A Multi-Agent System for Intelligent Environment Control *

Victor Lesser, Michael Atighetchi, Brett Benyo, Bryan Horling,
Anita Raja, Régis Vincent,
Thomas Wagner, Ping Xuan and Shelley XQ. Zhang

Computer Science Department
University of Massachusetts at Amherst
Amherst, MA 01003

UMass Computer Science Technical Report 1998-40

March 29, 1999

Abstract

Intelligent environments are an interesting development and research application problem for multi-agent systems. The functional and spatial distribution of tasks naturally lends itself to a multi-agent model and the existence of shared resources creates interactions over which the agents must coordinate. In the UMASS Intelligent Home project we have designed and implemented a set of distributed autonomous home control agents and deployed them in a simulated home environment. Our focus is primarily on resource coordination, though this project has multiple goals and areas of exploration ranging from the intellectual evaluation of the application as a general MAS testbed to the practical evaluation of our agent building and simulation tools.

1 Introduction

The intelligent home project (IHome) at the UMASS multi-agent systems lab is an exploration in the application of multi-agent systems technology to the problem of managing an intelligent environment. We have implemented a sophisticated simulated home environment, populated it with distributed intelligent home-control agents (including simulated robots) that control appliances and negotiate over shared resources, and begun experimentation with different coordination protocols and agent adaptability / responsiveness to changing environmental conditions.

Our work is akin to the Adaptive House [21] and [10, 14] in that the objective is for the environment to automate some of the tasks currently performed by humans – possibly with improvements in efficiency or quality of service. However, our focus is on resource coordination and temporally sequencing agent activities over shared resources. A broad spectrum of research falls into the general category of intelligent environments. For example, one class of work deals with collecting and integrating information about the activities that occur within the environment [3] while another class focuses on on identifying and tracking humans as they move about the environment [7, 23].

The UMASS simulated IHome environment is controlled by intelligent agents that are associated with particular appliances; a screen snapshot of a sample run is shown in Figure 1. The IHome population set

*Effort sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory Air Force Materiel Command, USAF, under agreement number F30602-97-1-0249 and by the National Science Foundation under Grant number IIS-9812755 and number IRI-9523419. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), Air Force Research Laboratory, National Science Foundation, or the U.S. Government.

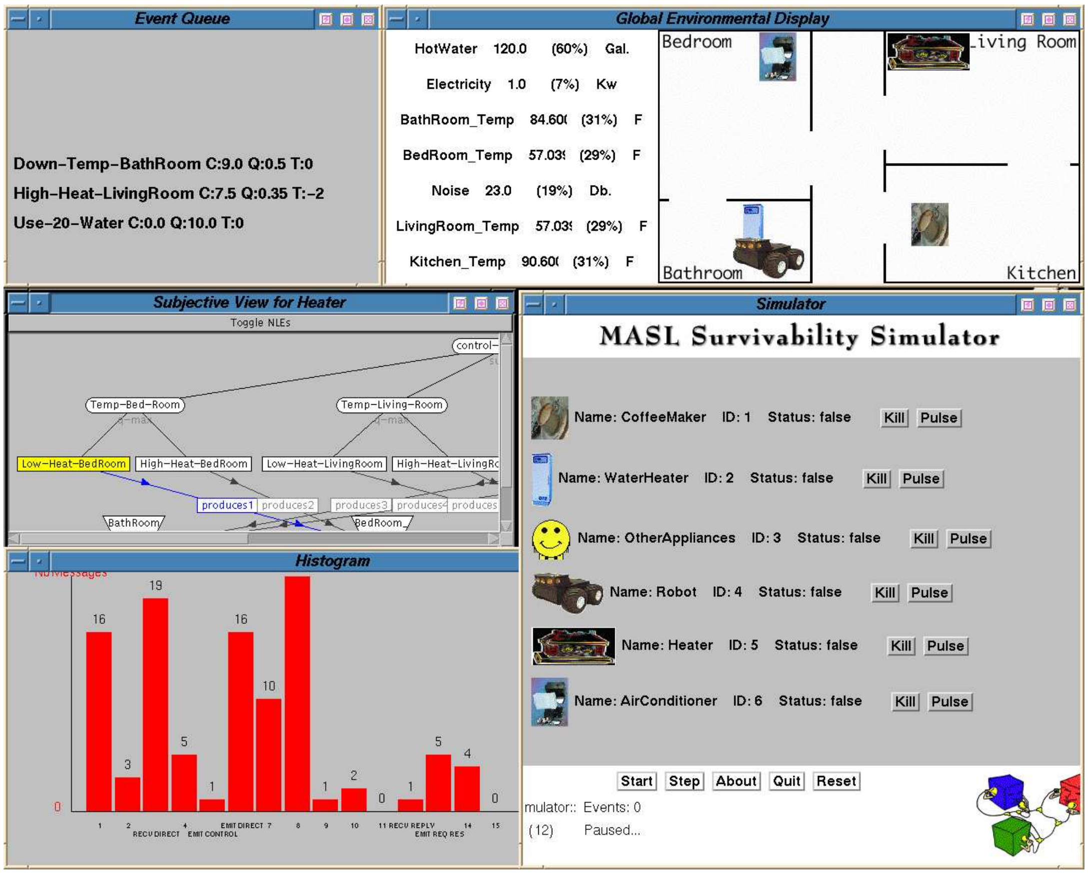

Figure 1: IHome Agents in Action

includes agents like an intelligent WaterHeater, CoffeeMaker, Heater, A/C, DishWasher, etc., and a robot for fetching items and moving physical goods from one location to another. The home agents reason about their assigned tasks and select candidate actions based on the occupant's preferences and the availability of resources. For example, if hot water is scarce, the DishWasher agent may elect to run a cold cycle, trading-off solution quality for resource consumption – the agent may also elect to wait until hot water becomes available. Agents coordinate over shared resources like noise, electricity, temperature, and hot water. Resources, resource interactions, task interactions, and the performance characteristics of primitive actions are all represented and quantified in the TÆMS [9] task modeling framework. This enables agents to reason about the trade-offs of different possible courses of action and to adapt behaviorally to the changing environment.

The research has several goals, among them are:

1. Examine the intelligent home domain as a general application testbed for research in multi-agent systems (Section 2).
2. Understand the distributed control issues of this particular multi-agent application and to relate these issues to research in general approaches to agent control, like the GPGP and $GPGP^2$ [19, 17] coordination systems and the Design-to-Criteria scheduling system [25]. Toward that end, the coordination protocols and agent control tools used in many of the home agents are the products of a bottom-up design process rather than a top-down process that would have occurred if the requirement had been to apply the generic technologies directly.
3. Apply the TÆMS [9] domain-independent task modeling framework to a new domain and evaluate its use in the rapid development of a new multi-agent application.
4. Test and refine our multi-agent simulation environment [24] that controls method execution and communication characteristics for a set of distributed agents. The environment employs a complex time mapping scheme and a process controller to resolve timing issues between the distributed agents and to ensure reproducibility.
5. Test and refine our java-based generic agent construction framework [13] that facilitates agent construction through an event-driven component architecture. The framework also enables agents to be decoupled from the simulator and executed in their application domain with a simple change in internal components, i.e., with a change to the makefile.

Space precludes discussing all of these points. We will focus on the challenges offered by the application domain, discuss the TÆMS modeling framework from a high-level view, describe the application and some of the agents, touch on the bottom-up resource-centric coordination protocols developed in the project, and present experimental results. For information about the agent development tools or more information about the project, readers should consult the project pages (mas.cs.umass.edu).

2 Intelligent Environments for MAS Exploration

Intelligent environments have long been an interesting application arena for many areas of computer science, in general, and AI in particular. From a multi-agent systems perspective, the domain poses several interesting challenges because of the existence of many forms of task or goal interactions, a natural physical and functional distribution of control, a requirement for actions to be taken in certain time intervals (soft real-time?), and learning/integrating potentially conflicting preference specifications. To understand the importance of these features, and from where they originate, let us discuss the domain characteristics in more detail.

Resources, like water, electricity, or money with which to purchase these, in intelligent environments are no more or less scarce than they are in unmanaged or unautomated environments. In both cases resources are often limited and different agents, be they human or software, must coordinate over the usage of said resources. Limited resources result in interactions between tasks or goals. For example, if the dishwasher and the washing machine are run during the same time interval in which someone is taking a shower, the person will probably have a cold shower, plus, the dishes and clothes may not be cleaned as well because the hot-water resource is inadequate for the demands during this period. In TÆMS terminology, this type of interaction is a *hinders* task inter-relationship, i.e., a soft interaction that has negative effects on the affected tasks. Other types of resource driven interactions exist. An example of a hard interaction effect is the attempt to use a single phone line for a `ppp` connection and for receiving a fax simultaneously.

The seemingly rich landscape of resource-driven task interactions leads to two areas of exploration, both of interest in MAS [15]. One area is in how to (equitably?) allocate a scare resource between multiple different agents. Another area is how to coordinate activities around the resource allocation, i.e., temporally sequencing actions. Often researchers focus primarily on one facet of this problem, either studying allocation methods or scheduling issues. For example, in [14] office temperature control agents participate in double-blind auctions to regulate the temperature in an office environment. Cool air is regarded as a limited resource and a monetary view is used to determine the allocation of said resource. In [2] air temperature is regarded as a shared resource but resource coordination focuses on determining the temperature of rooms when they contain more than one person. Our view of resources lies on the finer-grained end of the spectrum. We model and represent the individual tasks that operate on, or use, resources and quantify the use and the effects if there are insufficient resources to meet the demands. Agents then negotiate over the resources from a more detailed temporal perspective to determine which agent should execute and when. This is similar to the view taken in [8], though this GPGP-based work uses a more generic model-based approach to coordination in which agents exchange local task structures, detect interactions, and then coordinate over the interactions. In our work, interactions are detected through the announcement of resource requests rather than detection through exchange of private information about candidate tasks and actions.

The application is also naturally distributed, though the structure of the computation is debatable. Consider the producer/transporter/consumer application studied by many multi-agent systems researchers [11, 1]. In this problem domain, the producers, transporters, and consumers serve different functional purposes, and, perhaps more importantly, they are also tasks generally performed in the "real-world" by different corporate entities. For example, IBM might produce the equipment that is moved by Yellow Freight to General Motors. The distributed airport service problem [6] exhibits similar characteristics – functional decomposition as well as "corporate" or "entity" decomposition. Unlike these examples, in the distributed situation assessment [4] problem decomposition is generally data-driven, i.e., different agents are assigned different sensor regions to monitor; agents are homogeneous and coordination takes a different form (more centered on knowledge sharing) than coordination (more centered on temporal sequencing of actions) in the other two domains. Does an intelligent home or intelligent environment exhibit the same characteristics as either of these two problem domain classes? The answer to that is dependent on how the computation is structured.

One approach is is to group responsibility and function geographically, having agents responsible for particular regions in the environment, e.g., a bedroom manager, a living room manager, etc., all of which control or interact with agents in their geographic region (e.g., a clock radio agent, a TV agent). Another approach is to group responsibility and function according to the functional tasks being performed, e.g., a food preparation manager that either controls all food preparation devices directly or interacts with the intelligent (sub) devices and provides a common interface to the other manager entities. Still another model is to simply partition the computation functionally – resulting in a large organization of very specialized agents, e.g., the dishwasher agent, the coffee maker agent, etc. Obviously, there are many different possible structurings. Implicit in this discussion is the notion that it is unclear which structuring is particularly effective. What is clear is that this problem domain provides a rich landscape in which to study different computational organizations and different classes, if you will, of coordination and control.

Another interesting aspect of this problem domain is the implicit requirement that decisions be made in a timely fashion, i.e., in interactive-time if not real-time. Finally, because the general objective of an intelligent environment is to automate tasks or enhance the environment for its occupants, the need to learn, build, and manipulate or combine conflicting user profiles is ubiquitous in the application. This too is being addressed by the research community [5, 12, 21].

3 TÆMS Task Structures in the Intelligent Home

The simulator and the agents in our intelligent home model problem solving activities using the TÆMS domain independent task modeling framework [9, 17]. TÆMS models planned actions, candidate activities, and alternative solution paths from a quantified perspective; all primitive actions are described statistically

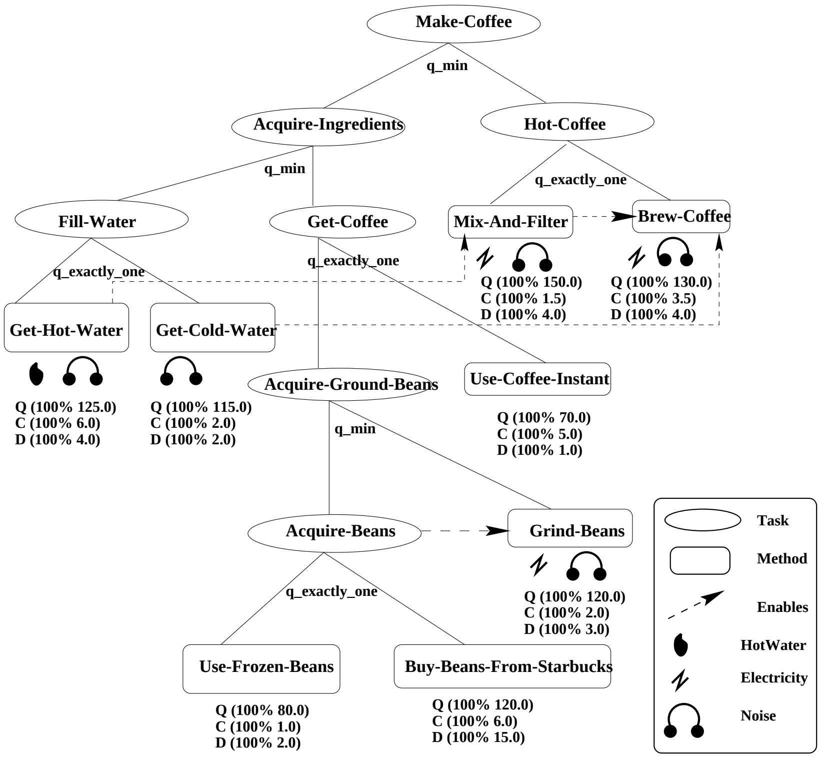

Figure 2: TÆMS Task Structure for Making Coffee

via discrete probability distributions in terms of quality, cost, and duration. A fourth dimension, uncertainty, is implicit in the probability distributions. Thus, TÆMS-based reasoners (e.g., [25]) can evaluate the quality, cost, and duration (and uncertainties in each of these) characteristics of each possible course of action and select the course of action that best[1] meets the current constraints and environmental conditions. For example, in a time constrained situation, an agent may sacrifice solution quality and possibly consume more resources to produce a result within the specified deadline.

Figure 2 shows a TÆMS task structure used by the CoffeeMaker agent to represent its process for making coffee. The simulator has a different *objective* view of the agent's *subjective* task structure and the views may be radically different – this enables experimentation with situations in which the agent's model is inaccurate. While TÆMS is a modeling framework, agents often use it internally to reason about the structure of their computation. In this case, it is akin to a process plan or other meta representations. It is a structure that describes how the primitive actions relate to accomplishing the overall objective and often the primitive actions in TÆMS correspond directly to underlying code. TÆMS task structures are models in the sense that they may be abstracted from some of the execution details, not in the sense that they are completely isolated from execution and can only be used in a simulated environment. In the IHome project, programmers describe the agents problem solving options in TÆMS, usually via a TÆMS graph-grammar-generator, and then build the tools, or use existing ones, for reasoning with the task structures. In this usage, the programmers take the place of a generative planner or problem solver that would normally produce the task structures (as in [18]) from its own internal representations. This enables programmers to rapidly create agents for applications where an off-the-shelf planner/problem solver is not available.

The task structure shown in Figure 2 describes alternative ways to obtain water, obtain coffee, and brew the coffee. Consider the *Acquire-Ground-Beans* task; it has two subtasks, one of which is another decomposable task, and another (*Grind-Beans*) which is a primitive action and is described in terms of quality, cost, and duration. The small icons under the action denote resource usage – resources and the resource interactions are absent from this figure to improve readability. The arc leading from *Acquire-Beans* to *Grind-Beans* is an *enables* non-local-effect (nle) and it denotes a hard constraint that *Acquire-Beans*

[1] Due to the combinatorics of the TÆMS scheduling problem, "best" does not necessarily denote optimal.

must have quality in order for *Grind-Beans* to execute, i.e., the agent must have beans before it can grind them. The *q_min* quality-accumulation-function (qaf) associated with *Acquire-Ground-Beans* denotes that its quality is computed by *min(Acquire-Ground-Beans,Grind-Beans)*, modeling the notion that poor beans or poor grinding produces poor ground beans. *Acquire-Beans* has two primitive action subtasks. Note that using frozen beans produces a lower quality result than buying beans from Starbucks, but that it also costs less and is considerably faster. If the CoffeeMaker is in a hurry, or has limited financial resources, it may thus choose to use frozen beans. However, if the agent is extremely time constrained, it will probably perform *Get-Coffee* by using instant coffee rather than obtaining ground beans of either form.

This task structure illustrates the notion of quantified choice in TÆMS and its facilitation of trade-off behaviors at run-time. However, it is not a good illustration of the use of uncertainty in TÆMS as the methods all have simple distributions and no represented probability of failure. If execution failure should occur the agent will reschedule accordingly. However, the lack of any representation of failure may keep the agent from working to reduce the probability of failure by choosing more conservative options. This can be important in cases when tight deadlines exist.

The quantifications of items in TÆMS is not regarded as a perfect science. Task structure programmers or problem solver generators *estimate* the performance characteristics of primitive actions. These estimates can be refined over time through learning and reasoners typically replan and reschedule when unexpected events occur. Quantification in TÆMS is not limited to the characterization of primitive actions. Interactions between tasks, actions, and resources are also described statistically. For example, agents describe their resource consumption behaviors in terms of a *consumes* non-local-effect and the effects of the resource on the task are described via a *limits* non-local-effect. The limits nle describes the negative effects of lacking sufficient resources to perform a task in terms of power-effects on quality, cost, and duration. These effects can model a range of behaviors, from an increase in duration in the case of a network resource to a complete reduction of expected quality to zero in the case of a hard resource like a locked file. For a non-consumable resource, e.g., network bandwidth, where the resource is diminished during the usage and then returned to its initial state, the definitions for consumes and limits are:

A resource-centered non local effect is a function of the form: *nle(M, R, t, q, c, d,* $R_{quantity}$*, p1, p2,...): [method* $\times$ *resource* $\times$ *current time* $\times$ *method quality* $\times$ *method cost* $\times$ *method duration* $\times$ *resource quantity* $\times$ *parameter1* $\times$ *other parameter2 ..] = [method quality* $\times$ *method cost* $\times$ *method duration* $\times$ *resource quantity]*

$$consumes(M, R, t, q, c, d, R_{quantity}, \alpha_{quantity}, M_{t\_exec}) =$$
$$\begin{cases} [q, c, d] & and \\ R_{quantity} = R_{quantity} - \alpha_{quantity} & t > M_{t\_exec} \\ R_{quantity} & otherwise \end{cases}$$

$$limits(M, R, t, q, c, d, R_{quantity}, \alpha_{quantity}, M_{t\_exec}, \phi_q, \phi_c, \phi_d) =$$
$$\begin{cases} [q - q * \phi_q, c + c * \phi_c, d + d * \phi_d] & t > M_{t\_exec} \ \& \ R_{quan} < \alpha_{quan} \\ [q, c, d] & otherwise \\ R_{quantity} & \end{cases}$$

4 The Intelligent Home

The intelligent home is a model of a small home constructed and executed using the generic multi-agent simulation environment [24]. The home consists of four rooms: a bedroom, a living room, a bathroom, and a kitchen, all joined by a common hallway. Though the home is more of an apartment, size is actually not necessary in this application to obtain interesting results; the interesting issues arise when agent controllers interact and a smaller space requires fewer agents to generate interesting interactions.

Expanding the size of environment may create an issue of scalability with respect to resource coordination protocols unless the expansion is achieved through composition of (primarily) independent sub-environments. If the intelligent environment were a large manufacturing factory, for instance, where hundreds of agents shared common resources like electricity and water, the simple peer-to-peer agent organization used in this project will probably lead to combinatorics and high coordination overhead. In situations such as these, the agents should be organized into work-groups or according to other partitioning schemes to reduce the scope of interaction.

In our model agents are associated with major appliances. As mentioned in Section 2, many different structurings of the agents are possible. We decided on the model of associating agents with appliances because we believe it is likely that in the future intelligent appliances will be packaged with their own intelligent control software. Different appliances will probably have different types of agent controllers and the agents will probably be heterogeneous, interfacing through a common protocol. This leads to either a peer-to-peer organization or a group-style organization where agents are perhaps clustered according to function (e.g., washer and dryer), spatial location, or resource usage. In the future, we plan to experiment with different organizational structurings.

Thus, agents are associated with major appliances and they interact directly to coordinate over shared resources. Currently, we model and coordinate over electricity, hot water, noise or sound levels, heating oil or natural gas, and room temperature in each of the modeled rooms. Agents coordinate using a resource coordination protocol discussed in Section 5.

In terms of modeling issues, we made some simplifying assumptions. Based on work ongoing in the community, we assumed the existence of supporting technology for: identification and tracking of individuals moving about the environment, obtaining client preference profiles that include things like deadlines on particular activities (e.g., dishes should be done by the time the client gets home from work), and assimilating different occupant preferences for parameters like room temperature. Since we currently employ only one fetching robot, we also did not address spatial constraint issues like two robots attempting to use the same door simultaneously (it is not clear that we will model robots at that fine a level of granularity in the future either).

The agents that populate the intelligent home are heterogeneous, each having its own internal problem solver that reasons using TÆMS task structures. Some of the agents make use of generic agent control tools like the Design-to-Criteria scheduler [25], but there is no requirement to do so as we are interested in examining the bottom-up production of agents for this application. All the agents were constructed using the generic Java Agent Framework [13], however the framework's role is to "glue" together disparate components and it does not impose any restrictions on the types of agents that can be constructed or how the agents approach particular problems. Interagent communication is done via KQML [16] routed through the simulation environment as discussed previously. The population of the intelligent home includes a mobile robot and appliance agents like the Dryer, TV, DishWasher, WaterHeater, VacuumCleaner, Heater, A/C, CoffeeMaker, and the OtherAppliances agent. The OtherAppliances agent is a place holder for other appliances not currently modeled by agents. It makes resource resource requests and otherwise stresses and exercises the system in much the same way as an additional x agents would. Space precludes discussing each agent in detail, though the agents are generally characterized according to the tasks they perform, the alternative ways to perform them, the resources they consume, and the agents with which they interact. For example:

A/C Agent Summary: Responsible for climate regulation. Has cooling ability, limited heating ability by routing air flow through home, and the ability to control humidity by routing air through the compressor. The agent's control flow is shown in Figure 3.

- **Task Performance Options** Different fan and compressor levels resulting in different cooling rates with different noise characteristics.
- **Shared Resources Noise:** interacts with the DishWasher, Dryer, VacuumCleaner, CoffeeMaker, and TV agents. **Electricity:** interacts with the DishWasher, Dryer, VacuumCleaner, TV, and CoffeeMaker agents. **Temperature:** interacts with the Heater agent.
- **Task Interactions** Task sharing with the Heater agent to control room temperature.

One of the agents in the home is actually a generic agent. It uses the Design-to-Criteria scheduler so that its behaviors are completely defined and described in TÆMS and a set of goal criteria for the scheduler. The generic agent can, in essence, become any of the other agents simply by changing its descriptive task structures and the scheduling criteria. The generic agent will not always perform identically to the agent it emulates because the agent may make different trade-off decisions than those made by the scheduler. In the future, we will compare the performance of the generic agent to the specialized agents to determine the differences and the relative strengths of each.

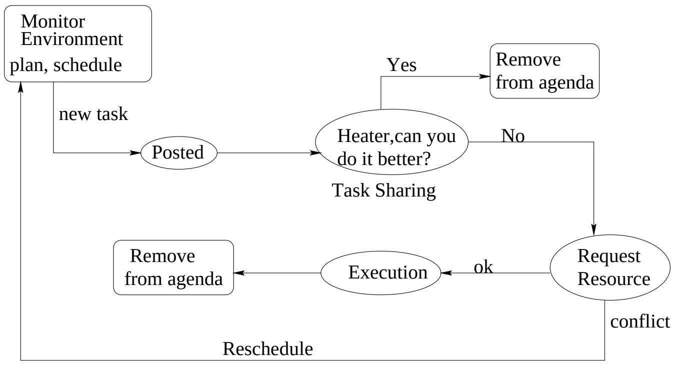

Figure 3: AirConditioner Agent's Control Flow

5 The *SHARP* Resource Coordination Protocol

In the IHome environment resources are either *centrally managed* or *decentralized.* Centrally managed resources are controlled by a particular agent which is often the agent responsible for producing the resource. For example, hot water is managed by the WaterHeater agent – the agent interacts with the other agents, determining who gets what quantity of hot water, and when, and it plans to produce hot water based on these commitments. In contrast, unmanaged or decentralized resources, e.g., noise or electricity, are not coordinated by a central entity and the consumers of the resource must coordinate with each other in a decentralized or unmoderated fashion. It is possible to "agentify" decentralized resources by assigning or electing an agent coordinator. However, we believe that certain resources are more naturally viewed from an agent perspective and other resources are more naturally framed as unregulated commodities. This heterogeneous model results in two slightly different resource coordination protocols.

Task based interaction also occurs in the IHome environment – coordination with the mobile robot is one instance of this and is handled using a contracting approach [22]. Another example of task interaction exists between the Heater agent and the A/C agent. This is a form of task overlap rather than task allocation and is handled with yet another cooperative task centered dialogue between the involved agents.

To handle coordination over shared resources we developed a simple protocol called *SHARP* (Simple Home Agent Resource Protocol) that assumes a cooperative agent model and uses a priority-based request and allocate scheme. In *SHARP*, each resource request contains a priority which denotes the importance of the task for which the resource is being requested. Priority ranges are associated with the different tasks and these reflect the client's preferences. For example, some people would rather have a hot shower than extra clean dishes if resources are limited. Conflicts occur when resources for a particular time period are insufficient to meet the demand. When this happens, agents holding resource confirmations for the time slot may be requested to release their reservation or may simply be told that their reservation is canceled (in the centralized version). The affected agents can then decide to raise their priority and negate the release request or they can simply release the resource.

Though *SHARP* operates from a priority perspective, it is important to note that the use of a priority scheme does not bind the protocol to a priority only framework. Priorities in the current incarnation are viewed as being derived from client preferences, however, priorities could also be determined through market mechanisms based on some medium of exchange. This would promote fairness and limit the potential for abuse by agents holding high priority tasks. The main characteristics of *SHARP* include:

- Supports for agent flexibility – agents can dynamically adjust the priorities of their resource requests to adapt to changes in the environment and the dynamic flow of requests coming from other agents. For example, if an agent faces resource starvation, it can raise the priorities of its resource requests and thus improve its chances of obtaining resource reservations. This enables agents to to use situation specific coordination strategies.
- *SHARP* is asynchronous and ongoing, i.e., there is not a single resource coordination phase followed by execution. *SHARP* thus supports dynamic conflict detection and resolution. Conflicts may occur when agents vie for resource reservations, but, resource conflicts may also occur *after* the reservation is made

(a new, higher priority task comes along or the agent is unable to obtain other, related resources). If *post-hoc* conflict occurs, the resource holder may employ the *decommit* aspect of the protocol to release its hold on the resource.

- The protocol is non-blocking. Agents do not need to wait for responses from other agents, rather, responses can arrive in an asynchronous fashion.

In the centralized protocol all reservations and priorities are kept at a moderating agent. In order for the requesting agent to chance its priority or cancel the reservation, it must contact the moderating agent. Let MA denote the moderator agent and RA to denote the agent issuing the resource request. In the centralized protocol communication only occurs between MAs and RAs. There is no communication between the RAs, though in the decentralized protocol the RAs communicate directly. There are six message types in the centralized protocol.

- *Need (from-agent, to-agent, resource, priority, amount, bound, duration)* where *from-agent* is an RA requesting allocation.
- *Accept (from-agent, to-agent, resource, priority, amount, bound, duration)* where *from-agent* is the resource MA granting a request.
- *NotAccept*[2] *(from-agent, to-agent, resource, priority, amount, bound, duration)* where *from-agent* is the MA denying a request.
- *Release (from-agent, to-agent, resource, priority, amount, bound, duration)* where *from-agent* is an RA announcing that it is willing to release the specified allocation.
- *Cancel (from-agent, to-agent, resource, priority, amount, bound, duration)* where *from-agent* is the MA.
- *Priority (from-agent, to-agent, resource, priority, amount, bound, duration)* where *from-agent* is an RA changing its priority.

Resource reservations are not static and not guaranteed – the MA may cancel the reservation even while a given task is executing. This would mean that the task is aborted or suspended, depending on the nature of the task (a washing machine may simply stop mid cycle). The agent executing the affected task would then have to make new resource request(s) in order to complete its task (or start over, as the case may be). Figure 4 shows how the state of a resource request evolves in this protocol. Possible extensions to this protocol are numerous. In the future when more complex resource issues are explored, the RAs could estimate their future resource needs, perhaps via approximate demand curves, and send these to the MA so that it can plan production activities from a rough perspective. Additionally, the MA could periodically broadcast or murmur, sending information about its current state and future estimates so that the RAs can revise their expectations.

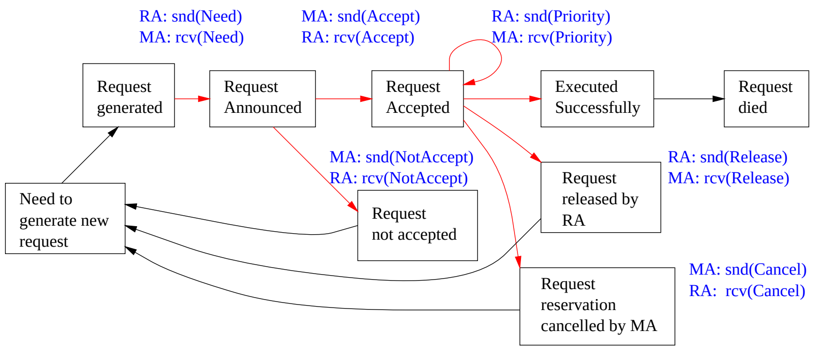

Figure 4: Centralized Protocol

Based on the small set of messages in the protocol, and the small number of states and transitions in Figure 4, it might appear that the protocol is restrictive and does not require much intelligence on the part of the agents. However, the protocol is actually quite flexible and expressive. To use the protocol, agents

[2] In more advanced versions of this protocol the *NotAccept* response could provide meta-level information about the estimated future availability of resources or suggest times and allocations that might be met with a positive response, as in [6, 20].

must decide: 1) whether to request a resource or not, 2) the amount of resource to request, 3) the time when resource is needed, 4) when to announce the request, 5) what priority to associate with the request, 6) whether to change priorities and when, 7) if changing, to determine the new priority, 8) whether/when to release/cancel a request, 9) what to do if reservation cannot be made, 10) what to do if reservation is cancelled. In Section 6 we will describe some of the different coordination strategies employed by the agents.

The decentralized protocol is closely related to the centralized protocol but it operates on a peer-to-peer basis. Space limitations preclude a detailed discussion, but, the general idea is that agents broadcast their requests and resource confirmation holders respond when conflicts occur. If no response is obtained, or "ok" messages are received from all involved parties, the requester assumes a resource confirmation for the time period. Conflicts in priorities are dealt with in the same adjust-and-reply fashion as in the centralized protocol.

Agent	# of Tasks		Final Quality		Resources Mes.		Conflict				Tasks Dropped	
	Alone	IHome	Alone	IHome	Alone	IHome	Alone	IHome			Alone	IHome
								E	R	N		
Dishwasher	4	**2**	135	**76**	10	10	0	21	1	5	0	**2**
Robot	5	5	10	10	0	0	0	0	0	0	0	0
WaterHeater		83		10				26	3	0	0	0
OtherAppliances	37	33	10	10	42	36	0	8	15	2	0	4
CoffeeMaker	4	**0**	80	**0**	3	3	0	1	11	24	0	**4**
	4	4	125	125	3	3	0				0	0
	4	4	125	125	3	3	0				0	0
Heater	8(41)	7 (77)	40	40	8	7	4	8	3	0	0	1
AirConditioner	8 (41)	12 (77)	35	20	16	24	4	12	4	0	0	0

Figure 5: Experiment 1: *Alone* indicates the performance when the agent executed alone in the environment with sufficient resources. The *IHome* column indicates performance when the agents are executed in a group and resources are shared. *E/R/N* indicates conflicts emitted, received, or nullified.

6 Sample Runs

Evaluation in a multi-agent system is always an interesting question. Since agents are distributed and autonomous the objective is to approximate some global utility measure via local-only views. Even if a centralized view exists, the optimal solution often cannot be directly computed due to exponential combinatorics. We are currently developing a definition of utility that relates quality, task achievement, meeting deadlines, and satisfying other constraints. This metric will facilitate rapid interpretation of experimental results. For discussion purposes, we will examine performance on an agent by agent basis, and compose an aggregate observation, using a working definition of optimal agent performance: the optimal performance of any agent is the performance achieved when it is run alone in the environment with ample resources with which to perform its tasks. Performance in this case denotes the quality the agent achieves and the constraints it meets, e.g., preference constraints or deadline constraints.

In the three experiments presented in this section, the IHome is populated by with seven agents, including the DishWasher, Robot, WaterHeater, CoffeeMaker, Heater, AirConditioner, and the OtherAppliances agent (that simulates the presence of multiple other agents in the environment). The communications patterns in each experiment are monitored, as is resource consumption and the behaviors of the agents. Communications statistics, such as the number of messages produced, provide a measure of the efficacy of coordination. The environment is held constant in each of the runs (in terms of communication bandwidth, execution performance of actions, etc.) while the availability of resources is varied.

In all three experiments, the preferred temperature setting is 76 degrees in all rooms and temperature change in the house is effected by the temperature-related agents, but also according to a curve that describes the heat exchange between the inside of the house and the outside environment and between the rooms of the house. The temperature related agents (AirConditioner/Heater) are reactive in nature, they respond to situations in which the temperature is not at its preferred point. In these experiments, the initial temperature is set at some point other than the preferred temperature and it is the task of the temperature control agents to bring it back into line. Like the temperature control agents, the WaterHeater agent works to keep the hot water level between a defined minimum and maximum capacities, and the tank is assigned an initial quantity of hot water.

The objective in the experiments is for the agents to carry out their assigned tasks, e.g., make coffee or wash the dishes in the alloted time. For reactive agents, like the A/C agent, the objective is to satisfy the expressed preference constraint, e.g., keep the temperature at 76 degrees, keep the water tank above the defined minimum, and so forth.

In the first experiment, the resources are configured as follows: 15Kw of electricity is available, 140 gallons of hot water initially reside in the water tank and the tank maximum is 200 gallons, the maximum allowable noise level at any time is 120 Db, and the initial temperatures in the different rooms are as follows: bedroom 50F, bathroom 90F, kitchen 90F, living room 50F.

The results for the first run are shown in Figure 5. In this experiment the agents that require multiple resources to carry out their tasks, and who have longer sequential chains of actions that must take place, like the CoffeeMaker and the DishWasher, perform poorly when compared to their independent performance. Because these agents require multiple resources at the same moment their performance requirements are higher and in this situation, where resources are constrained, they are generally unable to obtain the necessary resources. In both of the experiments the deadline for task completion is fairly tight in order to make the coordination problem non-trivial.

In contrast to the long-planning agents the more reactive agents (WaterHeater, Heater, AirConditioner) fair better. The only difference between their individual runs and the group run is that in the latter case they take longer to achieve the desired results. The Heater and the AirConditioner take until time 77 to reach their temperature goal of 76F, in contrast to the 41 clicks required in the individual case.

The behavior of the CoffeeMaker and DishWasher agents indicate a problem with our simple *SHARP* protocol. Though we have priority measures, higher priorities are not assigned to agents that are currently executing their plans. Thus tasks like making coffee are always superseded and interrupted by other higher priority tasks. Additionally, the priorities of tasks are not elevated as they are interrupted, thus they do not become less interruptible over time (a feature often found in priority based scheduling algorithms) or as they get closer to their deadlines. The problem also stems from a flawed implementation of personal preference – agent priorities in these experiments do not always reflect the client's personal preferences and thus the notion of a global utility function (even a local view of one) is somewhat muddied. This issue is currently being addressed.

The second experiment is identical to the first, with the exception that the coffee making tasks are assigned the highest overall priority in the system, enabling the CoffeeMaker to obtain the desired resources to carry out its tasks. However, its resource consumption pushed back temperature regulation tasks resulting in the A/C and Heater agents taking until time 90 (rather than 77) to reach their target temperatures.

In the third experiment, Figure 6, the resources are configured similarly except that the maximum capacity of the water tank is reduced to 60 gallons and it is empty at the start of the experiment. This decrease forces all agents using hot water to negotiate over the resource. In this case, the DishWasher is able to perform only one task out of its four assigned tasks. The 84 messages sent by the agent is testimony to its attempts to obtain the resources so that it could perform its other tasks (it was refused and canceled by the WaterHeater).

Interestingly given the tighter hot water constraints, the WaterHeater agent also performed fewer tasks than it did in the previous experiments. This is because the DishWasher was unable to execute, and the maximum capacity of the tank was reduced, thus the demand for water from a volume perspective also decreased. It is also interesting to note that the WaterHeater sent a large number of nullification or cancellation messages to all of the consumer agents because it was unable to fulfill all the requests it received.

In this run the AirConditioner and Heater agents also failed to reach their target temperatures. This is the result of the DishWasher's thrashing behavior. It would request and reserve electricity and thus interfere with the temperature control agents. When the DishWasher was unable to obtain the desired amount of hot water, it would release the electricity reservation but the thrashing behavior confused the (slow to respond to released resources) temperature control agents, resulting in diminished performance on their part.

Agent	# of Tasks		Final Quality		Resources Mes.		Conflict				Tasks Dropped	
	Alone	IHome	Alone	IHome	Alone	IHome	Alone	IHome			Alone	IHome
								E	R	N		
DishWasher	4	**1**	135	**40**	10	**84**	0	16	2	11	0	3
Robot	5	5	10	10	0	0	0	0	0	0	0	0
WaterHeater		**50**		10				**43**	0	**31**	0	0
OtherAppliances	37	28	10	10	42	37	0	1	13	9	0	**9**
CoffeeMaker	4	0	80	0	3	3	0	3	4	21	0	4
	4	4	125	125	3	3	0				0	0
	4	4	125	125	3	3	0				0	0
Heater	8(41)	9 (*)	40	40	8	9	4	6	2	0	0	0
AirConditioner	8 (41)	10 (*)	35	20	8	**22**	4	1	6	0	0	0

Figure 6: Experiment 3: *Alone* indicates the performance when the agent executed alone in the environment with sufficient resources. The *IHome* column indicates performance when the agents are executed in a group and resources are shared. *E/R/N* indicates conflicts emitted, received, or nullified.

7 Conclusions and Future Work

We have designed and implemented a simulated intelligent home environment and populated it with intelligent appliance agents. The agents interact and coordinate using the simple *SHARP* protocol over shared resources, contract over task-allocation interactions, and use a different coordination protocol for task overlap conditions. While we are pleased with this work, there is much room for improvement and expansion.

In the future, we plan to extend the protocols to cope with increasingly sophisticated IHome environment scenarios and situations requiring more complex negotiation between the agents. Temporal chains of multi-resource tasks is one example of this – particularly if member tasks are assigned to different agents. This leads to an interrelated multi-agent task and resource coordination problem. The introduction of multiple robots in the environment will motivate this area of exploration. We will also explore survivability, adaptability, and responsiveness issues in this context – deploying diagnoses and learning components currently under development.

A related area of future work is the integration of $GPGP^2$ into the IHome environment. This will facilitate comparison between the general coordination strategies employed in $GPGP^2$ and the specialized mechanisms currently used by the IHome agents. Experimentation with organizational design is another planned IHome extension. As mentioned, other areas of improvement include refining our evaluation metrics so that we can more easily evaluate experimental data and fully incorporating personal preference profiles into the agents' priority mechanisms.

In short, the intelligent home is proving to be an interesting environment for experimentation with MAS technologies. The multiple different types of resource and task interactions present in this application domain provide a rich landscape for work in coordination and local agent control. Even the simple resource protocol described in this paper opens interesting future research paths.

References

[1] Mihai Barbuceanu, Tom Gray, and Serge Mankovski. Coordinating with obligations. In *Proceedings of the Second International Conference on Autonomous Agents (Agents98)*, pages 62–69, 1998.

[2] Magnus Boman, Paul Davidsson, Nikolaos Skarmeas, Keith Clark, and Rune Gustavsson. Energy saving and added customer value in intelligent buildings. In *Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, pages 505–517, 1998.

[3] Jason A. Brotherton and Gregory D. Abowd. Rooms take note: Room takes notes! In *Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments*, pages 23–31, 1998.

[4] Norman Carver and Victor Lesser. The DRESUN testbed for research in FA/C distributed situation assessment: Extensions to the model of external evidence. In *Proceedings of the First International Conference on Multiagent Systems*, June, 1995.

[5] Sandeep Chatterjee. Sani: A seamless and non-intrusive framework and agent for creating intelligent interactive homes. In *Proceedings of the Second International Conference on Autonomous Agents (Agents98)*, pages 436–440, 1998.

[6] Michael Chia, Daniel Neiman, and Victor Lesser. Coordinating asynchronous agent activities in a distributed scheduling system. In *Proceedings of the Second International Conference on Autonomous Agents (Agents98)*, 1998.

[7] T. Darell, G. Gordon, J. Woodfill, and M. Harville. Tracking people with integrated stereo, color, and face detection. In *Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments*, pages 44–49, 1998.

[8] Keith Decker and Jinjiang Li. Coordinated hospital patient scheduling. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98)*, pages 104–111, 1998.

[9] Keith S. Decker. Task environment centered simulation. In M. Prietula, K. Carley, and L. Gasser, editors, *Simulating Organizations: Computational Models of Institutions and Groups*. AAAI Press/MIT Press, 1996.

[10] Werner Dilger. A society of self organizing agent in the intelligent home. Technical report, Technical Report SS-98-02 Standford, California, Menlo Park, March 1998.

[11] Edmund H. Durfee and Thomas A. Montgomery. Coordination as distributed search in a hierarchical behavior space. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1363–1378, 1991.

[12] David Franklin. Cooperating with people: The intelligent classroom. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 555–560, 1998.

[13] Bryan Horling. A Reusable Component Architecture for Agent Construction. Submitted to Agents99.

[14] Bernado Huberman and Scott H. Clearwater. A multi-agent system for controlling building environments. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS95)*, pages 171–176, 1995.

[15] Nicholas R. Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1):8–38, 1998.

[16] Yannis Labrou and Tim Finin. A Proposal for a new KQML Specification. Computer Science Technical Report TRCS-97-03, University of Maryland Baltimore County, February 1997.

[17] Victor Lesser, Keith Decker, Norman Carver, Alan Garvey, Daniel Neiman, Nagendra Prasad, and Thomas Wagner. Evolution of the GPGP Domain-Independent Coordination Framework. Computer Science Technical Report TR-98-05, University of Massachusetts at Amherst, January 1998.

[18] Victor Lesser, Bryan Horling, Frank Klassner, Anita Raja, Thomas Wagner, and Shelley XQ. Zhang. BIG: A resource-bounded information gathering agent. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, July 1998.

[19] Victor R. Lesser. Reflections on the nature of multi-agent coordination and its implications for an agent architecture. *Autonomous Agents and Multi-Agent Systems*, 1(1):89–111, 1998.

[20] T. Moehlman, V. Lesser, and B. Buteau. Decentralized Negotiation: An Approach to the Distributed Planning Problem. In K. Sycara, editor, *Group Decision and Negotiation*, volume 1, pages 161–192. Kluwer Academic Publishers, 1992.

[21] Michael C. Mozer. The neural network house: An environment that adapts to its inhabitants. In *Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments*, pages 110–114, 1998.

[22] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transctions on Computers*, 29(12):1104–1113, 1980.

[23] Rainer Stiefelhagen, Jie Yang, and Alex Waibel. Towards tracking interaction between people. In *Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments*, pages 123–127, 1998.

[24] Regis Vincent, Bryan Horling, Thomas Wagner, and Victor Lesser. Survivability simulator for multi-agent adaptive coordination. In *Proceedings of the First International Conference on Web-Based Modeling and Simulation*, 1998. Also available as UMASS CS TR-1997-60.

[25] Thomas Wagner, Alan Garvey, and Victor Lesser. Criteria-Directed Heuristic Task Scheduling. *International Journal of Approximate Reasoning, Special Issue on Scheduling*, 19:91–118, 1998. A version also available as UMASS CS TR-97-59.