Distributed and Centralized Task Allocation: When and Where to Use Them

Johannes van der Horst
SENSe
University of Southampton
Southampton, UK
jgvdh08r@ecs.soton.ac.uk

Jason Noble
SENSe
University of Southampton
Southampton, UK
jn2@ecs.soton.ac.uk

***Abstract*—Self-organisation is frequently advocated as the solution for managing large, dynamic systems. Distributed algorithms are implicitly designed for infinitely large problems, while small systems are regarded as being controllable using traditional, centralised approaches. Many real-world systems, however, do not fit conveniently into these "small" or "large" categories, resulting in a range of cases where the optimal solution is ambiguous. This difficulty is exacerbated by enthusiasts of either approach constructing problems that suit their preferred control architecture.**

We address this ambiguity by building an abstract model of task allocation in a community of specialised agents. We are inspired by the problem of work distribution in distributed satellite systems, but the model is also relevant to the resource allocation problems in distributed robotics, autonomic computing and wireless sensor networks. We compare the behaviour of a self-organising, market-based task allocation strategy to a classical approach that uses a central controller with global knowledge. The objective is not to prove one mechanism inherently superior to the other; instead we are interested in the regions of problem space where each of them dominates. Simulation is used to explore the trade-off between energy consumption and robustness in a system of intermediate size, with fixed communication costs and varying rates of component failure. We identify boundaries between regions in the parameter space where one or the other architecture will be favoured.

This allows us to derive guidelines for system designers, thus contributing to the development of a disciplined approach to controlling distributed systems using self-organising mechanisms.

***Index Terms*—self-organising control, centralised control, resource management, space vehicle control**

I. Introduction

The complexity of our modern distributed systems has led to increased interest in alternative methodologies such as self-organisation and self-adaptation for network configuration and control. These interrelated approaches promise increased robustness and scalability , while internally managing of the individuals in these networks. Self-organisation, which is the focus of this paper, achieves this through local interaction [1], while self-adaptive systems continuously regulate themselves while on-line. In recent years, however, these approaches have become an end in itself: an increasing number of publications claim merit primarily because they demonstrate the successful application of either technique. A parallel literature on traditional centralised approaches exists that assumes that global knowledge of the system is indispensable, usually motivated by claims of optimality. This schism is exacerbated by enthusiasts of the respective approaches constructing problems that suit their preferred control architecture. For self-organisation to become a respected tool available to system designers, we need to know when to use it and, just as importantly, when *not* to use it.

Distributed algorithms are implicitly designed for infinitely large systems. However, practical demonstrations of self-organising control methodologies frequently employ only modest numbers of agents. At the same time there is an assumption that small systems will be adequately addressed using centralised control. But this raises an important question: when can we describe a system as small? And when is it large? The clarity with which we think about the extremes obscures the vagueness that lies in between. This hazy middle-ground is also where a number of real-world systems can be found. In this paper we address this uncertainty by comparing the performance of two task allocation schemes. The objective is not to prove one superior to the other, but to identify the areas where their respective strengths makes one of them the control mechanism of choice. One is a self-organising, distributed mechanism that utilises market mechanisms to achieve good allocation [2]; the other is a centralised controller that uses a global model of the network to calculate allocations. The analyses of their respective behaviours illuminate points of interest to the wider self-* community: we need to carefully and responsibly identify the valid regions of application for these techniques.

Our interest in this area stems from research on ways of controlling distributed satellite systems [3]. Current spacecraft generally have monolithic structures, but a next generation is envisaged where a large number of specialised, free-flying modules will cooperate to complete mission objectives. The spatially distributed approach promises an increase in robustness, mission flexibility and lower system cost [4] [5]. However, these will be challenging systems to manage: failures and changing satellite positions mean that the communication topology is unstable, nodes have severe power limitations, and communication uses energy which translates to a measurable impact on system performance. Spacecraft engineering has traditionally been dominated by centralised control, but the characteristics of this problem point to a potential need for distributed control. Fractionated satellites fall in the problematic area where we have multiple components, but possibly not enough to warrant a self-organising approach. Although a wide range of task allocation strategies is available, we therefore limit our investigation to a centrally coordinated approach and a decentralised one.

The literature relating to decentralised task allocation spans several different disciplines: multi-robot coordination, distributed computing, wireless sensor networks, and operations research. The history and constraints of these fields have largely determined the approaches followed. A few attempts at classifying the space of task allocation problems have been made, but usually these are field-specific [6], [7]. [8], discussing multi-robot systems, defined the following three axes:

- Single-task vs multi-task agents: Agents can execute a single or multiple tasks simultaneously.
- Single-agent vs multi-agent tasks: Some tasks can be completed by one agent, while others need multiple agents.
- Instantaneous vs time-extended assignment: Instantaneous assignment permits only instantaneous allocation of tasks, with no information about the future. In time-extended allocation more information is available, such as the distribution of future tasks or the set of all possible tasks.

These dimensions are a good start in classifying multi-robot problems, but we have found the following additional parameters to be useful in describing the more general problem space of task allocation in multi-agent systems.

- Communication cost: In some networks bandwidth is effectively unlimited, which makes communication costs negligible, while in others communicating more than is necessary decreases system utility. The cost of transmitting information therefore determines the amount and accuracy of available information: global information (and control) is infeasible when communication is expensive, necessitating an approach that relies on local information.
- Group size: The number of agents in the group determines the rules and behaviour that apply, with a non-linear relationship between group size and the expected system-level behaviours, as argued elegantly by [9]. For a small group (less than ten agents) it is feasible to control agents either individually, or using centralised control; both of these methods are likely to run into problems with a very large group (thousands of agents or more). The space in-between is an area where the appropriate control mechanism can be contested.
- Agent heterogeneity: Diversity in agent types allows role specialisation, while homogeneous agents typically have greater redundancy.
- Volatility and node failure: If individual agents in a systems are extremely reliable, the control strategy need not consider the potential disruption to task allocation and routing: indeed in many models failure is not included as a possibility. In other systems it will be common for agents to malfunction or die, thus necessitating constant rearrangement of the communication network and reallocation of tasks.

Distributed satellite systems span a portion of this space. The majority of missions will involve single-task agents, with tasks requiring multiple agents to complete (either consecutively or concurrently). Task assignment can be regarded as instantaneous, as roles may change at any time due to component failure or changing mission objectives. For most applications, communication costs will be high due to the power it requires. Initially group sizes will be moderate, in the range of tens to hundreds, although massive systems have also been proposed [10]. The component spacecraft will be heterogeneous, but with some redundancy.

Our goal is to use both analytic and simulation methods to start to carve out the problem space defined by the dimensions listed above, separating it into regions where either distributed or centralized control is indicated as being more efficient and effective. We have focused in particular on communication cost, group size, and volatility. Previous attempts to compare distributed with centralized control on a level playing field are surprisingly rare: comparative studies primarily use the performance of centralised controller as a baseline against which to demonstrate the quality of the decentralised solution. Examples include [11] on computational grids, and [12] who compared a centralized climate control system to various market-based approaches. We suspect that the lack of fair comparisons is due to the dichotomy outlined by [13] between the centralized and decentralized mindsets. Centralised control is so ubiquitous in our way of thinking that it is rarely named explicitly, and decentralized approaches on the other hand can seem exotic and enchanting in the way that global order emerges from local interactions. It is our hope that the current paper will help to take some of the sound and fury out of this clash of architectures, simply through comparing each approach in terms of overall system efficiency at multiple points in the task allocation problem space.

II. Method

The task allocation problem is set on a network of specialised agents with multi-component tasks initially arriving at any point in the network from an external user. In our case the agents represent satellites, the external user is a ground station, and the links are wireless communication channels, but agents could also be robots or supply depots, while the connections between them could represent communication paths such as wired links or roads.

Tasks consist of multiple *task components*, each of which need to be executed by an agent with the required skill; agents are specialists in a single component. In the fractionated satellite case, for example, we have one type of satellite that is responsible for communication with the ground station, another with imaging capabilities, while yet another type handles data storage and processing. The task allocation mechanism is responsible for deciding which agents should execute task components: this must be done in a manner that maximises successful task allocation while minimising communication. It must also be done in a way that is sensitive to network topology, e.g., excessive local allocation of tasks could exhaust the energy reserves of local nodes and thus disrupt future communication to more distant nodes.

We distinguish between two types of communication packets:

- *Negotiation packets* are small, and require little energy to be transmitted (c_{tx}).
- *Task transfer packets* represent a substantial amount of data that must be transferred (e.g., image data for processing on a remote node). The energy required to transfer a task, c_{tf} is a multiple of the transmission cost for negotiation packets, i.e., $c_{tf} = \alpha c_{tx}$.

A. Self-organising task allocation

Market mechanisms have become an increasingly popular way of dealing with complicated allocation problems. Since Smith's definition of the Contract Net protocol [14] more than 30 years ago, auctions have been applied to a range of fields, ranging from distributed computation [15], to mobile robotics [16], to job shop scheduling [17]. The ability to use only local information is especially suited to cases where global communication is either impossible or very expensive, which encourages the use a distributed auction, similar to [18] or [19].

We use a distributed auction, similar to [18] or [19] for outsourcing labour as inspiration for our self-organising control (SOC) mechanism (see [3] for a previous analysis of robustness in this model). Bid values are used to convey the suitability of available agents for a potential task allocation. The auction protocol for a task component is shown in Fig. 1. When an agent becomes aware of a task component that need to be allocated, it sends an announcement to all nodes in its *auction community*. We define the auction community to be all agents within the time-to-live range (d_{TTL}) of the auctioneer. Relevant nodes, i.e., those with the appropriate component specialization and enough energy, calculate bids that reflect their fitness: the bid value (B) takes the ratio of maximum (e_{max}) to remaining energy (e_{rem}) into account, as well as a scaling factor for the size of the task (z).

$$B = z\frac{e_{max}}{e_{rem}}$$

This cost function leads to inexpensive bids from under-utilised agents, while those that receive more allocations increase their bids as their available energy decreases. Agents do not try to win tasks by underbidding others — our objective is to control the system by transmitting the minimum information; we are not modelling bidding strategies for a competitive real-world market. Bids are routed back to the auctioneer along the path of the original auction announcement. At every agent along the return route bids are aggregated and the only best bid is returned, in a manner analogous to [20]. The use of bid aggregation results in a significant energy saving across the network because fewer packets are transmitted.

The auctioneer assigns the task component to the agent with the lowest bid and transmits an allocation-offer message. If winning agent wants to accept the offer, it returns an acknowledgement message, upon which the auctioneer transfers the task and payment. If, however, the agent has changed state since bidding by accepting another task component from a different source, it transmits a negative acknowledgement to the auctioneer, who will then repeat the auction. If no relevant agents exist within the auction community (e.g., due to several node failures) allocation will certainly fail. The likelihood of this happening is dependent on the size of the auction community which is in turn determined by d_{TTL}. In this sense d_{TTL} serves as a robustness parameter because it governs the system's resilience to node failure.

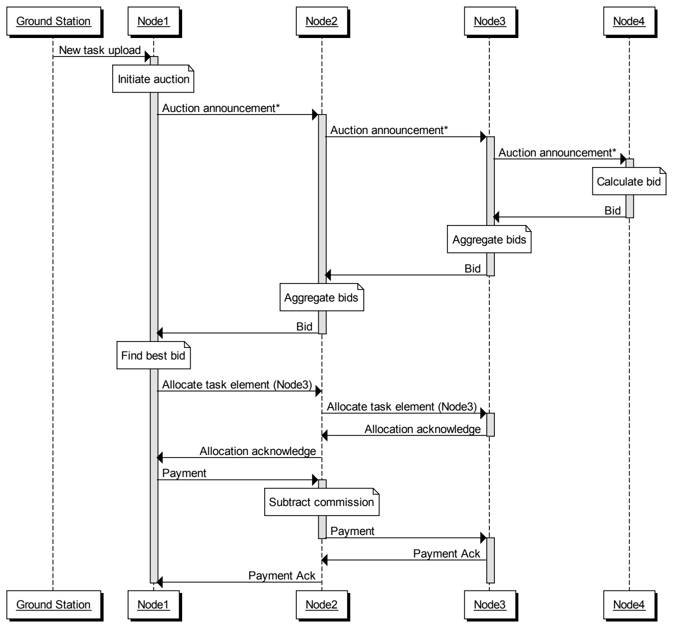

Fig. 1. Message sequence diagram describing the distributed task allocation flow. Node 1 acts as auctioneer, while node 3 is the successful bidder. The auction announcement messages are flooded thought the network, nodes capable executing the task respond with bids that convey their suitability. Bids are aggregated on the return path: only the best bid is forwarded. The task is allocated to the lowest bidder and payment and the task is transferred.

This strategy would work smoothly when a system contains a single auctioneer. However, multiple simultaneous auctioneers exist in our system: tasks can originate with any agent, multiple tasks will be in the system at any one time, and tasks will of course be passed around the system as each component is performed. With multiple auctioneers we potentially find conflicting allocation: the best available node will be allocated several different tasks concurrently. To remedy this we need a localising force that will counter the tendency to allocate all tasks in the system to the agent with the highest energy level. Thus, when an agent relays a bid to the auctioneer the bid is increased by a constant *commission* factor. Bids from far away will therefore appear more expensive to the auctioneer, causing it to favour nearby nodes.

When this process executes concurrently across the network, we find that it self-organises into zones of allocation around auctioneers "selling" similar types of task components. The sizes of the zones are determined by the distribution of bidding agents, their individual energy levels, the number of tasks injected into the system and the topology of the network. This auction mechanism is equivalent to a reverse, sealed-bid auction, but it is truly distributed. Not only do all agents have to act as temporary auctioneers, but all agents in the auction area help calculate the winner. Furthermore, agents do not maintain a model of the network beyond their immediate neighbours — routing occurs in an ad hoc manner through the auction mechanism itself.

B. Centralised control

In this section we develop a centralised allocation mechanism to compare SOC against. This centralised controller is designed to serve as a fair comparison: it is subject to the same transmission cost constraints as the SOC version, with reasonable fault tolerance mechanisms. The central controller uses global information to compute the best allocation: of particular interest in this study is the cost involved in building and maintaining the allocation model. As stated above, the objective is not to prove one approach superior to the other, but rather to find the regions of parameter space where either approach dominates.

The network consists of a manager agent that calculates allocations using a model of the network, and a number of worker agents that execute tasks on command from the manager. The workers have the same capabilities as for SOC: each has a special skill and limited energy, some of which will be spent on communication. The manager polls agents to keep track of the available energy of nodes, their respective skills and the connections between them. As the manager knows the effect of all communication and allocation in the network, it can accurately adjust its view of the network as it progressively allocates tasks. However, external influences on the system cannot be predicted and require detection. Workers do not maintain a map of the network: instead they only need to know how to reach the manager, who will in turn provide them with routing instructions for a task.

As with SOC, tasks are initially injected into the system by the ground station. Alternatively new task components appear in the system when the preceding task element has been completed. In both cases the relevant agent notifies the manager, who uses its model of the network to determine an allocation. The manager transmits the allocation information to the originating agent. The agent then proceeds to transfer the task along a route specified by the manager to the receiving agent. The receiving agent acknowledges receipt by transmitting a task-received message to the manager. The manager can now update its model to reflect the energy used in transmission and allocation, before relaying the acknowledgement to the originating node, thereby closing the allocation loop. If the originating node does not receive the allocation acknowledgement message in a predetermined time, it will repeat the allocation process. A message sequence diagram is shown in Fig. 2.

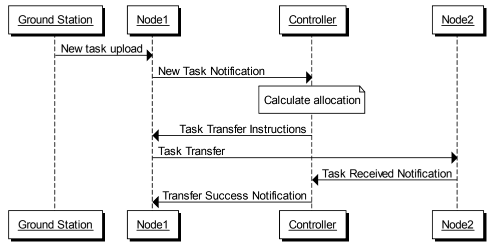

Fig. 2. Message sequence diagram describing the task allocation flow for central control. Nodes that relay messages are not shown. All messages are addressed.

The ability to allocate tasks successfully in a changing environment depends on the accuracy of the manager's network model and on the workers knowing the correct route to the manager. Both these factors are determined by the interval between status polling messages (τ_{status}) sent by the controller. These messages refresh workers' routes and update the manager's network model. With a large τ_{status} the manager is effectively implementing open-loop control, while smaller values of τ_{status} provide feedback resulting in a more accurate model, which also requires more communication. τ_{status} therefore acts as a robustness parameter, analogous to d_{TTL} for SOC.

C. Experimental setup

The test network initially consists of 225 agents in a 15 by 15 lattice topology. Agents can communicate directly only with their immediate neighbours to the north, south, east and west; communication to other agents must be relayed via these neighbours. Every agent has a skill or specialization randomly selected from the set of types $S = A, B, C, D, E$.

Two tasks are injected into the network every 100 time steps, over a total duration of 2000 time steps, for a total of 40 tasks. These tasks consist consist of five components, each of which requires one unit of energy to execute. Every task component is dependent on the successful allocation of the preceding component, as the causal dependencies in shown in Table I demonstrate. The execution of components is not tied to specific nodes: any node with the corresponding skill and sufficient energy is suitable.

Transmission cost was fixed at 0.01 units for a negotiation packet and 1 for a task transfer packet, i.e., the value of α was 100. Even though the basic model includes the possibility that agents will run out of energy, agents in the simulation runs described have effectively been supplied with infinite energy. This was done in order to isolate the effects of node failure from node exhaustion.

1) Node failure: An agent is removed from the network when it fails: not only can they not perform any work, but its connections with other nodes are broken, disrupting routing

TABLE I

COMPOUND TASK STRUCTURE USED IN EXPERIMENTS. THE EXECUTION OF TASK ELEMENTS (LEFT-HAND COLUMN) RESULTS IN ANOTHER TASK ELEMENT (RIGHT-HAND COLUMN) THAT MUST BE EXECUTED. TASKS ARE ONLY CONSIDERED COMPLETE IF ALL THE TASK ELEMENTS ARE SUCCESSFULLY EXECUTED.

$A \rightarrow$	B
$B \rightarrow$	C
$C \rightarrow$	D
$D \rightarrow$	E
$E \rightarrow$	.

and network topology. In this experiment an exponential distribution is used to model agent failures, where the shape parameter (λ) is the failure rate.

$$f(t, \lambda) = \begin{cases} \lambda e^{-\lambda t} & t \geq 0 \\ 0 & t < 0 \end{cases} \tag{1}$$

We vary λ from 0 for no failures, to 10^{-4} for a mean of 192 observed failures over a full run.

2) Robustness parameters: A higher failure rate requires greater robustness from the task allocation mechanism. With SOC this is achieved by modifying the size of the auction community. We therefore range d_{TTL} from 3 to 8, which results in a auction community size ranging from 24 up to 144, although communities limited by the edge of the network are smaller. Central allocation is more successful if the model of the network is frequently updated by decreasing the τ_{status} parameter. We explore a range of values from 10 to 200, as well as the case where only an initial update occurs and tasks are allocated in an open-loop fashion.

III. RESULTS

We want use the above task allocation mechanisms to find the regions in which they perform well, i.e., the role played by parameters such as group size, communication cost, and agent failure rates. To compare self-organising to centralised task allocation, we develop analytical descriptions of each approach under idealized circumstances. This is supplemented by simulation in which the relative performance of each method is compared by looking at the number of tasks successfully allocated, as well as the overall energy used. Finally we combine these approaches to return to our central question: what are the conditions under which each method will be preferred?

A. Analytic treatment of allocation cost

1) Self-organised task allocation: We start by finding a description for the total communication cost required to allocate a single task component. To derive an upper bound on cost we assume the most expensive topology for allocation: a linear network. From the message sequence diagram for SOC (Fig. 1) we see that, for a network with n agents, up to n auction-announcement messages will be broadcast. If all nodes place bids, this will result in n bid messages, because each node only forwards the best bid it receives. Assuming the worst case, i.e., that allocation goes to the most distant node, this will require n transmissions, for each of the allocation, task transferral/payment, and their respective acknowledgement messages. The worst case total transmission cost c_{auc} required for an auction is therefore:

$$c_{auc} = 5nc_{tx} + nc_{tf} \tag{2}$$
$$= 5nc_{tx} + \alpha nc_{tx} \tag{3}$$

Where

$$c_{tf} = \alpha c_{tx} \tag{4}$$

However, due to limited time-to-live (d_{TTL}) of messages, c_{auc} can be further constrained. With $d_{TTL} < n$, let the number of nodes that are within relay range of the auctioneer be n'. This is determined by the d_{TTL}, as well as the topology of the network. The worst case cost for a single auction as:

$$c_{auc} = 5n'c_{tx} + \alpha d_{TTL} c_{tx} \tag{5}$$

The first term is the negotiation overhead, while the second term describes the task transfer cost. The negotiation overhead is a linear function of the number of nodes within communication range of the auctioneer.

If t tasks of s components each and $c_{tf} = \alpha c_{tx}$, the total system cost over the duration of the experiment can be approximated as

$$c_{SOC} = tsc_{auc} \tag{6}$$
$$= tsc_{tx}(5n' + \alpha d_{TTL}) \tag{7}$$

From this expression we can see that the total cost of allocating tasks grows linearly with t, and linearly with n', which is in turn a function of d_{TTL}. Note that this expression is independent from n which means the allocation scheme will scale well. Communication cost c_{tx} has a linear relationship to c_{SOC}, making it a significant driver of overall system efficiency.

2) Central control: We can similarly find an expression for the worst-case cost of allocation for central allocation from Fig. 2. If t tasks of s components each are allocated, the allocation cost is described by:

$$c_{alloc} = ts(4c_{tx}n + c_{tf}n) \tag{8}$$
$$= tsc_{tx}n(4 + \alpha) \tag{9}$$

However, the centralised controller also requires status updates to maintain its model of the system:

$$c_{status} = \frac{c_{tx}n^2}{\tau_{status}} \tag{10}$$

where c_{status} is the total system expenditure on status updates, and τ_{status} is the status update interval.

The total communication expenditure over the duration (T) of the experiment is therefore

$$c_{CC} = c_{tx}\left(tsn(4 + \alpha) + \frac{Tn^2}{\tau_{status}}\right) \tag{11}$$

Note that c_{CC} is dominated by the n^2 term due to the status updates. This means that in systems with very large

n a great deal of energy will be expended on maintaining the manager's system model. In the case of open-loop control ($\tau_{status} \rightarrow \inf$), the allocation cost is linear with respect to all the parameters in the allocation term. As with SOC, the system is also sensitive to the value of c_{tx} — in fact it will be even more sensitive due to the linear relationship with system size (n) rather than the with the constant n'.

B. Simulation results

1) Task allocation success: The analytical description of the task allocation mechanism provides us with a description of the worst-case performance for a system without failure. To gauge the average performance and explore the effects of network volatility due to node failures, we simulate task allocation using the experimental setup described in Section III.

A contour plot of the number of tasks successfully completed is shown in Fig. 5 for SOC and Fig. 6 for central control. This is shown for a range of failure rates and a significant variation in the robustness parameter (d_{ttl} for SOC, τ_{status} for centralised control). Note the similarity in the shape of the landscapes formed by the allocation schemes. Both exhibit a strong deterioration as λ increases, which leads to a higher number of failures. The decrease in performance is more rapid for centralised control. A top to bottom deterioration is also visible — in both graphs the greatest robustness is shown at the top. We observe that for small values of d_{TTL} SOC fails to allocate successfully even when no failures occur. Centralised control is always successful in a network where no nodes fail.

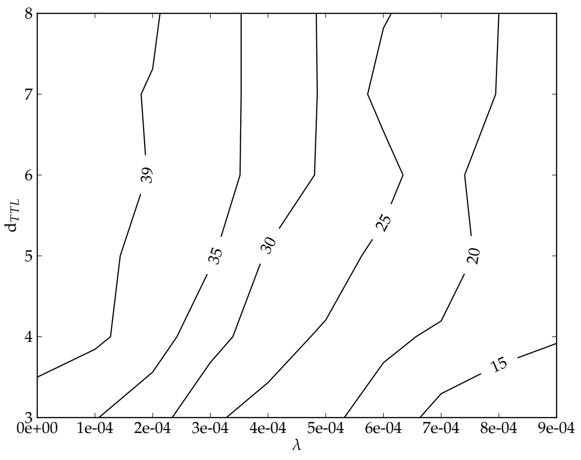

Fig. 3. Number of tasks successfully allocated using self-organising control by node failure rate (λ) and time-to-live range(d_{TTL}), i.e., the relevant robustness parameter. As failures increase, the number of tasks allocated decreases.

2) Energy consumption: Fig. 5 shows the total energy usage for a self-organising system over the same values of node failure rate and time-to-live range as shown in Fig. 3. Note that for a given failure rate, increasing robustness by increasing d_{TTL} results in higher energy usage. Fig. 6 shows

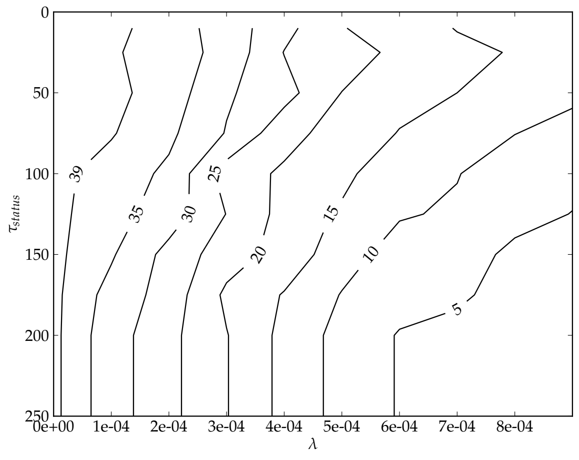

Fig. 4. Number of tasks allocated using central control by node failure rate (λ) and the status update interval (τ_{status}), the relevant robustness parameter. Note the y-axis is inverted to have the highest robustness at the top.

the total energy usage for a centrally controlled system over the same values of node failure rate and status update interval as shown in Fig. 4. Again, for a given failure rate increasing robustness results in higher energy usage. In both of these figures it may appear at first glance that a higher failure rate is good news. However, this decrease in energy usage only occurs because fewer tasks are being successfully allocated. The main distinction between the two figures is that high levels of robustness in the centralised control case are associated with extreme levels of energy consumption, nearly an order of magnitude greater than in the self-organising case.

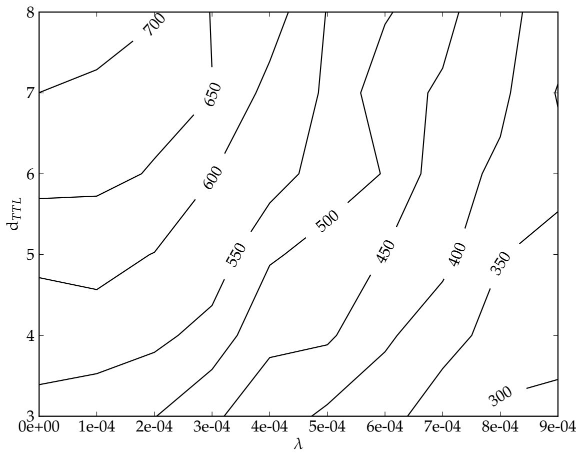

Fig. 5. Energy used in allocating tasks using self-organising control (Fig. 3) by node failure rate (λ) and time-to-live range(d_{TTL}), i.e., the relevant robustness parameter.

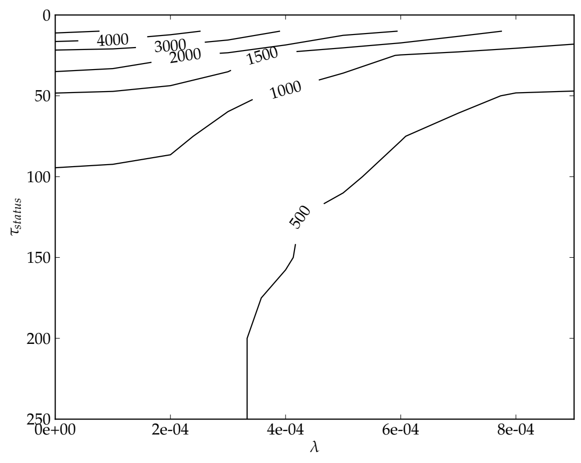

Fig. 6. Energy used in allocating tasks using central control (Fig. 4) by node failure rate (λ) and the status update interval (τ_{status}), the relevant robustness parameter. Note the y-axis is inverted to have the highest robustness at the top.

C. *Conditions favouring either approach*

When designing a task allocation system, some of the above parameters are imposed on the system designer, while others have to be selected. The cumulative effect of these parameters will determine whether a self-organising or centralised approach is to be preferred.

For example, suppose the above system is implemented in a space where $\lambda = 1.5e - 04$ and we require an average allocation success of 97.5% (39 out of 40 tasks). Referring to the task allocation graphs (Fig. 3 and 4), we find that an SOC approach requires $d_{TTL} \geq 5$ while the centralised approach requires $\tau_{status} \leq 50$. On the energy consumption graphs (Fig. 5 and 6), these points show expected energy consumption of 610 units for SOC and 1500 for centralised allocation. In this particular case, SOC is clearly the preferred approach.

However, this is not always so: we can use Eq. 7 and Eq. 11 to estimate the network size for which centralised control and self-organised control will be equivalent in performance. It follows that for smaller networks centralised control will be favoured. Note that this is an approximate threshold based on the expressions we derived for the upper bounds of allocation cost. We refine the upper bound given by Eq.11 to incorporate the effect of the lattice topology. The maximum distance between the manager node and any worker node is $\sqrt{n}$, while the maximum transfer distance is $2\sqrt{n}$. This gives:

$$c_{CC} = c_{tx}\left(ts\sqrt{n}(4+2\alpha) + \frac{Tn^2}{\tau_{status}}\right) \tag{12}$$

If all other parameters remain fixed as for the experiment above, then we find the intercept of Eq. 7 and Eq. 12 at 16 nodes. The effect of the topology is also demonstrated in this example: the root n term is determined by the lattice topology. For alternative topologies, e.g., scale-free networks, this factor would be different, shifting the point of ambivalence.

For both allocation approaches, total energy usage scales linearly with communication cost (c_{tx}). This means that there will be no point of ambivalence with respect to this dimension, except when $c_{tx} = 0$.

IV. Discussion

Central control is traditionally used for controlling small systems, while self-organising approaches have been proposed for large systems. In this paper we explored the parameters that determine whether we should regard a system as "small", or suited to central control; or "large", where distributed approaches will be better. The results demonstrate that both self-organising control and centralised control have a place in the tool set of a system designer. Although we focussed on self-organisation, the principle of identifying the most suitable control methodology is applicable to the wider self-* community.

Simulation was used to map the trade-off between successful task allocation and energy consumption for a range of failure rates. This allows for unbiased selection of the most efficient allocation approach. In addition, we demonstrated a method for estimating the boundaries in parameter space that divide regions where one architecture or the other is the most attractive. Although this method was applied to a specific problem, it can be adapted to serve the wider self-* community.

Which parameters turned out to be the most important in determining the appropriateness of each approach? The size of the network is a critical factor, because the allocation costs of centrally controlled systems scale with n^2. In our example system, we found that 16 agents was the point of ambivalence between self-organised and centralised approaches. However, there is no suggestion that 16 is any sort of magic number dividing small from large systems. Our point is only to demonstrate that it is possible to find this threshold for any given problem.

Network volatility is also an important parameter, because the robustness measures required to deal with high volatility consume a significant amount of energy. This is especially true for centrally controlled systems. On the other hand, if the system is very stable, e.g., agents are extremely reliable, robustness measures are not required which changes the point of ambivalence dramatically: centrally controlled systems become much more competitive. This is not only applicable to agent failures, but also to changes in topology, or nodes changing their capabilities in a manner that is unpredictable to the manager node.

Whether communication cost should be regarded as significant depends on the relative size of tasks and communication costs. If the total communication cost is orders of magnitude less than task sizes, or communicating does not directly impact the ability of an agent to do work, the choice of allocation system will be determined by other factors, such as implementation and verification effort. When communication comes at some cost we will prefer one approach to the other, but increases in communication cost will never lead to a switch in the preferred option.

Several limitations of this study directly suggest future extensions. One limitation was that the energy expended on communication was the primary limiting factor in determining performance. Many systems have other factors that constrain performance, such as bandwidth, processing power, or physical limitations on the speed of task transfer. Future work should include similar analyses of these factors. Task allocation problems become harder when multi-component tasks with logical interdependencies are involved, as presented by [21]. The current model uses only a single type of sequential task structure: the ability of the system to cope with variation in this regard will be investigated further. Additionally the response of the allocation methods to heavier task workloads will be of interest. Finally, our simulations considered only a regular lattice as a network topology. Our analytical results have highlighted the likely effects of alternative topologies, but this remains to be verified.

In closing, we want to reiterate that it is only by understanding the compounding influences of all the relevant parameters that we can make unbiased design decisions on the best control strategy for a particular job.

References

[1] S. Bullock and D. Cliff, "Complexity and emergent behaviour in ict systems," HP Labs, Tech. Rep. HPL-2004-187, 2004.

[2] D. Cliff and J. Bruten, "Animat market — trading interactions as collective social adaptive behavior," *Adaptive Behavior*, vol. 7, no. 3-4, pp. 385–414, January 1999.

[3] J. van der Horst, J. Noble, and A. Tatnall, "Robustness of market-based task allocation in a distributed satellite system," in *Advances in Artificial Life: Tenth European Conference on Artificial Life (ECAL '09) (in press)*. Springer, 2009.

[4] O. Brown and P. Eremenko, "Fractionated space architectures: A vision for responsive space," in *Responsive Space 4*, 2006.

[5] D. J. Barnhart, T. Vladimirova, and M. N. Sweeting, "Very-small-satellite design for distributed space missions," *Journal of Spacecraft and Rockets*, vol. 44, no. 6, pp. 1294–1306, 2007.

[6] T. L. Casavant and J. G. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems," *IEEE Transactions on Software Engineering*, vol. 14, no. 2, pp. 141–154, February 1988.

[7] G. Dudek, M. R. M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for multi-agent robotics," *Autonomous Robots*, vol. 3, no. 4, pp. 375–397, 1996.

[8] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, September 2004.

[9] P. W. Anderson, "More is different," *Science*, vol. 177, no. 4047, pp. 393–396, August 1972.

[10] I. Bekey, "Phase I Study: Extremely large swarm array of picosats for microwave / RF earth sensing, radiometry and mapping," NASA Institute of Advanced Concepts (NIAC), Tech. Rep., April 2005.

[11] T. Eymann, M. Reinicke, O. Ardaiz, P. Artigas, Díaz, F. Freitag, R. Messeguer, L. Navarro, D. Royo, and K. Sanjeevan, "Decentralized vs. centralized economic coordination of resource allocation in grids," *Grid Computing*, pp. 9–16, 2004.

[12] F. Ygge and H. Akkermans, "Decentralized markets versus central control: A comparative study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 301–333, 1999.

[13] M. Resnick, *Turtles, termites, and traffic jams : explorations in massively parallel microworlds*. MIT Press, March 1997.

[14] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions on Computers*, vol. 29, no. 12, pp. 1104–1113, 1980.

[15] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and S. W. Stornetta, "Spawn: A distributed computational economy," *IEEE Transactions on Software Engineering*, vol. 18, no. 2, pp. 103–117, February 1992.

[16] B. P. Gerkey and M. J. Matarić, "Sold!: auction methods for multirobot coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 758–768, 2002.

[17] W. Shen, "Distributed manufacturing scheduling using intelligent agents," *Intelligent Systems, IEEE*, vol. 17, no. 1, pp. 88–94, Jan 2002.

[18] P. B. Sujit and R. Beard, "Multiple MAV task allocation using distributed auctions," in *AIAA Guidance, Navigation and Control Conference and Exhibit*. AIAA, August 2007.

[19] R. Zlot and A. Stentz, "Market-based multirobot coordination for complex tasks," *The International Journal of Robotics Research*, vol. 25, no. 1, pp. 73–101, January 2006.

[20] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, February 2003.

[21] R. M. Zlot, "An auction-based approach to complex task allocation for multirobot teams," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, December 2006.