SPONSOR-BASED-ARCHITECTURE FOR RESOURCE MANAGEMENT IN MULTI-AGENT SYSTEMS

Zunera Jalil and Hajira Jabeen
National University of Computer & Emerging Sciences (FAST-NU), Islamabad, Pakistan

ABSTRACT

Collaboration and competition among agents in multi-agent systems extends to consumption of computational resources. Management of resources is critical to the performance of multi-agent systems. This paper presents a new concept for distributed resource management in multi-agent system based on the sponsor based approach of economics. Sponsor based architecture is an economic framework where autonomous agents with resource requirements look for sponsors of that resource. The notion of customer agent and sponsor agent has been used where customer agent is the one in need of resource and sponsor is the one offering that resource. Agents pay rent in the form of currency units (GCUs) to the sponsor and get resource units in return. Sponsors periodically advertise their packages. First we introduce the resource allocation problem in multi-agent systems. We then briefly analyze the existing traditional, agent based and actor based architectures for resource management. In the next section we propose a sponsor based architecture RASP (**R**esource **A**llocation using **SP**onsors).We then present some experimental results showing RASP performance.

KEYWORDS

Sponsor, GCUs, Optimization Problem, Actors

1. INTRODUCTION

Resource allocation in multi-agent systems is a problem that raises issues of reciprocity as well as performance and security concerns. In multi-agent systems, agents migrate from one node to another searching for computation environments suitable for them to complete their tasks with affordable costs. In distributed environment where large numbers of agents are distributed over the network, each trying to avail resources, a greedy agent with a low priority task can monopolize a resource for a long time, making agents with higher priority tasks to wait. “Malicious and erroneous agents may threaten the node not only by attempting to access specific resources, but also by the degree to which they use them.[A] A good resource management strategy is the one having efficient resource access and utilization, with load balancing and fault tolerant behaviour. The basic idea in our sponsor based approach in derived from social welfare framework of economics and process scheduling methods used in operating systems.

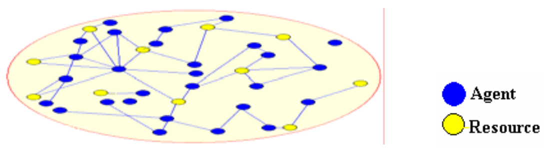

Figure 1. Agents and shared resources in multi-agent systems

This paper is organized as follows. We first introduce the distributed resource management problem, its importance and complexity. In the next section we discuss traditional, agent based, actor based and CyberOrgs architectures used for distributed resource management in past. In section 3, we propose and describe the sponsor based architecture for distributed resource management problem in multi-agent systems. Section 4 gives the simulation results and the final section concludes the paper highlighting the direction for future research.

2. RELATED WORK

In dynamic distributed resource allocation scenario, information, decisions and resources are distributed and the exact conditions e.g demand and availability of resources are not known in advance and are changing [D]. Researchers have previously used natural and economic models to design algorithms to solve different kinds of resource allocation or optimization problems. The architectures used before are as follows:

2.1 Traditional Architectures

Traditional distributed resource management systems use a system model to describe resources and a centralized scheduler to control their allocation. A central node has the entire responsibility of choosing and assigning the optimal resource to a task at hand. These approaches are both inefficient as well as error-prone, making the resources under-utilized and vulnerable to severe performance degradation under unexpected overload conditions [C]. The communication overhead is involved because of centralization. Also under heavy workload congestion may occur in the systems making it a bottleneck.

2.2 Agent-based Architectures

In agent based architectures used for resource management, control is distributed and concurrent and the interaction between agents is asynchronous. Agents are persistent, having long lived goals describing the functional aspects of what they are doing and have computational engines which serve as a mechanism for achieving these goals and these computational engines include a resource utilization strategy. In agent based approach, resources are assigned to agents initially and afterwards they can only use the resource they have in hand. In this architecture, agents do not have a global view of the state of the system so the resource allocation decisions made initially and overall resource utilization may not be optimal and good.

2.3 Actor-based Architectures

Resource management has also been performed using actors in past. Actors were initially developed by Carl Hewitt [G] for conceptual modeling of open systems. Actors are self-contained, interactive, autonomous components of a computing system that communicate by asynchronous message passing. Actor model does not require that the structure or shape of a computational problem be completely determined [A]. In a typical actor based model, all agents having resources becomes actors, they coordinate, negotiate and share resources with each other making autonomous decisions. In other actor based architectures [A], some actors are assigned duties of resource allocation and they act as “host' of resources. The host allocates the resource units to an agent interested in paying currency for it. These may include both physical (e.g processors, memory etc) and logical resources (threads). This architecture is quire realistic but it has some disadvantages.

1) Since one type of resource units are managed by a single host so all agents requiring that resource need to communicate to that particular host leading to network traffic congestion.
2) The choice of allocating resource to agents lies with the host agent and the host will always try to allocate the optimum resource to a task at hand which is a complex problem.
3) Loops and deadlock can occur in these architectures quite easily. For example TASK1 having resource A may require resource B captured by TASK2 and simultaneously TASK 2 having resource B with it may require resource A.

2.4 CyberOrgs Model

CyberOrgs is a model for resource control in an open multi-agent system organizing computational resources as a market, and their ownership and control hierarchically. Each cyberorg encapsulates a concurrent computation and resources available for its execution. Cyberorgs own *eCash* with which they buy resources according to a pre-negotiated contract. Even though ownership relationships are represented as a hierarchy, the ownership of a resource is absolute for the duration of an ownership. CyberOrgs has no cost management strategy and no centralized control to define relation ships between cyberorgs. CyberOrgs also provide no realistic description about ticks and their incorporation. Hence, we propose a new sponsor based model called RASP (**R**esource **A**llocation using **SP**onsors).

3. PROPOSED ARCHITECTURE

Complex networks of multi-agent systems share many of the characteristics of human economies, such as heterogeneity, decentralization, and dynamism. Thus, it is only natural to apply the principles developed in the study of complex human economies to the problem of resource allocation in complex multi-agent systems. Sponsor based approach is an approach from 'computational microeconomics'. Resource units are purchased by paying GCUs (Global Currency Units) to the sponsors of the required resource.

3.1 Customer Agent

Customer agents are the agents having task at hand and requiring resource(s) to complete their tasks. They cooperatively allocate their tasks to the sponsor agent with the capability to carry them out. Tasks are characterized by their id, priority and their GCUs. Agent having the task keeps its GCUs in its account. It is also assumed that every agent has only one task to perform with it and if the task requires more resources than its limit then it will be split before sending to sponsor. GCUs are deducted from customer agent's account when it gives some currency to the sponsor. In previous agent and actor based architecture all tasks were treated equal however, in real life we need to prioritize tasks. Three priority levels 1, 2 and 3 are introduced in our sponsor based architecture for high, medium and low priority tasks respectively.

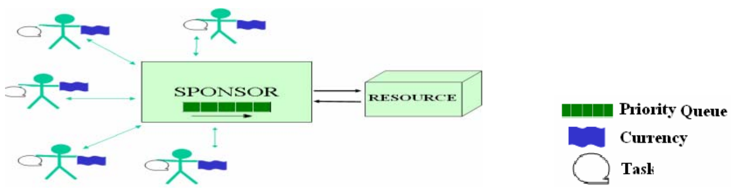

Figure 2. Customer Agents with tasks and currency units (GCUs) in hand communicating with sponsor of resource

The customer agent will be having task, its priority, required amount of resource units and some GCUs at hand. First the appropriate sponsor will be selected and then task and its GCUs will be sent to the sponsor agent. Sponsor agent will receive the task and will fulfill its resource requirement. Afterwards it will be sent back to the customer agent. Customer agents are shown in figure 2 below where many customer agents are communicating with a single resource sponsor which is having a priority queue for resource allocation.

3.2 Sponsor Agent

Sponsor is actors that manages physical & logical resources of a node and offer them to agents interested in paying for them. Every sponsor is manager of a single resource. The idea of sponsor is derived from the actor based economic model used in CyberOrgs. In proposed architecture a sponsor is associated with every single resource. Sponsor sets price for resource units and initializes its data structures. How much price to be set for resource items? This decision depends on many factors, for example, resource consumptive behavior of the system, number of agents or resources of same type existing in the environment. Two parameters are used:

1) *Maximum currency units a sponsor can accept for a task by customer agent. (C)*
2) *Minimum resource units to be allocated to any task (R)*

Sponsors maintain a priority queue with it to place pending jobs requiring resource and on the basis of their priority jobs are submitted to the resource agent and taken back on completion or when R units have been assigned. When a request for resource arrives at sponsor, it places the task in the priority queue. Three priority levels one, two and three are used for high, medium and low priority tasks respectively. Task scheduling afterwards is done quite similar as the process scheduling in operating systems. The tasks of same priority will be handled on FCFS basis. A constant amount of resource units (R) can be assigned at a time, after assigning R resource units; resource agent deducts its GCUs and places it in the queue. When the GCUs finish, the task is sent back to the customer agent. An agent with a low priority may give large amount of GCUs to the sponsor monopolizing resource. To avoid such scenarios, the parameter C is used.

3.2.1 GCUs Allocation

In previous actor based architectures, GCU were allotted fixed to all, randomly or by user selection. In proposed sponsor based architecture (RASP) the problem of allocation of GCUs is solved in an efficient way. GCUs are allotted by agents and there are no restrictions on the number of GCUs it can assign except one i.e. the number of GCUs will be less than the monetary value of all resources. However to avoid monopoly and starvation we have used constants C and R.

3.2.2 Resource Discovery

One of the central problems faced in multi-agent systems is keeping track of the resources being used and dynamic resource discovery. Rather than specifying the names of the machines you specify your task requirements with some GCUs and the systems will choose the sponsor having best offer for you. Sponsors advertise their services periodically to all agents (e.g 10 nano seconds of processor time for 1 GCU) and each agent maintaining list of sponsors in its knowledge base. When an agent will be having a task with it, it will look in its sponsors table of knowledge base and will choose an appropriate sponsor. Sponsors advertise their services and offers in after random time intervals and agent while performing their routine work receives these offers and saves in its knowledge base for future usage. A sponsor is managing a single resource item of one type of resource (e.g. 10 sponsors for 10 processors). Deadlock cannot occur in our RASP because the currency with customer agents will never be more than the total price of resources. So there will be no exceptionally rich agent. In order to minimize the number of lost requests caused by an overload, the allocation of resources is changed dynamically and to avoid resource monopolization two parameters C and R (mentioned above) are used. The complete algorithm used in sponsor based architecture is as under:

```
Step1:  Set resource units (R) & Maximum GCUs
Step2:  If Task Received then Place it in Queue else repeat step 2
Step3:  If Queue is Empty then goto step2
Step4:  Send task to resource agent; deduct GCUs and goto step 3
```

Another resource agent is used which actually assigns resource units to the jobs sent by sponsor. In RASP, the pending tasks are placed in a priority queue; sponsor sends the task requesting the resource to the resource agent. Resource agent receives the task, gives R resource units and returns it back to the sponsor. It is assumed that at a time only R resource units can be assigned to any task.

4. SIMULATION RESULTS

We introduce programming abstraction derived from the sponsor based model through a prototype implementation of RASP as a sponsor program. This proposed sponsor based architecture has been implemented in the Java programming language using JADE for agents' incorporation. Statistical analysis helps us to suggest how the algorithm works. In RASP, the performance of systems is analyzed by changing the values of two parameters C and R. Rest of the parameters that are not mentioned will remain constant.

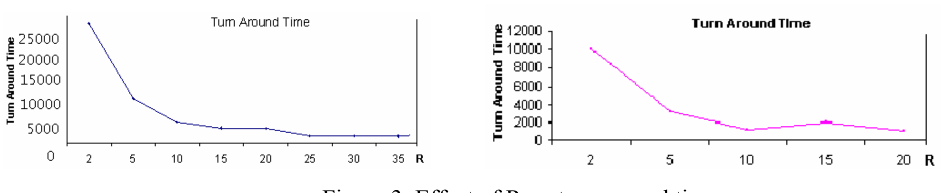

Figure 3. Effect of R on turn around time

We have measured the performance of systems analyzing the turn around times of tasks. Since tasks requiring greater than R resource units are placed in priority queue more than once, so the turn around time of heavy tasks with more GCUs gets affected. The resource allocation is fair and prevents starvation (Table 1). In table 1, tasks 5 and 6 with smaller number of GCUs are not starved. However the tasks with large number of GCUs are completed in parts resulting in greater turn around time. This turn around time will also not be too large because a task with greater than C number of GCUs will not be accepted at all. Rather task partitioning will be required. It was noted that the turn around time was lesser for the agents with R=0.5*C.

Table1. Task turn-around-times of task having different customer GCUs values

Task No.	Agent GCUs	P	C	R	Task Turn-Around-Time
1	50	1	50	10	15344
2	40	1	50	10	15531
3	30	1	50	10	12484
4	10	1	50	10	1406
5	5	1	50	10	6453
6	2	1	50	10	1422

RASP manages tasks of different sizes effectively, making efficient utilization of resource. Agents of same priority with different number of GCUs are treated equally. We tested RASP application with 25 customer agents each trying to access sponsor resources simultaneously, the experimental results are shown in figure 4.

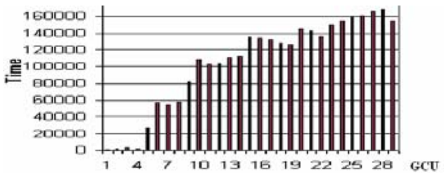

Figure 4: Fair Resource allocation by sponsor to different types of 25 tasks having diverse GCUs.

5. CONCLUSION

The resource allocation and management strategy proposed in RASP using sponsor based approach is quite efficient, fair, optimum and practical with increased resource utilization and less communication overhead. The multi-agent systems become more responsive, fair, flexible, robust and reliable. In current sponsor based architecture each agent can perform single task a time. We can extend it by allowing agent to perform many tasks requesting many sponsors simultaneously. In current implementation of sponsor based architecture (RASP), we have used CPU cycles as a resource and generated statistics; however it can be extended to simulate for other resources like file, memory, printer etc as well.

REFERENCES

[A]. Nadeem Jamali, Prasannaa Thati and Gul A. agha, 1999. An Actor-based Architecture for Customizing and Controlling Agent Ensembles. *IEEE Intelligent Systems, v.14 n.2, p.38-44.* Department of Computer Science, University of Illinois, Urbana, IL 61801, USA.

[B]. Nadeem Jamali and Gul A. Agha, 2003. CyberOrgs: A Model for Decentralized Resource Control in Multi-Agent Systems. *Second International Join Conference on Autonomous Agents and Multiagent Systems (AAMAS 03).* Department of Computer Science, University of Saskatchewan, Canada and Department of Computer Science, University of Illinois, Urbana, IL 61801, USA.

[C]. S. Groh and M. Pizka, 1997. A Different Approach to Resource Management for Distributed Systems. *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA'97.*Munich University of Technology, Department of Computer Science, Germany.

[D]. Paul Davidsson, Stefan J. Johansson, Jan A. Persson and Fredrik Wernstedt, 2003. Agent-based Approaches and Classical Optimization Techniques for Dynamic Distributed Resource Allocation: A preliminary study. *Proc AAMAS´03 workshop on Representations and Approaches for Time-Critical Decentralized Resource/Role/Task Allocation).* Department of Software Engineering and Computer Science, Blekinge Institute of Technology, Soft Center, Sweden.

[E]. Gul A. Agha, Prasannaa Thati, Reza Ziaci, 2001.Actors: A Model for Reasoning about Open Distributed Systems. *Formal methods for distributed processing: a survey of object-oriented approaches,* Cambridge University Press, New York

[F]. Jonathan Bredin, David Kotz ,Daniela Rus,2005, *Utility Driven Mobile-Agent Scheduling,, Dartmouth College Hanover, NH, USA*

[G]. C. Hewitt, 1977. Viewing Control Structures as Patterns of Passing Messages. *Journal of Artificial Intelligence*, vol 8-3, pp 323-364.