Understanding decentralised control of resource allocation in a minimal multi-agent system

Mariusz Jacyno, Seth Bullock, Terry Payne
School of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK
{mj04r,sgb,trp}@ecs.soton.ac.uk

Michael Luck
Department of Computer Science
King's College London
Strand, London WC2R 2LS, UK
michael.luck@kcl.ac.uk

1. INTRODUCTION

In response to the advent of new computational infrastructures, a number of initiatives, such as *autonomic computing* [5] and *utility computing* [8], have been announced by major IT vendors sharing the same underlying principles of provisioning distributed computational resources to a large number of users "on demand". Since, by their nature, such systems are large, open and dynamic, allocation of resources to users presents unique challenges that threaten to overwhelm existing centralised management approaches [2].

An efficient and fair mechanism of resource allocation must be scalable, reliable and adaptive to changing system conditions. Devolving responsibility from some central administrative authority to a population of autonomous agents, each representing a resource consumer or a resource provider, offers one way to address the challenge. However, it is far from obvious how to equip such agents with resource allocation mechanisms that can satisfy global system objectives despite operating on local and imperfect information.

We argue that the success of such an approach will hinge on understanding the fundamental properties common to any decentralised resource allocation system. By focusing initially on the dynamics of resource allocation associated with minimally complex agents, this task can be made tractable to empirical exploration and analysis. In doing so, we are committed to a working hypothesis: the dynamics of such a system, while simple, will share general properties with more realistic utility computing systems involving more complicated "intelligent" agents.

There is a considerable literature exploring the efficacy or optimality of particular candidate agent algorithms for a range of resource allocation and associated tasks [10, 11, 7] (e.g., coalition formation, load balancing, role allocation, etc.). Such work often relies on a central executive to handle the task, and tends to focus on assessing algorithm performance, rather than characterising the fundamental nature of the problem. Work on resource allocation in decentralised multi-agent systems is much less frequent [4, 9, 1]. Here we focus on identifying general principles that explain how performance scales under various kinds of systemic pressure. As such, we do not aim to develop or evaluate a working solution to a specific resource allocation problem for utility computing. Instead, we concentrate on characterising relationships between local mechanisms and global performance that are likely to generalise from the minimal system explored here to all real-world utility computing systems in general.

2. SIMULATION

2.1 Model design

The decentralised multi-agent system that we will explore in this paper comprises of a service registry that serves as an inventory of the resource providers within the system; a population of S_N agents representing resource providers (services); a population of U_N agents representing resource consumers (consumers).

Services are provided by agents that facilitate access to resources (disk storage, CPU time, etc.).

Consumers consume resources according to a fixed personal workflow defining the type, capacity and order of services required.

The registry is an agent tasked with maintaining an inventory of system services, and supplying it to consumers when queried. Since, in reality, the information obtained from such registries can become stale and unreliable in a dynamic system, in future work we will be interested in systems where this unreliability is sufficient to ensure that consumers prefer not to make use of it at all.

Service Allocation is decentralised such that each consumer first obtains from the registry a list of existing services capable of providing any of the resources required by its workflow. This action takes time T_x and incurs an execution cost, C_x. Repeatedly, services are chosen from this list and their availability determined (each time incurring a query cost, C_q, and consuming time, T_q). Services may be unavailable because they are busy to the extent that they do not have the spare capacity required by the workflow component, or because they are no longer part of the system. Once an available service has been located, the agent attempts to allocate the next component of its workflow. Once all components of the workflow are allocated to services in this way, the agent attempts to execute the workflow using these services. Since services are not locked during the allocation process, it is possible that a consumer agent may allocate a workflow component but find that the service is busy when it attempts to execute it. In such circumstances, the consumer must re-allocate this workflow component. Successfully executing a component also takes time, T_x, and incurs an execution cost, C_x. Should a service fail during execution, the consumer still pays the execution cost, but must also re-allocate the workflow component. If, during any allocation process, a consumer makes n attempts to locate an available service of a particular type, the allocation is deemed to have failed, as is the workflow it is part of. Here, for each workflow component to be allocated, we set n equal to the number of services of the required type returned by the registry. Whether successful or unsuccessful, upon completing a workflow, a consumer agent, U_i is inactive for some randomly determined period drawn from a uniform distribution $[0, \omega_i]$ after which the same workflow allocation process begins again.

Scenarios, unless stated otherwise, involve an equal number of agents repeatedly attempting to execute each of three different workflows ($W_1 = \{A\}$, $W_2 = \{A, B\}$, $W_3 = \{A, B, C\}$) for a period of time, T_N. Since we expect simple workflows to be requested more frequently, the maximum period of time for which a consumer will sleep after completing (or failing to complete) a workflow, ω, is workflow-dependent such that more complicated workflows tend to be associated with longer periods of sleep: $\omega_1 = 1\text{s}$, $\omega_2 = 2\text{s}$, $\omega_3 = 3\text{s}$.

For all agents, the following values are assigned to the costs and execution times of service interactions and service executions: $C_x = 20$ units, $T_x = 500\text{ms}$, $C_q = 10$, $T_q = 100\text{ms}$. These parameters define minimal costs for a consumer attempting to execute each workflow: $C^*(W_1) = 50$ units, $C^*(W_2) = 80$ units, $C^*(W_3) = 110$ units. Since the proportion of consumers attempting to execute each workflow is the same, the optimal cost for a system can be calculated as $C^* = 80$ units.

2.2 Consumer Strategies

Consumers rely on strategies to guide their individual behaviour. Here we explore minimally sophisticated strategies: **Random selection ($\mathcal{R}$)**: when attempting to allocate a workflow component to a service, the consumer makes random choices from the list of available resources obtained from the service registry. **Hybrid strategy ($\mathcal{RP}$)**: when attempting to allocate a workflow component to a service, the consumer preferentially returns to the last service employed for that component, if it was executed successfully during the last workflow, otherwise the random selection strategy is employed.

We do not expect these simple strategies to be employed within real utility computing systems. However, the simplicity of the randomising and canalising behaviours that they employ make them good candidates for examination, since any decentralised resource allocation strategies that *are* employed within utility computing will probably involve some more complicated form of randomisation and/or canalisation.

3. RESULTS

3.1 Consumer Heterogeneity

It is highly unrealistic to assume that agents attempting to allocate the same type of resource will also share exactly the same service preferences. For example, in the domain of utility computing, different amounts of CPU processing power, storage size, quality of service, etc., may be required. Hence, for each consumer, only a subset of services of a particular type will be capable of satisfying its particular demands. Since many of these attributes are dynamic properties that may change rapidly and unpredictably, it may be that a centrally maintained registry of services cannot be relied upon to provide the information required by consumers to identify appropriate services.

In order to manipulate the degree of heterogeneity in consumer demand, H, within the model we assign different capacity requirements to consumers and differing capacity provision to services. A consumer will be satisfied by any service of the required type with free capacity that either equals or exceeds its capacity requirements. As such, consumers with high capacity demands must necessarily have at least as difficult an allocation task as consumers with lower capacity demands. In all cases considered here, there exists an allocation of services to consumers where all available service capacity is utilised in executing every workflow component simultaneously (i.e., supply always exactly matches demand), and no service has capacity to simultaneously execute more than one workflow component. We define H as the number of unique levels of service capacity required by the workflows of a consumer population (or, equivalently, the number of unique levels of capacity provided by a population of services). Thus, for $H = 1$, all workflows share the same capacity requirements, whereas for $H = 2$, each workflow (and every service type) is present in a low-capacity and high-capacity variant such that each variant is assigned to an equal number of consumers. The scenarios reported below are otherwise identical to those described in Section 2.1, with systems comprising of 180 consumers and 360 services.

For consumers employing the random selection strategy, $\mathcal{R}$, there is a significant increase in cost, and in its variation, as the degree of consumer heterogeneity increases. While $\mathcal{R}$ preserves a uniform distribution of resource requests among a group of services of the same type, and thus effectively minimises the number of conflicts, it is *blind* to the various levels of capacity offered by services. By contrast, the $\mathcal{RP}$ strategy, which combines $\mathcal{R}$ with preferential selection of previously utilised services, delivers a significant improvement in performance. Here, the system converges to a near optimal allocation of services to consumers, effectively matching consumer capacity demands to service capacity provision from a global as well as local perspective. This convergence is achieved, even though there exists a potential for conflict between low- and high-capacity consumers over who gets to utilise high-capacity services.

These observations are confirmed by Figure 1 which depicts the mean workflow completion costs for a high degree of consumer heterogeneity ($H = 4$). For each workflow, four levels ofcapacity demand are introduced: $\{s1, s2, s3, s4\}$. While high-capacity workflows tend to attract higher allocation costs, irrespective of consumer strategy, the departure from optimal allocation costs is much reduced for $\mathcal{RP}$ strategists. Consumers adopting this hybrid strategy were thus able to form preferences for resources that not only satisfied their own resource demands but, in the case of low-capacity consumers, also contributed to satisfying the demands of their competitors, increasing overall system efficiency.

4. DISCUSSION

Convergence to an optimal allocation is a consequence of a balance between random selection and preferential selection. Where an inefficient allocation of services to consumers arises, the former tends to disturb preferences for resources that are overexploited, while preferential selection is able to increase pressure on underexploited services. In this way, the extent to which low-capacity consumers form preferences for high-capacity resources will tend to be matched by the pressure to “move on” exerted by unsatisfied high-capacity consumers.

At any point in time, a system of $\mathcal{RP}$ consumers can be represented as two interdependent populations of agents, one currently able to rely on a developed preference, while the other either has no such preference, or has a preference for a service that is currently busy, and must rely on random selection. While it is important to remember that every consumer in a $\mathcal{RP}$ population possesses the same strategy, we will refer to these temporary behavioural dispositions as $\mathcal{RP}$ strategy *elements*. Agents relying on the $\mathcal{R}$-element are more aggressive, selecting resources randomly and thereby dispersing their activity across all system resources. Agents relying on the $\mathcal{P}$-element, on the other hand, canalise their activity in a specific region of the systems resources. Where supply meets or exceeds demand, the former encourages system fairness, while the

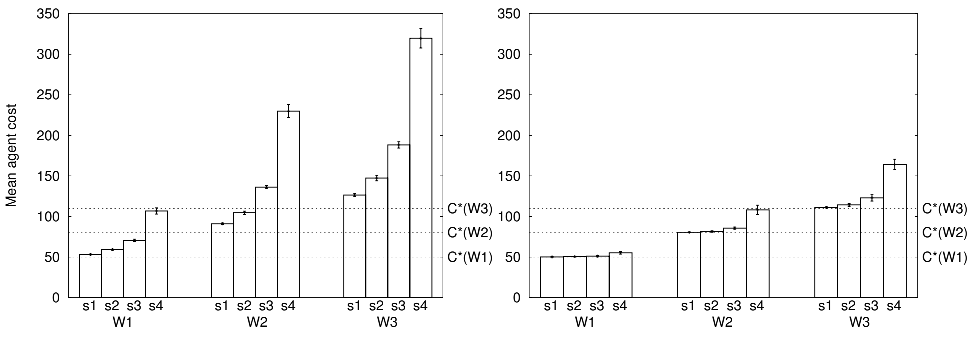

Figure 1: Mean workflow completion costs for agents relying on $\mathcal{R}$ (left) and $\mathcal{RP}$ (right) where $H = 4$. Within each workflow type, four subclasses are identified in order of increasing capacity requirement ($s1, s2, s3, s4$). Dotted lines correspond to the optimal cost for each workflow group. $U_N = 180$, $S_N = 360$, $T_N = 400$ seconds.

latter lowers system cost. Since there is no central controller deciding which agent should rely on what strategy element for a given level of system heterogeneity, it is interesting to explore by what means the balance between strategy elements is brought about.

Crudely, each "sub-population exerts a specific pressure on the other. By "stealing" the preferred resources of conservative $\mathcal{P}$-element agents, aggressive $\mathcal{R}$-element agents drive $\mathcal{P}$-element agents to switch strategy. At the same time, $\mathcal{R}$-element agents that successfully allocate resources also switch strategy element. In both cases, such switching prevents agents relying upon the same resource for a long time. This ensures that the costs of resource competition are distributed fairly among all agents. Furthermore, as system dynamism increases (with increasing load or heterogeneity, for instance), and the chance of developing useful preferences falls, the proportion of $\mathcal{R}$-element agents increases. Likewise, if system dynamism relaxes, the proportion of agents successfully exploiting preferences increases. This coupling between strategy elements drives the system behaviour, and its response to externalities such as load or heterogeneity.

The nature of this coupling resembles certain accounts of self-organisation within natural decentralised systems, where complex system-level dynamics are characterised in terms of the generic feedbacks that arise from local interactions between components [3, 6]. Within the minimal system presented here, the $\mathcal{R}$ strategy (or element) generates destablising positive feedback within the population of agents, whereas the $\mathcal{P}$-element induces canalising negative feedback. Since random selection dominates the weak negative feedback brought about by consumer preferences, where demand outstrips supply, the entire system reconfigures continually. While such reconfiguration is costly, it ensures fairness by preventing a subset of consumers from benefiting from stable preferences at the expense of their competitors.

However, where consumers are either unable or unwilling to make use of a centralised registry of services, ensuring efficiency and fairness will be a more complicated challenge. In such a case it is likely that agents will have to base their allocation decision on information that they learn themselves, or learn from other agents. As such, strategies for sharing (or not) information about resources with other agents will be a key consideration. In future work it will be important to explore how such consumer strategies might be encouraged to bring about allocation that is robust, efficient and fair. Addressing these questions will necessarily involve understanding how the feedbacks identified here generalise to more sophisticated agent behaviour and a richer "ecology" of differentiated resources and resource demands. However, as with the current study, it will be crucial to consider minimal agent strategies in the most simple agent ecologies, since introducing a more realistic degree of complexity is likely to render analysis intractable.

5. REFERENCES

[1] S. Brueckner and H. V. D. Parunak. Self-organizing MANET management. In G. D. Marzo, A. Karageorgos, O. F. Rana, and F. Zambonelli, editors, *Engineering Self-Organising Systems*, pages 1–16. Springer, 2003.

[2] S. Bullock and D. Cliff. Complexity and emergent behaviour in ICT systems. Technical Report HP-2004-187, Hewlett-Packard Labs, 2004.

[3] F. Heylighen and C. Joslyn. Cybernetics and second-order cybernetics. In R. Meyers, editor, *Encyclopedia of Physical Science and Technology*, volume 4, pages 155–170. Academic Press, New York, 2001.

[4] T. Hogg and B. A. Huberman. Controlling chaos in distributed systems. *IEEE Transactions on Systems, Man and Cybernetics*, 21:1325–1332, 1991.

[5] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEE Computer*, 36(1):41–50, 2003.

[6] H. V. D. Parunak and S. A. Brueckner. Engineering swarming systems. In F. Bergenti, M.-P. Gleizes, and F. Zambonelli, editors, *Methodologies and Software Engineering for Agent Systems*, pages 341–376. Kluwer, 2004.

[7] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[8] M. A. Rappa. The utility business model and the future of computing services. *IBM Systems Journal*, 43:32–42, 2003.

[9] S. Sen, S. Roychowdhury, and N. Arora. Effects of local information on group behavior. In *Proceedings of the Second International Conference on Multi-Agent Systems*, pages 315–321. AAAI Press, Menlo Park, CA, 1996.

[10] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101:165–200, 1998.

[11] P. Stone and M. Veloso. Layered learning and flexible teamwork in robocup simulation agents. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup*, pages 495–508. Springer, 2000.