, there may be a large number of Mission agents who could be the primary contractor for the task, and each Mission agent could choose from a variety of subcontractors. If each of these Mission agents announces subtasks and receives bids before bidding on the task, a great deal of communication and computation must be performed before each can generate a complete bid on the task. On the other hand, if each Mission agent bids on the task before soliciting subcontracts, there is a great deal of uncertainty in the accuracy of resulting bids. Finding a trade-off between subcontracting overhead and completeness of bids for decomposable tasks presents a major challenge for efficient and effective allocation.

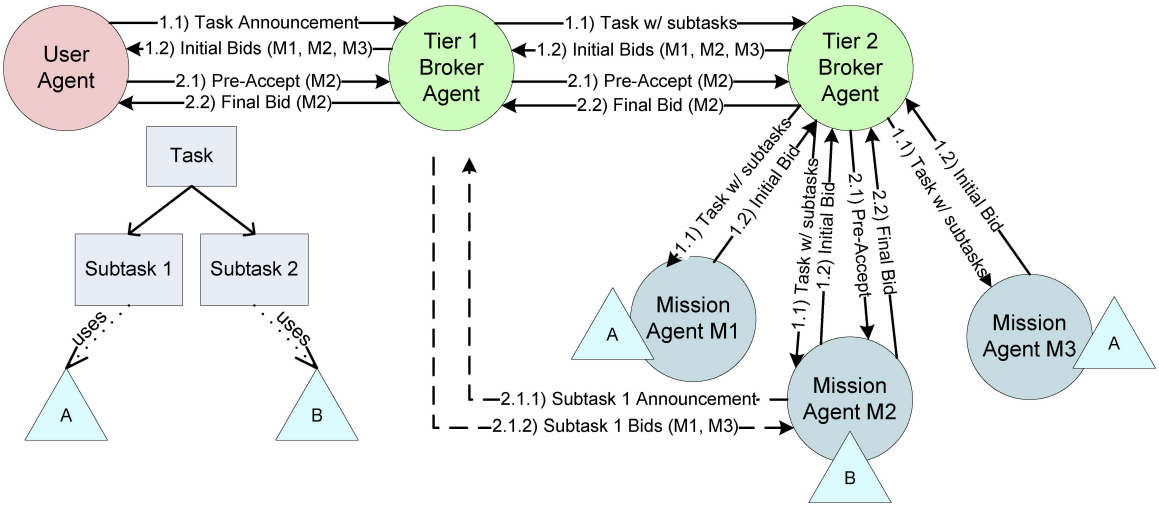

Figure 7: MACRO subcontracting

The MACRO CNP addresses this challenge by separating initial bidding from final bidding, extending the approach in the two-phase contract net protocol proposed by Aknine et al. [1]. This variation of the CNP breaks the contract net negotiations into an initial pre-commitment phase and a final commitment phase. During pre-commitment, the task is announced and initial bids are made, as illustrated by the steps 1.1 and 1.2 in Figure 7.

The announcing agent then pre-accepts what it determines to be the best initial bid, beginning the commitment phase, illustrated by the steps 2.1 and 2.2. In the MACRO CNP the pre-accepted agent can then announce subtasks it may not be able to, or want to, perform. As illustrated in step 2.1.1, a pre-accepted Mission agent is acting in an announcing capacity for these subtasks and communicates them to an assigned Tier 1 Broker, just as the User agent did for the initial task announcement. After receiving bids on its announced subtasks, the Mission agent makes a final bid on the task. This final bid includes relevant information on subcontracts and better estimates of applicable quality measures and time-to-completion. The announcing agent can then accept the final bid or pre-accept a different agent.

MACRO employs this two-phase CNP, including subcontracting, to limit subtask negotiations to a subset of initial bidders. Rather than allowing all potential contractors for the high-level task to announce subtasks, the MACRO CNP allows Mission agents to announce subtasks only after receiving a pre-accept from the User agent. Further, by limiting the number of pre-accepts a User agent can issue for a given task announcement, the MACRO CNP significantly reduces the total amount of communication and computation overhead in the contract net. One goal of the experiments in Section III.6.1 is to determine appropriate cutoffs for the number of pre-accepts allowed under different system configurations and operating conditions. Further, these experiments identify scalability trends for the MACRO pre-commitment subcontracting in terms of the major stress factors related to Mission agent capability overlap and task composition.

III.6 Experimental Evaluation

To evaluate the performance of MACRO task allocation and verify its efficiency and effectiveness, we performed two sets of experiments. The first set of experiments evaluates the efficiency (*i.e.*, overhead performance) and scalability of MACRO task allocation with

the pre-commitment task subcontracting presented in Section III.5. The second set of experiments evaluates MACRO allocation performance (*i.e.*, performance on the sensor web task allocation metric from Section III.3) using task auctions and broker task valuation, as described in Section III.4.

III.6.1 Subcontracting Experiments

This section presents the design and results of experiments that evaluate the overhead and scalability of the MACRO extended contract net protocol for subtask allocation under a variety of different potential system configurations and operating conditions. This experimental study of overhead performance with randomly-generated tasks allows us to determine realistic scalability trends and determine appropriate pre-accept cutoffs for real-world applications. These experiments validate our claims in Section III.5 that the MACRO extended contract net, with appropriate pre-accept cutoffs, provides an efficient, scalable solution to the challenges of allocating hierarchically-decomposable tasks in a sensor web or similar large multi-agent system. We determine reasonable limits on the number of pre-accepts required to find the best final bid under a variety of potential system configurations and conditions. Further, we identify scalability trends for the major factors affecting subtask allocation: 1) Mission agent capability overlap (defined as a density equal to the average number of Mission agents capable of performing requested subtasks), 2) number of alternative decompositions per task, and 3) number of subtasks per task decomposition.

III.6.1.1 Experimental Design

To maintain the generality of our results and their applicability to other large-scale multi-agent systems, we employed a simplified representation of subtasks and Mission agent capabilities. Specifically, each subtask is randomly generated in a generic XY plane, and each Mission agent is capable of achieving subtasks in a square region within that plane. We define density of Mission agents as the average number of agents capable of

requested subtasks in the system. In this setup, the Mission agent density is defined by their overlapping geographic regions, since subtasks are randomly generated in the XY plane. However, this simple representation allows our results to be easily extended to other applications where capabilities are significantly more complex and even unrelated to geography. Specifically, these results are applicable to other systems in which MACRO subcontracting can be applied and where capability overlap/density can be determined. In this experiment, 200 Mission agents were grouped into regions of 4 agents with overlapping capabilities. Each group of 4 Mission agents was assigned to a Tier 2 Broker for a total of 50 Tier 2 Brokers.

We also use a simplified representation of bid characteristics, rather than explicitly model the large range of utility functions a User agent could apply to bids. The experiments employ a single, randomly generated, quality value for each subtask based on the Mission agent contracted or subcontracted to perform the subtask. In general, Mission agents may be unaware of how the User agent determines bid utility from individual bid characteristics. Therefore a single quality value for each subtask suffices to represent the combined characteristics for that subtask in this experiment. Because Mission agents can include multiple possible task decompositions and subcontractors in their bids, the User agent can, in general, choose the one with the highest utility in any given bid.

In each trial, a single User agent and Tier 1 Broker agent were used to randomly generate a task and its decomposition into subtasks. With the MACRO Brokers' ability to aggregate task trees, any task can be decomposed into one or more sets of subtasks, where each subtask can be achieved by at least one Mission agent. Therefore, each task in this experiment had a corresponding set of randomly generated decompositions with a variable number of subtasks.

To determine pre-accept limits applicable to a variety of systems, we take a plausibly worst case approach in setting static experimental parameters. For example, in a real system, the different possible decompositions of a task would likely include some of the same

subtasks or, at least, many subtasks that could be performed by the same agents. Instead, this experiment considers a worst case scenario where decompositions are completely uncorrelated by independently generating random subtasks for each decomposition. Similarly, subtasks are generated randomly over the range of Mission agent capabilities, while a real-world system would likely exhibit greater clustering of subtasks based on geographic location and sensor types. The quality values for subtask bids are generated in the uniform, random range of 1 to 100, which is far more variability in quality than is likely in most systems. In this experiment, subtask bids are generated by all agents with the capabilities to perform the subtask. This results in more messages than in most systems where agents may not be interested in all subtask announcements or may have already committed the requisite resources to another task.

Further, subtask quality can be aggregated in a variety of ways to yield overall task quality, such as using the minimum or maximum value of subtasks (*e.g.*, when quality is determined by timeliness for a set of parallel subtasks). In these experiments, User agents employ a sum quality aggregation function (qaf) to generate the quality of the task from the subtask qualities. In other words, the quality of the task is equal to the sum of qualities for each subtask. Use of a sum qaf in these experiments represents situations in which the utility of a task bid to the User agent depends on the characteristics of each subtask, rather than being determined by a single, limiting subtask, as with the minimum and maximum qafs. Further, our experiments indicated that the sum qaf was the more difficult parameter setting, requiring more pre-accepts (and messages) to find the best bid than the minimum and maximum qafs.

In these experiments, we define performance in terms of message overhead rather than time overhead because messages are the major factor in the workload for the system infrastructure (*i.e.*, broker agents mediating the contract net negotiations). Sensor webs and similar applications that deal with many task announcements can face significant problems

due to communication and computation overhead when allocating each task requires numerous subtask announcements and bids, as with pre-bid subcontracting. Although the sequential pre-accepts in the proposed algorithm require more time to allocate a task than with the pre-bid approach, this provides a disincentive for User agents to increase system workload by making additional pre-accepts after they have received an acceptable final bid. While the increase in time overhead could still be a issue for some applications, it can be significantly mitigated by a simple modification to the limited pre-accept contracting approach. For these applications, the allowed number of pre-accepts could be performed in parallel rather than sequentially, and the presented performance and scalability results for message overhead remain the same.

III.6.1.2 Experimental Results

Each experimental run involved 2000 trials (*i.e.*, 2000 randomly generated tasks). In each trial, the User agent announced a task through the Tier 1 Broker. The Tier 1 Broker generated task decompositions and passed this information to the applicable Tier 2 Brokers, who forwarded the task announcement and decompositions to the applicable Mission agents.

In the baseline pre-bid subcontracting approach, each Mission agent receiving the task announcement then announced all subtasks that it could not perform through the Tier 1 Broker. In response, it received bids from all other applicable Mission agents for those subtasks. Each Mission agent combined these subtask bids and generated a task bid, which was forwarded through its Tier 2 Broker and the Tier 1 Broker to the User agent. The User agent ranked the complete bids and chose the one with the highest aggregated quality value. Because the bids were complete, this agent was pre-accepted and contracted. These results were compared to the results with MACRO pre-commitment subcontracting.

In the pre-commitment subcontracting approach, each Mission agent receiving the task announcement made a preliminary bid, including only the quality values for the subtasks

it could perform. The User agent ranked the preliminary bids by aggregated quality value. Starting with the highest preliminary bid, the User agent pre-accepted the corresponding Mission agent. The pre-accepted Mission agent then announced all subtasks that it could not perform through the Tier 1 Broker. In response, the Mission agent received bids from all other applicable Mission agents for those subtasks. It combined the subtask bids and generated a task bid, which was forwarded through its Tier 2 Broker and the Tier 1 Broker to the User agent. The User agent continued to pre-accept each preliminary bid in this manner, in order of decreasing quality, to generate the results presented below.

To illustrate the scalability of the MACRO approach, we compare the average number of messages required to reach the best final bid in MACRO pre-commitment subcontracting with the messages required in the baseline pre-bid subcontracting. These results are presented in Figures 8, 9, and 10. In each figure, the circle data points indicate pre-commitment subcontracting and the square data points are the result of pre-bid subcontracting. The 95% confidence interval is shown for each data point, although it is smaller than the data point marker in many cases.

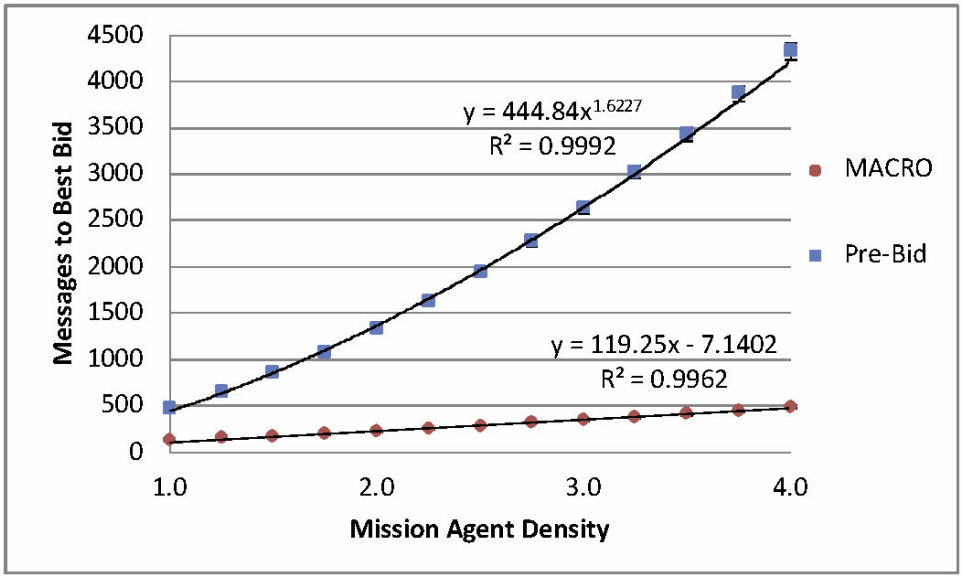

Figure 8: Scalability with respect to Mission agent density

Density	Decompositions	Subtasks	Pre-Bid Messages	To 75% Cutoff Pre-Accepts	To 75% Cutoff Messages
1.50	1-3	1-3	113 (84)	1	41 (20)
1.50	1-3	4-6	606 (326)	2	140 (41)
1.50	1-3	7-9	1505 (728)	2	226 (58)
1.50	4-6	1-3	276 (117)	3	109 (29)
1.50	4-6	4-6	1494 (408)	3	277 (47)
1.50	4-6	7-9	3714 (840)	4	520 (81)
2.00	1-3	1-3	168 (132)	1	51 (25)
2.00	1-3	4-6	956 (523)	2	172 (51)
2.00	1-3	7-9	2411 (1197)	2	282 (76)
2.00	4-6	1-3	414 (187)	3	136 (38)
2.00	4-6	4-6	2369 (673)	4	400 (72)
2.00	4-6	7-9	5988 (1381)	5	756 (129)

Table 1: Subtask allocation results (2000 trials each)

The appropriate cutoff for allowed number of pre-accepts depends on the configuration and requirements of the system. For example, the cutoff may be determined based on the desired percentage of tasks that must reach the best final bid. Consequently, this percentage also determines how many messages will be required, on average, in MACRO pre-commitment subcontracting with limited pre-accepts. For a given percentage of trials/tasks to reach the best final bid, the resulting cutoff requires an average number of messages closely correlated with the average number of messages required to reach the best final bid. Therefore, the pre-commitment subcontracting is illustrated by this average number of messages in Figures 8, 9, and 10. Table 1 provides specific pre-accept cutoff values for at least 75% of trials reaching the best final bid. These results cover a large range of potential system configurations/conditions and illustrate the dramatic reduction of messages in MACRO pre-commitment subcontracting compared to the pre-bid approach.

Figure 8 illustrates the scalability of MACRO pre-commitment subcontracting in terms of Mission agent density/overlap. The average number of messages scales linearly with Mission agent density. These results were generated with 2-6 (average 4) decompositions

per task and 2-6 (average 4) subtasks per decomposition. In comparison, the pre-bid subcontracting approach required an average number of messages that increased with Mission agent density to a power of 1.6, for the same parameter settings. For the highest Mission agent density of 4.0, pre-bid subcontracting required nearly 10 times the messages of pre-commitment subcontracting.

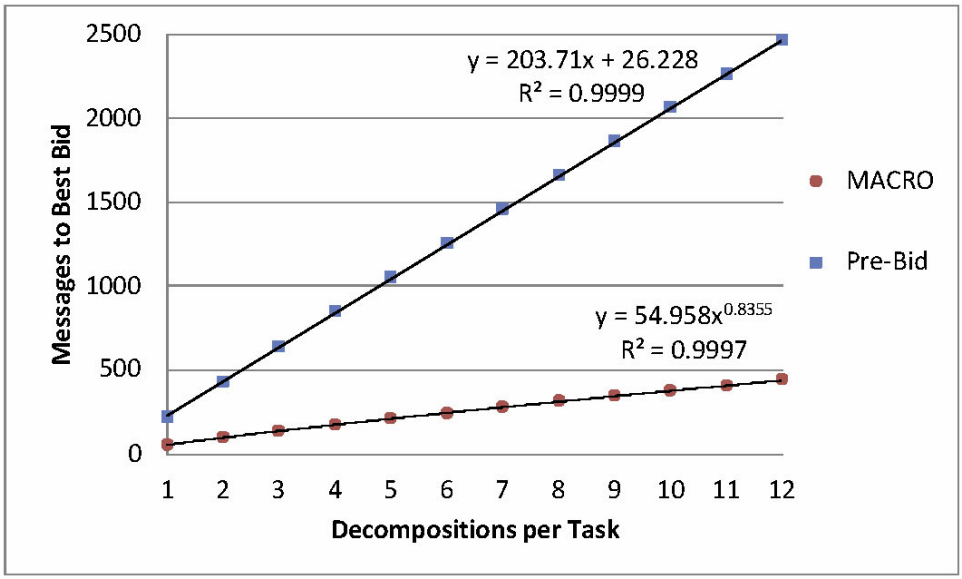

Figure 9: Scalability with respect to task decompositions

Figure 9 illustrates the scalability of MACRO pre-commitment subcontracting in terms of decompositions per task. The average number of messages scales slightly less than linearly (to the power of 0.8) with the number of decompositions per task. These results were generated with a Mission agent density of 1.5 and 2-6 (average 4) subtasks per decomposition. The pre-bid subcontracting approach, on the other hand, scaled linearly in terms of decompositions per task. For the highest decompositions of 12, pre-bid subcontracting required over 5 times the messages of pre-commitment subcontracting.

Figure 10 illustrates the scalability of MACRO pre-commitment subcontracting in terms of subtasks per task decomposition. The average number of messages scales slightly worse than linearly (to the power of 1.2) with number of subtasks. These results were generated

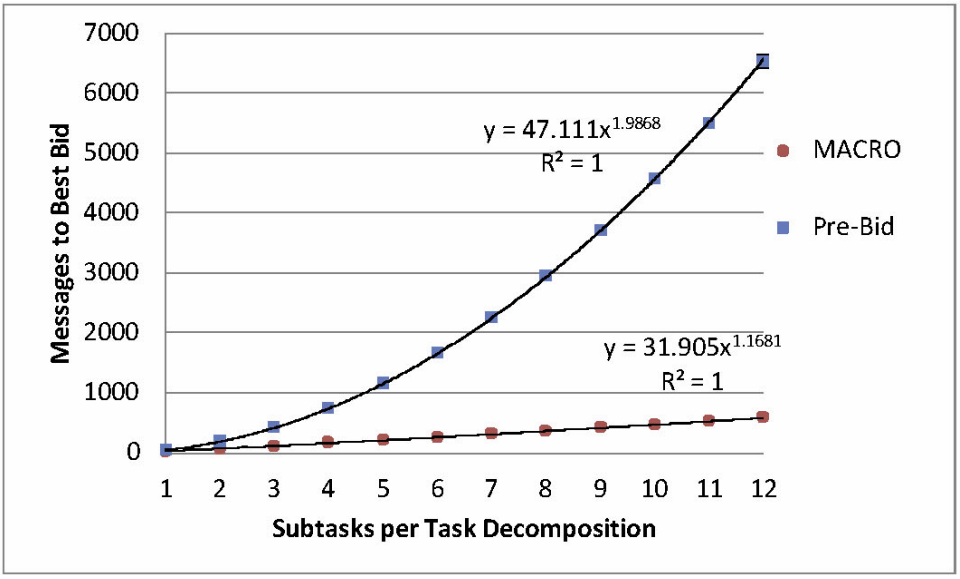

Figure 10: Scalability with respect to number of subtasks

with a Mission agent density of 1.5 and 2-6 (average 4) decompositions per task. In comparison, the pre-bid subcontracting approach required an average number of messages that increased with Mission agent density to a power of 2.0. For the highest subtasks of 12, pre-bid subcontracting required over 10 times the messages of pre-commitment subcontracting.

These results show that the MACRO limited pre-commitment subcontracting approach scales significantly better than the pre-bid subcontracting approach, and that the number of subtasks is the largest stress factor in terms of scalability. Further, the required number of messages to reach the best final bid in pre-commitment subcontracting are on the order of five times fewer than pre-bid subcontracting for likely sensor web system configurations and operating conditions. Considering that each trial illustrates subcontracting of a single task, the number of messages required for pre-commitment subcontracting (*e.g.* 50 to 200 for likely scenarios) are still higher than would be desired. This point is mitigated by the fact that these numbers were determined with plausibly worst case settings of static experimental parameters (*e.g.*, uncorrelated subtasks and task decompositions, all capable agents bidding on task/subtask announcements, and a large range of subtask quality with

the application of "sum" quality aggregation function) and by the increasingly low latency and high bandwidth connections available to agents communicating over the internet.

III.6.2 Allocation Optimization Experiments

This section presents the design and results of experiments that evaluate the performance of the MACRO task allocation mechanism under a variety of different potential system configurations and operating conditions. Performance is measured by evaluation of the MACRO task allocation using the fair and efficient sensor web metric proposed in Section III.3 and comparison with the performance of offline allocation generation under the same metric. These experiments validate our claims in Section III.4 that the evaluation of task marginal utility by MACRO Brokers during task auctions yields fair and efficient task allocation in the sensor web system. Further, we identify relevant trends for the major factors affecting allocation performance: 1) maximum ratio between User agent shares of sensor web resources, 2) number of announced tasks, 3) scenario length (system execution time), and 4) task execution time.

III.6.2.1 Experimental Design

As in the subcontracting experiments presented in Section III.6.1, we employed a simplified representation of subtasks and Mission agent capabilities to maintain the generality of our results and their applicability to other large-scale multi-agent systems Specifically, each subtask is randomly generated in a generic XY plane, and each Mission agent is capable of achieving subtasks in a square region within that plane. We define density of Mission agents as the average number of agents capable of requested subtasks in the system. In this setup, the Mission agent density is defined by their overlapping geographic regions, since subtasks are randomly generated in the XY plane. However, this simple representation allows our results to be easily extended to other applications where capabilities are significantly more complex and even unrelated to geography. Specifically, these results

are applicable to other systems in which MACRO allocation can be applied and where capability overlap/density can be determined.

With overlapping capabilities, multiple task decompositions, and a large number of total tasks, there are many possible task allocations in these experiments. Therefore, generation of the baseline results for comparison to the MACRO allocation mechanism required significant processing time, so we limit the number of Mission agents in these experiments to 20. However, we test a wide variety of system configuration parameters and performed enough trials for each data point to generate statistically significant results. In this experiment, the 20 Mission agents were grouped into regions of 4 agents with overlapping capabilities. Each group of 4 Mission agents was assigned to a Tier 2 Broker for a total of 5 Tier 2 Brokers.

The full implementation of task allocation in MACRO allows task announcements at any time with specified time-out values, and there is a potential lag time in updating system performance information, as mentioned in Section III.4. To simplify the generation and execution of scenarios in these experiments, task announcements are compressed into rounds, and each broker receives a complete update on allocations from the previous round at the beginning of a round. Further, the broker time windows were equal to the length of the trial, so all previously allocated tasks were considered in each round.

We also employ a simplified representation of bid characteristics, rather than explicitly model the large range of utility functions a User agent could apply to bids. The experiments employ a single, randomly generated, quality value for each subtask based on the Mission agent contracted or subcontracted to perform the subtask. In general, Mission agents may be unaware of how the User agent determines bid utility from individual bid characteristics. Therefore a single quality/utility value for each subtask suffices to represent the combined characteristics for that subtask in this experiment. Because Mission agents can include multiple possible task decompositions and subcontractors in their bids, the User agent can, in general, choose the one with the highest utility to itself.

In each trial, 20 User agents were employed to announce randomly-generated tasks and were divided into groups of 4. Each group of User agents was assigned to a Tier 1 Broker agent, for a total of 5 Tier 1 Brokers to evaluate and forward task announcements. With the MACRO Brokers' ability to aggregate task trees, any task can be decomposed into one or more sets of subtasks, where each subtask can be achieved by at least one Mission agent. Therefore, each task in this experiment had a corresponding set of randomly generated decompositions with a variable number of subtasks.

To maintain applicability to a variety of systems, most experimental scenario parameters are randomly generated within a uniform range of values. For example, the number of decompositions and each decomposition's number of subtasks are randomly chosen from the range [1,3] (*i.e.*, each task can be performed in one to three different ways and each way requires resources from one to three different Mission agents). Similarly, subtasks are generated randomly over the range of Mission agent capabilities. For simplicity, each Mission agent can only execute one subtask per time step, and each subtask requires the same quantity of resources.

The quality values for subtask bids are generated in the uniform, random range of [50, 100] to represent the variety in bid characteristics (*e.g.*, data rate and completion time) from different Mission agents capable of performing the same subtask. User agents employ a sum quality aggregation function to generate the quality of the task from the subtask qualities. In other words, the quality of the task is equal to the sum of qualities for each subtask, and this total quality is used as the agent's utility for the task/bid. This represents a situation in which the user utility of a bid depends on the characteristics of each subtask, rather than being determined by a single, limiting subtask, as with the minimum and maximum aggregation functions. Further, this results in decompositions with more subtasks having a higher possible utility. Since allocating more subtasks requires more resources, this follows the common pattern of higher utility being achievable by applying more resources to the achievement of the task.

III.6.2.2 Experimental Results

Each experimental run involved 100 trials (*i.e.*, 100 randomly generated scenarios, including User agent tasks over the entire length of the trial). In each trial, the User agents announced the tasks for that round (*i.e.*, the tasks with start times for the next time step) through their assigned Tier 1 Broker. The Tier 1 Broker generated task decompositions and calculated marginal utility values for each task. The Broker then passed this information to the applicable Tier 2 Brokers, who forwarded the task announcement and decompositions to the applicable Mission agents. The task auctions proceeded with each User agent providing pre-accepts to their three preferred initial bids and ultimately contracting to the most preferred final bid, if any.

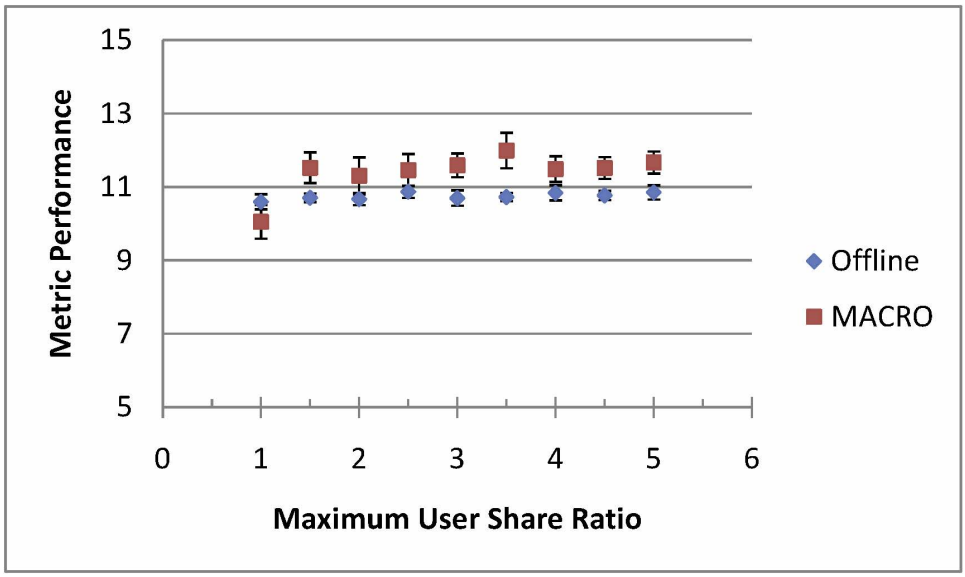

Figure 11: Performance with respect to User agent share ratio

To provide a value for comparison, each trial in the experiment included the random generation of 10,000 allocations with an offline algorithm (*i.e.*, random allocation of tasks to Mission agents using knowledge of all tasks to be announced over the entire course of the trial). The best allocation (under the sensor web metric) from those randomly generated allocations was used as the comparison data point and are labeled “Offline” in the figures

below. Of course, such a mechanism is far more computationally expensive than MACRO's task valuation and auction, as well as inappropriate for use in the *online* setting of sensor web task allocation (*i.e.*, allocation of tasks as they are announced at runtime). However, the offline algorithm provides a useful baseline for allocation performance.

During generation of the offline allocations, the same trend in (best) performance values was observed across all parameter settings. Specifically, the best performance value increased rapidly at first (*e.g.*, the first 500 randomly-generated allocations), largely leveling off by the first quarter of the allocation generation and only increasing slightly (*e.g.*, 1% to 5%) over the remaining 7,500 allocations. The sparsity of data points for any individual trial precludes quantitative assessment of individual patterns, but the consistent trend observed across these trials suggests that the "Offline" data points presented below are at least in the neighborhood of the optimal allocation.

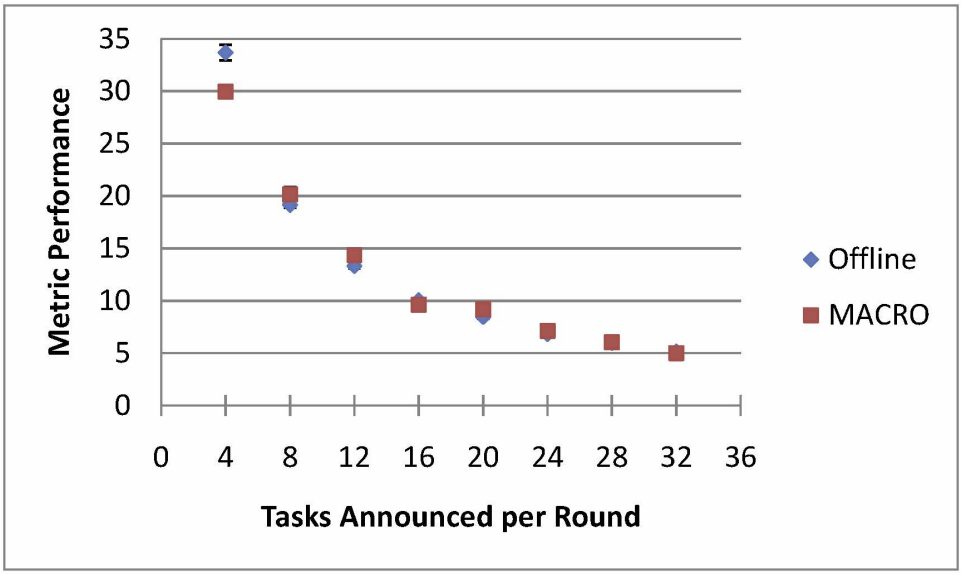

Figure 12: Performance with respect to tasks per round

Except for the parameter being varied in each experiment, the parameter settings in these experiments were: *Number of User Agents* = 20, *User Share Ratio* = [1, 1.5], *Number of Mission Agents* = 20, *Mission Agent Density* = 1.5, *Number of Tasks Announced per*

Round = 15, *Execution Time (steps) per Task* = [1, 2], *Number of Decompositions per Task* = [1, 3], *Number of Subtasks per Decomposition* = [1, 3], *Subtask Quality* = [50, 100], *Trial Length (steps)* = 20.

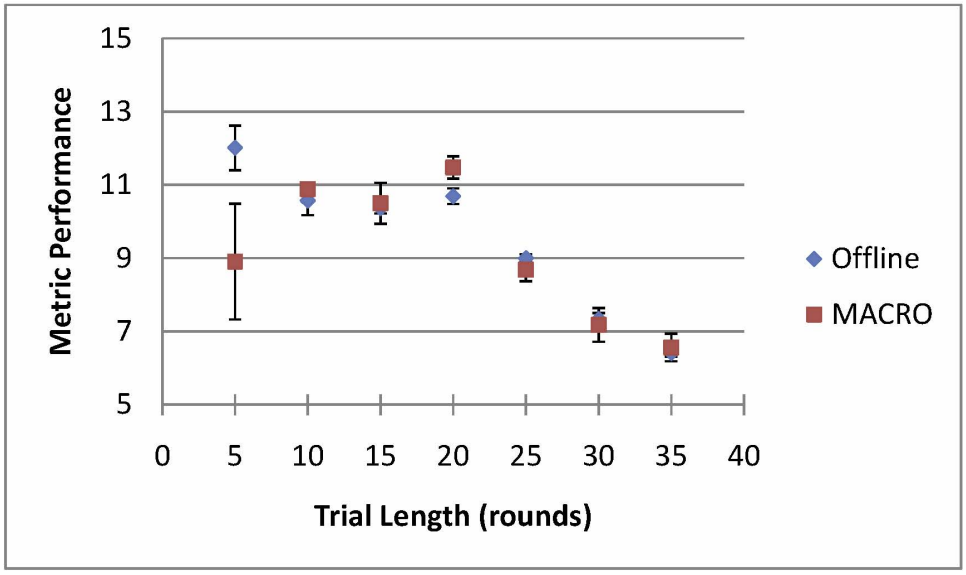

Figure 13: Performance with respect to trial length

Figures 11, 12, and 13 compare the mean performance of MACRO and Offline allocation with the 95% confidence interval shown for each set of trials. Figure 11 illustrates that performance under the sensor web metric does not vary significantly by the distribution of User agent shares of the sensor web. Further, Figures 11, 12, and 13 all indicate that the performance of MACRO task allocation is very close to that of the best Offline allocation under a wide variety of conditions. The one exception to this characterization is that MACRO performance for the shortest trials (5 rounds) is significantly lower than Offline performance. This suggests that MACRO marginal utility task valuation requires a minimum amount of past performance information (*e.g.* 5 to 10 rounds in these experiments) in order to be most effective in optimizing allocations.

Figures 12 and 13 show a decreasing metric value for both MACRO and Offline allocations as the number of tasks per round or length of trials increases. This trend illustrates that

the sensor web metric is sensitive to the number of announced but unallocated user tasks. Because the utility value of a user's tasks is normalized with respect to all announced tasks, the normalized value of allocated tasks can be decreased by large numbers of unallocated tasks. However, MACRO performance *relative* to the Offline allocations remained consistently high. Still, the sensitivity of the metric, and consequently of the MACRO marginal utility task valuation, to unallocated tasks could be detrimental to performance in situations where some User agents announce significantly more tasks than others. In future work, we could address this issue by weighting unallocated tasks during normalization to reduce their impact on marginal utility calculations.

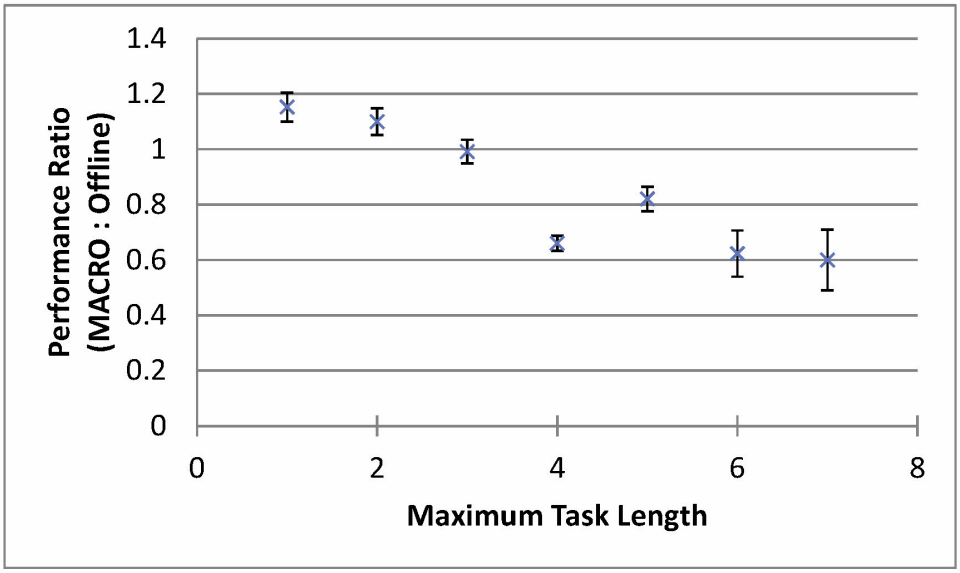

Figure 14: Performance with respect to task length

Although MACRO performance closely matched Offline performance across a range of user share ratios, tasks per round, and trial lengths, Figure 14 illustrates one parameter that had a significantly greater impact on MACRO performance than Offline performance. Figure 14 shows the ratio of MACRO to Offline performance across a range of task execution times. With longer execution times, MACRO performed significantly worse in comparison to Offline allocations. The MACRO task allocation performs poorly with long

execution times in this experiment because no decommitments were allowed. Even when an announced task was of significantly greater value, the Mission agents were not allowed to decommit from currently contracted and executing tasks in favor of the new task. Future work, detailed in Section VI.2, includes exploration of decommitment schemes to improve MACRO allocation performance with long-running tasks.

III.7 Summary

Large multi-agent systems with limited resources and heterogeneous users require an appropriate task allocation mechanism to simultaneously address issues of allocation efficiency (*i.e.*, utility to the system as a whole) and fairness (*i.e.*, individual user satisfaction). In this chapter, we presented a novel approach to fair and efficient allocation of complex tasks spanning multiple agents' capabilities and resources, while employing limited infrastructure computational resources. We presented the definition of a task allocation utility metric combining user satisfaction and system utility that is applicable to a global sensor web. We illustrated how this metric is employed by the MACRO Brokers to value tasks using an approximation of their marginal utility to the system. We also presented the details of MACRO's task auctions in a brokered, two-phase contract net, which includes a novel subcontracting approach to minimize message overhead. Overall, MACRO's task allocation mechanism provides a scalable method of allocating complex tasks to the resources of multiple agents and results in fair and efficient allocations. Finally, we presented results of experiments with MACRO task allocation that verify both its allocation performance and scalability. The generality of representation in these experiments ensures that these results can be extended to other systems in which hierarchically-decomposable tasks must be allocated to agents with overlapping capabilities.

CHAPTER IV

PLANNING AND SCHEDULING

A global sensor web is a large-scale, distributed system that spans multiple domains of agent operation. More specifically, the global sensor web requires global coordination among sensor networks to cooperatively achieve complex user tasks and adaptive operation of individual sensor networks situated in a variety of physical environments. In such systems, choosing a single planning and scheduling mechanism for all agents may be inefficient and impractical. In MACRO, complex task/goal achievement with resource constraints and time deadlines presents different planning and scheduling challenges at the two levels of agent operation.

As described in Section II.3, MACRO is structured as a two-level agent hierarchy with different planning and scheduling needs at each level: (1) at the *mission level*, Mission agents representing individual sensor networks coordinate to cooperatively achieve complex tasks spanning the resources and capabilities of multiple agents, and (2) at the *resource level*, Exec agents and other domain-specific agents adapt local operations within a sensor network to efficiently achieve goals given current conditions in dynamic, uncertain and resource-constrained environments. Therefore, agents at these different levels of the system operate in different contexts that imply different planning and scheduling requirements.

At the mission level of a sensor web MAS, new tasks are allocated based on user requests, as discussed in Chapter III, as well as from a sensor network's internal aims and priorities. Many tasks have a high degree of complexity a variety of ways by which they may be accomplished. Typically such tasks must employ resources from multiple sensor networks. In particular, hierarchical analysis helps deal with this complexity, both for problem/task representation by domain experts and for coordinated planning and scheduling among multiple agents. While the sensor web dynamically performs multiple tasks

simultaneously, the relative importance of completion metrics differs among tasks. For example, achievement of some tasks may need to be optimized for earliest completion time while others may need to be optimized for a particular data quality of service characteristic.

At the resource level of a sensor web MAS, many sensor networks incorporate distributed, real-time, embedded (DRE) systems functioning in environments where operating conditions and resource availability cannot be completely characterized *a priori*. Further, goals and priorities may change frequently due to evolving phenomena, changes in operating conditions, and the allocation of tasks at the mission level. Appropriate autonomous planning and scheduling capabilities at this level must produce plans to maximize expected utility with their limited computational resources for achieving local goals in a dynamic, uncertain environment. Moreover, they must provide re-planning/re-scheduling that can quickly revise scheduled plans during execution and prevent more expensive (in time and computational resources) re-planning/re-scheduling at the mission level.

With the appropriate autonomous planning and scheduling capabilities at each level, MACRO agents can facilitate the real-time collection and analysis of sensor data, even under changing environmental conditions and many concurrent science objectives. This chapter describes the planning/scheduling representations and mechanisms employed in MACRO. Section IV.1 discusses the related work in planning and scheduling relevant to the MACRO mission level and the choice of a distributed planning and scheduling representation/mechanism for Mission agents. Section IV.2 covers the related work in planning under uncertainty and incorporation of scheduling with planning relevant to the MACRO resource level. Section IV.3 identifies the unresolved challenges in planning and scheduling for MACRO resource-level agents, and Section IV.4 presents the MACRO solution to these challenges in the form of the Spreading Activation Partial Order Planner (SA-POP). Section IV.5 illustrates the use of SA-POP and the extension of MACRO resource-level planning with plan schemas for field agents in a representative sensor network. Section IV.6 presents an experimental verification of SA-POP planning performance. Finally,

Section IV.7 summarizes the MACRO research on planning and scheduling presented in this chapter.

IV.1 Mission-level Planning and Scheduling

At the MACRO mission level, Mission agents representing individual sensor networks negotiate with User agents to allocate user tasks. These tasks are often complex and may require the resources of multiple Mission agents for successful completion. In order to perform the allocated tasks, Mission agents must have appropriate distributed planning and scheduling capabilities to create effective plans and schedules for task/subtask achievement. Further, they must coordinate those scheduled plans with other Mission agents that have been allocated parts of the same overall task.

IV.1.1 Planning/Scheduling Problem and Task Representation

The representation of tasks and goals is an important first step in designing or choosing a system for planning and scheduling. At the MACRO mission level, the representation language must be expressive enough to represent important characteristics of the sensor web domain, including scheduling/resource information, utility with varying degrees or types of completion (*e.g.*, in terms of data rates and quality of service (QoS) criteria), and assignment to different agents with resulting constraints and commitments between them. Classic planning representations, such as STRIPS [43] and hierarchical task networks (HTN) [94], have been extended in various ways to allow representation of more complex domains/planning problems, although scheduling and resource constraints are usually an afterthought, or handled as a separate problem, when such languages have been employed. More recent representations, such as the planning domain definition language (PDDL) [75] have been extended to include scheduling and utility/solution-quality aspects of the problem as a primary part of its representation (*e.g.*, PDDL 2.1 includes temporal and numeric quality properties in its representation of tasks [45]). However, none of these representations are

designed to take into consideration issues primary to multi-agent planning/scheduling, such as assignment and synchronization.

Recently some multi-agent planning/scheduling representations have been created, such as MAPL [14], which allows efficient representation of partially ordered, scheduled, multi-agent plans. However, MAPL does not easily allow representation of plans/tasks at multiple levels of abstraction. The complex nature of tasks and plans at the mission level of a sensor web MAS makes hierarchical analysis important for dealing with this complexity, both for problem/task representation by domain experts and for coordinated planning among multiple agents. The Task Analysis, Environment Modeling, and Simulation (TÆMS) [55] language provides such a hierarchical task representation for multi-agent planning and scheduling.

The TÆMS language represents tasks as a hierarchical structure with a number of possible decompositions into subtasks, and ultimately to individual methods that can be directly implemented by agents, as illustrated in Figure 15. The TÆMS representation has many features making it applicable to planning and scheduling in a sensor web MAS. For example, rather than simply expressing task completion as a black or white (success/failure) outcome, it allows for shades of gray, *i.e.*, levels of quality in outcomes [69]. Given that there are many options for performing a high-level task, information on expected quality combined with resource and time considerations allows decision-theoretic criteria to guide task choice/decomposition by agents. Recent extensions to TÆMS also allow the specification of discrete probability distributions for task/subtask characteristics including outcomes and quality [116]. The probability distributions allow better preliminary planning decisions for uncertain environments and task execution, as well as replanning during the execution of a plan.

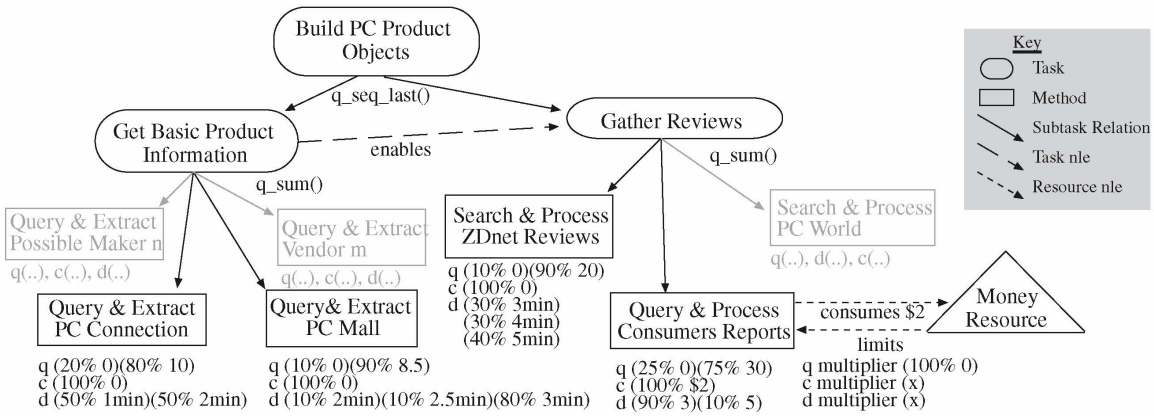

Figure 15: Example TÆMS task decomposition tree [116]

IV.1.2 Distributed Planning/Scheduling Algorithms

While an appropriate task/plan representation is vital to effective performance of distributed planning/scheduling in MACRO, the choice of how to perform planning and scheduling is equally important. Some of the key characteristics of an algorithm for this problem in sensor webs are: 1) scheduling under resource constraints is integral to the planning process; 2) subtasks will be distributed among agents, so synchronization and other communication activities must be planned; 3) multiple goals or tasks will be planned and execute concurrently; 4) utility should be (at least approximately) optimized when not all goals can be satisfied; and 5) replanning/rescheduling are necessary in a dynamic environment.

In recent years, numerous multi-agent planning algorithms with most or all of these characteristics have been proposed for various domains (*e.g.*, [29, 30, 32, 110]. One important division between these algorithms is whether they are designed for inherently cooperative agents willing to share sufficient details of their goals and plans to maximize a group/system utility measure or more self-interested/untrusting agents with private goals/plans [29, 32]. Mission agents in the sensor web may not wish to divulge all details of their internal goals or, in particular, the personal importance/utility of those goals. However, there is little reason for them to require complete privacy of their "personal" plans (*e.g.*, sharing resource and scheduling information, if not actual subtasks/subgoals) or

other constraints imposed by their internal goals. Since Mission agents can reasonably share pertinent plan information and constraints, it is not necessary to limit sensor web planning/scheduling with the privacy restrictions imposed by some algorithms, such as the sequential goal plan repair scheme proposed by van der Krogt and de Weerdt [110].

Another important restriction in some domains is limited communication, such as in the Shared Activity Coordination (SHAC) framework [20]. Again, this specialization and its resulting limitations on the planning/scheduling algorithm are not necessary at the mission level of a sensor web MAS, where Mission agents can communicate at any time within reasonable bandwidth restrictions. Instead, a more efficient, cooperative, multi-agent planning and scheduling algorithm can be applied in a sensor web MAS. Generalized Partial Global Planning (GPGP) [30] is such an algorithm, which has been successfully implemented for numerous real-world systems. The GPGP coordination technique shares partial plans between agents to build up (partial) global plans in order to recognize constraints, conflicts, and opportunities between individual agents' plans. This technique, introduced by Partial Global Planning (PGP) [35], is extended in GPGP to include hierarchical tasks and more flexibility in coordinating planning/scheduling activities between agents.

IV.1.3 GPGP/TÆMS for a Sensor Web MAS

GPGP uses the TÆMS representation to schedule tasks across agents and coordinate task interdependencies through inter-agent commitments. Although its name refers to "planning," GPGP is focused primarily on optimizing this coordination and scheduling of tasks across agents to maximize group utility. The traditional planning aspects of the problem are solved primarily through parsing of TÆMS task networks and other domain/design-dependent internal activities of each agent. With its emphasis on coordination and scheduling, GPGP is particularly useful in the face of interdependent tasks, such as those that result from joint high-level goals across agents, and limited time/resources. While agents may be more or less self-interested and have independent goals in addition to joint goals, GPGP is

only effective in increasing utility to the extent that agents are willing to make commitments to other agents. In particular, most of the coordination protocols designed for GPGP rely on the agents being willing to make commitments without any direct benefit/reciprocity for themselves. Further GPGP uses the interdependence of tasks (or potential decomposition of tasks) across agents to drive its coordination. This interdependence results from subtasks that contribute to the solution of a joint, higher-level goal/task as specified in TÆMS hierarchical task graphs.

Within the realm of systems designed to achieve global solutions with highly cooperative agents, GPGP/TÆMS has a great degree of domain-independence, making it applicable to a wide array of applications. It is a particularly powerful solution for applications in domains with: interdependent tasks that can be distributed across agents, local autonomy issues, resource constraints, time constraints, and dynamic environments. Further, through the choice of GPGP coordination protocols and the design of the agents' domain problem-solving engine, the framework can be highly tailored to specific applications. For example, GPGP/TÆMS has the flexibility to allow meta-level or additional coordination activities to take place between agents based on their domain-specific problem-solving strategies [69]. As such, it should be possible to effectively integrate it with coordination and negotiation via contract net for task/subtask allocation. Lesser et al. also suggest the potential for significantly reduced communication/coordination overhead through application of organizational roles implicitly or explicitly used to shape GPGP coordination protocols [69]. In the sensor web, teams organized by the allocation of subtasks from a particular user task can provide an important reduction in overhead and complexity of GPGP. Because agents with independent goals generally cannot achieve a significant increase in utility by sharing their plans, many sensor web planning/scheduling interactions can be limited to the members of each task team.

The GPGP family of coordination protocols relies heavily on the TÆMS task representation. A TÆMS task structure is a hierarchical task network that includes meta-data on

the composition/decomposition of subtasks and on implementable "methods" that form the leaves of the task decomposition tree. At the root of a TÆMS task tree is the high-level task or goal to be achieved by one or more agents. The root task may be decomposed into one or more subtasks that can be performed to achieve the root task. Similarly subtasks can be decomposed into simpler subtasks, and so on, until the decomposition tree reaches the methods that can be directly implemented by the agent. A quality accumulation function (QAF) defines how the decomposition is performed (*e.g.*, decomposition by choosing any one of the subtasks, any subset of the subtasks, all of the subtasks, or all of the subtasks in sequence) and how the quality of subtasks is combined to give the quality of the parent task (*e.g.*, sum, minimum, or maximum). Meta-data for the methods in the task tree include quality, cost, and duration, each of which is described by a discrete probability distribution allowing uncertainty in the outcome of method execution. Execution time constraints/windows from the goal specification, coordination with other agents, and scheduling can also be associated with the tasks and methods.

In addition to method meta-data and decomposition meta-data, TÆMS task structures include relationships between tasks/subtasks called nonlocal effects (NLEs). NLEs can define hard constraints between tasks, such as "enables" or "disables," defining causal or other precedence constraints. However, "soft NLEs" can define other relationships between tasks where the execution of one affects the outcome of another but does not indicate a hard constraint. For example, "facilitates" and "hinders" indicate the specific effects of one task on another where the first task increases or decreases (respectively) the probable quality or execution speed of the second task. Finally, TÆMS task structures can include consumable and non-consumable resources and their relationships to the tasks/methods. For example, a method may require a quantity of a resource to execute or the absence of a resource may affect the quality or duration of a method without being strictly required for it to execute.

With a task structure represented in TÆMS, GPGP provides a general approach and specific set of mechanisms for coordination of plans and scheduling among a set of agents.

The GPGP approach is for each agent to build up a local view of its tasks/plan include non-local tasks performed by other agents and their relationship to its local tasks through coordination with other agents. This partial view of the global MAS plan/schedule allows agents to reason about their local activities in light of related activities performed by other agents. The GPGP coordination mechanisms allow an agent to recognize important patterns in the relationships between local and non-local tasks and specifies an appropriate coordination response in terms of modifying task structures, communicating information, and making commitments to other agents. Because the coordination mechanisms are applicable to specific task relationships or patterns, multiple coordination mechanisms can be used concurrently. Generally, more than one mechanism is necessary for effective and efficient coordination and the specific set of coordination mechanisms appropriate to a MAS are domain-dependent.

A number of GPGP coordination mechanisms have been designed for specific domains since the original introduction of GPGP. However, the most generally applicable coordination mechanisms are the original five introduced with the GPGP concept [30]:

- *Updating non-local views* – When a relationship between a local task and a non-local task is detected, communicate the relevant task structures to the other agent. This is the basic coordination mechanism for building up partial global views in the MAS. Detection of the inter-agent task relationships is domain-dependent. In a sensor web MAS, such as MACRO, subtasks of a high-level user task are determined by agents from the known task decomposition trees and can be allocated via the contract net protocol. In such a system, inter-agent task relationships are known by the agent managing the high-level task and can be communicated to the other agents contracted to perform subtasks. Upon further decomposition/scheduling of subtasks, additional inter-agent task relationships may be identified and communicated.

- *Communicate results when they will be used by others* – When another agent can use the result of a local task known by the relationship between the local task and a

non-local task or a commitment to provide the result, communicate the result when execution of the local task completes. In sensor web MAS, many of the "results" communicated between tasks will actually be in the form of a sensor data stream being transmitted for additional processing or analysis by another agent. Extending this mechanism to allow data streams in addition to one time result communication is an obvious improvement for sensor web MASs. Further, with agents organized into teams to perform a high-level task, it is unlikely that a task result will be useful to another agent where there is not already a commitment to provide that result between team members. Therefore, the detection step of this mechanism need not search for relationships where no commitment has been made.

- *Handling simple redundancy* – When multiple agents intend to execute the same method, one agent is randomly chosen to execute the method and commit to sending the result to the other agents. When tasks and subtasks are allocated via contract net, as in MACRO, this coordination mechanism is not likely to be useful. An agent already intending to perform a method would generally have the lowest incremental cost for its execution and would therefore have been contracted to perform it for any other tasks as well, so redundant execution of methods would be rare.

- *Handling hard relationships from the predecessor side* – When a hard relationship/-NLE exists between two tasks at different agents, the agent with the predecessor task should schedule the task locally and then communicate a commitment to complete the task by the scheduled time to the other agent. Clearly this coordination mechanism is important for coordination in a sensor web MAS so that the agent with the successor task can accurately schedule its execution in light of the expected completion time for the predecessor task.

- *Handling soft relationships from the predecessor side* – Similar to a hard relationship, when a soft relationship/NLE exists between two tasks at different agents, the agent

with the predecessor task should schedule the task locally and then communicate a commitment to complete the task by the scheduled time to the other agent. In this case, the agent with the successor task may determine through scheduling that it is preferable to execute its task before the predecessor task will be completed, but it can make that decision with knowledge of the expected completion time for the predecessor task.

GPGP coordination relies on the use of existing or generated TÆMS task structures and an independent scheduler/planner that can generate an appropriate task decomposition and schedule for execution. The scheduler designed for use with TÆMS and successfully employed in many systems using GPGP is the Design-To-Criteria (DTC) scheduler [114, 115]. The DTC scheduler is designed to provide soft real-time scheduling for the combinatorial problem of optimally decomposing and scheduling a TÆMS task. Although DTC does not use an anytime algorithm, it does provide soft real-time results by using goal criteria and heuristics to reduce the search space of scheduled plans.

To provide a tractable search space of scheduled decompositions, the DTC scheduler employs four major techniques [115]:

- *Criteria-direct focusing* – The goal criteria (*e.g.*, reducing uncertainty in task outcome, maximizing quality, and minimizing completion time) is used to focus processing on partial solutions likely to achieve the provided goal criteria. For example, if the goal criteria is to reduce uncertainty in task outcome, DTC attempts to schedule possible decomposition of a task with highly certain outcomes and only considers less certain decompositions if that scheduling fails.

- *Schedule approximations* – Alternative decompositions are generated from the task tree with approximate quality, cost, and duration distributions before actually ordering methods in those decompositions. Combined with criteria-directed focusing, this

allows DTC to attempt the more computationally complex task of scheduling only for decompositions that are likely to meet goal criteria.

- *Heuristic action ordering* – Heuristics for leveraging positive task interactions in scheduling and avoiding negative ones are used to minimize the work performed in scheduling potential decompositions. Although the resulting algorithm is approximate, this allows the scheduling complexity to be reduced to polynomial time in the worst case.

- *Heuristic error correction* – The action ordering heuristics and schedule approximations used by DTC can result in schedules that do not achieve the task or meet goal criteria. Additional heuristics are used to repair correctable scheduling errors or else identify invalid schedules and prevent them from being returned.

The combination of GPGP coordination mechanisms, TÆMS task representation, and DTC scheduling provides a powerful solution to distributed planning/scheduling problems. For a sensor web MAS, the identified GPGP coordination mechanisms (updating non-local views, communicating results, predecessor handling of hard relationships, and predecessor handling of soft relationships) allow effective coordination of local planning/scheduling across multiple agents working together to achieve a high-level task. Moreover, the hierarchical task representation in TÆMS limits the complexity of codifying and reasoning about sensor web domain information. Finally, the DTC scheduler provides an effective method for local planning/scheduling of TÆMS tasks that is required for GPGP coordination.

However, the research on TÆMS and similar hierarchical task representations assumes a system with a unified design team. In an open MAS, an agent's domain knowledge, including TÆMS task trees, is specific to their individual purposes and capabilities, without full knowledge of goals and tasks throughout the system. Therefore, interoperability among independently-designed agents requires an enhancement to existing hierarchical task representations to allow translation and aggregation of task information across agents. In

MACRO, TÆMS is extended with the OGC SensorML standard [9] for describing sensor capabilities and processing of sensor data. This allows Broker agents to aggregate Mission agent TÆMS task trees and translate between User agent requests and Mission agent capabilities, as described in Section V.4.

Moreover, MACRO resource level agents use a different planning/scheduling representation and mechanism (SA-POP decision-theoretic planning and scheduling described in Section IV.4). Employing the two different representations and forms of planning and scheduling requires the definition of an appropriate plan/schedule coordination mechanism between the mission and resource level. This coordination between Mission and Exec agents is detailed in Section V.5.

IV.2 Resource-level Planning and Scheduling

At the MACRO resource level, Exec agents and other domain-specific agents operate in environments where operating conditions and resource availability cannot be completely characterized *a priori*. Further, goals and priorities may change frequently due to evolving phenomena, changes in operating conditions, and the allocation of tasks at the mission level. Autonomous planning and scheduling for resource-level agents must produce plans to maximize expected utility for achieving local goals in a dynamic, uncertain environment while operating on shared processors with limited computational resources. To adapt local operations for action failure and changes in resources availability, agents must also have re-planning/re-scheduling capabilities that can quickly revise scheduled plans during execution.

IV.2.1 Planning under Uncertainty

In real-world domains, such as a sensor network, obtaining complete knowledge about the state of the environment is often impractical or impossible. Therefore, many planners

for uncertain environments make simplifying assumptions about the environment and attempt to construct effective plans using probabilistic domain knowledge. Some planners assume a known initial environment and utilize actions with probabilistic effects. Others assume that actions have deterministic effects, but the starting environment is unknown, and the challenge is to determine or approximate the initial environment [73]. Overall, planning under uncertainty can be categorized by the observability of the environment: (1) no-observability planning, (2) partial-observability planning and (3) full-observability planning. Full-observability planners assume a known initial environment and actions whose effects can be immediately observed. No-observability planners, or conformant planners, try to generate a plan that has the highest possibility of success assuming no feedback from the environment about plan success. Partial-observability planners assume partial feedback. Partial- and full-observability planning, which are most applicable to the sensor network environments in a sensor web, are usually approached in one of two ways: (1) contingent planners construct plans that include contingency subplans or actions based on the effects of actions, and (2) re-planners construct a single, complete plan and re-plan during execution, if observed conditions invalidate part of the plan.

Probabilistic planning problems are often represented as a Markov Decision Process (MDP) or a Partially Observable Markov Decision Process (POMDP) [8]. Many early probabilistic planners attempted to find optimal solutions using MDP/POMDP representations, but, in practice, such planners do not scale [88]. Even for moderately-sized POMDP problems, optimal solutions require very efficient mechanisms and/or additional assumptions about the soluble set of problems. For example, Incremental Pruning [16] was shown to significantly outperform other exact POMDP algorithms. However, in general, these algorithms are not competitive on planning problems [73]. Newer approaches like Grid-Based algorithms proposed by Bonet [8] have shown promise, but remain untested on benchmark International Planning Competition (IPC) domains.

Another common solution to probabilistic planning is to adapt the successful Graphplan framework for a probabilistic environment (*e.g.*, PGraphplan and TGraphplan [7]). PGraphplan is an optimal dynamic programming planner; unfortunately, PGraphplan could not fully leverage the plan graph structure that provides the efficiency of Graphplan [71]. Other optimal probabilistic planners suffer similar performance problems for large planning problems. Another approach is to sacrifice optimality by using a relaxed problem that can be more easily solved. For example, TGraphplan is an online planner which solves a relaxation of the probabilistic problem via a traditional Graphplan algorithm; therefore it is not an optimal planner, but runs at essentially the same speed as deterministic Graphplan. Like the probabilistic extensions of Graphplan, planning-as-satisfiability algorithms have been extended to stochastic satisfiability (SSAT) planners. For example, ZANDER [73] is capable of solving finite-horizon problems in partially-observable environments and has performed well against contemporary probabilistic planners [73].

Many probabilistic planners, (*e.g.*, derivatives of Graphplan like PGraphplan and TGraphplan [7]) only allow uncertainty about action outcome. However, some planners also incorporate uncertainty about the environment or observations of the environment (*e.g.*, C-SHOP [10] using hierarchical planning and Drips [51], which produce contingent plans). In some sensor networks, the outcomes of data sensing and processing actions may depend on environmental characteristics that cannot be directly measured or can only be approximated. Therefore, the ability to represent and utilize probabilistic information about both the environment and action outcome is an important feature in a planner for MACRO resource-level agents. Of course, the ability to handle unexpected occurrences, such as failures and transient phenomena, in uncertain sensor network environments is also vital.

Many of the most successful IPC planners for fully- and partially-observable domains have been re-planners, rather than contingency planners [123]. FF-Replan [122] and RFF [108] employ the efficient deterministic planner Fast Forward on a deterministic relaxation

of the probabilistic environment for planning and re-planning. When an action has a consequence not predicted by the deterministic relaxation, these planners produce a new plan based for the current situation. This is often a very computationally efficient approach, and works especially well in domains with no dead-end states. In domains with dead-end states, a re-planner can end up in situations where there is no longer a valid solution, but which a contingent planner could have avoided. However, in most sensor networks, sensors are far more common than actuators. Consequently, there are few or no actions that have direct effects on the local environment, making dead-end states rare. Therefore, re-planners are a promising option for the MACRO resource level.

Another important characteristic for sensor webs is the production of highly parallel plans because of the variety of long-running data gathering and analysis actions that must operate concurrently. Partial-order planning (POP) produces partially-ordered plans that generally exhibit a high degree of parallelism as opposed to plans generated by other techniques like POMDP and state-space planners [79]. The parallelism of partial-order plans also promises greater flexibility in scheduling for limited resource environments and tight time deadlines. While the use of partial-order causal-link (POCL) planners is promising for sensor networks, state-space planners have often outperformed POCL planners. Fortunately, REPOP [79] and VHPOP [124] have demonstrated that POP can utilize some of the advanced heuristics of contemporary state-space planning to be competitive.

IV.2.2 Integrating Planning and Scheduling

Many complex, real-life, automated planning domains, such as sensor networks, have to satisfy both functional (*i.e.*, planning) and scheduling constraints. In such situations, many have chosen to separate the planning and scheduling/resource aspects of the problem (*e.g.*, [105] and [36]). This approach works well when the resource/time constraints are relatively loose or there are relatively few alternatives in the planning process that could

use fewer or different resources. However, in some domains, interleaving planning and scheduling can provide significant performance improvements [5].

In particular, domains with tight resource constraints and multiple software implementation options (with different resource requirements) for actions may benefit from the interleaving of planning and scheduling. In sensor networks, this approach has the potential to improve planning efficiency by utilizing resource constraints to limit the planning search and choice of action implementations. Interleaving planning with scheduling also allows for the translation of certain constraints in the problem description into resources, which schedulers are usually better at handling than planners. Of course, scheduling must be leveraged at the appropriate point during planning for its integration to be beneficial. For example, creating complete schedules at each step of the planning search, when they will likely have to be re-scheduled at the next step, is unlikely to be an efficient approach.

IxTeT [67] uses partial-order planning and allows interleaving resource conflict resolution with the planning process. However, its incorporation of scheduling/timing information into the action representation makes it difficult to separate the issues of action choice and action implementation that are important to sensor network domains where data gathering and processing actions may be implemented with a variety of software components. Garrido *et al.* present a more flexible integrated planning and scheduling framework [48], which can employ a variety of planning strategies (*e.g.*, POCL planning, hierarchical task network (HTN) decomposition, or planning graphs). However, the existing planning capabilities in this framework do not provide probabilistic planning for uncertain environments. Further, the choice of a scheduling mechanism that complements the chosen planning mechanism is important to performance and applicability to a particular domain, such as sensor networks.

Given the limited computational resources of sensor network systems, a least commitment strategy of only scheduling tasks in highly constrained areas (by schedule and

resource usage) of the plan is a promising approach. In particular, Laborie's energy precedence and balancing constraint propagation techniques [66] allow just such a strategy for scheduling choices. Further, such constraint propagation techniques can leverage the ordering information available in a partial-order plan to efficiently schedule actions even without concrete time limits (*e.g.*, in sensor network data processing activities whose end times depend on the evolving phenomena in the environment).

IV.3 Unresolved Challenges

Section IV.2 illustrates the promise of interleaving partial-order planning (POP) and resource-constraint propagation for MACRO resource-level agents. Further, re-planning and re-scheduling abilities are necessary for effective operation in a sensor network environment. Because multiple resource-level agents may share the same limited computational resources in a MACRO sensor network, their planning and scheduling needs are best resolved by providing a shared planning/scheduling service. The appropriate design of this planning/scheduling service presents significant research challenges.

Given the limited computational resources of many sensor networks, designing and incorporating an efficient heuristic to guide the POP process in the search for valid, high expected utility plans is a significant challenge. Section IV.4.1 resolves this challenge by extending a spreading activation mechanism to calculate a heuristic for use in partial-order planning. Further, efficiently integrating scheduling with POP presents another significant challenge in the design of a resource-level planning/scheduling service. Section IV.4.2 resolves this challenge by interleaving resource-constraint propagation with planning, while ordering tasks to resolve scheduling conflicts only in highly constrained areas of the plan.

To effectively adapt sensor network operations to current goals and conditions, MACRO resource-level agents must be able to both execute traditional, individual actions, as well as assemble and deploy applications comprising configured software components for data processing and analysis. Representing and producing plans combining both of these aspects of

sensor network operation presents another challenge in the design of SA-POP. Section IV.4 describes how the functional and resource-usage aspects of both actions and parameterized software components are represented in SA-POP task networks and task maps. Section IV.5 illustrates how MACRO resource-level agents can employ SA-POP-generated plans with both actions and software components in a representative sensor network. Further, Section IV.5 illustrates how MACRO resource-level agents can employ both SA-POP and plan schemas in sensor networks where resource-limited field hardware precludes the use of the full planning and scheduling capabilities in SA-POP.

IV.4 The Spreading Activation Partial-Order Planner

To resolve the planning and scheduling challenges for MACRO resource-level agents operating in a dynamic, uncertain, resource-constrained, sensor network environment, we designed and implemented the Spreading Activation Partial Order Planner (SA-POP) [59]. SA-POP provides autonomous planning and scheduling capabilities as a service for resource-level agents operating on shared computational resources. SA-POP starts with a goal specified by a MACRO agent and generates a plan with a high expected utility based on the current and expected conditions in the environment. The resulting plan is made up of *tasks*, which are (1) actions that can be directly executed by the agent, or (2) parameterized software components that can be deployed using the middleware infrastructure in MACRO. A *goal* can include multiple goal conditions each with an associated utility value as well as scheduling (time) constraints. Each task may have more than one possible implementation, especially for software component tasks, where different implementations have different time and resource requirements. Figure 16 illustrates the SA-POP mechanism in conjunction with MACRO agents and a middleware infrastructure.

The planner uses the links in the task network to generate a plan for the given goal. It also employs a spreading activation mechanism [4] to compute a heuristic values for each task that guide its choice of tasks. However, the plan must also execute within the time and

resource constraints of the system. As each task is added to the nascent plan, it is associated with an implementation from the task map based on resource and time constraints. If scheduling constraints cannot be met using the highest expected utility tasks, lower expected utility tasks will be used. The complete plan, including assemblies of configured software components (*i.e.*, applications) and executable actions is captured as an *operational string*, which specifies the necessary tasks, a suggested implementation for each task, the control (ordering) dependencies, the data (producer/consumer) dependencies, and any necessary start and end times for tasks.

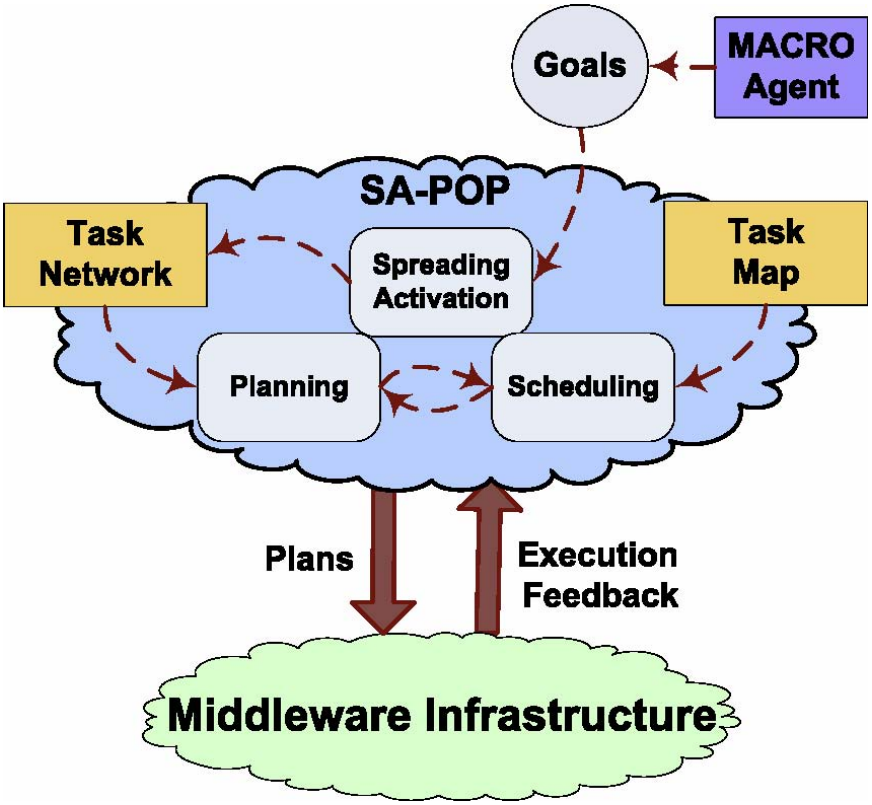

Figure 16: Planning/re-planning structure of SA-POP

To ensure planned action execution and applications do not violate resource and scheduling constraints, SA-POP requires knowledge of the expected resource consumption and

execution time of each possible implementation of a task, *i.e.*, its *resource signature*. SA-POP includes a *task map* to associate each task with a set of implementations and their individual resource signatures. As each task is added to the nascent operational string, SA-POP performs scheduling in critical areas of the plan using Laborie's resource constraint propagation [66].

For SA-POP to choose appropriate tasks to achieve a goal, it must know which preconditions must be satisfied for each task, its input/output data dependencies, if any, and the pertinent effects that result from its operation. Uncertainty as to whether tasks will execute successfully and produce the relevant output or effects is captured via conditional probabilities associated with the preconditions and effects of a task, respectively. Together, the input/output definitions, preconditions/effects, and related conditional probabilities define the *functional signature* of the task.

The functional signature of each task, and consequently all task dependencies, are captured in a domain-specific *task network*, as illustrated in Figure 16. A task network is a directed graph that represents both tasks and conditions (preconditions, data input, effects, and data output) with the links encoding the requisite probability information. In general, the task network can be constructed by a domain expert using domain-specific modeling tools, such as the SA-POP Modeling Language (SAML), which is implemented in the Generic Modeling Environment (GME) [58]. Given the task network, current values for conditions, and a goal, SA-POP's spreading activation mechanism computes a heuristic value for each task that guides the search for a high expected-utility (EU) plan.

IV.4.1 The Spreading Activation Mechanism

Figure 17 illustrates an example task network, which contains condition nodes and task nodes with directed links to indicate the preconditions and effects of each task. Condition nodes represent Boolean variables and include a probability that reflects their believed likelihood of being true/false. Environmental/system conditions (*e.g.*, a particular sensor

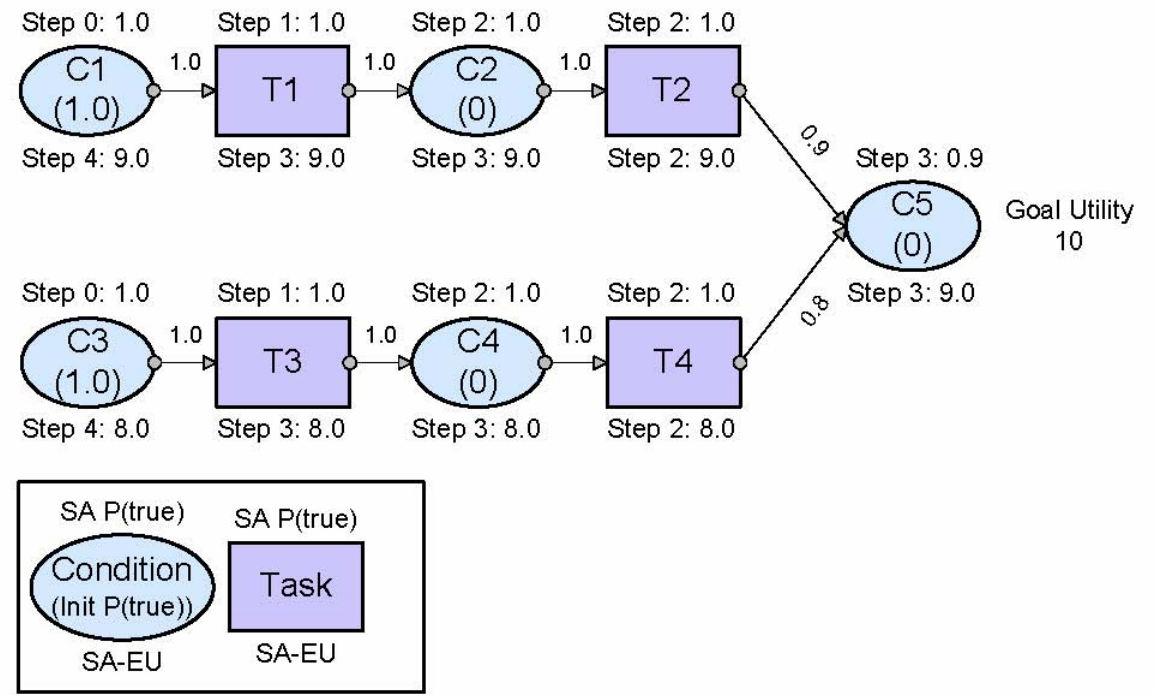

Figure 17: Example spreading activation task network

is active or environmental condition has been observed) and data input/output (*e.g.*, a data stream from a particular sensor or software component) are represented as condition nodes.

The condition nodes representing preconditions or required data input for a particular task have links to it, which encode relevant probabilistic information. The weight, w_{ij}, of the link from a condition node, c_i, to a task node, t_j, defines the likelihood that t_j executes successfully given c_i:

$$w_{ij} = \frac{P(t_j^s|c_i = true) - P(t_j^s|c_i = false)}{P(t_j^s|c_i = true) + P(t_j^s|c_i = false)}, \tag{IV.1}$$

where t_j^s indicates that task t_j executes successfully. This encoding supports *hard constraints* (weight = 1 (-1)), *i.e.*, the condition must be true (false) for the task to succeed, and *soft constraints* (weight < 1 (> -1)), *i.e.*, a true (false) value of the condition increases the probability of task success. Soft constraints can be used to model inferred conditions in uncertain environments, especially in situations where actual preconditions cannot be

sensed directly but are probabilistically related to other conditions that can be sensed. For example, an imperfect (noisy) sensor for detecting an environmental condition necessary for the success of a task can be modeled using a soft constraint.

The task nodes have outgoing links to their effects and/or data outputs. The weight, w_{jk}, of the link from a task node, t_j, to a condition node, c_k, indicates the probability that c_k will be true/false after t_j executes, *i.e.*, t_j^x:

$$w_{jk} = \begin{cases} P(c_k = true|t_j^x) & \text{if } t_j \text{ sets } c_k = true \\ -P(c_k = false|t_j^x) & \text{if } t_j \text{ sets } c_k = false, \end{cases} \tag{IV.2}$$

Agent goals are expressed as desired conditions with associated utility values. SA-POP first uses the spreading activation mechanism to generate heuristic values for individual tasks that can contribute to achieving the specified goal conditions. The spreading activation mechanism simultaneously performs a forward propagation of probabilities and a backward propagation of utilities to produce this spreading activation expected utility (SA-EU) heuristic for each task node.

Forward propagation uses an iterative approach of calculating the likelihoods of task success and the probability of resulting effect conditions. For a given state of the network S_t, the set of conditions in the network $[c_1, c_2, ..., c_N]$ have an initial probability value indicating the believed probability that they are true. In order to efficiently calculate the SA-EU heuristic, the spreading activation mechanism makes the simplifying assumption of independence among the preconditions of a task, *i.e.*, $P(c_1, c_2, ..., c_N) = \prod_{i=1}^{N} P(c_i)$ and $P(c_1, c_2, ..., c_N|t_j^{success}) = \prod_{i=1}^{N} P(c_i|t_j^{success})$.

With this assumption, Bayes rule gives the probability of successful task execution as:

$$P(t_j^{success}|S_t) = P(t_j^{success}) \prod_{i=1}^{N} \frac{P(t_j^{success}|c_i)}{P(t_j^{success})}. \tag{IV.3}$$

Equation IV.3 can be rewritten with respect to a current condition, c_i, as

$$\frac{P(t_j^{success}|c_i)}{P(t_j^{success})} = \frac{\begin{array}{c}P(t_j^{success}|c_i = true)P(c_i = true|S_t)+ \\ P(t_j^{success}|c_i = false)(1-P(c_i = false|S_t))\end{array}}{P(t_j^{success})}. \tag{IV.4}$$

This ratio represents the likelihood of a task succeeding given a condition's current value versus the prior probability of task success. If the value of c_i increases the probability of a task being successful, the ratio will be > 1. If the condition decreases the likelihood, the ratio will be < 1. If the condition must be true for the task to execute successfully, but its current value is 0 (*i.e.*, false), the task's posterior probability will be 0.

Equation IV.3 defines the probability of a task's success. Spreading activation propagates this probability forward to the effects of a task. The future probability of a condition due the execution of a task is calculated as:

$$P(c_k|t_j^{executed}, S_{t+1}) = w_{jk}\frac{P(t_j^{success}|S_t)}{P(t_j^{success})}. \tag{IV.5}$$

When multiple effect links apply to a condition, the maximum probabilities for both true and false values are maintained for the current state:

$$P_{max}(c_k = true|S_{t+1}) = \max_j P(c_k|t_j^{executed}, S_t) \tag{IV.6}$$

$$P_{max}(c_k = false|S_{t+1}) = -\min_j P(c_k|t_j^{executed}, S_t). \tag{IV.7}$$

Thus spreading activation propagates the probabilities from S_t, maintaining the maximum ones for each condition in S_{t+1}, and so on.

Only keeping the maximum probabilities for true and false at each condition node allows spreading activation to efficiently compute the SA-EU heuristic for each task. However, this optimistic calculation is no longer valid when the task producing the highest

probability for a condition cannot be employed due to scheduling constraints. Further, orderings enforced on the execution of tasks during planning and scheduling can invalidate the optimistic probability values maintained by spreading activation. It is these issues that motivate the experiment detailed in Section IV.6 to prove the effectiveness of the SA-EU heuristic, even when scheduling constraints prevent the use of the highest SA-EU valued set of tasks.

In order to calculate the SA-EU heuristic value of a task, spreading activation combines the forward-propagated probability values with backward-propagated utility values from the goal condition(s). The backward-propagation of utility highlights potential subgoals for achievement of the goal condition(s). The utility values in spreading activation can be either positive, indicating a (sub)goal of the condition as true, or negative, indicating a (sub)goal of the condition as false. Conditions that can have no impact on goal condition achievement will have utility values of 0.

During backward-propagation of utility, the utility value of a condition is first used to determine the reward that tasks affecting it receive. We represent the initial probability (for true) of a condition, c_k, as $P(c_k = true|S_t)$. If this probability equals 1, then tasks that set the value to true are left without a reward, since they are redundant. Similarly, if the probability of false is 1, any task that would change the value to false will not receive any award. The reward is computed as:

$$R(t_j|c_k,S_t) = \begin{cases} w_{jk}P(c_k = false|S_t)U^+(c_k) & \text{if } (w_{jk} > 0) \\ w_{jk}P(c_k = true|S_t)U^-(c_k) & \text{if } (w_{jk} < 0) \end{cases} \quad \text{(IV.8)}$$

This reward is multiplied by the task's probability of success to give the spreading activation expected utility (SA-EU) value for the task's effect on c_k:

$$U(t_j|c_k,S_t) = P(t_j^{success}|S_t)R(t_j|c_k,S_t). \quad \text{(IV.9)}$$

These values are summed for all effects of a task to produce its total SA-EU value:

$$U(t_j|S_t) = \sum_k U(t_j|c_k, S_t). \tag{IV.10}$$

Spreading activation then propagates a task's SA-EU value backward to its preconditions, making them potential subgoals. Since positive and negative utilities for conditions indicate utility for different values of the condition, they are maintained separately. For example, the positive SA-EU of condition c_i is calculated as a sum of the SA-EU from all tasks that require it to be true as a precondition.

$$U^+(c_i|S_t) = \sum_j w_{ij} U(c_i|t_j, S_t) \quad \text{if } (w_{ij} > 0). \tag{IV.11}$$

Figure 17 illustrates spreading activation probability and utility (SA-EU) values in a simple network. At each step of spreading activation both forward propagation of probabilities and backward propagation of utilities occur (first for conditions then for tasks). In the first step of spreading activation, tasks T1 and T3 acquire a probability of 1.0 from their preconditions, C1 and C3, respectively. In step 2, this probability is propagated to their effects, C2 and C4, and then to tasks T2 and T4. At this point, because T2 has a probability of 1.0 and a reward of 9.0 from C5, its utility (SA-EU) becomes 9.0. Similarly, T4 has a utility of 8.0. The utility values of T2 and T4 are then propagated back through their preconditions, to tasks T1 and T3.

Since T2 has a higher SA-EU value than T4, SA-POP would prefer to use it to satisfy the goal condition C5. This would ultimately result in the highest expected utility plan of executing T1 and then T2. However, in an alternative scenario, the initial probability of C1 being true might only be 0.8 instead of 1.0. In this scenario, the propagated probability would result in an SA-EU value for T2 of 7.2 instead of 9.0. Since T4 would have the

higher SA-EU value of 8.0, it would instead be chosen to satisfy C5. The resulting SA-POP plan of executing T3 followed by T4 would have a higher expected utility than the plan chosen in the original scenario because of the uncertainty in condition C1's value.

IV.4.2 Integrated Planning and Scheduling

To generate a plan, SA-POP performs partial-order causal-link (POCL) planning, while preferring tasks with higher SA-EU heuristic values to satisfy open conditions. It also leverages ordering information from the nascent partial-order plan to propagate resource constraints at each planning step. Specifically, SA-POP applies a modified version of Laborie's energy precedence and balancing constraint propagation techniques [66] to detect potential resource violations and constrict potential execution time windows for tasks. To speed the search for a plan, SA-POP makes scheduling decisions only in critical regions of the plan (*i.e.*, where time and resources are highly constrained). This allows it to recognize insoluble scheduling conflicts early, while leaving other scheduling decisions until a complete plan is generated.

A plan (or operational string) generated by SA-POP is constrained by four different kinds of "links," which specify an ordering between two task instances. A *Causal Link* indicates one task must execute before the other based on a system/environmental condition. These links are imposed during planning when a task is chosen to satisfy an open, non-data condition and are only applicable between task instances within a single operational string. A *Data Link* indicates both tasks must execute simultaneously because they both operate on the same data stream. These links are imposed during planning when a task is chosen to satisfy an open, data condition, and are only applicable within a single operational string. *Threat Links* indicate one task must execute before the other to avoid resource violations. These links are imposed during planning to resolve causal link threats, and are valid within, as well as across, operational strings. *Scheduling Links* indicate one task must execute before the other to avoid resource violations. Scheduling links are imposed during scheduling

to prevent potential resource violations, and are valid within, as well as across, operational strings. We also define one additional type of constraint on task instances in a plan, a *Time Constraint*. This constraint specifies a required start-by or end-by time, and it is specified in the goal input for a required condition.

In addition to links and time constraints in the nascent plan, SA-POP also maintains some additional time and ordering information A *Time Window*, which consists of an earliest time and latest time. Start and end time windows are maintained for each task instance. A *Ranking*(a, b) is a comparison between task instances a and b, which describes the order in which they will be executed given the current knowledge of the plan. There are four rankings used by SA-POP: *Before* (a will complete its execution before b begins executing), *After* (reciprocal of the Before relation), *Simultaneous*(both a and b will start and end their executions strictly at the same times), and *Unranked* (a and b potentially overlap in their execution).

The rankings between all pairs of tasks are maintained in a precedence graph [66], determined by the links and time windows in the current plan. The precedence graph maintained by SA-POP differs from Laborie's definition primarily in that it is defined between pairs of task instances rather than *events*, which are the individual start and end times of task instances. This simplification allows more efficient scheduling calculations for *discrete resources*, *e.g.*, memory which is used during a task execution and then freed. SA-POP currently does not deal with *reservoir resources*, which are resources that can be arbitrarily produced or consumed. The existing framework could be extended, however, to apply resource constraint propagation and search for reservoir resources with some modifications to the Laborie balance constraint [66].

Before planning and scheduling, SA-POP uses spreading activation as a preprocessing step to generate the SA-EU heuristic values for tasks. In addition to guiding SA-POP's search for high expected utility plans, SA-EU values also speed up the SA-POP planning and scheduling process by pruning portions of the planning search space. Any task with an

SA-EU of 0 cannot be successfully executed (or is not on any path to the goal conditions, but such tasks would not be considered during the partial-order planning anyway).

In the worst case, the computational complexity of spreading activation is $O(n^3)$ because, at each step, each of n nodes may update probability and utility values from its neighbors ($O(n)$ for highly connected networks), and to fully propagate probabilities and utilities can require $O(n)$ steps. However, SA-POP only performs spreading activation once as a pre-processing step, and when time or processing power available for planning is scarce, SA-POP can limit the number of spreading activation steps. Spreading activation is an anytime algorithm, and stopping after x steps corresponds to generation of SA-EU values that account for potential task sequences with lengths up to $\frac{x}{2}$.

After generation of the SA-EU heuristic values, SA-POP uses four mutually-recursive algorithms to produce a scheduled plan. The first two algorithms, *Plan* and *ResolveThreats*, correspond to steps in traditional POCL planning. The other two algorithms, *Schedule* and *ResolveRes*, perform scheduling through constraint propagation and resolution of the most critical potential resource violations.

Algorithm 1 summarizes the SA-POP planning and scheduling process. Each step in the generation of a plan involves all four mutually-recursive algorithms:
(1) *Plan* first chooses an *open condition* (which is a goal or subgoal unsatisfied in the current plan), then chooses a task that can achieve the open condition, and next instantiates the task. (2) *ResolveThreats* resolves causal link threats by promotion or demotion of task instances in a recursive manner. This operates in the same manner as traditional POCL planning algorithms.
(3) *Schedule* chooses an implementation for the task instance and propagates resource constraints to constrict time windows.
(4) *ResourceRes* adds scheduling links to resolve potential resource violations.

Algorithm 1 (*Plan*) begins with the provided goal conditions as the set of open conditions. Since data manipulation tasks are resource intensive and execute concurrently with

Algorithm 1 Plan()

```
1: Add all goal conditions to the list of open conditions
2: if ∃ open condition then
3:    cond ⇐ Choose an open condition (prefer data conditions)
4:    while ∃ task to achieve cond in TaskNetwork do
5:       task ⇐ Choose a task
6:       Remove cond from open conditions
7:       Add preconditions of task to open conditions (prefer high EU)
8:       while ∃ instance of task do
9:          inst ⇐ Choose existing instance of task
10:         Add causal links for inst
11:         if ResolveThreats(inst) then
12:            return true
13:         end if
14:      end while
15:      inst ⇐ New instance of task
16:      Add causal links for inst
17:      if ResolveThreats(inst) then
18:         return true
19:      end if
20:   end while
21: else
22:   for all res ⇐ Resources do
23:      for all inst ⇐ Task instances do
24:         if ¬ResolveRes(inst, res, 0, false) then
25:            return false
26:         end if
27:      end for
28:   end for
29:   return true
30: end if
31: return false
```

other tasks on the same data stream, SA-POP gives priority to data flow conditions, which enables early detection of irresolvable resource violations in a nascent plan, thereby pruning the search space.

In choosing a task to satisfy the current open condition, SA-POP prefers tasks with higher expected utility values However, there is also a threshold on the probability of a task achieving the open condition, and those tasks falling below the threshold are ranked strictly by probability rather than expected utility. This ranking represents a tradeoff between the total expected utility, which may accumulate from multiple goals, and the likelihood of achieving a particular subgoal under consideration.

The total expected utility of a task may be accumulated from multiple goals, so a high expected utility task may have only a low probability of achieving the given subgoal. Conversely, simply choosing the task with the highest probability of achieving the given subgoal does not consider the potential benefit of using another higher total expected utility task that could aid in the achievement of other goals. In some cases, such a task may be directly used by the plan in the course of achieving multiple goals. Even when this is not the case, the incorporation of such a task makes the plan more robust because the task might become necessary to achieve other goals due to unexpected changes in the environment.

Algorithm 2 (*ResolveThreats*) recursively resolves causal link threats, as in traditional POCL planning. Specifically, a causal link is of the form $T1$-$(C1 = ValueX) \rightarrow T2$, meaning task instance $T1$ achieves condition $C1 = ValueX$ as a precondition for task instance $T2$. A causal link threat occurs when another task instance, $T3$, has an effect of $C1 = ValueY$, where $ValueX \neq ValueY$, and is not ordered (by the current set of causal, data, threat, and scheduling links) with respect to $T1$ and $T2$. To resolve this threat, $T3$ must be ordered either before $T1$ (demotion) or after $T2$ (promotion). *ResolveThreats* thus recursively resolves all causal link threats by promotion or demotion.

With Algorithm 3 (*Schedule*), SA-POP moves from POCL planning to scheduling that meets available/expected resource requirements. SA-POP first determines the change in

Algorithm 2 ResolveThreats(Task Instance *inst*)

```
1: if ∃ causal link threats involving inst then
2:    threat ⇐ Choose a causal link threat
3:    Resolve threat by promotion
4:    if ResolveThreats(inst) then
5:       return true
6:    else
7:       Resolve threat by demotion
8:       if ResolveThreats(inst) then
9:          return true
10:      end if
11:   end if
12: else
13:   if Schedule(inst) then
14:      return true
15:   end if
16: end if
17: return false
```

Algorithm 3 Schedule(Task Instance *inst*)

```
1: while ∃ implementation of task in TaskMap do
2:    impl ⇐ Choose an implementation
3:    Add or update inst in the Precedence Graph
4:    for all res ⇐ Resources used by impl do
5:       ENERGY ↪ constrict time windows
6:       for all Task start and end events do
7:          BALANCE ↪ compute min resource level
8:       end for
9:       BALANCE ↪ constrict time windows
10:      BALANCE ↪ impose scheduling links
11:      if Irresolvable res violation then
12:         Try next implementation
13:      end if
14:      if ¬ResolveRes(inst, res, Thresh, true) then
15:         Try next implementation
16:      end if
17:   end for
18: end while
19: return false
```

potential resource usage for each implementation (from the task map) of a task instance, given current rankings from the precedence graph. The *resource impact score* of an implementation is the sum across all resources of the percentage of resource capacity that would be utilized if all potentially overlapping task instances were to be executed concurrently. The implementation with the smallest impact on potential resource availability, as measured by the resource impact score, is chosen to implement the task instance. This preference for low resource impact implementations is analogous to the least constraining value heuristic often used in general constraint satisfaction problems.

For example, there may be multiple compression components that are associated with a "Compress Data" task, each with a different tradeoff between memory and CPU usage requirements. If other potentially overlapping task instances in the current plan (*i.e.* those that are *Unranked* or *Simultaneous* in the precedence graph with respect to the given task instance) would require nearly, or even more than, the full capacity of the memory resource, but the CPU resource is relatively underutilized, SA-POP would prefer a component that uses less memory, even if it has a high CPU usage profile.

In Algorithm 3 (*Schedule*), SA-POP also employs Laborie's energy precedence and balance constraint propagation techniques [66] modified for planning in DRE systems. These techniques are largely complementary and apply to different precedence sets with respect to a given task instance. The energy precedence constraint propagation can constrict time windows (and consequently derive more accurate rankings), even with relatively loose time windows that are prevalent early in planning. It applies to a task instance's start (end) time window based on the resource usage of all other task instances in its *Before* (*After*) precedence set.

The balance constraint propagation applies to a task instance based on the other task instances in its *Unranked* and *Simultaneous* precedence sets. For the discrete resources, precedence graph, and links utilized by SA-POP, the constraint propagation differs from the Laborie calculations [66] with one major simplification: for discrete resources, only the

start and end events of (potentially) overlapping task instances (*Unranked* and *Simultaneous* precedence sets) must be considered in the balance constraint. With this simplification, SA-POP uses Laborie's balance constraint propagation [66] to constrict time windows and detect irresolvable resource violations.

Algorithm 4 ResolveRes(Task Instance *inst*, Resource *res*, Criticality *thresh*, Flag *plan*)

```
1:  if ∃ res violation in Unranked(inst) > thresh then
2:     inst_x ⇐ Choose a task instance
3:     BALANCE ↪ compute min resource level for x
4:     while ∃ instance set ⊆ Unranked(inst_x) do
5:        inst_set ⇐ Choose a set of task instances
6:        for all inst_y ∈ inst_set do
7:           Add link inst_y → inst_x (inst_x → inst_y)
8:        end for
9:        if ResolveRes(inst, res, thresh, plan) then
10:          return true
11:       end if
12:    end while
13: else
14:    if plan then
15:       if Plan() then
16:          return true
17:       end if
18:    else
19:       return true
20:    end if
21: end if
22: return false
```

Algorithm 4 (*ResolveRes*) implements SA-POP's search for resolutions to potential resource violations. SA-POP employs two significant simplifications in the calculation of resource levels for events: (1) a potential resource conflict can only be resolved by imposing an ordering constraint (scheduling link) between two task instances (*i.e.*, by ordering an end event before a start event) and (2) given (1), only a worst case (minimum) resource level and best case (maximum) resource level need be calculated for each task instance (corresponding to the level when its start event occurs).

For a given task instance, the minimum resource level is the resource capacity less the sum of: its resource usage, the resource usage of all unranked task instances that *may* overlap its start (*i.e.*, those whose minimum start time is less than the given task instance's maximum start time), and the resource usage of all simultaneous task instances. The maximum resource level is the resource capacity less the sum of: its resource usage, the resource usage of all unranked tasks that *must* overlap its start (i.e. those whose minimum start time is less than its maximum start time and whose minimum end time is greater than its maximum start time), and the resource usage of all simultaneous task instances.

The heuristic for choosing the most significant resource violations is provided by the scheduling criticality score [66] for each task instance: $crit(x) = max(0, -L_{min}(x))/(L_{max}(x) - L_{min}(x)Q\Delta t_{start}(x)$ where, L is a resource level, Q is a resource capacity, and Δt is the length of a time window. After choosing the most critical task instance, x, a set of task instances from $Unranked(x)$ that can be ordered before x is chosen to reduce the criticality of x below a specified threshold. The heuristic for choosing these task instances is provided by preferring those with highest pair-wise criticality values [66]: $crit(x,y) = -commit(y,x)/R(y)$ where, R is a resource usage value, and $commit(y,x)$ is a measure of the commitment implied by ordering the end event of y before the start event of x. This heuristic provides a least commitment strategy (consistent with SA-POP's preference for highly-parallel, minimally-constrained operational strings) by balancing the preference for low commitment with the preference for high reduction in potential resource violations.

During execution of these four mutually-recursive algorithms, SA-POP employs backtracking whenever an irresolvable resource violation is discovered, or an attempt is made to impose a link inconsistent with the rankings in the precedence graph. This makes SA-POP a *complete* planning algorithm, because it will always find a valid plan if one exists. Further, SA-POP's use of heuristic to guide choices for tasks (SA-EU value), implementations (resource impact score), and early resolution of potential resource violations (scheduling criticality score). Moreover, the use of task SA-EU values to guide task choice results in

plans of high expected utility even when scheduling conflicts prevent SA-POP from employing the tasks with the highest SA-EU values. The experiments in Section IV.6 verify SA-POP's expected utility performance even under tight scheduling constraints.

IV.5 Case Study: SEAMONSTER Sensor Network

Recent work on the MACRO resource level has been driven by collaboration with the *South East Alaska MOnitoring Network for Science, Telecommunications, Education, and Research* (SEAMONSTER) [42], which is a glacier monitoring sensor network. This sensor web monitors and collects data regarding glacier dynamics and mass balance, watershed hydrology, coastal marine ecology, and human impact/hazards in and around the Lemon Creek watershed. The collected data is used to study the correlation between hydrology, glacier velocity, and temperature variation at Lemon Creek.

The SEAMONSTER sensor network, illustrated in Figure 18, includes sensors with weatherized computer platforms that are deployed on the glacier and throughout the watershed to collect data of scientific interest. The data collected by the sensors is relayed to a cluster of servers primarily via wireless networks for processing, correlation, and analysis. The data processing applications (*e.g.*, GPS data analysis for glacier dynamics and hydrology data analysis for watershed monitoring) run on the server cluster, while small computational platforms for controlling sensors and acquiring data with minimal processing are deployed in the field. As opposed to the multi-processor spacecraft constellation in MMS described in Section II.3.2, SEAMONSTER is an example of a sensor network in which relatively powerful servers are connected to a highly distributed set of sensors with extremely limited computational resources.

This type of sensor network provides a different set of challenges for MACRO resource agents than a spacecraft constellation sensor network. Specifically, differences between the servers and field nodes require a different organization of MACRO resource-level agents and agent services. In the server cluster, significant computational resources are available

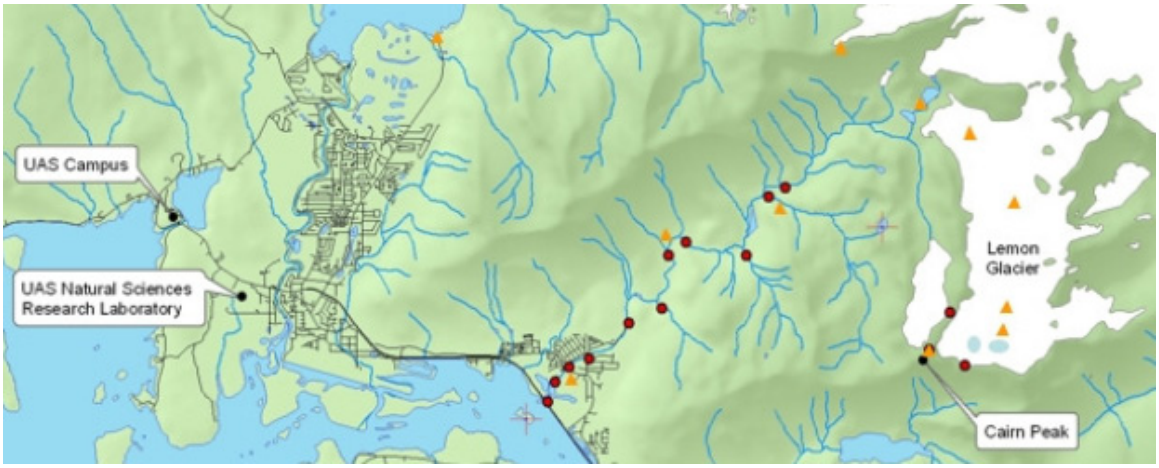

Figure 18: SEAMONSTER field sensor deployment

to direct the tasks performed by computationally limited field resources. These servers are shared among the data processing applications, sensor web agents, and other SEAMONSTER applications, such as a database and web server. As such, they are controlled as a resource group similar to that in the spacecraft scenario. In the field, however, computational resources are severely limited and consequently require software solutions with a smaller footprint and lower computational complexity. A full resource group implementation of multiple agents and powerful agent services, such as SA-POP and RACE, is impractical for the field nodes.

However, to address the problem of rapid, effective reaction to local changes in environmental conditions and resource availability—while respecting system-wide science goals—the field nodes must be capable of some autonomous adaptation and action. Since local field agents have limited computational resources, extensive planning and scheduling (*e.g.*, the SA-POP service), is not feasible for rapid reaction to local changes. Instead, in SEAMONSTER and similar sensor networks with servers and extremely limited field platforms, MACRO field agents are provided with a set of template plan schemas that cover a range of conditions and local goals to which they are applicable [60]. Server-based agents can then provide the field agents with the current set of local goals to pursue, and the task

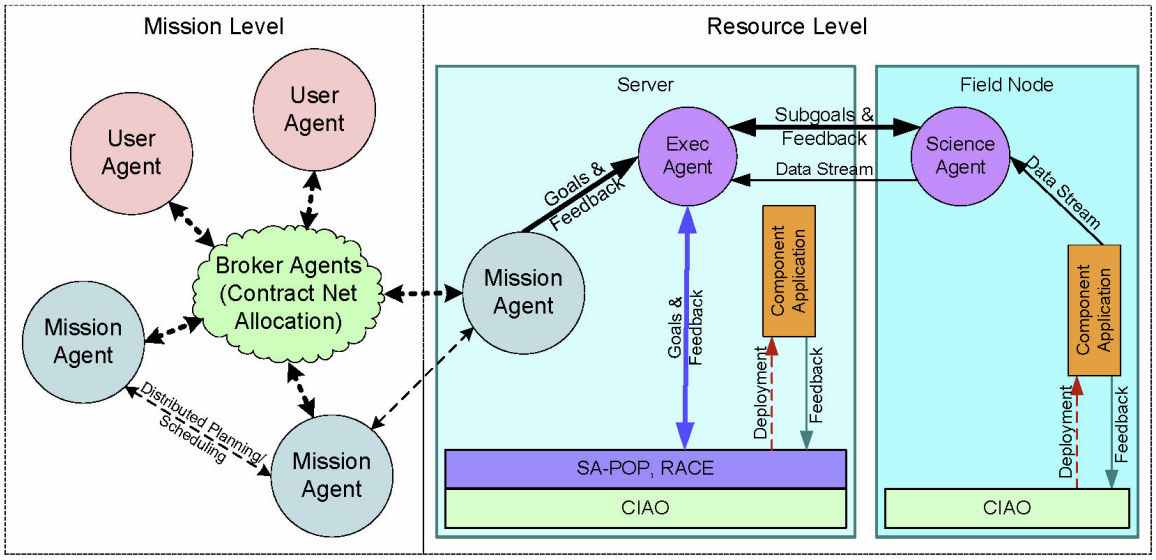

Figure 19: MACRO configuration for SEAMONSTER

of the field agent becomes the simpler choice of an appropriate set of schemas to follow given current conditions and resource constraints. When conditions change unexpectedly during schema execution, MACRO field agents stop executing an instantiated schema that is no long applicable and instantiate and execute a new one based on current conditions and the relevant goal.

This alternative MACRO resource-level architecture is illustrated in Figure 19. The Exec agents on the server use SA-POP, described in Section IV.4, to decompose goals into subgoals to be achieved at the server or by individual field nodes and to plan/schedule their achievement. With information from field agents about current conditions and local activities, SA-POP produces scheduled, high expected utility plans to achieve all subgoals, including both the selection/configuration of software components for data processing on the server and actions/reconfiguration at the field nodes.

In the nominal case, where an appropriate set of actions and software deployments at a field node is already represented in a schema available to the field node's Science agent, the subgoal and scheduling information is simply passed to that field agent. Otherwise, the plan can be packaged as a new schema and distributed to the field agent along with the applicable

subgoal, conditions, and scheduling information. This process allows the server-based agents to do extensive planning and optimization for the current set of science objectives, as well as entirely new objectives, while the computationally limited field agents can choose among their pre-packaged schemas to intelligently react to changing local conditions and resource availability in light of current science objectives.

The SEAMONSTER-inspired work on MACRO resource-level agents resulted in the addition of field agents using planning schemas and the Action/Effector framework designed to allow extensible implementation of low-overhead hardware-dependent actions. The design and overhead results for the initial Action/Effector framework implementation are described in papers [C-5], [C-3], and [C-4]. These results illustrate the overhead and extensibility benefits of a middleware-based agent infrastructure combining server agents employing SA-POP and RACE services with field agents using planning schemas and the Action/Effector framework on resource-limited field hardware.

IV.6 Experimental Evaluation

This section presents the results of SA-POP performance in terms of the expected utility (EU) of its plan compared to the expected utility of the best possible plan in a variety of randomly-generated domains. These results validate our claims that SA-POP's use of the SA-EU heuristic to guide its planning process results in high expected utility plans even when scheduling requires SA-POP to forgo its first-choice plan.

IV.6.1 Experimental Design

To illustrate the general effectiveness of the SA-EU heuristic, these experiments used 500 randomly-generated spreading activation task networks for each set of domain parameters. For each of these task networks, 10 trials were run with a randomly chosen goal condition and random initial conditions, providing 5,000 trials for each set of domain parameters.

It is difficult to simulate the tight resource and time constraints in real-world sensor webs across a variety of randomly-generated domains. Random generation of resource usage and duration for tasks generally resulted in either no schedulable plan or SA-POP's first-choice plan being schedulable. Therefore, we simulate the tight, real-world, scheduling constraints by rejecting the partially-ordered set of tasks that made up SA-POP's *first-choice plan* as unschedulable. This forced SA-POP to continue planning until it found a schedulable plan (its *preferred plan* for the purposes of these experiments).

For comparison to SA-POP's preferred plan, we exhaustively generated every possible plan that could achieve the randomly-chosen goal condition with a non-zero probability. We applied the same schedulability criteria in this exhaustive compilation of plans by rejecting any plan that contained the "unschedulable" set of tasks from SA-POP's first-choice plan. SA-POP's performance on each trial is measured as the ratio of the expected utility of SA-POP's preferred plan to the highest expected utility of any possible plan.

The domain parameters used to generate task networks were: 1) size of the network, 2) average number of precondition links per task (node in-degree), and 3) average number of effect links per task (node out-degree). For each network size, nodes were split equally between tasks and conditions. Each link was generated by randomly choosing the condition and task node for the link with an Erodos-Renyi random graph generator.

To simplify the generation of networks, all links had a positive weight and all preconditions were hard preconditions (*i.e.*, a precondition link weight of 1.0). Effect link weights were generated from a random, uniform distribution of [0.8, 1.0] because most real-world tasks are only included in a system if they have a reasonable chance of successfully achieving their intended effects. Since all links in the network were positive, initial conditions for each trial were chosen by randomly assigning half of the conditions in the network to true (other than the goal condition), and the rest to false.

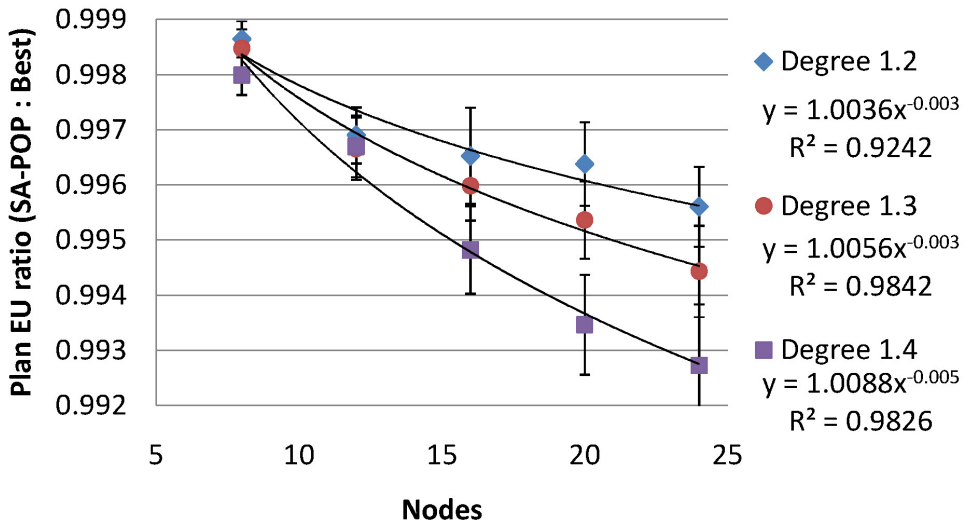

Figure 20: SA-POP plan expected utility performance

IV.6.2 Experimental Results

The experimental results in Figure 20 illustrate SA-POP's performance across a variety of network sizes and node in/out degrees. The 95% confidence interval is shown for each data point. Although SA-POP is capable of quickly generating plans for much larger networks, in order to exhaustively generate all possible plans, we had to limit network size to 24 nodes in these experiments. Similarly, as node in and out degree increases, the number of possible plans increases exponentially. However, the highest node in/out degree in these experiments (1.4/1.4) is close to the average node in/out degree of 1.44/1.69 on International Planning Competition problems, after conversion to spreading activation task networks.

Since the greater connectivity of networks with higher node degree generally results in more complex problems with more potential plans, SA-POP's performance is higher for the lower node degrees. However, as illustrated in Figure 20, SA-POP's average performance is above 99% of the best plan EU for all runs. More importantly, for larger real-world domains, the experimental results show a power regression to be a closer fit than a linear regression for all data sets. If the power curve holds for larger network sizes, it forecasts

SA-POP EU performance to be over 96% of the optimal EU for 10,000 node networks with 1.4/1.4 in/out node degrees.

IV.7 Summary

In MACRO, User agents request tasks in OGC SensorML format. Broker agents translate these tasks into the TÆMS hierarchical task representation and manage negotiations for task allocation. Mission agents employ GPGP coordination and DTC planning/scheduling to cooperatively achieve allocated tasks at the mission level. Mission agents implement their plans to achieve tasks/subtasks by providing goals to resource-level agents for achievement in the local environment.

To operate efficiently and effectively at the resource level requires the use of available probabilistic domain information for generation of high expected utility plans and the integration of an efficient scheduling mechanism. For MACRO resource-level agents, the shared Spreading Activation Partial Order Planner (SA-POP) service provides decision-theoretic planning and scheduling to support their autonomous adaptation of sensor network operations. SA-POP employs the SA-EU heuristic in an integrated planning and scheduling process designed to produce high expected utility plans in uncertain, resource-limited domains. Together, the planning and scheduling capabilities of MACRO mission- and resource-level agents facilitate the real-time collection and analysis of sensor data, even under changing environmental conditions and many concurrent science objectives.

This chapter presented the related work, challenges, and solution for the planning and scheduling needs of MACRO agents. Section IV.1 discussed the related work in planning and scheduling relevant to the MACRO mission level and the choice of a distributed planning and scheduling representation/mechanism for Mission agents. Section IV.2 discussed the related work in planning under uncertainty and incorporation of scheduling with planning relevant to the MACRO resource level. Section IV.3 identified the unresolved challenges in planning and scheduling for MACRO resource-level agents, and Section IV.4

presented the MACRO solution to these challenges in the form of the SA-POP service. Section IV.5 illustrated the use of SA-POP and the extension of MACRO resource-level planning with plan schemas for field agents in a representative sensor network. Finally, Section IV.6 presented the experimental results and verification of SA-POP planning performance.

CHAPTER V

SYSTEM INTEGRATION

Section I.1 introduced the coordination and control requirements for a global sensor web. MACRO provides the necessary capabilities to meet these requirements through its agent architecture, task allocation mechanism, and planning/scheduling mechanisms detailed in Chapters II, III, and IV, respectively. However, implementing even a prototype of a complete MACRO system based on these capabilities is a non-trivial task. Building a computational scheme that captures integration of the agents, autonomous planning/scheduling, and coordination mechanisms in MACRO presents additional research challenges. This chapter covers the major implementation and integration challenges in building a proof of concept (PoC) MACRO system.

Section V.1 discusses existing work related to the design of the MACRO PoC implementation and Section V.2 identifies the challenges not resolved by existing work. Section V.4 describes MACRO's translation of User tasks in SensorML to Mission agent TÆMS tasks and aggregation of Mission agent TÆMS task trees augmented with SensorML meta-data. Section V.3 details the object-oriented design of an extensible, prototype Mission agent designed to handle multiple coordination and execution activities at the MACRO mission level. Section V.5 covers coordination of planning and scheduling between the MACRO mission and resource level. To illustrate the capabilities of the integrated MACRO system, Section V.6 provides a case study of MACRO system operation for a set of simulated sensor networks, and Section V.7 presents the results of experiments on planning/scheduling coordination between the MACRO mission and resource levels. Finally, Section V.8 summarizes the MACRO implementation and integration work presented in this chapter.

V.1 Related Research

Integration of the MACRO PoC implementation leverages research in a variety of fields. Related work in multi-agent systems is presented in Section II.1. Section III.1 presents related work in task/resource allocation. Research related to the task-oriented planning/scheduling and plan/schedule coordination at the MACRO mission level is presented in Section IV.1. Further, Section IV.2 presents research related to decision-theoretic planning and integration of resource-constraint-propagation scheduling at the MACRO resource level. In this section, we present additional research related to the design of the MACRO Mission agent and the coordination of planning and scheduling between the MACRO mission and resource levels.

V.1.1 Agent Internal Architecture

Effective execution of the disparate activities (*e.g.*, inter-agent negotiation, planning, scheduling, and plan execution or delegation) performed by a sensor web agent, such as a MACRO Mission agent, requires an appropriate internal architecture to facilitate and control these activities. Autonomous agent architectures range from primarily *reactive* designs (*e.g.*, the Subsumption Architecture [15]) to highly *deliberative* designs (*e.g.*, architectures based on the popular Belief-Desire-Intention (BDI) theory [23, 93]). Further, a popular architectural approach combining reactive and deliberative elements is the *layered* architecture [120]. Layered architectures combine both a reactive layer for quick reaction to perceptions and one or more deliberative layers for more complex control.

Because MACRO is designed to include agents designed by multiple parties, complete consideration of the internal structure of sensor web agents is beyond its scope. However, implementation and integration of the MACRO PoC system requires the design of a prototype Mission agent. Since MACRO Mission agent reasoning operates at the mission level, including direction of subordinate resource-level agents and global coordination across sensor networks, its control needs fall primarily into the deliberative category. For

a deliberative agent, the Belief-Desire-Intention (BDI) framework [23, 93] provides a standard context for discussing many internal control and reasoning activities pertinent to the allocation, planning/scheduling, and inter-agent coordination in a sensor web MAS.

The BDI theory suggests three primary concepts central to an agent's reasoning and actions: its *beliefs* about the world, including beliefs about other agents; its *desires*, which are often loosely construed as inherent or persistent goals, not all of which may be applicable or achievable at a given time; and its *intentions*, which are commitments to achieving certain goals/desires through action. A BDI agent uses its beliefs and reasoning mechanism(s) to determine the relevance and feasibility of its desires in the current situation. This reasoning about beliefs and desires results in an updated set of intentions, for which it produces a plan to achieve. The correspondence between the prototype MACRO Mission agent and the BDI framework is discussed further in Section V.3.

Although the BDI theory provides a context for general discussion of some Mission agent activities, in order to perform effectively in a MACRO system implementation, a Mission agent must be able to coordinate a wider variety of reasoning and domain activities. Many recent advances in agent architectures go beyond traditional reactive, deliberative, and layered architectures to allow more complex design and interaction among different agent activities (*e.g.*, aspect-oriented software engineering for agents [47]). In particular, the category of *meta-control* [2] refers to the coordination of a variety of agent activities including both domain activities (*e.g.* action execution) and deliberative reasoning mechanisms.

Raja and Lesser suggest a broad division of agent activities into four categories [91]: (1) domain activities that are the actual actions executed by the agent; (2) planning/scheduling activities to determine what actions to take in the domain and when to execute them; (3) coordination activities, which includes all communication and negotiation with other agents; and (4) meta-level control activities, such as determining when and how to perform planning/scheduling and coordination. Meta-control is especially important for agents with

multiple deliberative reasoning mechanisms (*e.g.*, a planning mechanism and inter-agent negotiation mechanisms), such as the MACRO Mission agent.

Some have suggested meta-control should be performed by additional meta-agents incorporated into a MAS, providing guidance to ordinary agents for their individual reasoning activities and inter-agent coordination (*e.g.*, [17, 87]). Given the independent design and goals of sensor web agents, providing incentives to follow the guidance of meta-agents could present more problems than the meta-agents solve. The more common approach to meta-control is to incorporate it directly into the architecture of individual agents (*e.g.*, [3, 90, 92]).

The approach of integrating meta-control in the agent architecture is a good solution for sensor web agents who must perform planning/scheduling of announced and contracted tasks, as well as multiple forms of coordination with other agents. The limited meta-control of the prototype Mission agent is described in Section V.3. More advanced meta-control and meta-reasoning techniques are left for future work described in Section VI.2.

V.1.2 Planning and Scheduling Coordination

MACRO's approach to planning and scheduling builds upon and extends a significant body of related work. As discussed in Section IV.1, MACRO Mission agents employ task decomposition with criteria-directed scheduling [114, 115], which operates on a task tree based on the Task Analysis, Environment Modeling, and Simulation (TÆMS) [55] representation. Employing criteria-directed scheduling with the TÆMS task representation allows Mission agents to efficiently optimize schedule/plan generation for the criteria relevant to assigned subtasks. Moreover, Mission agents executing inter-dependent tasks coordinate their plans and schedules through the Generalized Partial Global Planning (GPGP) [69] coordination mechanisms. At the resource level, Exec agents employ

the Spreading Activation Partial Order Planner (SA-POP) [59] for decision-theoretic planning with constraint-propagation scheduling. Details of SA-POP design and operation are presented in Section IV.4, with related work covered in Section IV.2

In order to efficiently coordinate the two planning/scheduling mechanisms in MACRO, agents must communicate the most useful information at the appropriate abstraction level. To limit communication and computation overhead, this coordination should only occur when the plan/schedule information may be relevant to the receiving agent for use during planning and scheduling. The translation from resource-level plans to mission-level method parameters has some similarities to research that uses plan summary information to coordinate between agents employing hierarchical task network (HTN) planning (*e.g.*, [21, 22]). MACRO Mission and Exec agents, however, employ *different* representations for planning and scheduling, making translation and coordination more difficult. Moreover, the resource and scheduling constraints employed in MACRO require summary information beyond the pre-, in-, and post-conditions used in Clement's task summary approach [21].

V.2 Unresolved Challenges

To coordinate task achievement, MACRO Mission agents employ multiple protocols (*e.g.*, a broker-mediated contract net protocol and GPGP distributed planning protocols). Results of planning and scheduling can affect bidding on tasks in the contract net, and contracted tasks require further planning, scheduling, and GPGP coordination. Although design and implementation of advanced meta-control and meta-reasoning techniques is beyond the scope of this work, an appropriate Mission agent design should lend itself to future extension with such techniques. Therefore, a significant integration challenge in MACRO is defining a flexible, extensible Mission agent architecture that can effectively coordinate its allocation, planning/scheduling, plan/schedule coordination, and task execution activities. Section V.3 discusses how the MACRO prototype Mission agent architecture addresses this challenge.

At the mission level, MACRO combines broker-mediated allocation with GPGP coordination and criteria-directed scheduling to effectively achieve user tasks. Extending and integrating these techniques in a sensor web MAS poses additional challenges because the TÆMS representation and GPGP coordination were designed for systems built by a single group of designers sharing a conception of the entire system and its function. In that original context, task decomposition trees can be built with a top-down goal-oriented structure combined with bottom-up knowledge of the functionality available in the system. However, at the mission level in MACRO, Mission and User agents are implemented by the various groups they represent, such as organizations operating sensor networks or applications and researchers using sensor web data. Typically, the agent's domain knowledge, such as appropriate internal goals and TÆMS task trees, is specific to their individual purposes and capabilities, without full knowledge of goals and tasks throughout the system. The independence of task tree design in the sensor web presents a challenge in combining Mission agent tasks and determining potential decompositions of requested tasks that require resources of multiple Mission agents. MACRO's broker-based solution to this challenge is presented in Section V.4.

To achieve the high-level tasks allocated and coordinated at the mission level, individual Mission agents must communicate appropriate subtasks or subgoals to their resource-level agents for execution. A Mission agent uses a TÆMS task decomposition tree and criteria-directed scheduling to plan and schedule tasks for execution. However, at the resource level, Exec agents use SA-POP for detailed, first-principles planning and scheduling to achieve their assigned goal conditions in a dynamic environment. Efficiently employing both representations and forms of planning and scheduling requires careful design of the protocol and format for communication between Mission and Exec agents. For example, the quality and completion time of subtasks in TÆMS format, may include a probability distribution for a range of possibilities. However, the expected quality and completion time of subtasks could change significantly after resource-level planning and scheduling,

reducing the utility of, or invalidating, the Mission agent's original plan and schedule. Achieving efficient coordination and interoperability between the two different levels/forms of planning and scheduling is another design challenge in MACRO, which is addressed in Section V.5.

V.3 MACRO Mission Agent Architecture

A MACRO Mission agent must perform a variety of intertwined reasoning and coordination activities to effectively participate in the global sensor web. Section V.2 identified the challenge of integrating these activities in an efficient and effective manner. The prototype MACRO Mission agent follows an object-oriented design, in which objects encapsulate a particular set of related functionality and state information. The event-driven (*e.g.*, messages from other agents, including task announcements from Broker agents, GPGP coordination messages from other Mission agents, and environmental condition or plan/schedule updates from subordinate Exec agents) interaction of these objects results in the desired overall agent behavior. The major objects and their interactions in the Mission agent architecture are illustrated in Figure 21.

The Belief-Desire-Intention (BDI) framework, discussed in Section V.1.1, provides a context for discussing internal control and reasoning activities pertinent to the allocation, planning/scheduling, and inter-agent coordination in a sensor web MAS. By definition, intelligent agents reason about their beliefs (*e.g.*, their knowledge of environmental conditions and other agents plans) in order to make decisions. In MACRO Mission agents, the *Knowledge Base* object maintains the current set of beliefs (*e.g.*, conditions in the sensor network environment and scheduled completion times for related tasks assigned to other Mission agents). Reasoning in MACRO Mission agents, includes the task decomposition and scheduling performed by the *Planning and Scheduling* object, as well as the meta-level control integrating the agent's activities in the *Meta-Control* object. For MACRO Mission

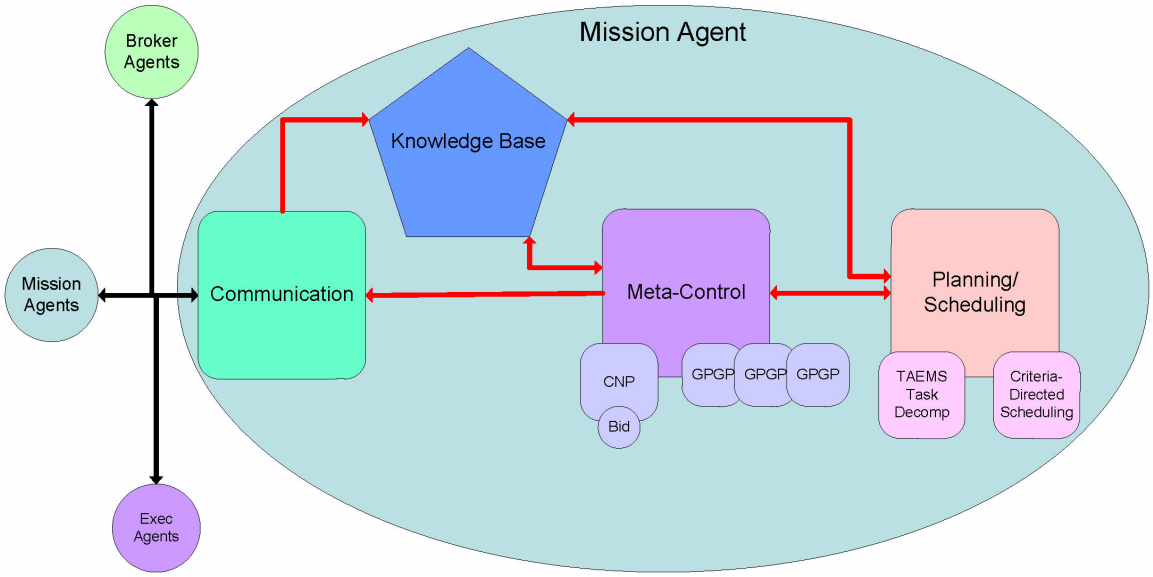

Figure 21: MACRO Mission agent architecture

agents, desires are maintained in the Meta-Control object in the form of internal, sensor-network goals as well as announced tasks. Intentions include the achievement of both contracted tasks/subtasks and any internally-generated tasks (as a result of sensor-network-specific goals).

Meta-Control. The central component of a Mission agent is the meta-control object. In general, this object controls the allocation, planning/scheduling, plan/schedule coordination, and domain activities of the Mission agent. Advanced meta-control strategies are beyond the scope of this work, but the centrality of the Meta-Control object with its interface to all other objects allows for future extension with such strategies.

Meta-control in the prototype MACRO Mission agent, performs three major sets of activities: (1) task allocation is handled by generating bids for announced tasks and choosing among mutually-exclusive tasks based on their broker-assigned values; (2) task execution is performed by informing Exec agents of relevant goals (based on the methods in its plan for achievement of an allocated task); and (3) coordination with other Mission agents is defined by the five GPGP coordination mechanisms described in Section IV.1.3 for allocated subtasks of the same overall task.

Knowledge Base. The Mission agent knowledge base provides the central repository for information about the state of its sensor network environment, communications with other agents, and its allocated tasks. In general, other agent objects request notification when a particular pattern of knowledge is present in the Knowledge Base. In the prototype Mission agent this is limited to specifying particular message types and environmental conditions. For example, the meta-control object is informed of any of the situations (identified by communications from other Mission agents) relevant to GPGP coordination protocols. Similarly, the planning/scheduling object is informed whenever a resource-level agent indicates changes in environmental conditions relevant to existing plans.

Planning and Scheduling. The Mission agent's planning/scheduling object performs the actual decomposition and scheduling of TÆMS tasks. The mission-level approach to planning and scheduling is discussed in Section IV.1.3. With direction from the meta-control object, scheduled plans are generated for announced tasks in order to produce appropriate bids. Further, the meta-control object employs the planning/scheduling object to generate and update plans for allocated tasks.

Communication. The Mission agent communication object handles formatting, addressing, and interpretation of agent messages. For example, the meta-control object uses the communication object to format and address a message to the appropriate Exec agent for execution of a method in an allocated task. The communication object maintains the header information (based on the FIPA Agent Communication Language [44]) for all ongoing agent conversations in order to properly address or interpret agent messages. For example, an incoming message from an Exec agent with current conditions in the sensor network environment is interpreted by the communication object, and the parsed data is provided to the Knowledge Base.

V.4 Mission-level Goal/Task Representation and Interoperability

MACRO employs the Open Geospatial Consortium (OGC) SensorML standard [9] to represent sensor characteristics and data processing activities in a standardized, interoperable format. SensorML represents sensors and data analysis as processes with standardized descriptors applicable to sensor webs. Therefore, User agents can succinctly and effectively express their desired tasks based on this format. However, MACRO Mission agents utilize the TÆMS format for specifying tasks/subtasks in sensor networks, allowing criteria-directed scheduling and GPGP distributed planning coordination among Mission agents. Section V.2 identified the challenge of providing interoperability among Mission agent task trees and between User agent SensorML task requests and Mission agent TÆMS tasks/subtasks.

The MACRO solution approach to this challenge is to define Mission agent capabilities and task meta-data with the relevant SensorML attributes. In general, MACRO incorporates SensorML attributes that are identified using the relevant XML tags and their parameters. SensorML tag and parameter values (*i.e.*, numbers or standard OGC string values) are specified by a range or list in MACRO task meta-data and agent communications.

Listing V.1 provides a SensorML excerpt partially specifying a temperature sensor. In order to specify the use of a particular sensor or sensor type, a MACRO User agent provides a constraint based on the relevant SensorML tags. Specifically, the hierarchy of header tags, the constraining tag(s) and tag parameter(s), and the relevant values or value range for each tag or parameter. To indicate the specific temperature sensor from the example, MACRO specifies a constraint as:

```
HeaderTags=[]
Tag=<Detector>
Parameter="id"
ParameterValues=["EXAMPLE_THERMOMETER"]
Values=[]
```

To indicate any temperature sensor providing measurements in either Celsius or Farenheit, MACRO specifies a constraint on the output type of the detector as the OGC Uniform

```
<Detector id="EXAMPLE_THERMOMETER">
  <identification>
    <IdentifierList>
      <identifier name="longName">
        <Term qualifier="urn:ogc:def:identifier:longName">
          Example Thermometer</Term>
      </identifier>
    </IdentifierList>
  </identification>
  <!-- INPUT DEFINITION -->
  <inputs>
    <InputList>
      <input name="temperature">
        <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"
          uom="urn:ogc:def:unit:celsius"/>
      </input>
    </InputList>
  </inputs>
  <!-- OUTPUT DEFINITION -->
  <outputs>
    <OutputList>
      <output name="urn:macro:def:dataType:temperature">
        <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"
          uom="urn:ogc:def:unit:celsius"/>
      </output>
    </OutputList>
  </outputs>
</Detector>
```

Listing V.1: SensorML sensor example (thermometer)

Resource Name (URN) for temperature and a constraint on the unit-of-measure (UOM) as a list of the URNs for Celsius and Farenheit:

```
HeaderTags=[<Detector>, <outputs>, <OutputList>, <output>]
Tag=<swe:Quantity>
Parameter="definition"
ParameterValues=["urn:ogc:def:phenomenon:temperature"]
Parameter="uom"
ParameterValues=["urn:ogc:def:unit:celsius", "urn:ogc:def:unit:farenheit"]
Values=[]
```

Although the MACRO PoC implementation only recognizes a small subset of SensorML, the generality of the MACRO approach in using constraints on SensorML tags and parameters allows it to be directly extended to the full SensorML language, as it evolves.

V.4.1 Aggregation of TÆMS Task Trees

One responsibility of MACRO Broker agents is to aggregate domain knowledge across Mission agents. Tier 2 Brokers collect task/subtask information from their assigned Mission agents to allow a complex user task spanning multiple sensor networks to be translated into subtasks, each of which can be performed by a single Mission agent. Since Mission agent task trees may be designed independently, they are enhanced with SensorML [9] meta-data to provide standard descriptions of task/subtask input, output, and type classifiers. This allows MACRO Brokers to determine appropriate join points and overlap between independent task trees. Tier 1 Broker agents employ the aggregated information to translate User agent task requests into one-level task decompositions, as described in Section V.4.2.

In SensorML, data acquisition and processing is defined in terms of processes. In fact, SensorML even defines sensors as a special type of process. A SensorML process (or *process chain*) is composed of *process models*, defining atomic processes, with associated *process methods*, defining the behavior and interface of process models. For example, a

process model (and associated process method) for calculation of wind chill is partially specified in Listing V.2.

```
<ProcessModel id="WINDCHILL_PROCESS">
  <description>
    <swe:Discussion>Wind chill computation</swe:Discussion>
  </description>
  <!-- INPUTS DEFINITION -->
  <inputs>
    <InputList>
      <input name="urn:macro:def:dataStreamType:temperature">
        <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"
          uom="urn:ogc:def:unit:celsius"/>
      </input>
      <input name="urn:macro:def:dataStreamType:windSpeed">
        <swe:Quantity definition="urn:ogc:def:phenomenon:windSpeed"
          uom="urn:ogc:def:unit:milesperhour"/>
      </input>
    </InputList>
  </inputs>
  <!-- OUTPUTS DEFINITION -->
  <outputs>
    <OutputList>
      <output name="urn:macro:def:dataStreamType:temperature:windChill">
        <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"
          uom="urn:ogc:def:unit:celsius"/>
      </output>
    </OutputList>
  </outputs>
  <ProcessMethod>
    <classification>
      <ClassifierList>
        <classifier name="intendedApplication">
          <Term qualifier="urn:ogc:def:property:intendedApplication">
            weather</Term>
        </classifier>
        <classifier name="processType">
          <Term qualifier="urn:macro:def:processType">windChill</Term>
        </classifier>
      </ClassifierList>
    </classification>
  </ProcessMethod>
</ProcessModel>
```

Listing V.2: SensorML process example (wind chill)

The MACRO PoC implementation allows TÆMS tasks/subtasks to specify input and output meta-data (based on a SensorML process model), as well as classifier meta-data

(based on a SensorML process method). For example, a task for calculating wind chill values would include the meta-data:

```
Input=("urn:ogc:def:phenomenon:temperature", "urn:ogc:def:unit:celsius")
Input=("urn:ogc:def:phenomenon:windSpeed", "urn:ogc:def:unit:milesperhour")
Output=("urn:ogc:def:phenomenon:temperature", "urn:ogc:def:unit:celsius")
ProcessType="windChill"
```

By comparing the SensorML descriptions of task input and output data types, MACRO Broker agents can identify possible join points between tasks and subtasks from different Mission agents to combine and extend their task decomposition trees. Further, MACRO adds a Uniform Resource Name (URN) for process type to allow classification of data processing tasks. This is a special classifier whose applicable values are defined by MACRO because SensorML does not currently include any standard way to identify equivalent types of data processing/analysis. By comparing *processType* classifiers, MACRO Broker agents can identify substitutable tasks/subtasks across Mission agents.

V.4.2 Translation of SensorML Requests to TÆMS Tasks

In addition to aggregation of domain information across Mission agents, MACRO Broker agents also provide a matchmaking/locater service (*i.e.*, identifying agents capable of performing all or part of an User requested task and forwarding appropriate messages). Tier 2 Broker agents cluster Mission agents by geographic region and maintain a directory of sensor and computational capabilities based on SensorML meta-data from the Mission agents in their region.

In SensorML, a *sensor* is defined as a *measurement system* made up of one or more *detectors*. Each detector, defines the input (physical phenomena) and output (measurement of the phenomena) types, as well as its sampling and response characteristics. SensorML measurement systems include information on their physical *position*. A SensorML position can include a location (as latitude, longitude, and altitude) and an orientation. For example, a measurement system position for the thermometer example is specified in Listing V.3.

```
<System id="THERMOMETER_SYSTEM">
  <!-- POSITION in EPSG4329 -->
  <positions>
    <PositionList>
      <position name="thermometerPosition">
        <swe:Position localFrame="#THERMOMETER_FRAME"
          referenceFrame="urn:ogc:def:crs:EPSG:4329">
          <swe:location>
            <swe:Location definition="urn:ogc:def:phenomenon:location">
              <swe:coordinate name="latitude">
                <swe:Quantity definition="
                    urn:ogc:def:phenomenon:latitude"
                  uom="urn:ogc:def:unit:degree">34.72450</swe:Quantity>
              </swe:coordinate>
              <swe:coordinate name="longitude">
                <swe:Quantity definition="
                    urn:ogc:def:phenomenon:longitude"
                  uom="urn:ogc:def:unit:degree">86.94533</swe:Quantity>
              </swe:coordinate>
              <swe:coordinate name="altitude">
                <swe:Quantity definition="
                    urn:ogc:def:phenomenon:altitude"
                  uom="urn:ogc:def:unit:meter">5.1169</swe:Quantity>
              </swe:coordinate>
            </swe:Location>
          </swe:location>
          <swe:orientation>
            <swe:Orientation definition="
                urn:ogc:def:phenomenon:orientation">
              <swe:coordinate name="trueHeading">
                <swe:Quantity definition="
                    urn:ogc:def:phenomenon:angleToNorth"
                  axisCode="Z" uom="urn:ogc:def:unit:degree">87.0</
                    swe:Quantity>
              </swe:coordinate>
            </swe:Orientation>
          </swe:orientation>
        </swe:Position>
      </position>
    </PositionList>
  </positions>
</System>
```

Listing V.3: Measurement system location example (thermometer)

In MACRO, sensor capabilities are represented by their output type and position. Sensor output type is defined in the detector output portion of a SensorML system format (*e.g.*, the thermometer output defined by the $< output >$ tag in Listing V.1). Sensor position is defined by the position portion of a SensorML system (*e.g.*, the thermometer location and orientation defined by the $< swe : location >$ and $< swe : orientation >$ tags in Listing V.3).

Similarly, computational capabilities are represented by the SensorML classifiers. Specifically, computational capabilities are defined by MACRO process type classifiers ($urn : macro : def : processType$), discussed in Section V.4.1. For example, Listing V.2 classifies the wind chill calculation process as a *windChill* process type. Similarly, a JPEG compression algorithm for images could be classified by multiple levels of classifier, such as *imageCompression* and $imageCompression : JPEG$, in MACRO.

A MACRO User agent announces a task using the relevant sensor types and locations, along with the relevant process chain information (as identified by MACRO defined process types and input/output data types, if any). For example, a task could specify use of a temperature sensor in a particular geographic region as:

```
  HeaderTags=[<Detector>, ... <output>]
  Tag=<swe:Quantity>
  Parameter="definition"
  ParameterValues=["urn:ogc:def:phenomenon:temperature"]
  Values=[]
AND
  HeaderTags=[<System>, ... <swe:Location>, <swe:coordinate>]
  Tag=<swe:Quantity>
  Parameter="definition"
  ParameterValues=["urn:ogc:def:phenomenon:latitude"]
  Parameter="uom"
  ParameterValues=["urn:ogc:def:unit:degree"]
  Values=[34.6 - 35.8]
AND
  HeaderTags=[<System>, ... <swe:Location>, <swe:coordinate>]
```

```
Tag=<swe:Quantity>
Parameter="definition"
ParameterValues=["urn:ogc:def:phenomenon:longitude"]
Parameter="uom"
ParameterValues=["urn:ogc:def:unit:degree"]
Values=[86.8 - 87.0]
```

The sensor location requirements of an announced task are used by Tier 1 Broker agents to forward task announcements to appropriate Tier 2 Broker agents, who then relay the announcement to applicable Mission agents. Tier 1 Broker agents are also responsible for defining the possible decompositions of the user task into TÆMS subtasks, each of which can be achieved by a single Mission agent. This is achieved by matching process type classifications in the task request to the process type meta-data of TÆMS tasks/subtasks collected by Tier 2 Broker agents. For example, a simple user task could request an image in a compressed format by specifying a process type of *imageCompression* : *JPEG* and appropriate parameters for a JPEG image compression process. However, if the User did not care which type of image compression was used, it could simply specify a process type of *imageCompression*.

With such general specification of process types, the number of possible task decompositions could be very large in MACRO. However, the MACRO PoC implementation uses only the first level of decomposition in Mission agent TÆMS task trees for aggregation and user task decomposition by MACRO Brokers. This limits the possible decompositions of a user task request to a manageable number, while ensuring that each subtask in any decomposition can be performed by at least one Mission agent. Further, whenever possible, MACRO Brokers limit the data processing portions of a task to Mission agents in the same region(s) (defined by their Tier 2 Broker agent) as the sensor (data acquisition) requirements of the task.

The use of the SensorML standard in MACRO agent communications and task meta-data allows for interoperability among independently-designed Mission and User agents.

Broker translation of User agent task requests and identification of relevant Mission agents, allows MACRO User and Mission agents to use different task representations, appropriate to their different roles in the sensor web. Further, use of SensorML meta-data in TÆMS task trees and Broker aggregation of Mission agent tasks, allows MACRO to support Mission agents independently designed by a variety of sensor network domain experts.

V.5 Context-sensitive Coordination of Planning and Scheduling

In large-scale, distributed, multi-agent systems (MAS) that span multiple domains of agent operation, choosing a single planning and scheduling mechanism for all agents may be inefficient and impractical. For example, a global sensor web must select and coordinate an appropriate subset of many heterogeneous, distributed sensors and computational resources for user tasks. Further, sensor web tasks often operate under significant resource constraints and require collaboration among multiple constituent sensor networks. In MACRO, accomplishing such complex tasks requires that planning and scheduling be performed at multiple levels of the sensor web.

As described in Section II.3, MACRO is structured as a two-level agent hierarchy: (1) *mission level* and (2) *resource level*. Agents have different responsibilities related to task achievement at each level:

- At the mission level, Mission agents representing individual sensor networks must coordinate to cooperatively achieve complex tasks spanning the resources and capabilities of multiple agents.

- At the resource level, Exec agents and other domain-specific agents adapt local operations within a sensor network to efficiently achieve goals given current conditions in dynamic, uncertain and resource-constrained environments.

Therefore, agents at these different levels of the system operate in different contexts that imply different planning and scheduling requirements.

At the mission level of a sensor web MAS, user tasks and scheduled plans may span multiple sensor networks and have a high degree of complexity. MACRO, therefore, employs a modified implementation of the TÆMS language [55], which provides a hierarchical task network representation for multi-agent planning and scheduling. The TÆMS language represents tasks as a hierarchical tree structure, decomposing tasks into applicable sets of subtasks, which can be further decomposed into their subtasks, and so on. The TÆMS representation also allows the specification of discrete probability distributions for task/subtask characteristics including potential outcome quality and duration [69]. As illustrated in Section IV.1, MACRO Mission agents employ hierarchical task network decomposition and criteria-directed scheduling [115] to generate an appropriate task decomposition and schedule from a TÆMS task tree.

At the resource level, Exec agents use the *Spreading Activation Partial Order Planner* (SA-POP), which generates high-utility, scheduled, partial-order plans that respect local resource constraints. Section IV.4 describes the SA-POP service, which allows the Exec agents to use their limited computational resources to maximize expected utility for achieving local goals in dynamic, uncertain environments. First-principles planning [6] and scheduling with SA-POP requires a set of goal conditions that correspond to the desired outcome. These goal conditions are specified as desired environmental and system conditions with associated utility values and time deadlines. Given these goal conditions, SA-POP uses current/expected conditions to generate a scheduled plan of high expected utility.

MACRO achieves efficient and effective autonomous planning/scheduling by combining hierarchical task network planning with criteria-directed scheduling at the mission level and decision-theoretic planning with resource-constraint propagation at the resource level. Mission agents efficiently generate and coordinate plans and schedules based on the TÆMS task decomposition trees As shown in Figure 22, the leaves of a TÆMS task tree are *methods*, which in standard TÆMS usage can be directly executed by the agent. In

MACRO, however, Mission agents must communicate these methods to their Exec agents for resource-level planning/scheduling and ultimate execution. As illustrated in Figure 22,

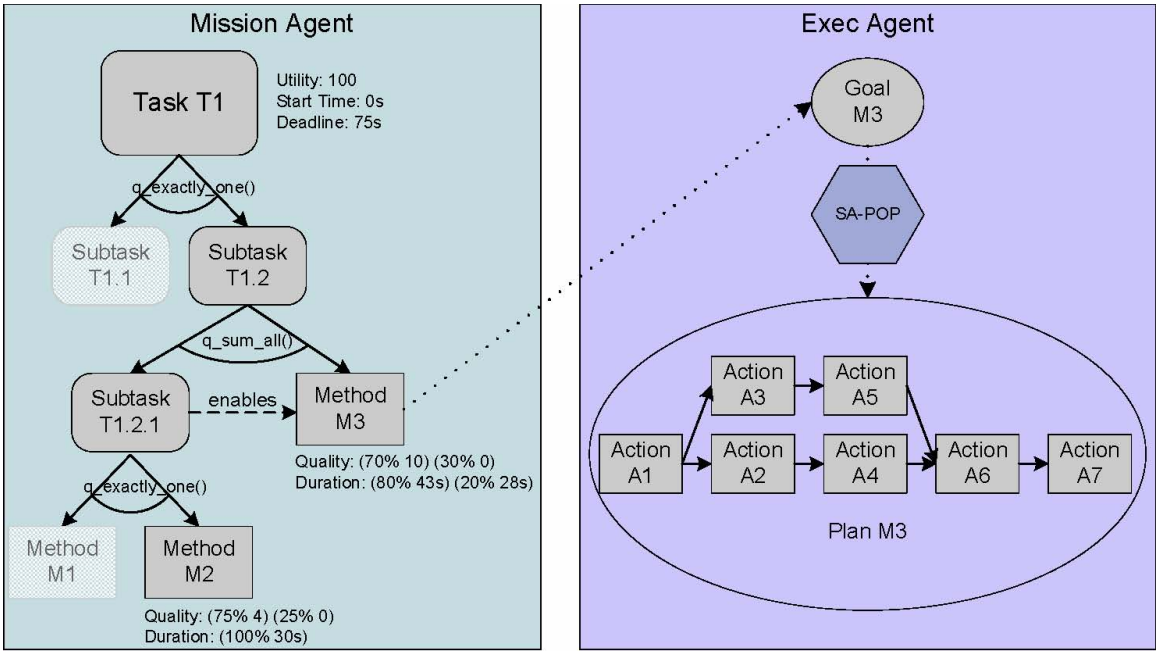

Figure 22: Planning/scheduling representations in MACRO

Exec agents employ the planning and scheduling service, SA-POP, to achieve *goals* in the dynamic sensor network environment. Effectively employing both representations and forms of planning and scheduling requires an appropriate translation of the task, plan, and schedule representations between levels in MACRO. Further, MACRO agents require a co-ordination mechanism for deciding when to exchange information between levels during plan execution.

V.5.1 Translation: Top-Down

Problem. For an Exec agent to implement a TÆMS method, the Mission agent must translate it into the goal format used by SA-POP. SA-POP goals include one or more goal conditions with associated utility values and time deadlines. To generate an effective plan

for a goal, SA-POP requires knowledge of expected system and environmental conditions at the time the plan will be executed. Although current conditions and other Exec agent plans provide most of this information, other expected conditions may be the result of methods assigned to other Exec agents in the Mission agent's current plan (*i.e.*, other methods that *enable* the method in question by satisfying some of its preconditions).

Solution → Cross-references in task/goal modeling. In MACRO, TÆMS methods are associated with necessary resource-level preconditions and goal conditions, which in turn are represented in the action network model employed by the Exec agent and SA-POP. Moreover, a domain expert can derive method distributions for duration and outcome by employing SA-POP with its domain-specific modeling language, discussed in Section IV.4. Provided with initial condition settings (including their associated probabilities) SA-POP can produce scheduled plans. Aggregating the expected utilities, durations, and resource usages for these plans provides the information to specify TÆMS method characteristics (*e.g.*, duration and outcome quality).

Instead of directly executing a method, the Mission agent uses the encoded translation information from the model to provide a goal to the Exec agent. This top-down translation is shown by the Mission agent to Exec agent information transfer in Figure 23. The Mission agent awards overall task utility to methods based on the quality aggregation functions (QAFs) and expected quality in the TÆMS task tree.

In the chosen decomposition of the TÆMS task tree, parents with a QAF that requires execution of all child subtasks/methods pass the full parent utility to each child, while QAFs that allow any subset of children pass a percentage of parent utility based on the child's percentage of total expected quality for the parent. For example, a task with an overall utility of 100 that is decomposed into two subtasks of expected quality 3 and 7 with a sum QAF would assign utility of 30 and 70, respectively, to its subtasks. Future work will investigate more advanced methods of reward assignment in the decomposition of TÆMS task trees.

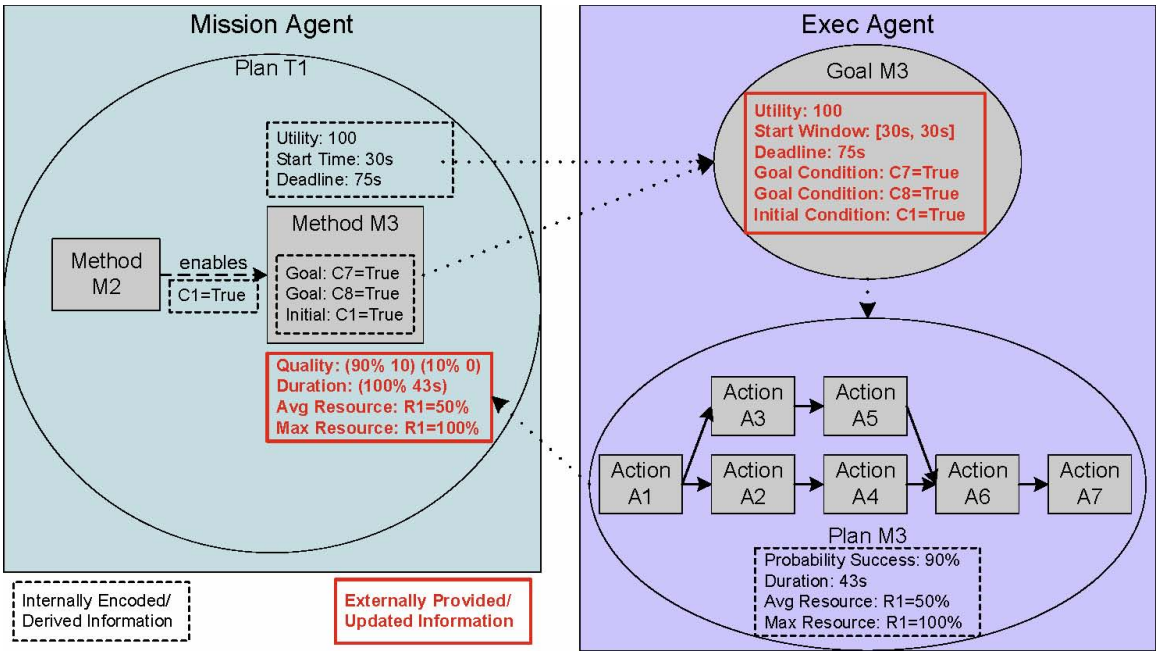

Figure 23: Planning/scheduling translation in MACRO

V.5.2 Translation: Bottom-Up

Problem. Another important challenge is codifying the bottom-up translation between SA-POP plans and TÆMS method parameters. Standard TÆMS methods include *a priori* probability distributions for duration and outcome quality, which are used during initial criteria-directed scheduling by the Mission agent. After an Exec agent plans to achieve a goal, the resultant scheduled plan may imply significantly different probability distributions for the corresponding method. Similarly, as a plan is being executed by the Exec agent, there may be further changes to the expected duration or probability of outcomes for the plan and its corresponding method. To improve the efficiency of future criteria-directed scheduling and to trigger appropriate mission-level re-scheduling, information about the Exec agent's plan must be communicated to the Mission agent.

Solution → Summarize resource-level plans. Instead of providing the complete resource-level plan to the Mission agent (whose format is ill-suited to its planning and scheduling capabilities), a MACRO Exec agent summarizes its plan by providing relevant

information only, including (1) expected duration, (2) probability of achieving the goal, and (3) average and maximum resource usage over expected execution. The Mission agent uses these values to update method parameters with more accurate information, based on the resource-level planning and scheduling for the current and expected environmental/system conditions. For example, Figure 23 illustrates updating a TÆMS method with a new outcome distribution (*i.e.*, 90% probability of achieving the method's maximum utility based on the 90% probability of success for the resource-level goal condition), duration (*i.e.*, 43 seconds from the expected completion time of the resource-level plan), and resource usage (*i.e.*, as both a maximum and average resource usage over the scheduled execution of the resource-levle plan). The updated method parameters allow the Mission agent to more effectively perform any further planning and scheduling for its task(s).

V.5.3 Context-Sensitive Updates

Problem. In addition to translating between the Mission and Exec agent planning/-scheduling representations, MACRO agents must also decide *when* to update and communicate the translated information. In particular, during execution of Exec agent plans, deviations may occur (*e.g.*, differences between actual and expected duration of actions). Only some variations, however, will impact the rest of the mission-level plan—or other plans—in a manner that would be of interest to the Mission agent.

Solution → Leverage mission-level task context. Given the hierarchical relationship between Mission and Exec agents, the top-down decision to communicate (*i.e.*, when the Mission agent should communicate information to an Exec agent) is relatively straightforward. Specifically, whenever a new task is decomposed/scheduled or method parameters in the plan are changed by re-planning/re-scheduling, the Mission agent communicates the new or revised goals (translated from the methods) to the assigned Exec agents.

For bottom-up updates, however, an Exec agent can use its knowledge of a Mission

agent's overall goals/interests to guide its decision of when to communicate. Without Mission agent guidance, an Exec agent would be forced to communicate on a periodic basis or whenever the execution deviates from the scheduled plan, which may happen frequently in a dynamic sensor network environment. When tasking an Exec agent with a goal, therefore, the MACRO Mission agents also provide guidance and contextual information, such as the optimization criteria for the related task. Knowledge of the optimization criteria allows the Exec agent to configure SA-POP's planning and scheduling to prefer plans based on that criteria.

In addition to optimization criteria, the Mission agent can specify acceptable deviations (in either direction), success probability, expected utility, duration, and resource usage of an executing plan. This information provides the Exec agent with guidance on the *context* for the corresponding method in the Mission agent's plan, which allows the agent to more intelligently determine when to update its scheduled plan and provide the revised summary to the Mission agent. Specifically, during execution of a plan, the Exec agent will only replan and re-schedule if the expected utility falls below, or if the duration surpasses, specified thresholds. When other thresholds are exceeded, the Exec agent simply communicates updated summary information to the Mission agent.

Figure 24 shows the execution of the resource-level plan from Section V.5.2. To demonstrate the benefit of the guidance/context provided by the Mission agent, we focus on deviations of action duration from expected duration in the critical path (*i.e.*, the linked sequence of actions that requires the longest time to complete). Although the planning and scheduling in MACRO does not rely on identification of the critical path, such a path(s) always exist, and it constrains the expected completion time of the plan.

Without the context provided by duration thresholds, the Exec agent would have no knowledge of what deviations were important to the Mission agent and would have to communicate updates based on each deviation. The Exec agent would recalculate its schedule every time an action did not complete with exactly the expected duration. It would also

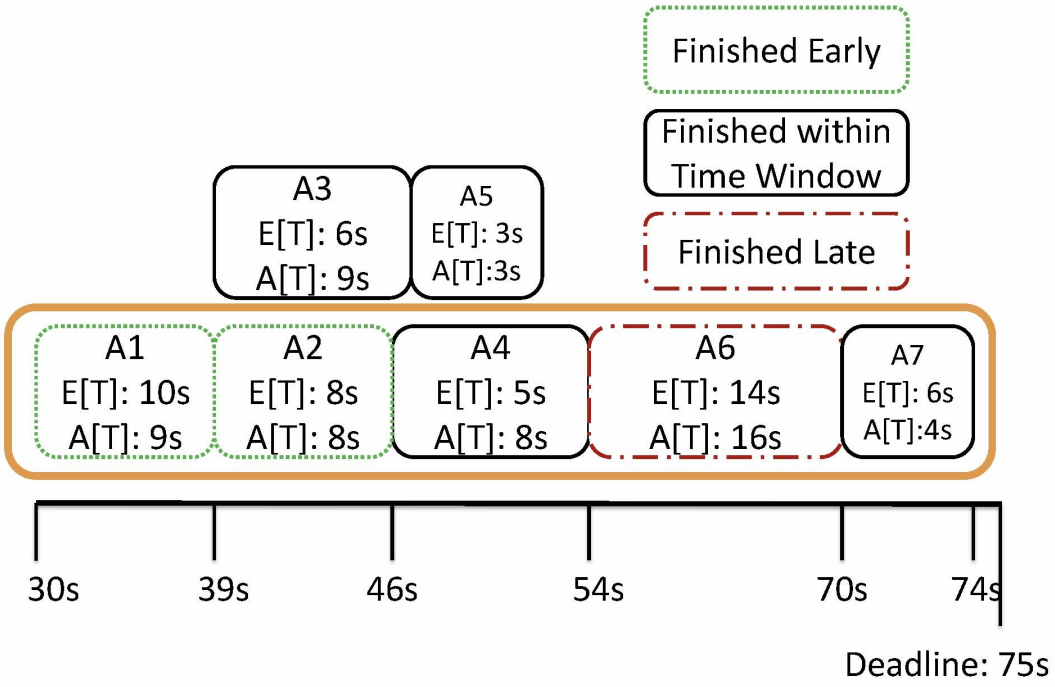

Figure 24: Example resource-level plan with critical path highlighted

transmit the new expected duration of the plan either with every recalculation or at least every time an action finished outside of its scheduled end window (either before or after that window).

The example execution in Figure 24 shows a typical case in which the Mission agent provides an over-threshold on duration equal to the difference between the expected end-time of the plan and the original deadline. In other words, the Mission agent is only interested in changes to the resource-level schedule that would result in its finishing later than the deadline. In this example, the Exec agent would have to re-plan/re-schedule only when execution of action A6 goes beyond its scheduled end window. Without the appropriate context (*i.e.*, the duration threshold), the Exec agent would have also had to unnecessarily recalculate or re-plan/re-schedule three times (after completion of A1, A4, and A3) and communicate unnecessary updates twice (after A1 and A4).

V.6 Case Study: Southeast Alaska

Chapters II, III, and IV described the MACRO agents/architecture, task allocation mechanism, and planning/scheduling mechanisms, respectively. Further, this chapter has detailed the major challenges in integrating the adaptation/coordination mechanisms and protocols of the MACRO system in Sections V.4, V.3, and V.5. In order to illustrate the full complement of MACRO's capabilities and their integration, this section describes an end-to-end case study scenario. The SouthEast Alaska (SEA) case study includes three simulated sensor networks, described in Section V.6.1, and two simulated user applications, described in Section V.6.2. The MACRO mission level for this scenario includes two User agents, a Tier 1 Broker agent, a Tier 2 Broker agent, and three Mission agents, illustrated in Figure 25. The MACRO resource level in this scenario includes four Exec agents with no secondary resource-level agents to minimize extraneous detail.

V.6.1 MACRO for Southeast Alaska Sensor Networks

Th Southeast Alaska case study includes three simulated sensor networks in southeast Alaska: (1) the SEAMONSTER glacial watershed research sensor network, which is based on a real sensor network; (2) the hypothetical MONSEA marine sensor network; and (3) the hypothetical WReNSEA weather monitoring sensor network. The MACRO agent organization for these sensor networks is illustrated in Figure 25. For comprehensibility of the overall scenario, each simulated network includes only four field nodes with three to four sensors each. All simulated sensors have three data rates, defined as *low*, *medium*, and *high*.

SEAMONSTER is the simulation of the *South East Alaska MOnitoring Network for Science, Telecommunications, Education, and Research*. The SEAMONSTER network in this scenario is based on the actual SEAMONSTER network [42], described in Section IV.5 and illustrated in Figure 18. The SEAMONSTER sensor network includes sensors attached to weatherized computer platforms in the vicinity of Juneau, AK. SEAMONSTER field

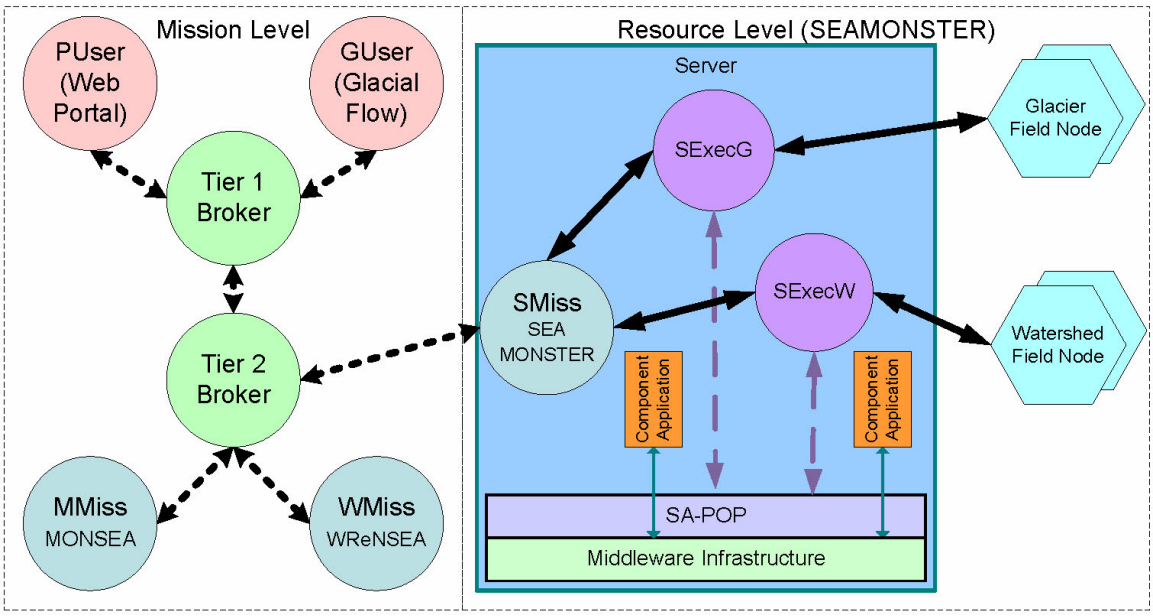

Figure 25: MACRO configuration for southeast Alaska sensor networks

computer nodes are deployed on Lemon Glacier and throughout the Lemon Creek watershed to collect data of scientific interest. The data collected by the sensors attached to the weatherized field nodes is communicated via wireless networks through a field relay node to a server cluster for processing, correlation, and analysis. In this scenario, we identify the MACRO Mission agent representing the SEAMONSTER network as *SMiss*. SMiss has two Exec agents, resident on SEAMONSTER servers, for resource-level planning and execution: (1) *SExecW* is responsible for the field nodes along the watershed and (2) *SExecG* is responsible for the field nodes on the glacier. For this scenario, we include two glacier field nodes with sensors for air temperature, humidity, and wind speed. One of the glacier field nodes is located adjacent to a glacial lake, which occasionally drains into the watershed river, and has an additional sensor for monitoring the lake level. The simulated SEAMONSTER system also includes two watershed field nodes with sensors for air temperature, water temperature, and water turbidity. The relay node in SEAMONSTER has a limited bandwidth that can not accommodate high data rates from more than two field nodes at a time.

MONSEA is the simulation of the hypothetical *Marine Observation Network in South East Alaska*. The MONSEA sensor network includes small computational nodes with sensors in the coastal waters around Juneau, AK. In this scenario, we identify the MACRO Mission agent representing the MONSEA network as *MMiss*. MMiss has a single Exec agent, *MExec*, which is responsible for resource-level planning and execution for all MONSEA field nodes. The simulated MONSEA system includes four field nodes with water temperature, air temperature, and wind speed sensors.

WReNSEA is the simulation of the hypothetical *Weather Research Network of South East Alaska*. The WReNSEA sensor network includes land-based field nodes/sensors in and around Juneau, AK. In this scenario, *WMiss* is the MACRO Mission agent representing the WreNSEA network. WMiss has a single Exec agent, *WExec*, which is responsible for resource-level planning and execution for all WReNSEA field nodes. The simulated WReNSEA system includes four field nodes with sensors to monitor air temperature, humidity, and wind speed.

V.6.2 Southeast Alaska Scenario

Figure 26, illustrates the three major phases of the Southeast Alaska scenario: (1) the *task coordination* phase begins with the announcement of three tasks by the User agents, (2) the *task execution* phase begins at the specified start time for the two allocated tasks, and (3) the *dynamic reaction* phase begins when a glacial lake drainage event is detected in the SEAMONSTER network. Task allocation and planning/scheduling occur during the task coordination phase. During the task execution phase, the subtasks (of the allocated user tasks) are executed by the Exec agents of the sensor networks to which they were allocated. Finally, during the dynamic reaction phase, SEAMONSTER activates an internal task due to the lake drainage event, re-plans for the new task, and begins executing it.

The two User agents in the SEA scenario are *PUser*, which is a web portal providing sensor web access to registered users, and *GUser*, which is an application for glacial flow

modeling. PUser has two tasks: *PTask1* is visualization of local meteorological conditions and *PTask2* is visualization of local marine currents. GUser's task (*GTask1*) is extraction of relevant features of glacial weather conditions for use in the glacial flow modeling application.

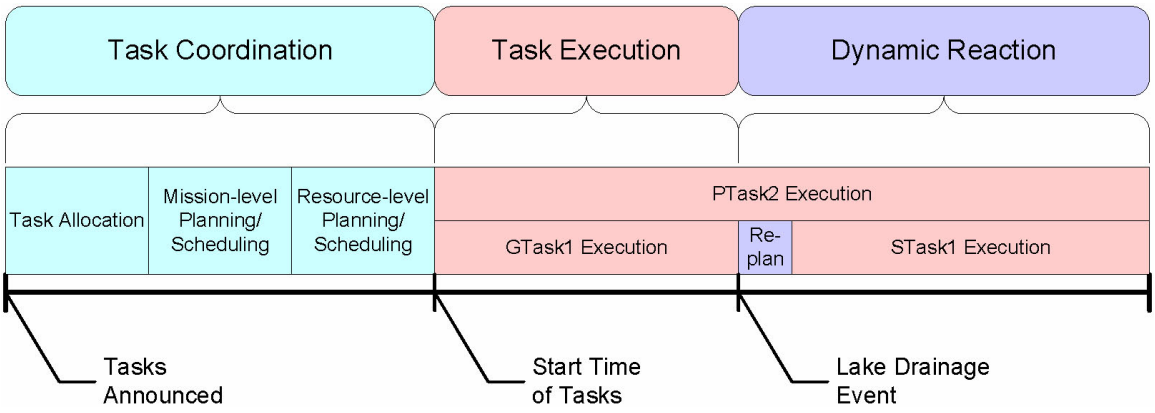

Figure 26: Timeline and phases of operation in SEA scenario

The three sensor network Mission agents capture domain information in TÆMS task trees, as discussed in Section IV.1.3. The relevant senors and subtasks are illustrated in Figure 27. SMiss's subtasks include: (S1) extracting glacial flow features and (S2) generating river temperature/turbidity map. MMiss's subtasks include: (M1) extracting marine current features, (M2) generating marine current map, and (M3) JPEG image compression. WMiss's subtasks include: (W1) generating meteorological map and (W2) JPEG image compression.

Before the end-to-end scenario in Section V.6.3, the MACRO Broker agents must gather appropriate domain information from the Mission agents. As detailed in Section V.4, MACRO Broker agents aggregate Mission agent task and sensor capability info, in order to provide matchmaker and translation services for User agent task announcements. During system initialization, the MACRO Tier 2 Broker requests the information on the top-level decompositions of TÆMS tasks from its Mission agents, including the subtasks illustrated

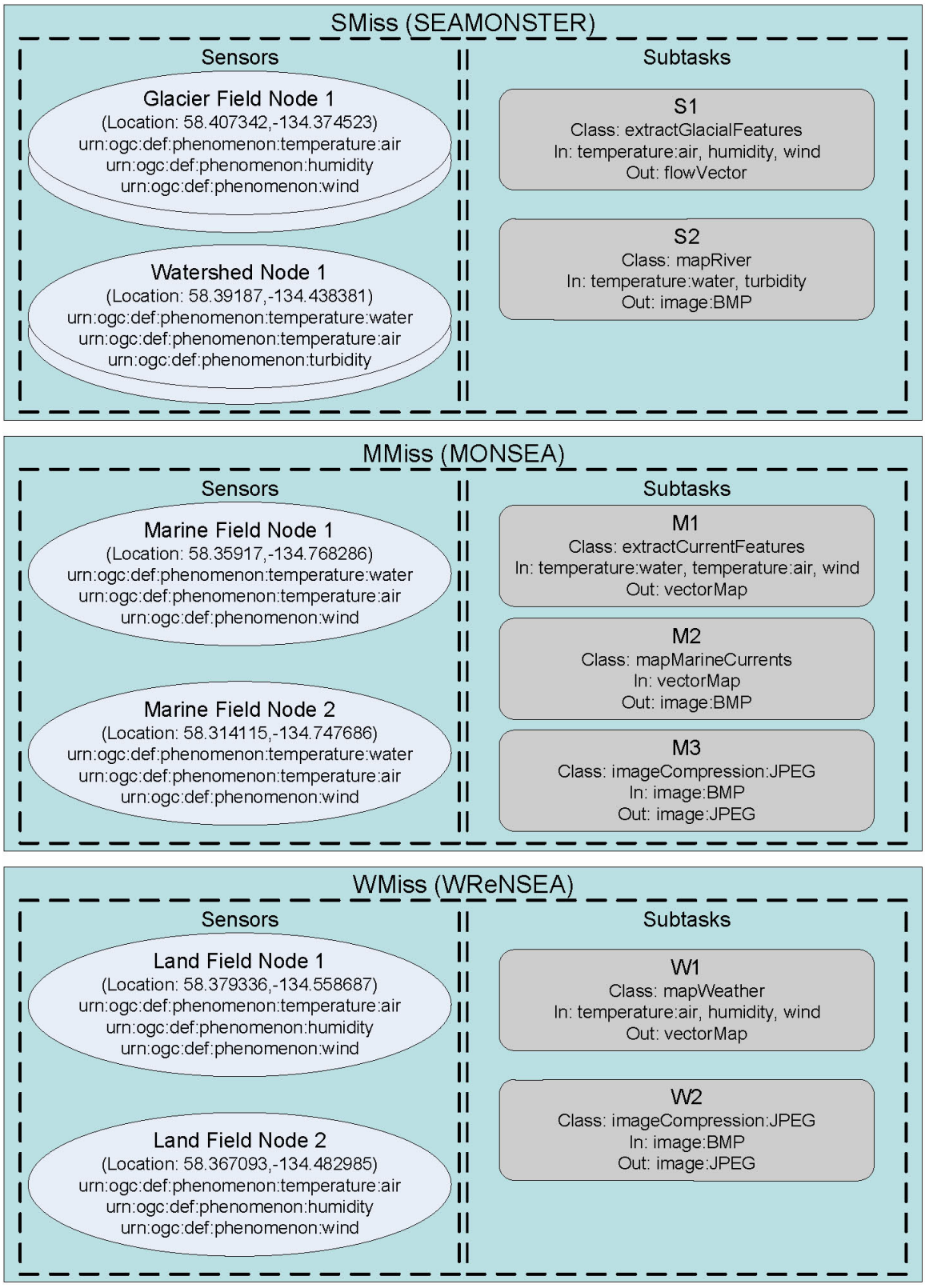

Figure 27: Mission agent sensors and subtasks in SEA scenario

in Figure 27. The Tier 2 Broker agent also requests sensor capability information from the Mission agents, and forwards the task and sensor information to the Tier 1 Broker.

V.6.3 End-to-End Scenario Evolution

The SEA scenario begins with the announcement of tasks by PUser and GUser. The Tier 1 Broker receives these announcements and translates them into potential TÆMS task decompositions, as described in Section V.4.2. The resulting, first-level task decompositions are illustrated in Figure 28. PTask1 and PTask2 each require image compression, which can be provided by either MMiss or WMiss. Therefore, each of those tasks have two possible decompositions. The Tier 1 Broker also assigns a system value to each task, as described in Section III.4.2. Because neither User agent has any previously announced or allocated tasks in this scenario, the Broker agent assigns a higher value to GTask1, which is from the User agent of greater importance to the overall sensor web.

The Tier 1 Broker forwards the translated task announcements to the Tier 2 Broker agent, which passes them to the Mission agents that can perform at least one of the relevant subtasks. As described in Section III.4.1 and Section III.5, each Mission agent produces an initial bid based on its own capabilities for the received task announcements. WMiss and MMiss each bid on PTask1 and PTask2 because they can perform at least one subtask of each task. Similarly, SMiss produces an initial bid for both PTask1 and GTask1. PUser sends pre-accepts for the more complete bids from WMiss for PTask1 and MMiss for PTask2, and GUser sends a pre-accept to SMiss for its bid on GTask1. At this point, WMiss attempts to subcontract the subtasks of PTask1 that it cannot perform to SMiss. SMiss must choose between making a final bid on GTask1 or PTask1, which cannot both be performed due to its bandwidth constraints. GTask1 has the higher broker-assigned value, so SMiss chooses to make a final bid to GUser for that task. Since SMiss rejects PTask1, WMiss cannot make a final bid on PTask1. PUser attempts to pre-accept MMiss for PTask1. However, when MMiss attempts to subcontract the same portions of the task,

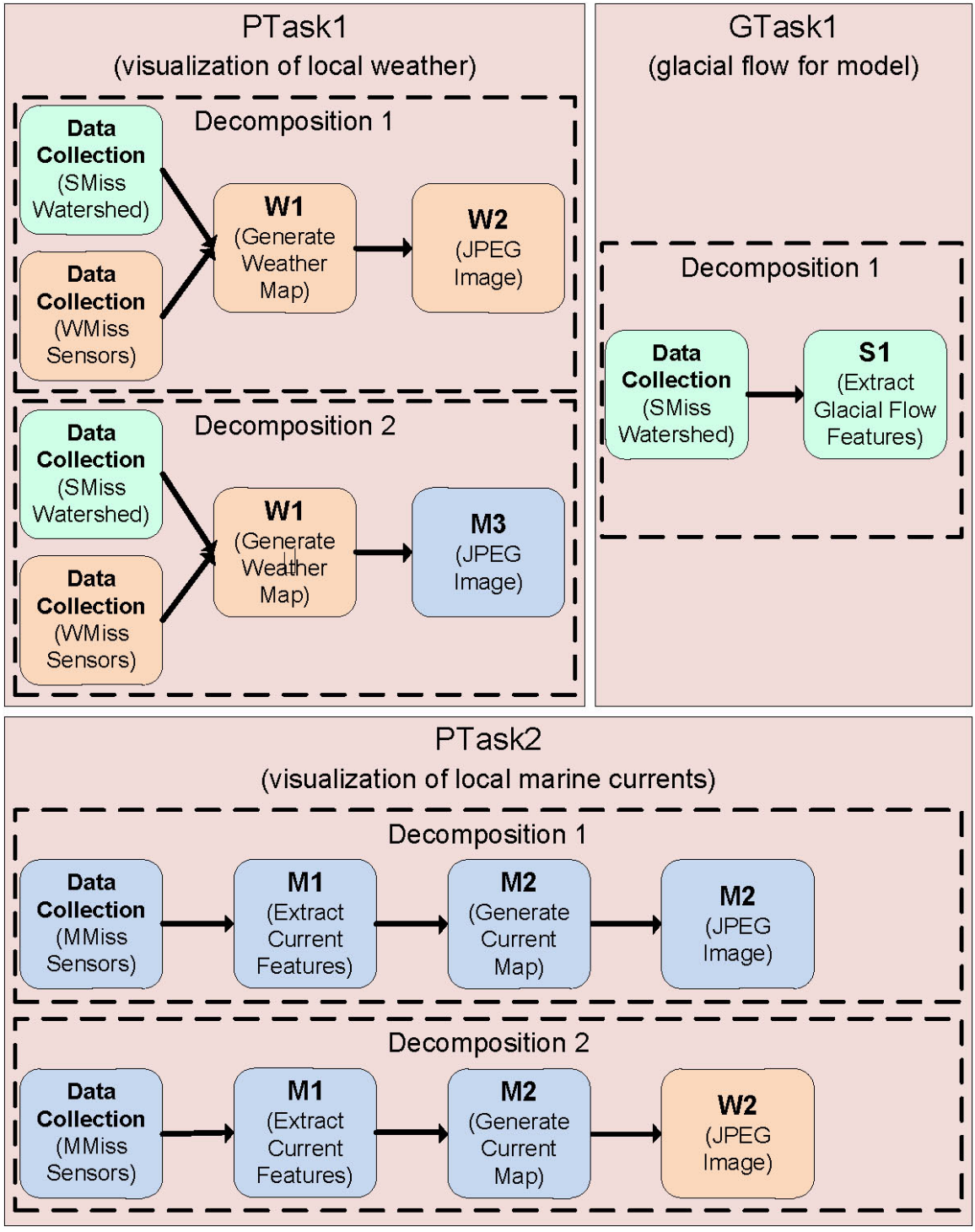

Figure 28: Broker translations of User agent tasks

SMiss again rejects them. Similarly, when PUser makes its final pre-accept attempt to SMiss, it is rejected. Ultimately, GUser accepts the final bid from SMiss for GTask1, and PUser accepts the final bid of MMiss for PTask2.

After the tasks have been allocated, SMiss and MMiss employ criteria-directed scheduling for their respective tasks, as described in Section IV.1.3. As described in Section V.5, SMiss passes the appropriate goals to SExecG for resource-level planning and scheduling to achieve GTask1. Similarly, MMiss passes the goals specified by methods in its plan for PTask2 to its Exec agent. At the start time specified in their tasks, the Mission agents instruct their Exec agents to begin execution of their plans for the specified tasks.

During execution of GTask1, the glacial lake begins draining. This event causes SMiss to activate an internal SEAMONSTER task, STask1, which includes high data rate monitoring of the watershed during lake drainage. However, due to SEAMONSTER's limited bandwidth, it cannot find valid task decompositions to achieve both this internal task and GTask1. The internal task has a higher priority than the external task, GTask1, so SMiss revokes the GTask1 goals from SExecG and provides goals for watershed monitoring to SExecW. This illustrates the dynamic adaptability of the MACRO resource level for transient environmental phenomena. However, since SMiss is no longer attempting to achieve GTask1, it informs GUser it has broken the contract for GTask1. Since no other sensor network can perform GTask1, GUser must wait until SMiss is available again to complete its task.

V.7 Experimental Evaluation

Section V.2 identified the challenge of achieving efficient coordination and interoperability between the two different forms of planning and scheduling employed at the MACRO mission and resource levels. MACRO's guided, context-sensitive coordination

between Mission agent and Exec agent planning/scheduling resolves this challenge, as discussed in Section V.5. This section presents the results of Mission and Exec agent coordination through the simulated execution of randomly-generated resource-level plans with a variety of duration distributions for actions. These experiments are intended to illustrate some of the benefits of MACRO's guided, context-sensitive coordination in planning/scheduling. In particular, these experiments show that MACRO's planning/scheduling translation and coordination can significantly decrease communication and computation overhead in comparison to a baseline of unguided coordination approach. In the baseline coordination, no context information (*i.e.*, no threshold or indication that the Mission agent is only interested in changes that exceed that threshold) is provided to the Exec agent.

V.7.1 Experimental Design

Our experiments simulate a scheduled, partial-order plan generated by SA-POP for an Exec agent at the resource level of MACRO. These plans include a set of actions with expected start and end time windows, as well as ordering links (specifically, causal links and the links generated by scheduling and causal threat resolution). For these experiments, we only simulate cases in which a valid plan can be generated and consider several variations on the execution context in terms of action duration.

An important parameter in these experiments is the variability of actual action durations. Action durations are generated from a probability distribution parameterized by a sigma value. These experiments included both uniform distributions and Gaussian (Normal) distributions. The uniform distributions showed the same trends observed in the Gaussian distributions. Moreover, the Gaussian distributions are likely to more accurately represent real-world action durations and proved the more difficult cases for MACRO plan/schedule coordination. Therefore the results presented here are based on the the Gaussian distributions. In these experiments, the action duration distributions have a mean of

100 seconds and "low" and "high" variance scenarios were based on a 95% likelihood that durations would be within 25 seconds or 75 seconds of the mean, respectively.

Another experimental variable is the length of the critical path in resource-level plans. The critical path is the ordered sequence of actions in the plan that constrains the end time of the plan (*i.e.*, the sequence of actions that requires the longest time to complete). Since each action has an expected duration of 100 seconds, the expected time for completion of the plan is determined by the number of actions in the critical path. In these experiments, the critical path length was varied between one action and 40 actions.

The final experimental variable is the time threshold provided by the Mission agent in MACRO context-sensitive coordination, which determines how far actions can surpass their expected end times before the Mission agent must be notified for potential mission-level re-planning and re-scheduling. To assess computation and communication overhead of the coordination mechanism, we employed random generation of plans across a range of parameters rather than using a few example problems. These experiments do not assess the quality or utility of plans or potential plan changes during coordination. MACRO coordination will not result in any degradation of plan quality in comparison to the baseline coordination, however, since plan and schedule information that triggers mission-level re-planning and re-scheduling is provided by both MACRO coordination mechanism and the baseline mechanism at the same time.

Since these experiments employ randomly-generated plans to cover a range of potential applications, they do not allow changes to resource-level or mission-level plans during execution. Whenever an action execution exceeded its scheduled end window, the schedule was updated and communicated to the Mission agent, but no changes to the plan or threshold were made. Without re-planning, the MACRO coordination overhead is an overestimate of the real overhead. After a critical path action's end window is exceeded, execution of further actions will continue to exceed action end windows. Re-planning reduces this possibility.

V.7.2 Experimental Results

Each experimental run included 10,000 trials with the given parameter settings. In each trial, a series of (n) actions formed the critical path, and each action had an expected duration of 100 seconds. Using the chosen distribution, random values are generated that correspond to actual execution times. The number of updates and messages are calculated using those values.

V.7.2.1 Investigating Critical Path Length

These experiments were performed under the assumption that the Mission agent simply requires a method to be completed by the provided deadline and should only be notified if the expected execution time will exceed that deadline. The threshold value is therefore set to the difference between the deadline and the expected duration of the plan. This threshold is varied in the experiments between 0 and 200 seconds in 5 second increments.

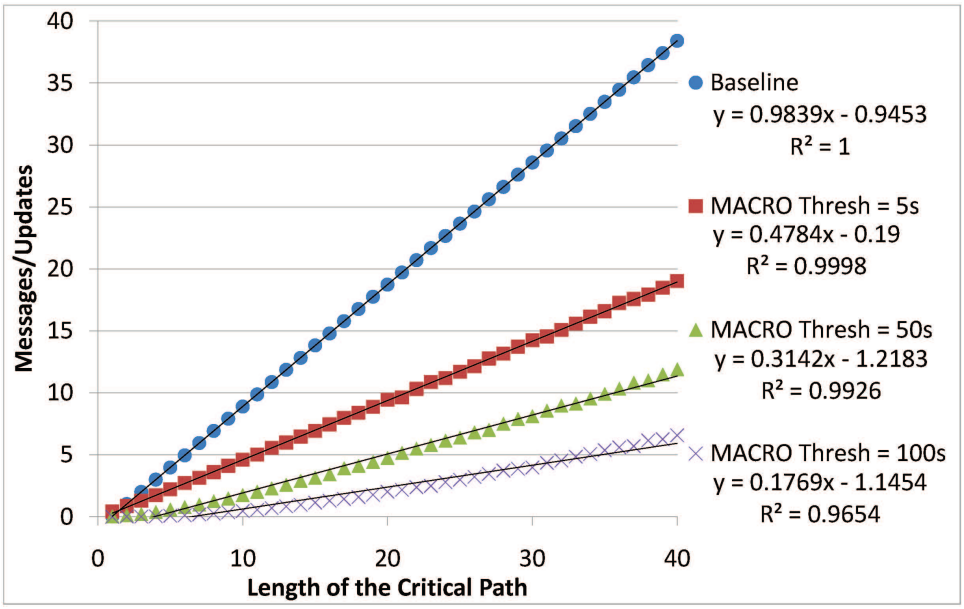

Figure 29: Effect of critical path length with a low variance Gaussian

Figure 29 shows the information from the Mission agent results in significantly less

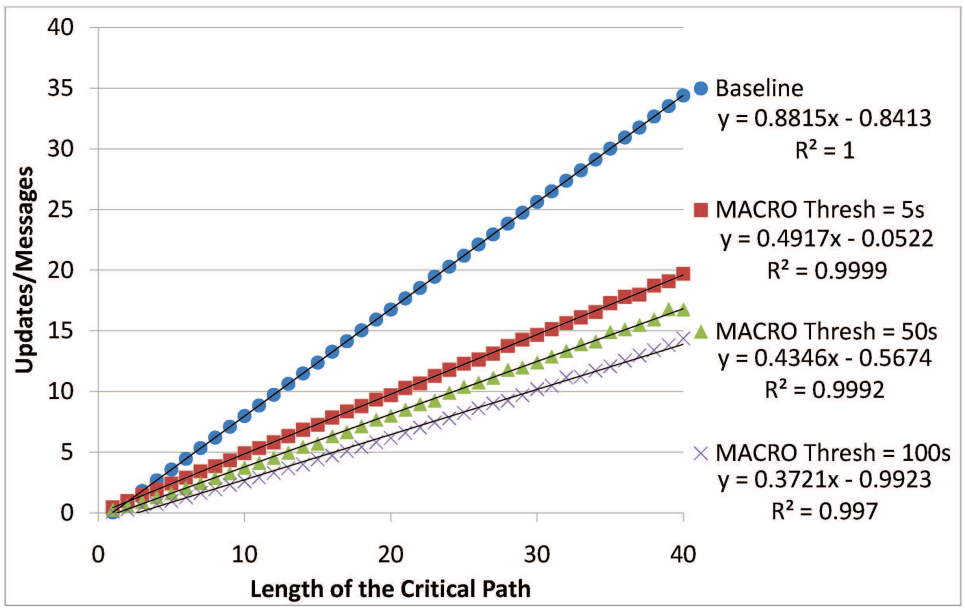

Figure 30: Effect of critical path length with a high variance Gaussian

computation and communication than the baseline condition for all but the smallest of critical paths. The linear trend suggests that in the worst case (*i.e.*, a tight threshold/deadline), MACRO sends about half as many messages as the baseline machanism. As the threshold increases, MACRO performs even better, whereas the baseline performance does not change. This illustrates MACRO's ability to leverage the context information of deadline thresholds to minimize coordination overhead.

A comparison of the low variance action duration distribution in Figure 29 to a high variance one in Figure 30 shows that with the smallest thresholds a ratio of approximately 1 update per 2 actions in the critical path is required for both distributions. The 1:2 ratio is thus an approximate upper limit on the average number of updates required in MACRO, even when re-planning and re-scheduling is not possible.

The baseline mechanism shows a slight, relative improvement in the high variance case, but MACRO's context-sensitive coordination still requires far fewer updates. However, the number of updates required in MACRO with different thresholds are much closer in the high variance case than the low variance case. This result suggests that when action

durations are less certain, the critical path length is significantly more important than the threshold, because even large thresholds can be exceeded by a series of actions that begins with an unexpectedly long-running action.

V.7.2.2 Investigating Time Thresholds

Figure 31 and Figure 32 show the trends in communication and computation with respect to the duration threshold. The baseline results are not included in these figures because they do not use of the threshold value, therefore, they would produce a horizontal line close to the number of actions in the critical path.

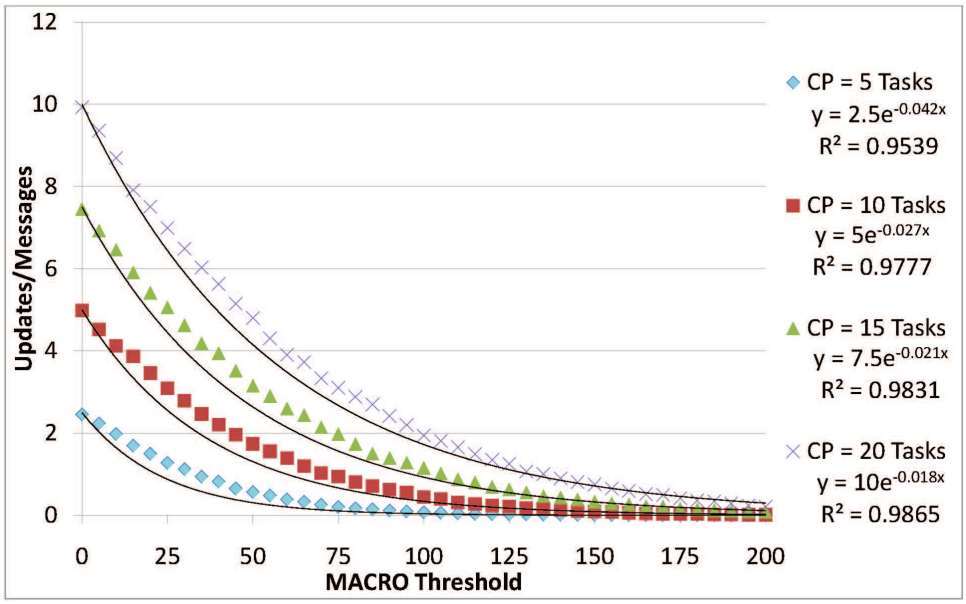

Figure 31: Effect of time threshold with a low variance Gaussian

These results show that as the threshold increases, the number of MACRO updates decreases. Figure 31, shows a steep initial decrease which levels off. Qualitatively, this trend occurs since longer thresholds allow a series of actions to exceed their expected duration by a greater amount before requiring an update. Extreme variation, however, from expected durations can occur and will still require some updates, even with relatively large

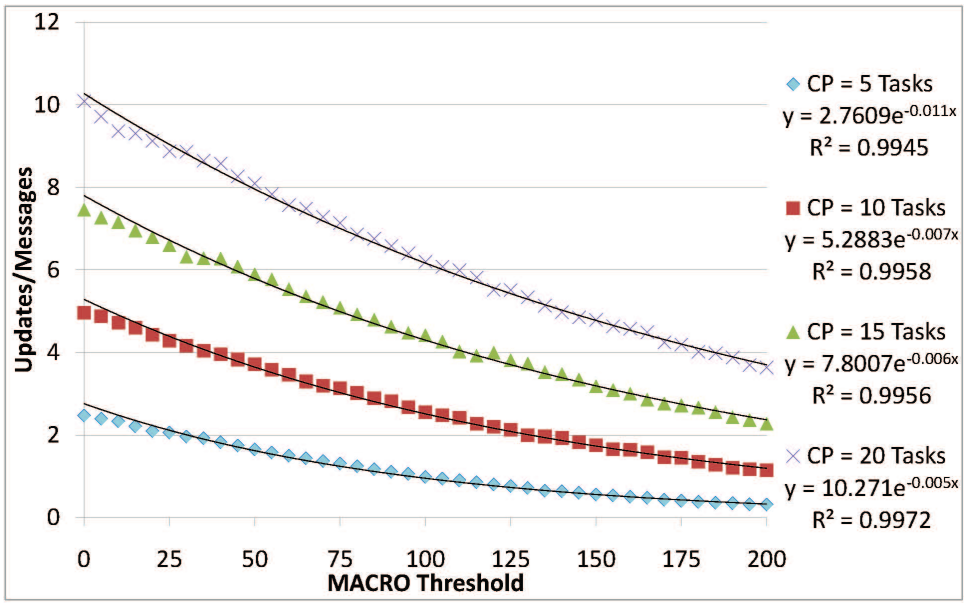

Figure 32: Effect of time threshold with a high variance Gaussian

thresholds. These results also show, that even when uncertainty of action duration is high, the Exec agent can leverage the contextual information provided by the Mission agent to minimize coordination overhead.

V.8 Summary

This chapter presented the challenges and solution approach for implementing and integrating a proof-of-concept (PoC) system employing MACRO task allocation and planning/scheduling mechanisms. Section V.1 discussed existing work related to the design of the MACRO PoC implementation and Section V.2 identified the challenges not resolved by existing work.

Employing OGC SensorML [9] standards allows MACRO to support interoperability with other external tools and systems employing these standards. Further, SensorML provides a standard representation for User agent tasks and a way to standardize Mission agent task trees across sensor networks. Section V.4 described MACRO's translation of

User tasks in SensorML to Mission agent TÆMS tasks and aggregation of Mission agent TÆMS task trees augmented with SensorML meta-data.

To perform allocation and distributed planning/scheduling, MACRO Mission agents employ multiple coordination protocols (*e.g.*, the contract net protocol and GPGP distributed planning protocols), which they must perform simultaneously. Further, a Mission agent may execute on the same set of computational resources as some resource-level agents. Therefore, the MACRO prototype Mission agent has an extensible, object-oriented design including meta-control to coordinate its diverse activities, which was described in Section V.3.

To achieve the high-level tasks allocated and coordinated at the mission level, individual Mission agents must communicate appropriate subtasks or subgoals to their resource-level Exec agents for execution. However, MACRO Mission and Exec agents employ different representations and forms of planning and scheduling, which complicates their integration in the overall MACRO system. Section V.5 detailed the context-sensitive coordination of planning/scheduling and translation of planning and scheduling representations between the MACRO Mission and Exec agents.

To illustrate the capabilities of the integrated MACRO system, Section V.6 provided a case study of MACRO system operation for a set of simulated sensor networks. Further, Section V.7 presented experimental results verifying the efficacy and efficiency of MACRO planning and scheduling coordination between the mission and resource levels.

CHAPTER VI

CONCLUDING REMARKS

Effective coordination and control of a global sensor web requires an open, efficient, scalable system to allocate, plan/schedule, and coordinate the operation of its many heterogeneous, independent components in a dynamic, effective, and equitable manner. Section I.3 identified specific research challenges in task allocation, planning/scheduling, and system integration to be addressed in this work on sensor web coordination and control. This dissertation has described the Multi-agent Architecture for Coordinated Responsive Observations (MACRO), designed to resolve those research challenges. MACRO provides a powerful computational infrastructure for enabling the deployment and adaptive operation of large, distributed systems that require high-level coordination of complex tasks among agents, as well as local, dynamic adaptation for effective use of limited resources in dynamic, uncertain environments.

VI.1 Summary and Research Contributions

Chapter II defined the Multi-agent Architecture for Coordinated Responsive Observations (MACRO) and provided an overview of its agents' roles and relationships, agent services, and the middleware infrastructure on which MACRO is built. MACRO uses QoS-enabled component middleware to help automate many system configuration and management tasks for sensor web agents and applications. Atop the middleware infrastructure, MACRO's dynamic resource management service, RACE, provides efficient allocation and control of computational resources, while MACRO agents employ a decision-theoretic planning and scheduling service, SA-POP, to autonomously adapt system functionality to changing science objectives and environmental conditions. MACRO agents are organized

in a two-level hierarchy: (1) the mission level spanning the sensor web's constituent sensor networks and (2) the resource level for adaptive operation of local sensor network resources. This separation of concerns enables tractable solutions to the coordination and control problems facing a system with the scope of a global sensor web.

In Chapter III, we presented a novel approach to achieving both fairness (*i.e.*, individual user satisfaction) and efficiency (*i.e.*, value to the system as a whole) in allocation of sensor web tasks. To integrate the allocation concerns of fairness and efficiency, we presented the definition of a sensor web task allocation metric, which combines a measure of user satisfaction and a measure of total system utility for a set of allocated tasks. We illustrated how this metric is employed by the MACRO brokers to value tasks and described MACRO's task auctions in a brokered, two-phase contract net, including a novel subcontracting approach to minimize message overhead. Finally, we presented results of experiments with MACRO task allocation that verify both its allocation performance and scalability. The major contributions of this chapter to the research in multi-agent systems are:

- A novel task allocation metric: To allow a tradeoff between system-wide value and user satisfaction, the metric weights allocation efficiency by an agent satisfaction level. Unlike utilitarian, egalitarian, or weighted combination metrics, this metric provides a preference for allocations in which most agents are highly satisfied and the allocated tasks provide a high value to the overall system, but does not require parameter tweaking for different system configurations (*e.g.*, system load and relative importance of user agents).

- A novel task allocation mechanism: To achieve scalable allocation of tasks while respecting allocation fairness and efficiency criteria, we employ brokered task auctions including efficient subcontracting and task valuation by an approximation of marginal utility. For effective design and implementation of necessary broker agents, we include two tiers of brokers that provides a separation between specific broker

roles and responsibilities. Generally, this mechanism is applicable to large multi-agent systems with heterogeneous users and hierarchically-decomposable tasks requiring multiple agents' resources. In such systems, the task allocation mechanism produces fair and efficient allocations of tasks while requiring only minimal infrastructure computational capabilities and not limiting user agent preferences on task achievement criteria. Experimental evaluation demonstrated the allocation mechanism's performance and scalability across a range of User agent share ratios, Mission agent densities, and task characteristics.

Chapter IV described the planning and scheduling mechanisms employed by MACRO agents. Mission agents employ the TÆMS hierarchical task-tree representation with GPGP coordination and criteria-directed scheduling/task-decomposition to cooperatively achieve allocated tasks. At the resource level, planning requires the use of available probabilistic domain information for generation of high expected utility plans and the integration of an efficient scheduling mechanism. We presented the details of SA-POP, which is a novel, decision-theoretic planning and scheduling service for MACRO resource-level agents to support adaptation of sensor network operations. Further, we presented experimental results that verify SA-POP's ability to produce high expected utility plans even under significant scheduling constraints. Together, the planning and scheduling capabilities of MACRO mission- and resource-level agents facilitate the real-time collection and analysis of sensor data, even under changing environmental conditions and many concurrent science objectives. The major contribution of this chapter to the research in autonomous planning and scheduling is:

- A novel planning and scheduling mechanism: Existing decision-theoretic planning mechanisms do not provide targeted scheduling during planning to produce high expected utility plans of autonomous action and application adaptation for distributed,

real-time embedded systems. To address this deficiency, SA-POP provides decision-theoretic planning and scheduling for agents operating in a dynamic, uncertain environment with limited, shared resources. This planning and scheduling mechanism employs a decision-theoretic heuristic to guide planning and efficiently integrates scheduling by leveraging a scheduling criticality measure for targeted scheduling during the planning process. Therefore, agents can adapt system operation in uncertain, resource-limited domains by executing the high expected utility, scheduled plans produced by this mechanism. Experimental evaluation demonstrated the planning and scheduling mechanism's performance across a range of task network sizes and connectedness.

Chapter V presented the challenges and solution approach for implementing and integrating a proof-of-concept (PoC) system employing MACRO task allocation and planning/scheduling mechanisms. We presented the MACRO Mission agent internal architecture, which allows Mission agents to integrate their multiple allocation, planning/scheduling, coordination, and domain operation activities. Chapter V also illustrated MACRO's use of the OGC SensorML [9] standard, which can enable interoperability with future sensor web tools and systems. Further, SensorML provides a standard representation for User agent tasks and a way to standardize Mission agent task trees across sensor networks. We presented MACRO's aggregation of domain information (*i.e.*, TÆMS task trees augmented with SensorML meta-data) across sensor networks and translation of User tasks from SensorML descriptions to one-level decompositions of TÆMS tasks. We also presented the guided, context-sensitive coordination of planning/scheduling and translation of planning and scheduling representations between the MACRO Mission and Exec agents. Experimental results verified the overhead reduction achieved with this coordination mechanism. Finally, to illustrate the capabilities and operation of the integrated MACRO system, we

provided a case study of MACRO execution for a scenario including three simulated sensor networks. The major contributions of this chapter to the research in multi-agent systems and autonomous planning and scheduling are:

- A sensor web task translation and aggregation mechanism: Implementation of a sensor web system with heterogeneous users and providers requires task translation and aggregation in order to leverage both SensorML standards and a hierarchically-decomposable task representation. Therefore, we employ a distributed set of broker agents to aggregate and translate tasks for other sensor web agents. This mechanism allows sensor web users to express tasks as constraints based on standard SensorML processes while providers can employ a hierarchically-decomposable task representation appropriate to their distributed planning and scheduling requirements. A case study illustrated the effectiveness of this mechanism in a MACRO sensor web.

- A novel planning/scheduling coordination mechanism: Existing work does not provide an efficient mechanism for coordinating scheduled plans between the two different planning and scheduling representations employed in MACRO. Therefore, this mechanism leverages mission-level context information, such as deadline thresholds and optimization criteria, to provide efficient coordination between MACRO Mission and Exec agents. In general, this mechanism is applicable to agents in a hierarchical relationship where the top level employs hierarchical task decomposition and (re)scheduling while the bottom level employs decision-theoretic, first-principles (re)planning and (re)scheduling. In such systems, this mechanism enables efficient coordination of scheduled plans for adaptation to local system conditions while obeying top-level goals and constraints. Experimental evaluation demonstrated the coordination mechanism's performance and scalability across a range of time thresholds and plan lengths.

VI.2 Future Research Directions

Given the complexity and scope of a global sensor web, there are many opportunities for future research with the MACRO framework. Promising research directions for extending the capabilities and performance of MACRO include:

Task decommitment. Section III.6.2 presented experimental results verifying MACRO near-optimal allocation performance across a range of user share ratios, tasks per round, and trial lengths. However, with longer task execution times, results indicated a decline in allocation performance. This decline was due to the fact that MACRO Mission agents could not decommit from an allocated task even when an announced task was of significantly greater value. Decommitment from tasks in a sensor web is particularly difficult because of the impact on real-world user satisfaction. In future work, we would investigate decommitment schemes to improve MACRO allocation performance with long-running tasks. In particular, we could define an appropriate threshold (between committed task value and announced task value), enforced by MACRO Broker agents, that will allow decommitments in extreme cases to increase overall allocation performance, while minimizing user dissatisfaction from decommitted tasks.

Dynamic subcontracting. Section III.6.1 presented experimental results showing that the MACRO limited pre-commitment subcontracting approach scales significantly better than the pre-bid subcontracting approach The required number of messages to reach the best final bid in pre-commitment subcontracting are on the order of five times fewer than pre-bid subcontracting for likely sensor web system configurations and operating conditions. However, the large number of tasks likely to be announced in a global sensor web suggests that further reduction in subcontracting message overhead would be worthwhile in sensor web task allocation. In future work, we could include dynamic adjustment of pre-accept cutoff values based on announced task and current configuration/conditions, as well as caching of subtask bids by broker agents, to decrease the number of messages required. Further, we can explore allowing User agents to include information on the utility

of specific bid quality criteria to yield smaller, more directed, Mission agent bids. These extensions should further diminish the message overhead required in pre-commitment subcontracting, while maintaining the scalability of this approach.

Mission agent meta-reasoning. To coordinate task achievement, MACRO Mission agents employ multiple protocols (*e.g.*, a broker-mediated contract net protocol and GPGP distributed planning protocols). Results of planning and scheduling can affect bidding on tasks in the contract net, and contracted tasks require further planning, scheduling, and GPGP coordination. The prototype MACRO Mission agent described in Section V.3 attempts to perform planning and scheduling for all announced tasks. However, a Mission agent may execute on the same set of computational resources as some resource-level agents, such as in the SEAMONSTER sensor net described in Section IV.5. Therefore, MACRO would benefit from Mission agents that can balance the amount of computation expended on each of its various internal activities to maximize the utility of its sensor network. In particular, with decomposable tasks and criteria-directed scheduling, additional computation may yield a better estimate of whether the task can be accomplished given current commitments, as well as more accurate schedules and estimates of the amount of work required for the task. For deciding whether to bid on announced tasks and determining the expected quality and duration of a task for bidding, there is a trade-off between amount of computation and accuracy of the required information. In future work, we could investigate advanced meta-reasoning and meta-control strategies to allow the Mission agent to determine the expected benefit of further computation across its different computationally-intensive activities.

APPENDIX A

LIST OF PUBLICATIONS

Our research on MACRO and SA-POP has lead to the following journal, conference, and workshop publications.

A.1 Refereed Journal Publications

J-1 Nishanth Shankaran, John S. Kinnebrew, Xenofon Koutsoukos, Chenyang Lu, Douglas C. Schmidt, and Gautam Biswas, "An Integrated Planning and Adaptive Resource Management Architecture for Distributed Real-time Embedded Systems", *IEEE Transactions on Computers: Special Issue on Autonomic Network Computing* 58(11):1485–1498, November, 2009.

A.2 Refereed Conference Publications

C-1 John S. Kinnebrew, Daniel L.C. Mack, Gautam Biswas, and Douglas C. Schmidt, "Coordination of Planning and Scheduling Techniques for a Distributed, Multi-level, Multi-agent System", *The International Conference on Agents and Artificial Intelligence (ICAART 2010)*, Valencia, Spain, January 22-24, 2010.

C-2 John S. Kinnebrew and Gautam Biswas, "Efficient Allocation of Hierarchically-Decomposable Tasks in a Sensor Web Contract Net", *The IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2009)*, Milan, Italy, September 15-18, 2009.

C-3 John S. Kinnebrew, William R. Otte, Nishanth Shankaran, Gautam Biswas, and Douglas C. Schmidt, "Intelligent Resource Management and Dynamic Adaptation in a Distributed Real-time and Embedded Sensor Web System", *The 12th International*

Symposium on Object/Component/Service-oriented Real-time Distributed Computing (ISORC 2009), Tokyo, Japan, March 17-20, 2009.

C-4 William R. Otte, John. S. Kinnebrew, Douglas C. Schmidt, and Gautam Biswas, "A Flexible Infrastructure for Distributed Deployment in Adaptive Sensor Webs", *The 2009 IEEE Aerospace Conference*, Big Sky, Montana, March 7-14, 2009.

C-5 William R. Otte, John. S. Kinnebrew, Douglas C. Schmidt, Gautam Biswas, and Dipa Suri, "Application of Middleware and Agent Technologies to a Representative Sensor Network", *The Eighth Annual NASA Earth Science Technology Conference*, College Park, Maryland, June 24-26, 2008.

C-6 Nilabja Roy, John S. Kinnebrew, Nishanth Shankaran, Gautam Biswas, and Douglas C. Schmidt, "Toward Effective Multi-capacity Resource Allocation in Distributed Real-time and Embedded Systems", *The 11th IEEE International Symposium on Object/Component/Service-oriented Real-time Distributed Computing (ISORC 2008)*, Orlando, Florida, May 5-7, 2008.

C-7 John S. Kinnebrew, Gautam Biswas, Nishanth Shankaran, Douglas C. Schmidt, and Dipa Suri, "Integrating Task Allocation, Planning, Scheduling, and Adaptive Resource Management to Support Autonomy in a Global Sensor Web", *The NASA Science Technology Conference*, College Park, Maryland, June 19-21, 2007.

C-8 John S. Kinnebrew, Ankit Gupta, Nishanth Shankaran, Gautam Biswas, and Douglas C. Schmidt, "A Decision-Theoretic Planner with Dynamic Component Reconfiguration for Distributed Real-Time Applications", *The 8th International Symposium on Autonomous Decentralized Systems (ISADS 2007)*, Sedona, Arizona, March 21-23, 2007.

C-9 Dipa Suri, Adam Howell, Douglas C. Schmidt, Gautam Biswas, John S. Kinnebrew, Will Otte, and Nishanth Shankaran, "A Multi-agent Architecture for Smart Sensing in

the NASA Sensor Web", *The 2007 IEEE Aerospace Conference*, Big Sky, Montana, March 3-10, 2007.

C-10 John S. Kinnebrew, Nishanth Shankaran, Gautam Biswas, and Douglas Schmidt, "A Decision-Theoretic Planner with Dynamic Component Reconfiguration for Distributed Real-Time Applications", poster paper at *The 21st National Conference on Artificial Intelligence (AAAI 2006)*, Boston, Massachusetts, July 16-20, 2006.

C-11 Dipa Suri, Adam Howell, Nishanth Shankaran, John S. Kinnebrew, Will Otte, Douglas C. Schmidt, and Gautam Biswas, "Onboard Processing using the Adaptive Network Architecture", *The Sixth Annual NASA Earth Science Technology Conference*, College Park, Maryland, June 27-29, 2006.

A.3 Refereed Workshop Publications

W-1 Nishanth Shankaran, John S. Kinnebrew, Xenofon D. Koutsoukos, Chenyang Lu, Douglas C. Schmidt, and Gautam Biswas, "Towards an Integrated Planning and Adaptive Resource Management Architecture for Distributed Real-time Embedded Systems", *Proceedings of the Workshop on Adaptive and Reconfigurable Embedded Systems (APRES)* at the *14th IEEE Real-Time and Embedded Technology and Applications Symposium*, St. Louis, MO, United States, April 22 - April 24, 2008.

W-2 John S. Kinnebrew, Nishanth Shankaran, Gautam Biswas, and Douglas C. Schmidt, "A Decision-Theoretic Planner with Dynamic Component Reconfiguration for Distributed Real-time and Embedded Systems," *Proceedings of the Workshop on Artificial Intelligence for Space Applications at IJCAI 2007*, Hyderabad, India, January 6-12, 2007.

APPENDIX B

LIST OF ACRONYMS

ACE	Adaptive Communication Environment
CCM	CORBA Component Model
CIAO	Component Integrated ACE ORB
CNP	Contract Net Protocol
CORBA	Common Object Request Broker Architecture
DAnCE	Deployment and Configuration Engine
DRE	Distributed, Real-time, Embedded
EU	Expected Utility
GPGP	Generalized Partial Global Planning
HTN	Hierarchical Task Network
IPC	International Planning Competition
MACRO	Multi-agent Architecture for Coordinated, Responsive Observations
MMS	Magnetospheric MultiScale mission
OGC	Open Geospatial Consortium
OMG	Object Management Group
ORB	Object Request Broker
PDDL	Planning Domain Definition Language

POCL Partial-Order Causal-Link

POP Partial-Order Planning

QAF Quality Aggregation Function

QoS Quality of Service

SA-EU Spreading Activation Expected Utility

SA-POP Spreading Activation Partial Order Planner

SAML SA-POP Modeling Language

RACE Resource Allocation and Control Engine

TÆMS Task Analysis, Environment Modeling, and Simulation

TAO The ACE ORB

REFERENCES

[1] S. Aknine, S. Pinson, and M. Shakun. An Extended Multi-Agent Negotiation Protocol. *Autonomous Agents and Multi-Agent Systems*, 8(1):5–45, 2004.

[2] G. Alexander, A. Raja, E. Durfee, and D. Musliner. Design Paradigms for Meta-Control in Multiagent Systems. In *Proceedings of AAMAS 2007 Workshop on Metareasoning in Agent-Based Systems*, pages 92–103, Hawaii, USA, May 2007.

[3] G. Alexander, A. Raja, and D. Musliner. Controlling deliberation in a markov decision process-based agent. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '08)*, pages 461–468, Estoril, Portugal, 2008.

[4] S. Bagchi, G. Biswas, and K. Kawamura. Task Planning under Uncertainty using a Spreading Activation Network. *IEEE Transactions on Systems, Man, and Cybernetics*, 30(6):639–650, Nov. 2000.

[5] R. Barták. Conceptual models for combined planning and scheduling. *Electronic Notes in Discrete Mathematics*, 4:1, 2000.

[6] A. Blum and M. Furst. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence*, 90(1):281–300, 1997.

[7] A. Blum and J. Langford. Probabilistic Planning in the Graphplan Framework. *Proceedings of the Fifth European Conference on Planning*, pages 319–332, 1999.

[8] B. Bonet. New grid-based algorithms for partially observable markov decision processes: Theory and practice, 2006.

[9] M. Botts et al. Sensor Model Language (SensorML). Technical Report OpenGIS Implementation Specification Document 07-000, Open Geospatial Consortium, July 2007.

[10] A. Bouguerra and L. Karlsson. Hierarchical Task Planning Under Uncertainty. *3rd Italian Workshop on Planning and Scheduling (AI* IA 2004). Perugia, Italy*, 2004.

[11] S. Brams, P. Edelman, and P. Fishburn. Paradoxes of Fair Division. *Journal of Philosophy*, 98(6):300–314, 2001.

[12] S. Brams, P. Edelman, and P. Fishburn. Fair Division of Indivisible Items. *Theory and Decision*, 55(2):147–180, 2003.

[13] S. Brams and A. Taylor. Fair Division. *The Oxford Handbook of Political Economy*, 2006.

[14] M. Brenner. Multiagent Planning with Partially Ordered Temporal Plans. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 1513–1514, 2003.

[15] R. Brooks. Intelligence without representation. *Artificial intelligence*, 47(1-3):139–159, 1991.

[16] A. Cassandra, M. L. Littman, and N. L. Zhang. Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. In *In Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 54–61, 1997.

[17] D. Chelberg, L. Welch, A. Lakshmikumar, and M. Gillen. Meta-reasoning for a distributed agent architecture. In *Proceedings of the 33rd Southeastern Symposium on System Theory*, pages 377–381, Mar 2001.

[18] Y. Chevaleyre, P. Dunne, U. Endriss, J. Lang, M. Lemaitre, N. Maudet, J. Padget, S. Phelps, J. Rodriguez-Aguilar, and P. Sousa. Issues in Multiagent Resource Allocation. *Informatica*, 30(1):3, 2006.

[19] Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. A Short Introduction to Computational Social Choice. *Lecture Notes in Computer Science*, 4362:51, 2007.

[20] B. Clement and A. Barrett. Continual Coordination through Shared Activities. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 57–64. ACM New York, NY, USA, 2003.

[21] B. Clement and E. Durfee. Theory for Coordinating Concurrent Hierarchical Planning Agents using Summary Information. *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Innovative Applications of AI Conference*, pages 495–502, 1999.

[22] B. Clement and E. Durfee. Performance of Coordinating Concurrent Hierarchical Planning Agents using Summary Information. *Proceedings of the Fourth International Conference on Multi-Agent Systems*, pages 373–374, 2000.

[23] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, pages 213–261, 1990.

[24] J. Collins, M. Tsvetovat, B. Mobasher, and M. Gini. Magnet: A multi-agent contracting system for plan execution. *Proceedings of Workshop on Artificial Intelligence and Manufacturing: State of the Art and State of Practice*, pages 63–68, 1998.

[25] S. Cox. Observations and Measurements. Technical Report OpenGIS Implementation Specification Documents 07-022r1 and 07-002r3, Open Geospatial Consortium, December 2007.

[26] S. Curtis. The Magnetospheric Multiscale Mission...Resolving Fundamental Processes in Space Plasmas. *NASA STI/Recon Technical Report N*, pages 48257–+, Dec. 1999.

[27] B. Dasarathy, S. Gadgil, R. Vaidhyanathan, K. Parmeswaran, B. Coan, M. Conarty, and V. Bhanot. Network QoS Assurance in a Multi-Layer Adaptive Resource Management Scheme for Mission-Critical Applications using the CORBA Middleware Framework. In *IEEE RTAS*, 2005.

[28] S. de Jong. Fairness in Multi-agent Systems. *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems: Doctoral Mentoring Program*, pages 1742–1743, 2008.

[29] M. de Weerdt and C. Witteveen. Multiagent planning: problem properties that matter. In E. Durfee and D. Musliner, editors, *Proceedings of the AAAI Spring Symposium on Distributed Plan and Schedule Management*, number SS-06-04, pages 155–156, 2006.

[30] K. Decker, V. Lesser, and V. Lesser. Designing a Family of Coordination Algorithms. *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*, pages 73–80, 1995.

[31] K. Decker, K. Sycara, and M. Williamson. Middle-agents for the internet. In *In Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pages 578–583, 1997.

[32] M. E. Desjardins, E. H. Durfee, C. L. Ortiz, and M. J. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, 20:13–22, 1999.

[33] M. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[34] E. Durfee. Scaling up agent coordination strategies. *Computer*, 34(7):39–46, 2001.

[35] E. Durfee and V. Lesser. Partial global planning: a coordination framework for distributed hypothesis formation. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(5):1167–1183, 1991.

[36] A. El-Kholy and B. Richards. Temporal and Resource Reasoning in Planning: The parcPLAN Approach. *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-96)*, pages 614–618, 1996.

[37] U. Endriss and N. Maudet. Welfare engineering in multiagent systems. *Lecture Notes in Computer Science*, pages 93–106, 2004.

[38] U. Endriss, N. Maudet, F. Sadri, and F. Toni. On optimal outcomes of negotiations

over resources. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2003)*, pages 177–184. Citeseer, 2003.

[39] U. Endriss, N. Maudet, F. Sadri, and F. Toni. Resource allocation in egalitarian agent societies. *Secondes Journées Francophones sur les Modeles Formels d'Interaction (MFI-2003)*, pages 101–110, 2003.

[40] C. Excelente-Toledo, R. Bourne, and N. Jennings. Reasoning about Commitments and Penalties for Coordination between Autonomous Agents. *International Conference on Autonomous Agents: Proceedings of the Fifth International Conference on Autonomous Agents*, 2001:131–138, 2001.

[41] P. Faratin, C. Sierra, and N. Jennings. Negotiation Decision Functions for Autonomous Agents. *Int. Journal of Robotics and Autonomous Systems*, 24:159–182, 1998.

[42] D. R. Fatland, M. J. Heavner, E. Hood, and C. Connor. The SEAMONSTER Sensor Web: Lessons and Opportunities after One Year. *AGU Fall Meeting Abstracts*, pages A3+, Dec. 2007.

[43] R. Fikes and N. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3/4):189–208, 1971.

[44] FIPA. Communicative Act Library Specification. Technical Report Technical Report SC00037J, Foundation for Intelligent Physical Agents, December 2002.

[45] M. Fox and D. Long. PDDL2. 1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20(2003):61–124, 2003.

[46] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the Computational Complexity of Combinatorial Auctions: Optimal and Approximate Approaches. *International Joint Conference on Artificial Intelligence*, 16:548–553, 1999.

[47] A. Garcia and C. Lucena. Taming Heterogeneous Agent Architectures. *Communications of the ACM*, 51(5):75–81, 2008.

[48] A. Garrido, F. Barber, D. Sistemas, and I. ComputaciÃşn. Integrating planning and scheduling. *Applied Artificial Intelligence*, 15:2001, 2001.

[49] B. Gerkey and M. Mataric. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research*, 23(9):939, 2004.

[50] X. Gu and K. Nahrstedt. Dynamic QoS-Aware Multimedia Service Configuration in

Ubiquitous Computing Environments. In *Proceedings of IEEE International Conference on Distributed Computing Systems*, 2002.

[51] P. Haddawy, A. Doan, and R. Goodwin. Efficient Decision-Theoretic Planning: Techniques and Empirical Analysis. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 1995.

[52] G. T. Heineman and B. T. Councill. *Component-Based Software Engineering: Putting the Pieces Together*. Addison-Wesley, Reading, Massachusetts, 2001.

[53] P. Hildebrand, W. Wiscombe, M. Albjerg, J. Booth, R. Miller, T. Miller, M. Mlynczak, G. Paules, D. Peterson, C. Raymond, et al. NASA Earth Science Vision 2030: Working Group Report. Technical Report NP-2003-2-611-GSFC, NASA, 2004.

[54] B. Horling and V. Lesser. A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19(04):281–316, 2005.

[55] B. Horling, V. Lesser, R. Vincent, T. Wagner, A. Raja, S. Zhang, K. Decker, and A. Garvey. The TAEMS White Paper. Technical report, Multi-Agent Systems Lab, University of Massachusetts, 1999.

[56] L. Hunsberger and B. Grosz. A combinatorial auction for collaborative planning. *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000)*, pages 151–158, 2000.

[57] N. Jennings, P. Faratin, A. Lomuscio, S. Parsons, M. Wooldridge, and C. Sierra. Automated Negotiation: Prospects, Methods and Challenges. *Group Decision and Negotiation*, 10(2):199–215, 2001.

[58] G. Karsai, J. Sztipanovits, A. Ledeczi, and T. Bapty. Model-Integrated Development of Embedded Software. *Proceedings of the IEEE*, 91(1):145–164, Jan. 2003.

[59] J. S. Kinnebrew, A. Gupta, N. Shankaran, G. Biswas, and D. C. Schmidt. Decision-Theoretic Planner with Dynamic Component Reconfiguration for Distributed Real-time Applications. In *Proceedings of the 8th International Symposium on Autonomous Decentralized Systems (ISADS 2007)*, Sedona, Arizona, Mar. 2007.

[60] J. S. Kinnebrew, W. R. Otte, N. Shankaran, G. Biswas, and D. C. Schmidt. Intelligent Resource Management and Dynamic Adaptation in a Distributed Real-time and Embedded Sensor Web System. In *Proceedings of the 12th International Symposium on Object/Component/Service-oriented Real-time Distributed Computing (ISORC '09)*, Tokyo, Japan, Mar. 2009.

[61] M. Klusch and K. Sycara. Brokering and matchmaking for coordination of agent societies: A survey. *Coordination of Internet agents: models, technologies, and*

applications, pages 197–224, 2001.

[62] T. Knabe, M. Schillo, and K. Fischer. Improvements to the fipa contract net protocol for performance increase and cascading applications. In *Proceedings of the International Workshop for Multi-Agent Interoperability at the Annual German Conference on AI (KI-2002)*, 2002.

[63] M. Kolp, P. Giorgini, and J. Mylopoulos. Multi-agent architectures as organizational structures. *Autonomous Agents and Multi-Agent Systems*, 13(1):3–25, 2006.

[64] S. Kumar, P. Cohen, and H. Levesque. The adaptive agent architecture: achieving fault-tolerance usingpersistent broker teams. In *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*, pages 159–166, 2000.

[65] E. Kutanoglu and S. Wu. On combinatorial auction and Lagrangean relaxation for distributed resource scheduling. *IIE Transactions*, 31(9):813–826, 1999.

[66] P. Laborie. Algorithms for Propagating Resource Constraints in AI Planning and Scheduling: Existing Approaches and New Results. *Artif. Intell.*, 143(2):151–188, 2003.

[67] P. Laborie and M. Ghallab. Planning with Sharable Resource Constraints. *Proc. 14th Int. Joint Conf. on AI*, pages 1643–1649, 1995.

[68] H. Lau and L. Zhang. Task Allocation via Multi-Agent Coalition Formation: Taxonomy, Algorithms and Complexity. In *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 346–350, 2003.

[69] V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. Prasad, A. Raja, et al. Evolution of the GPGP/TÆMS Domain-Independent Coordination Framework. *Autonomous Agents and Multi-Agent Systems*, 9(1):87–143, 2004.

[70] L. Lin and Z. Zheng. Combinatorial bids based multi-robot task allocation method. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '05)*, pages 1145–1150, 2005.

[71] I. Little and S. Thiebaux. Concurrent probabilistic planning in the graphplan framework. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 263–273, 2006.

[72] R. Mailler, V. Lesser, and B. Horling. Cooperative Negotiation for Soft Real-time Distributed Resource Allocation. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 576–583. ACM New York, NY, USA, 2003.

[73] S. M. Majercik and M. L. Littman. Contingent planning under uncertainty via

stochastic satisfiability. *Artificial Intelligence*, 147(1-2):119–162, 2003.

[74] F. Maturana and D. Norrie. Distributed decision-making using the contract net within a mediator architecture. *Decision Support Systems*, 20(1):53–64, 1997.

[75] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL-the planning domain definition language. *The AIPS-98 Planning Competition Comitee*, 1998.

[76] M. Mikic-Rakic, S. Malek, and N. Medvidovic. Improving Availability in Large, Distributed Component-Based Systems Via Redeployment. In *3rd International Working Conference on Component Deployment (CD 2005)*, Grenoble, France, 2005.

[77] P. Modi, H. Jung, M. Tambe, W. Shen, and S. Kulkarni. A Dynamic Distributed Constraint Satisfaction Approach to Resource Allocation. In *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*, pages 685–700, 2001.

[78] NASA. Report from the Earth Science Technology Office (ESTO) Advanced Information Systems Technology (AIST) Sensor Web Technology Meeting. Technical Report ATR-2007(2105)-1, The Aerospace Corporation, May 2007.

[79] X. Nguyen and S. Kambhampati. Reviving partial order planning. In *IJCAI*, pages 459–466, 2001.

[80] H. Nwana, L. Lee, and N. Jennings. Coordination in Software Agent Systems. *The British Telecom Technical Journal*, 14:79–88, 1996.

[81] Object Management Group. *Deployment and Configuration Adopted Submission*, OMG Document mars/03-05-08 edition, July 2003.

[82] Object Management Group. *Light Weight CORBA Component Model Revised Submission*, OMG Document realtime/03-05-05 edition, May 2003.

[83] E. Ogston and S. Vassiliadis. Matchmaking among Minimal Agents without a Facilitator. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 608–615. ACM Press, New York, NY, USA, 2001.

[84] D. Ouelhadj, C. Hanachi, B. Bouzouia, A. Moualek, and A. Farhi. A Multi-Contract Net Protocol for Dynamic Scheduling in Flexible Manufacturing Systems (FMS). *Proceedings of the IEEE International Conference on Robotics and Automation*, 2, 1999.

[85] D. Parkes and L. Ungar. Iterative Combinatorial Auctions: Theory and Practice. *Proceedings of the National Conference on Artificial Intelligence (AAAI '00)*, pages 74–81, 2000.

[86] S. Paurobally, J. Cunningham, and N. Jennings. Verifying the Contract Net Protocol: A Case Study in Interaction Protocol and Agent Communication Language Semantics. *Proceeding 2nd International Workshop on Logic and Communication in Multi-Agent Systems*, 2004.

[87] M. Pechoucek, O. Stepankova, V. Marik, and J. Barta. Abstract architecture for meta-reasoning in multi-agent systems. *Lecture Notes in Computer Science: Multi-Agent Systems and Applications III*, 2691:84–99, 2003.

[88] J. Pineau, N. Roy, and S. Thrun. A hierarchical approach to pomdp planning and execution. In *Workshop on Hierarchy and Memory in Reinforcement Learning (ICML)*, 2001.

[89] D. Porter, S. Rassenti, A. Roopnarine, and V. Smith. Combinatorial auction design. *Proceedings of the National Academy of Sciences*, 100(19):11153–11157, 2003.

[90] A. Raja, G. Alexander, and V. Mappillai. Leveraging Problem Classification in Online Meta-Cognition. In *Proceedings of AAAI 2006 Spring Symposium on Distributed Plan and Schedule Management, Stanford*, pages 97–104, 2006.

[91] A. Raja and V. Lesser. Real-time meta-level control in multi-agent systems. In *In Proceedings of Multi-Agent Systems and Applications - ACAI 2001 and EASSS 2001 Student Sessions. Also Adaptability and Embodiment Using Multi-Agent Systems: AEMAS 2001 Workshop*, 2001.

[92] A. Raja and V. Lesser. A framework for meta-level control in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 15(2):147–196, 2007.

[93] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. *Proceedings of Knowledge Representation and Reasoning (KRandR-91)*, pages 473–484, 1991.

[94] E. Sacerdoti. *A structure for plans and behavior*. PhD thesis, Stanford University, Stanford, CA, USA, 1975.

[95] T. Sandholm. An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 256–256, 1993.

[96] T. Sandholm and V. Lesser. Coalition formation among bounded rational agents. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 662–671, 1995.

[97] T. Sandholm and V. Lesser. Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Protocol. *Proceedings of the First International Conference on Multiagent Systems (ICMAS-95)*, 1995.

[98] M. Schillo, C. Kray, and K. Fischer. The Eager Bidder Problem: A Fundamental Problem of DAI and Selected Solutions. *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '02)*, pages 599–606, 2002.

[99] T. C. Service and J. A. Adams. Coalition formation for task allocation: Theory and algorithms. *Autonomous Agents and Multi-Agent Systems*, 2010.

[100] N. Shankaran. *Adaptive Resource Management Agorithms, Architectures, and Frameworks for Distributed Real-time Embedded Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN, USA, Dec 2007.

[101] N. Shankaran, D. C. Schmidt, X. D. Koutsoukos, Y. Chen, and C. Lu. Design and Performance Evaluation of Resource-Management Framework for End-to-End Adaptation of Distributed Real-time Embedded Systems. *Journal on Embedded Systems: Special issue on Operating System Support for Embedded Real-Time Applications*, 2008.

[102] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, 1998.

[103] M. Sims, C. Goldman, and V. Lesser. Self-organization through Bottom-up Coalition Formation. *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '03)*, pages 867–874, 2003.

[104] R. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, 29(12):1104–1113, 1980.

[105] B. Srivastava and S. Kambhampati. Scaling up Planning by Teasing Out Resource Scheduling. *Proceedings of the European Conference on Planning*, 1999.

[106] D. Suri, A. Howell, D. Schmidt, G. Biswas, J. Kinnebrew, W. Otte, and N. Shankaran. A Multi-agent Architecture for Smart Sensing in the NASA Sensor Web. In *Proceedings of the 2007 IEEE Aerospace Conference*, Big Sky, Montana, March 2007.

[107] D. Suri, A. Howell, N. Shankaran, J. Kinnebrew, W. Otte, D. Schmidt, and G. Biswas. Onboard Processing using the Adaptive Network Architecture. In *Proceedings of the Earth-Sun Science Technology Conference*, College Park, MD, June 2006.

[108] F. Teichteil-Koenigsbuch and G. I. U. Kuter. Rff: A robust, ff-based mdp planning algorithm for generating policies with low probability of failure. In *Proceedings of the Sixth International Planning Competition (IPC) at ICAPS '08*, 2008.

[109] E. Tsang, T. Gosling, B. Virginas, C. Voudouris, G. Owusu, and W. Liu. Retractable Contract Network for Empowerment in Workforce Scheduling. *Multiagent and Grid Systems*, 4(1):25–44, 2008.

[110] R. van der Krogt and M. de Weerdt. Coordination Through Plan Repair. In *MICAI 2005: Advances in Artificial Intelligence: 4th Mexican International Conference on Artificial Intelligence, Monterrey, Mexico, November 14-18, 2005: Proceedings*. Springer, 2005.

[111] L. Vig and J. Adams. Market-based multi-robot coalition formation. *Distributed Autonomous Robotic Systems 7*, pages 227–236, 2006.

[112] L. Vig and J. Adams. Coalition formation: From software agents to robots. *Journal of Intelligent and Robotic Systems*, 50(1):85–118, 2007.

[113] J. Vokrınek, J. Bıba, J. Hodık, J. Vybıhal, and M. Pechoucek. Competitive Contract Net Protocol. *Proceedings of the 33rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 656–668, 2007.

[114] T. Wagner, A. Garvey, and V. Lesser. Complex Goal Criteria and Its Application in Design-to-Criteria Scheduling. In *Proceedings of the National Conference on Artificial Intelligence*, pages 294–301, 1997.

[115] T. Wagner and V. Lesser. Design-to-Criteria Scheduling: Real-Time Agent Control. *Lecture Notes in Computer Science*, pages 128–143, 2001.

[116] T. Wagner, A. Raja, and V. Lesser. Modeling Uncertainty and its Implications to Sophisticated Control in Tæms Agents. *Autonomous Agents and Multi-Agent Systems*, 13(3):235–292, 2006.

[117] G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, 2000.

[118] D. Weyns, N. Boucke, T. Holvoet, and K. Schelfthout. DynCNET: A Protocol for Flexible Task Assignment Applied in an AGV Transportation System. *Proceedings of the Fourth European Workshop on Multi-Agent Systems*, pages 359–370, 2006.

[119] H. C. Wong and K. Sycara. A taxonomy of middle-agents for the internet. In *Proceedings of the Fourth International Conference on MultiAgent Systems*, pages 465 – 466, July 2000.

[120] M. Wooldridge. Intelligent Agents: The Key Concepts. *Multi-Agent-Systems and Applications*, pages 3–43, 2001.

[121] L. Xu and H. Weigand. The Evolution of the Contract Net Protocol. *Lecture Notes in Computer Science*, pages 257–266, 2001.

[122] S. W. Yoon, A. Fern, and R. Givan. Ff-replan: A baseline for probabilistic planning. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS '07)*, pages 352–359, Providence, Rhode Island, USA, Sep 2007.

[123] S. W. Yoon, A. Fern, R. Givan, and S. Kambhampati. Probabilistic planning via determinization in hindsight. In *AAAI*, pages 1010–1016, 2008.

[124] H. Younes and R. Simmons. Vhpop: Versatile heuristic partial order planner. *Journal of Artificial Intelligence Research*, 20:2003, 2003.

[125] Z. Zhang and C. Zhang. An improvement to matchmaking algorithms for middle agents. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '02)*, pages 1340–1347, Bologna, Italy, 2002.