Deliverable D3.2

Report on rule-based modules for unit differentiation using cross-inhibition and/or resource competition

Deployment and maintenance of a distributed service framework through component self-aggregation and biologically-inspired differentiation

Status and Version:	Final Version
Distribution:	Public
Author(s):	WP3 Partners
Checked by:	F. Zambonelli, A. Manzalini

Table of Contents

Document History

Version	Date	Comment
0.1	29/06/2007	First version.
1.0	11/07/2007	Final deliverable
2.0	31/03/2008	Revised version

1 Introduction

Orchestrating the deployment and maintenance of a complex distributed application requiring frequent interactions between its many constituents is a well-identified but difficult and as yet unsolved problem in highly dynamic environments. New components may appear and old ones become obsolete while new usage patterns develop and the demand for individual or aggregated services fluctuates over time, making the whole system partly unpredictable and calling for high levels of adaptability.

This is in contrast with traditional and proprietary "top-down" applications, which are amenable to a "stop and restart" approach to reorganisation, patching and/or upgrade. Indeed, with the current trend towards largely open service platforms, involving multiple providers, software developers and hardware manufacturers, this management strategy has already started to crumble.

In recent work, we have shown that decentralised mechanisms could enable the emergence of efficient collaborative overlays through self-organised aggregation (Saffre et al., in press) and facilitate division of labour in a fixed-topology network through co-ordinated individual specialisation (Saffre et al., 2006). In this paper, we seek to combine both approaches to create a highly plastic distributed application, able to spontaneously react to various local or global changes affecting its fully distributed execution environment.

2 Load-balancing and biologically-inspired differentiation

2.1 The limits of load-balancing through overlay management

In the previous deliverable, D3.1, we have successfully demonstrated that a pre-existing heterogeneous workload could be efficiently processed by a community of pre-specialised peers provided that they were able to reorganise their relationships (i.e. the structure of the overlay) so as to create the right conditions for co-operation. Although these results could in principle be generalised to a continuous inflow of new service requests (as opposed to a large but fixed initial workload to be processed), the rewiring algorithms involved would not be able to correct a hypothetical imbalance between offer and demand in the system *as a whole*. Our solution was effectively designed to solve the problem of localised imbalance, by allowing components faced with an inadequate workload, either qualitatively (i.e. "wrong service type") or quantitatively (i.e. "overload"), to identify and establish one or more partnerships with (an) adequate "subcontractor(s)".

In the case of a global mismatch between the offer and demand for one particular service however, rewiring of the overlay is useless. For instance, if fewer resources than necessary are allocated system-wide, requests will accumulate indefinitely and average queuing time will keep increasing. In the opposite situation (over-provisioning), underutilised resources will be frequently idling. In neither case is the reorganisation of co-operative relationships between components of any help.

This is illustrated by the example shown in figure 1, where demand for the 4 represented services is different but the resources are equally distributed (similar numbers of each component have been deployed).

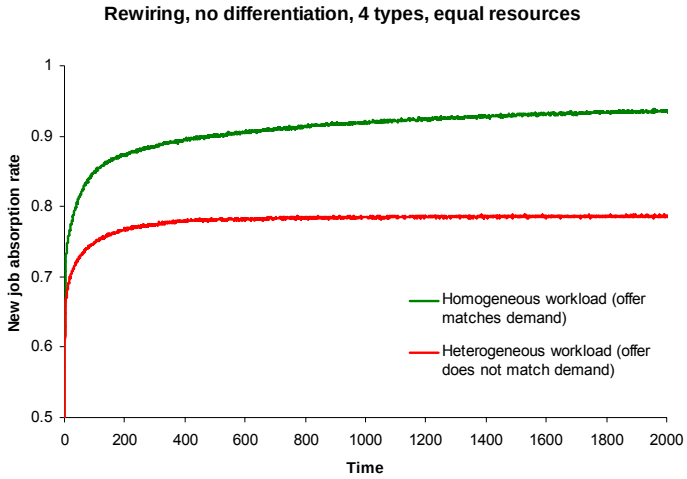

Fig. 1: Effect of resource "misallocation" on overall processing capability (job absorption rate). One hundred independent realisations, 1024 nodes, random graph topology (average node degree of 4). In the heterogeneous case, workload for job type 1 is (statistically) twice that for job type 2, which is twice that for job type 3 etc. Total workload (overall demand, summed over all types) is identical for both numerical experiments.

In order for the system to be able to adapt to a change in the *overall* demand (assuming that the total processing capability is sufficient), it is necessary that resources can be repurposed. This problem is similar to that of transferring servers between hosted applications in a data centre in response to fluctuations in the relative workload, with the key difference that it has to be achieved without centralised scheduling or global information on the existing balance between offer and demand (no centralised accounting of resources).

We refer to this "self-(re)allocation" of individual resources based on locally available information and local interactions as "differentiation", because it shares many characteristics with differentiation in biological morphogenesis.

2.2 Fundamental dynamics of differentiation

2.2.1 Concepts and methodology

The question of differentiation at all levels from cells to organisms, social systems or ecosystems, is ubiquitous in biology. This vast issue can be subdivided according to the context of differentiation. Here we focus on systems where identical (or nearly identical) individuals differentiate through interactions among themselves and/or with their environment, a problem that has been addressed many times in theoretical biology. One current vision of such differentiation phenomena is that the system (or the individuals) can

exist in multiple states between which individuals (or the whole system) can switch. A key discussion in biology revolves around the problem of identifying the necessary conditions to obtain dynamical and emergent differentiation properties.

Biological complex systems can often be described as an interaction graph defined as a finite oriented graph including signs for every edge. The vertices correspond to individual components of the system (genes, cells, individuals, species…) and the edge between A and B is positively (or negatively) labeled when A activates (or represses) the activity of B. However, edges can also be positive or negative depending on some properties of the vertices and depending on the context. A conjecture formulated by R. Thomas (Thomas, 1981) states that a necessary condition for multistability is that the graph (G) has a circuit (C) which is positive. C is positive when the product of the signs of the edges of C is positive, at least in one area of the phase space of the corresponding dynamical system. This conjecture has been proven mathematically in several special cases and in the general case (Soulé, 2003, Cinquin & Demongeot, 2002; Gouzé 1998; Plathe, Mestl & Omholt, 1995; Snoussi, 1998). Besides these structural considerations on the nature of interaction graphs, the general framework used both quantitatively and qualitatively is dynamical system theory.

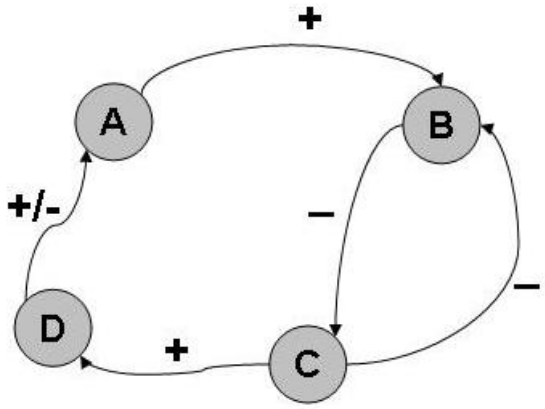

Fig. 2: Example of a small interaction graph between agents (nodes A-D). The oriented edges correspond to the nature of their interactions i.e. positive (activation) or negative (inhibition).

A brief outline the mathematical framework used in the present work (i.e. dynamical system theory and its corresponding terminology) is provided below. Given a positive integer *m*, we define a differentiable map *F*: R^m-> R^m and the corresponding dynamical system:

$$\frac{dx}{dt} = F(x)$$

Where $x=(x_1(t), \ldots x_m(t))$ is a trajectory in the *m*-dimensional Euclidian space. The interaction graph *G(x)* of *F* at point *x* is the oriented graph with *{1,…,m}* as vertices and such that there is a positive (or negative) arrow from *j* to *i* if and only if the partial derivative

$$\frac{\partial f_i}{\partial x_j}(x)$$

is positive (or negative). Each edge in *G(x)* is thus oriented and endowed with a sign. The variable *x* is viewed as the phase space location of the graph *G(x)*. A circuit in the graph *G(x)* is a sequence of distinct vertices $i_1, i_2, \ldots, i_k$ such that there is an edge from i_α to $i_{\alpha+1}$, with $1 <= \alpha <= k-1$, and from i_k to *i*. The sign of a circuit is the product of the signs of it edges. A circuit is thus determined by a set of non zero coefficients in the jacobian matrix

$$J(x) = \left(\frac{\partial f_i}{\partial x_j}(x) \right)$$

the rows and columns of which are in cyclic permutation. Its sign is the sign of the product of these coefficients. Several circuits are defined as disjoint when they do not share any vertex. Following Thomas and Kaufman we present some definitions on which we can then state their conjectures for the necessary conditions to have multistability in the system. A **nucleus** is a union of one or more disjoint circuits which involve all the vertices of *G(x)* (also defined as "Hamiltonian hooping" in Soulé 2003). The sign of a nucleus is $(-1)^{p+1}$ with *p* being the number of positive circuits in the nucleus (Eisenfeld, De Lisi, 1985).

Based on the analysis of this type of small graphs a number of conjectures have been formulated to predict the existence of multistationarity, and hence the possibility for the system to present interesting properties such as memory effects or differentiation.

Conjecture 1 (formulated by Thomas in genetics, proved mathematically by Soulé, 2003): The presence of a positive circuit in a region of the phase space is a necessary condition for multistationarity.

This type of results can be taken as design principles for self-organized properties in artificial systems (even in artificial biological systems). The design of dynamical, emergent differentiation in a population of initially identical (or almost identical) individuals requires building at least one positive circuit in the interaction graph between individuals and their environment. It is these principles that guided us qualitatively or quantitatively when designing the algorithms for self-organised differentiation developed in WP3.

In this framework, without even writing the equation, one can start to ponder the type of behaviour that will emerge in the systems solely by considering the interaction graph. Although in general, models based on differential equations rapidly become intractable for realistic complex systems, this framework is a good starting point for engineering emergent properties in artificial systems. The design process can fruitfully be guided by toy models or a simplified description of the system (Saffre and Inglesby, 2007).

However, today, there is no systematic procedure or methodology to effectively design an artificial system from model to implementation. This is partly due to the nature of complex systems, which implies a multi-level and multi-framework description. The main difficulty is often to cast the problem in terms of attractors of a dynamical system. Attractors correspond to stable states that will be reached after a transient time, whatever the initial conditions. Attractors are steady states (with various dynamical properties like hysteretic bistability or excitability), limit cycles (encompassing a wide variety of nonlinear oscillations ranging from simple to complex) or strange attractors (deterministic chaos).

As these attractors represent the predictable outcome of system dynamics, the associated interaction graph must be designed in such a way that they correspond to the desired features of the engineered system. The ideal design methodology is therefore to describe the target system state(s) as a type of attractor, design suitable interaction graphs to produce such attractors, write the corresponding mathematical model taking into account realistic constraints and, finally, make the transition from modeling to implementation.

2.2.2 Basic differentiation model

2.2.2.1 Analytical model version I

We formulate a basic model for differentiation, featuring a set of processing units (agents or ACEs) and a set of tasks to be performed (jobs or requests). The dynamics of such a system can be described using a simple set of differential equations:

$$\frac{dx_i}{dt} = \alpha - \beta X_i \frac{x_i}{\varepsilon + x_i} \qquad i = 1,..., N$$

$$\frac{dX_i}{dt} = \sum_{j=1}^{N} \gamma (X_j \frac{x_i^n}{\sum_{k=1}^{N} x_k^n} - X_i \frac{x_j^n}{\sum_{k=1}^{N} x_k^n}) = \gamma S \frac{x_i^n}{\sum_{k=1}^{N} x_k^n} - \gamma X_i$$

where X_i is the number of agents performing task *i* and x_i is the number of task *i* (or job) to be performed. *N* is the total number of the different tasks and *S* the total number of agents.

The rate of task performing per agent is:

$$\frac{\beta x_i}{\varepsilon + x_i}$$

such that β is the maximum achievable processing rate per agent.

The probability for an agent to adopt state *i* is given by:

$$P_i = \frac{x_i^n}{\sum_{k=1}^{N} x_k^n}$$

This model exhibits only one stationary state $(\frac{dx_i}{dt} = 0, \frac{dX_i}{dt} = 0)$, corresponding to:

$$\alpha - \beta X_i \frac{x_i}{\varepsilon + x_i} = 0$$

$$\gamma S \frac{x_i^n}{\sum_{k=1}^{N} x_k^n} - \gamma X_i = 0$$

Which, in the case where all jobs arrive at the same rate (α is identical for all tasks) translates into:

$$x_1 = x_2 = ... = x_N = \frac{\alpha N \varepsilon}{(\beta S - \alpha N)}$$

$$X_1 = X_2 = ... = X_N = \frac{S}{N}$$

Note that this stationary state only exists if: $\beta S - \alpha N > 0$, which simply means that, if the rate of arrival of new tasks (α) is greater than the total processing capability (individual rate of execution multiplied by the number of agents), jobs keep accumulating in the system.

Stability analysis of the stationary state:

$$\frac{d\delta x_i}{dt} = -\beta \delta X_i \frac{x_i}{\varepsilon + x_i} - \beta X_i \frac{e}{(\varepsilon + x_i)^2} \delta x_i$$

$$\frac{d\delta X_i}{dt} = -\gamma S \frac{n x^{-1}}{N^2} \sum_{\substack{k=1 \\ k \neq i}}^{N} \delta x_k + \frac{\gamma S n x^{-1} (N-1)}{N^2} \delta x_i - \gamma \delta X_i$$

$$\frac{d\delta x_i}{dt} = -\delta X_i \frac{\alpha S}{N} - \frac{(\beta S - \alpha N)}{(\beta S \varepsilon) N} \delta x_i = -A \delta X_i - B \delta x_i$$

$$\frac{d\delta X_i}{dt} = -\frac{\gamma S n (\beta S - \alpha N)}{\alpha \varepsilon N^3} \sum_{\substack{k=1 \\ k \neq i}}^{N} \delta x_k + \frac{\gamma S n (\beta S - \alpha N)(N-1)}{\alpha \varepsilon N^3} \delta x_i - \gamma \delta X_i$$

$$\frac{d\delta X_i}{dt} = -C \sum_{\substack{k=1 \\ k \neq i}}^{N} \delta x_k + C(N-1) \delta x_i - \gamma \delta X_i$$

In case of two tasks (N=2)

$$\begin{vmatrix} w+B & 0 & A \\ 0 & w+B & 0 \\ -C & C & w+\gamma \end{vmatrix} = 0$$

$$(w+B)(w+B)(w+\gamma) + AC(w+B) = 0$$

$$(w+B)(w+\gamma) + AC = 0$$

$$w = -B$$

$$w = 0.5(-(B+\gamma) \pm ((B+\gamma)^2 - 4(B\gamma + AC)))$$

with

$$A = \frac{\alpha S}{2}; B = \frac{(\beta S - 2\alpha)}{2\beta S \varepsilon}; C = \frac{\gamma S n (\beta S - 2\alpha)}{8\alpha\varepsilon}$$

The stationary state is always stable. It may exhibit oscillations that are always stable if $B^2 + \gamma^2 - 2B\gamma - 4AC < 0$. When oscillations are not observed, the stationary state is a stable node.

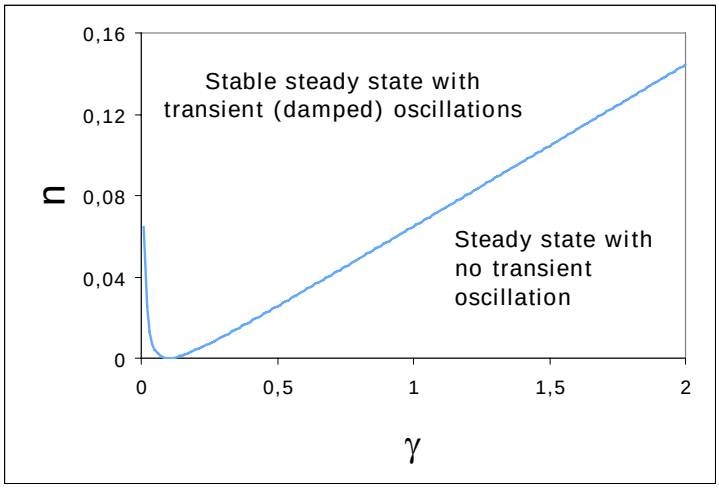

Fig. 3. Parameter space showing the zone with damped oscillations and without oscillations for γ and n. α = 1, β= 0.25, S=10, ε=1.

2.2.2.2 Analytical model version II

An alternative version featuring similar properties is:

$$\frac{dx_i}{dt} = \alpha - \beta X_i \frac{x_i}{\varepsilon + x_i} \qquad i = 1,\ldots,N$$

$$\frac{dX_i}{dt} = -\gamma X_i \left(1 - \frac{x_i^n}{\sum_{k=1}^{N} x_k^n}\right) + \frac{1}{N-1} \sum_{\substack{j=1, \\ j \neq i}}^{N} \gamma X_j \left(1 - \frac{x_j^n}{\sum_{k=1}^{N} x_k^n}\right)$$

The difference with model I being that the probability for an agent of type *i* to leave its current state is now:

$$L_i = 1 - \frac{x_i^n}{\sum_{k=1}^{N} x_k^n}$$

The rate at which agents "electing" to change type adopt any of the other (*N*-1) available states is still given by the same expression as in model I. Model II also exhibits only one stationary state $(\frac{dx_i}{dt} = 0, \frac{dX_i}{dt} = 0)$ and is basically undistinguishable from Model I in terms of its properties (stable states and trajectories).

2.2.3 Numerical results for the mean-field model:

In order to confirm, visualise and expand the analytical results presented in the previous section, we developed a numerical equivalent (discrete time-step deterministic simulation) of the differential equation-based model. We confirmed the presence of dampened oscillations when the system starts far from equilibrium (i.e. mismatch between the frequency of different job types and of the corresponding specialised agents), Figure 4 shows an example of such behaviour, for the case in which there are three job types, all equally represented.

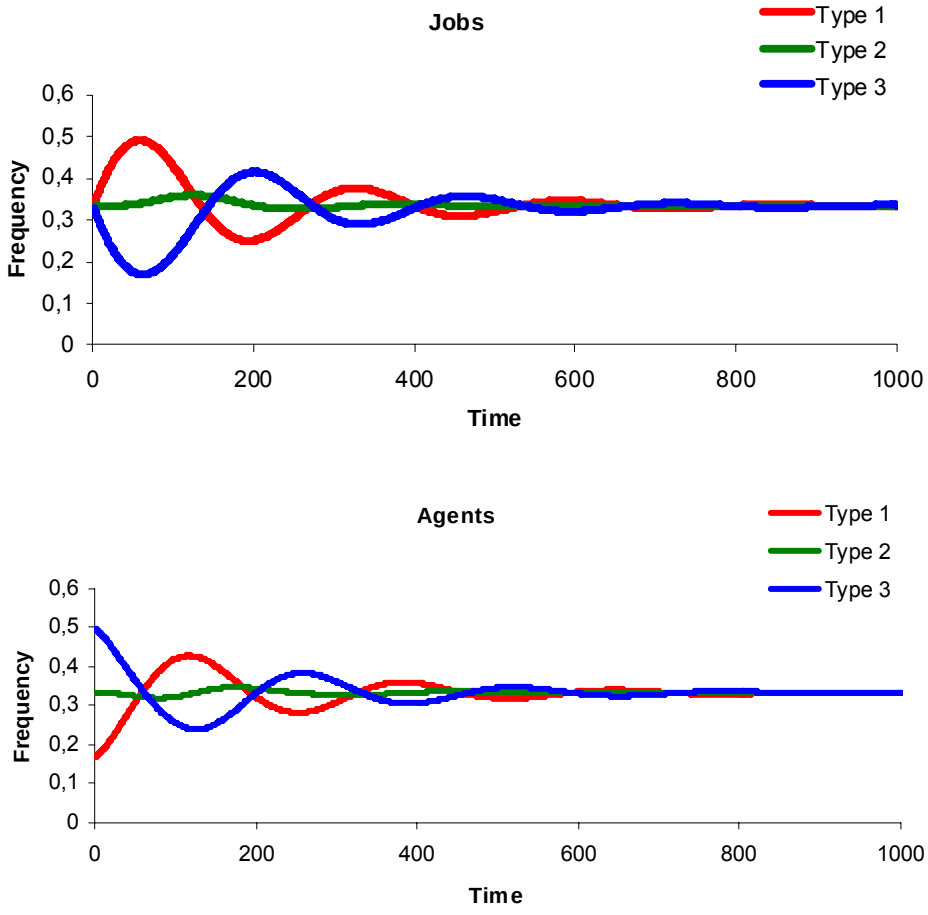

Figure 4. The frequency of jobs (upper panel) and agents (lower panel) over time, when the three job types arrive at equal rates but the initial frequencies of agents are far from equal (see frequencies at time 0 in lower panel).

We also used the deterministic simulation engine to conduct a preliminary study of the system's behaviour when job frequency (i.e. the structure of the demand) varies over time. The results confirmed the analytical prediction that after a while (and possibly further oscillations), the system would re-stabilise at the new optimal solution (i.e. the distribution of specialist agents exactly reflecting the frequency of the corresponding jobs). Figure 5 shows the early stages of the transition between two equilibrium states, after, at time t =

500, job type 1 and 3 switch from representing 1/3rd of the demand each to representing 1 half and 1/6th respectively.

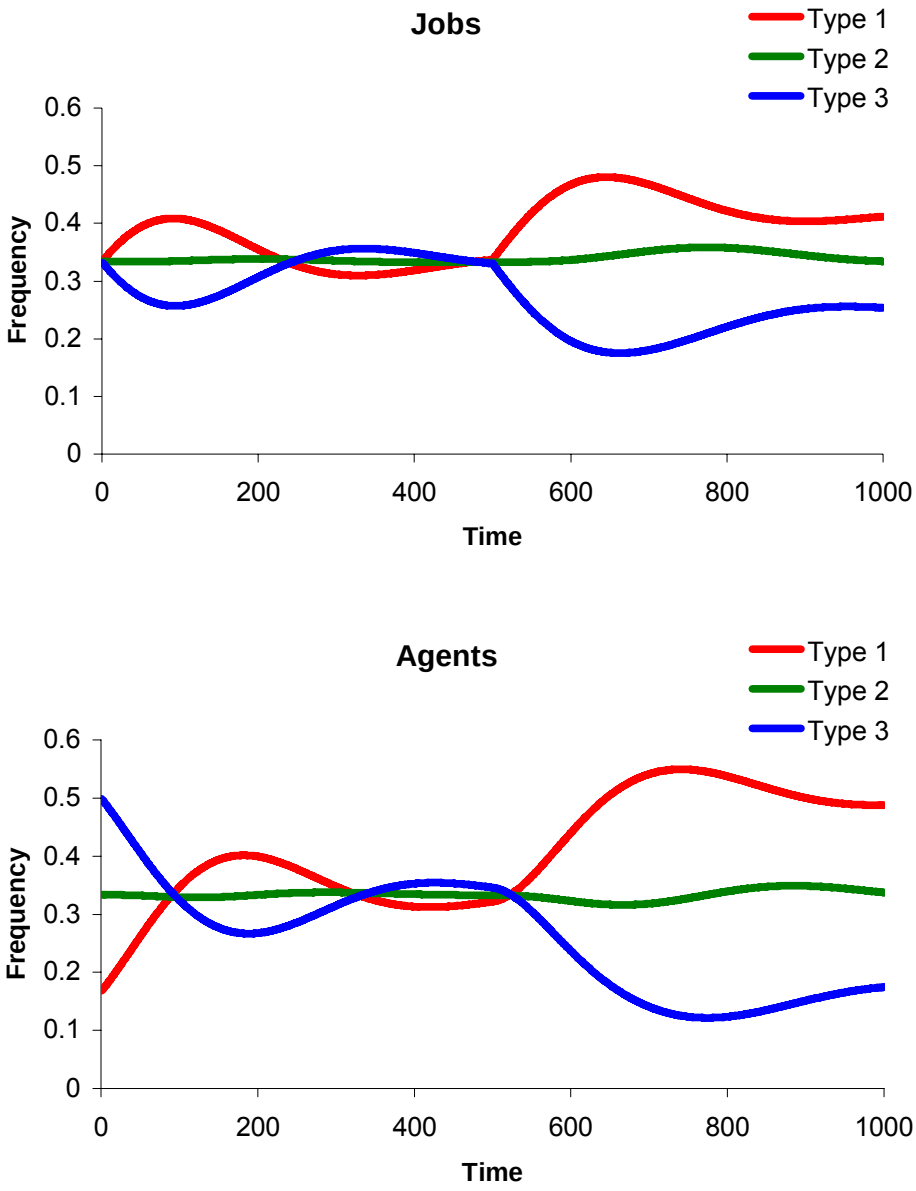

Figure 5. The frequency of jobs (upper panel) and agents (lower panel) over time, when the three job types arrive in the ratio 2:2:2 from time 0 to 499 but then abruptly change to a ratio of 3:2:1 at time 500.

2.3 Individual node capabilities

In order to be able to usefully repurpose themselves, resources have to make informed decisions, based on their knowledge of the local demand. We hypothesised that nodes would keep a record of the type of requests reaching them over a sliding window. This would allow them to estimate the relative frequency of different request types, which would approximate the model described in section 2.2.2 in which the system-wide distribution of jobs (x) is an explicit variable. Individual resources can then use this information to decide which type of job to specialise into (presumably, this means loading the corresponding software components).

At present, nodes compute the probability of "choosing" a given type using the perceived distribution of the demand, using equation:

$$P(x_i) = \frac{\gamma x_i^n}{\sum_{k=1}^{N} x_k^n}$$

Where x_i is the number of encounters with job type i over the sliding window, N is the diversity (number of job *types* actually encountered), γ is a non-specific differentiation rate and n is a nonlinearity parameter. The width of the sliding window (memory) is an implicit parameter. The wider it is, the more precise and accurate the estimate of the relative frequency of the different job types will be (assuming that the demand doesn't fluctuate over a similar time-scale).

Cooperation is supported by a similar "subcontracting" process to the one described in deliverable D3.1. On every time step, every node is allowed to send one job request to each of its first neighbours in the overlay, where it will have a chance of being processed (assuming the types match) on the following time-step. There are currently two versions of this delegation process:

- **Random diffusion:** a randomly selected job is sent to every neighbour. This "unrealistic" version was used for benchmarking and for comparison with the "mean-field" analytical results.
- **Targeted diffusion:** if possible (i.e. if the corresponding local queue is not empty), one job of the correct type is sent to every neighbour. However, if no jobs of the correct type are available, a randomly selected request is sent instead. This is designed to allow jobs for which the right type of neighbour is unavailable to eventually "percolate" through successive redirections.

As in deliverable D3.1, nodes are also allowed to rewire their connections, using the "on-demand" algorithm with specific neighbour type request. In this case, the probability of requesting a given type of neighbour is a linear function of the relative length of the corresponding local queue. It should be noted that, although the two variables are not independent, the relative length of a given local queue is *not* identical to the frequency of the corresponding job type in the sliding window.

2.4 Results

2.4.1 Homogeneous Load, Random Diffusion

The first batch of results uses a constant (per node, per time step) addition to the job load. Each node gets 4 new jobs in each timestep. This load is split equally among a number of types (which varies in the range [2,4,8,16] in different experimental runs). We call this 'homogeneous load'.

Two initial topologies are used for different runs: random graph and regular torus. In all runs there are initially 1024 nodes and 2048 links. In the random graph each link is initially assigned to two nodes chosen drawn at random, without replacement, from the set of all nodes. When all links are assigned, any nodes without any links are deleted. In the regular torus every node has exactly 4 links, 1 to each of its four 'neighbours' in the lattice.

In all runs there are initially equal numbers of nodes of all types (reminder: by definition, a node of type X will only directly process jobs of type X).

In the torus topology the types are assigned in a regular fashion, so if a node is of type 0, its neighbour to the right will be of type 1, and that node's neighbour to the right will be of type 2 and so on.

First we examine the 'null' situation where the nodes are allowed to neither change their type (it is this change of type which we refer to as 'differentiation') nor change their links ('rewiring').

Figure 6 shows how excess jobs accumulate in the system.

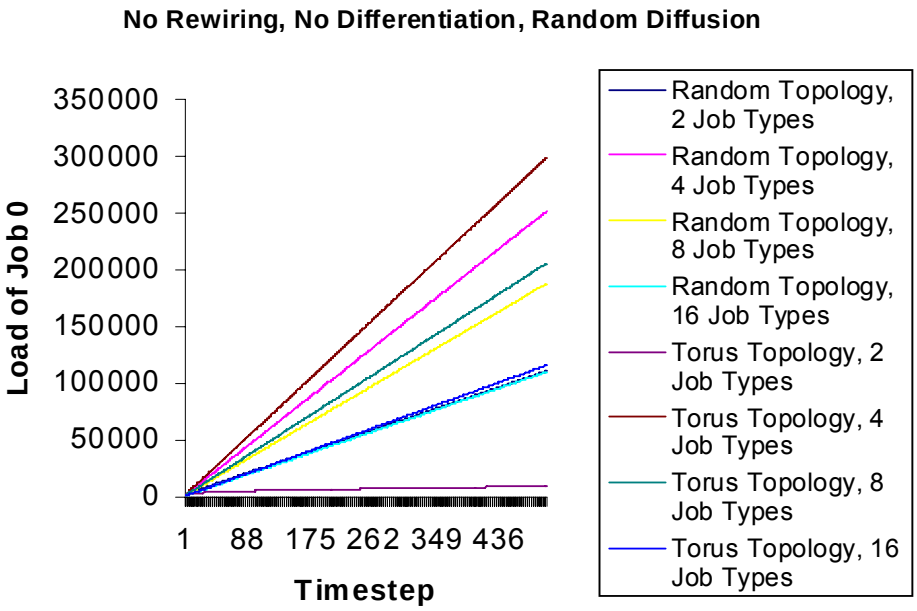

Fig 6 The number of jobs of a single type queuing in the entire network. Curves are shown for both topologies tested, and for 2, 4, 8 and 16 different job types.

The results are somewhat hard to interpret from these curves, since they relate to a single job type and absolute numbers of those jobs. Focusing on a single job type makes interpretation harder because as the number of different types increases the absolute number of each of those types of jobs goes down (since the total number of all types of jobs added to each node in each timestep is always 4, so in a run with two types of job a node expects to get an average of two type 0 and two type 1 jobs each timestep. In a run with 4 types of job a node expects one of each type on average.) Figure 7 shows the rate of accumulation of all jobs once the network has reached steady state. Note that the maximum rate at which jobs could accumulate is 4096. This would be the case if no jobs are processed at all in the maximum network size of 1024 nodes (each receiving 4 jobs per timestep). Clearly the minimum steady accumulation rate is zero, where as many jobs are processed as are arriving.

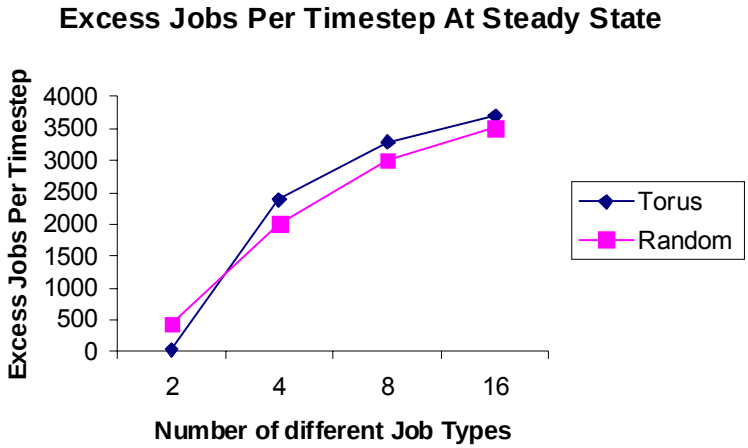

Fig 7 When the system has reached steady state (the gradient of the curve in Fig 6 is no longer changing) how many more jobs are added in every timestep than are processed?

From the figure we see that as the number of types rises the ability of the system to cope with the imposed load degrades significantly. Recall that there is plenty of capacity in the system – each node can process up to 8 jobs of its own type in each timestep. So the failure to process the load is due to the time jobs linger in the system without encountering a node of the correct type.

Only the torus with 2 job types is able to come close to coping with the demand. Even here we must refer back to figure 6 and note that steady state (as depicted in figure 7) is only achieved after a considerable build-up of jobs in the system.

For the rest of this section we shall depict results from the various simulated networks in terms of the total, normalised rate of job processing occurring in each timestep. That is to say, what fraction of the number of jobs (of all types) arriving in a timestep are 'absorbed' in that timestep. A network of 1024 nodes receives 4096 new jobs every timestep and so would have a job absorption rate of 0.5 in a given timestep if 2048 jobs were processed in that timestep. Note that the job absorption rate tells us nothing about the total number of jobs queuing in the system. An absorption rate of 1 tells us only that jobs are being

processed as fast as they are arriving – it does not tell us that all new jobs are processed immediately. The minimum absorption rate possible is zero – where no jobs are processed. The maximum sustainable absorption rate is 1 – where the rate of processing matches the rate of arrival. It is of course possible to have an absorption rate greater than 1 while working through a backlog of queued jobs but once the backlog is cleared jobs can only be processed as fast as they arrive.

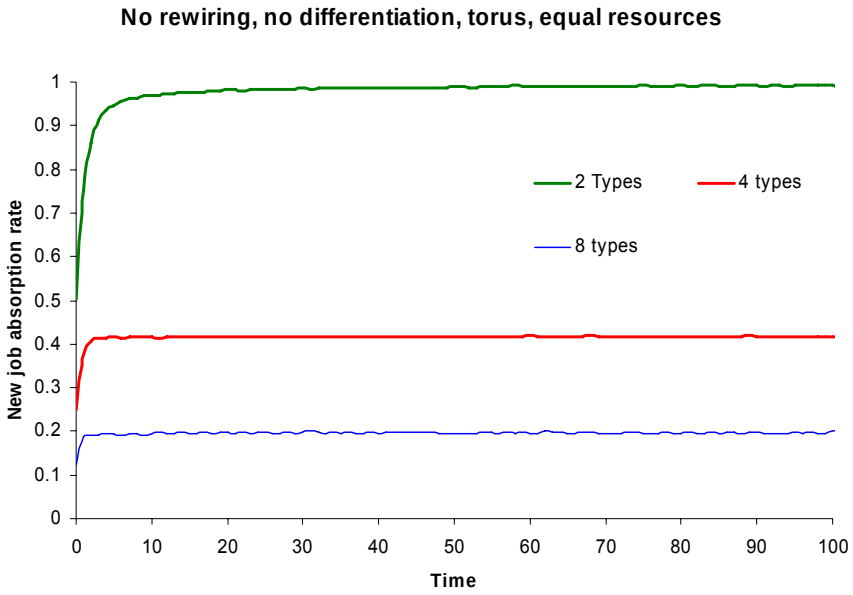

Fig 8. With no rewiring and no differentiation occurring in these simulations the only change over time is the increasing backlog of jobs. As time goes on each node has a queue of zero for jobs of its own type and increasing queues for all other job types. The increasing backlog increases the absorption rate because it increases the rate at which jobs diffuse on to nodes of the corresponding type. In the case of only 2 types of job it is possible for the network to reach an absorption rate of 1, once backlogs are really large. In these circumstances in each timestep every node receives, on average, 2 new jobs of its own type (which it processes instantly) and 2 other jobs of its own type diffusing from its two different-type neighbours. The node also processes these two diffusing jobs and is hence processing a total of 4 jobs in each timestep leading to an absorption rate of 1 across the network.

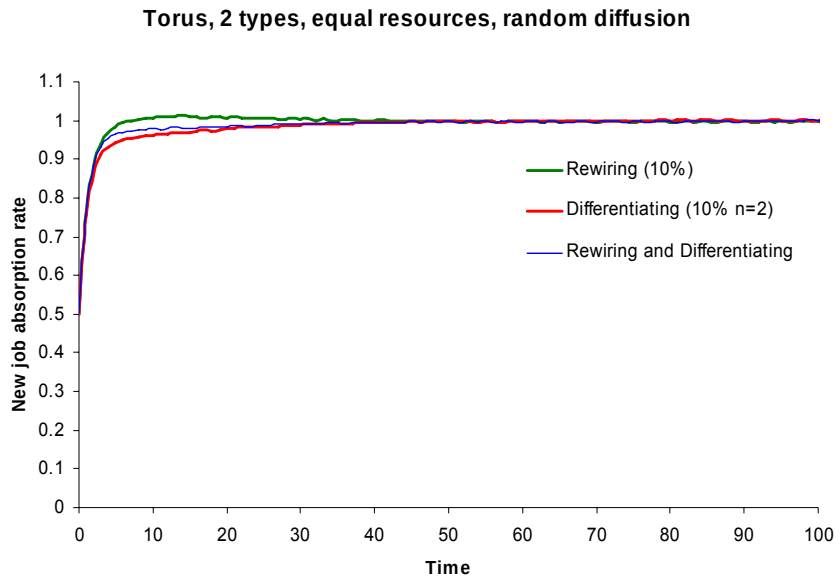

Fig 9 As noted above the torus is able to reach an absorption rate of 1 without any rewiring or differentiation. Rewiring does allow that rate to be achieved more quickly however.

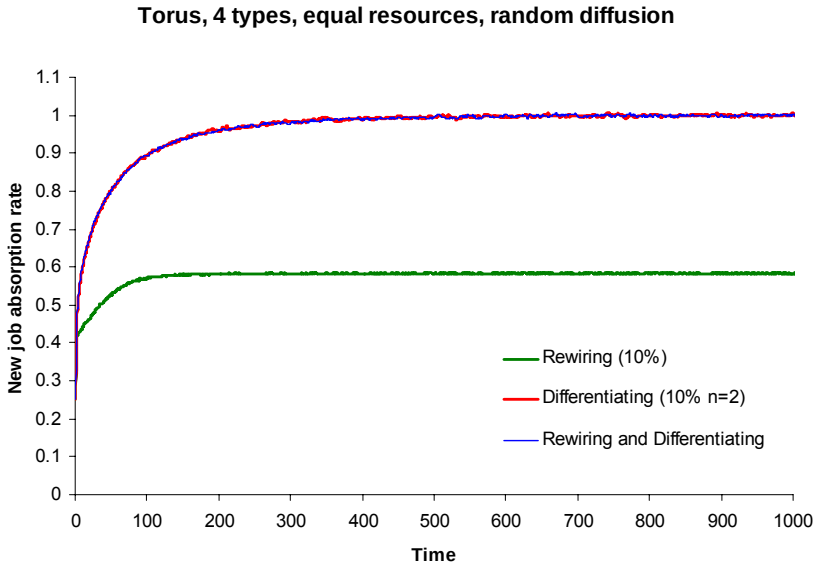

Fig 10 Rewiring alone allows some improvement to the ability of the network to absorb new jobs (compare with fig 8). Differentiation (with or without rewiring) results in a much greater improvement eventually reaching an absorption rate of 1. Note that the x axis has been extended to 1000 timesteps to show steady state.

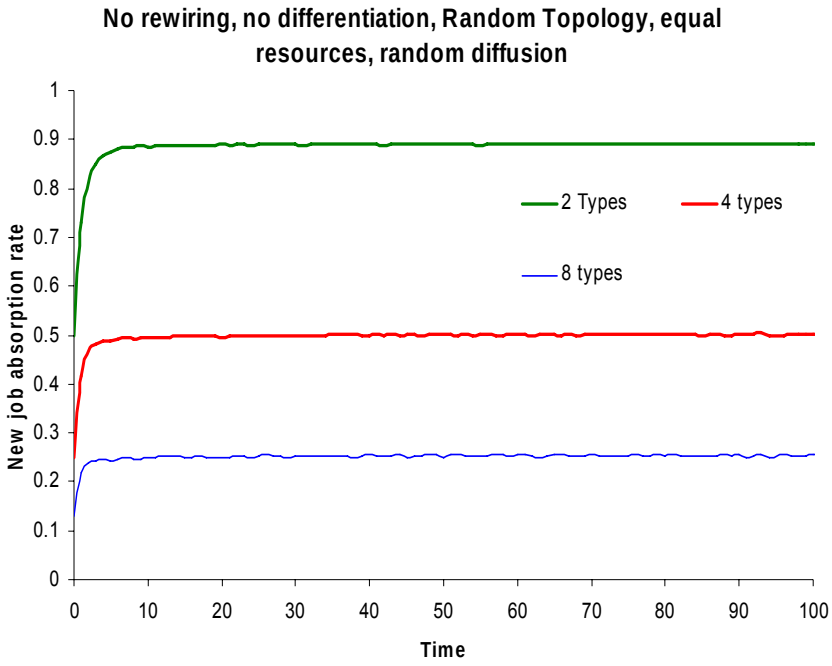

Fig 11 Identical to the situation depicted in Fig 8 except that the topology here is a random graph rather than a regular, toroidal lattice. Even when there are only two types of job (and two corresponding types of node) in the system, the random graph topology cannot achieve an absorption rate of 1 in the absence of rewiring or differentiation.

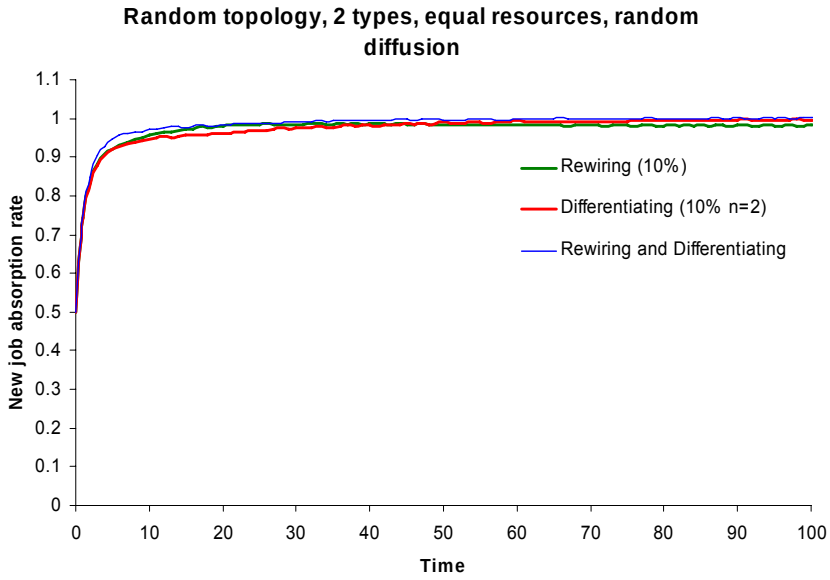

Fig 12 Rewiring and/or differentiation do allow the system to reach a point where job processing rate matches job arrival rate

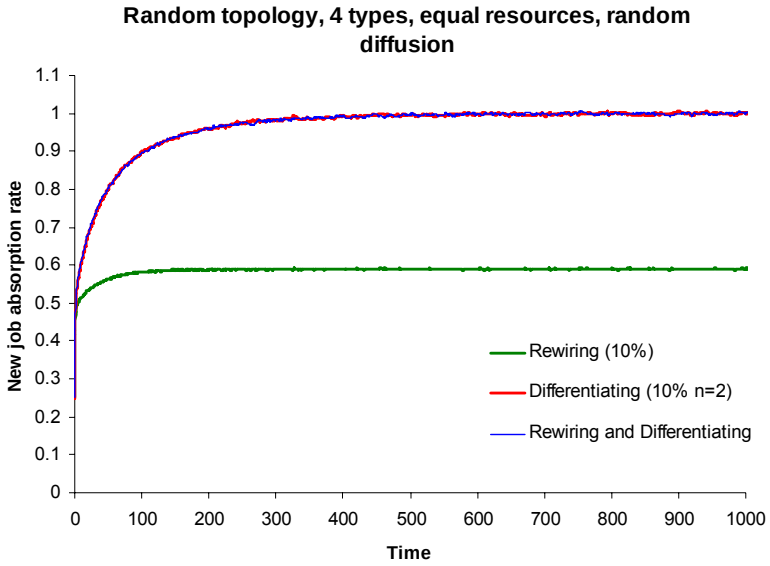

Fig 13 Note that this plot runs on to timestep 1000 to allow a steady state to be shown. Differentiation (with or without rewiring) allows the system to meet demand, eventually. Rewiring alone does not

In summary our simplest set of simulations, in which the load was statistically homogeneous and 'diffusion' of queuing jobs away from nodes was random, shows:

- Certain scenarios can allow the load to be processed (the torus deals with two job types even when no 'rewiring' or 'differentiation' is used to alter the initial network) but in most cases tested the initial network is not able to process the load
- Differentiation allows networks to alter such that they are able to process the load impinging on them

2.4.2 Homogeneous Load, Targeted Diffusion

In this set of simulations the load placed on the system is the same as in the preceding simulations (uniform across the different job types) and the initial network set-up is the same (either a toroidal lattice or random graph topology with equal numbers of nodes specialising in each job type).

However, the 'diffusion' by which queuing jobs are handed on to network neighbours has changed to the 'directed diffusion' described above, allowing nodes to selectively send the right job to the right neighbour if such an option exists.

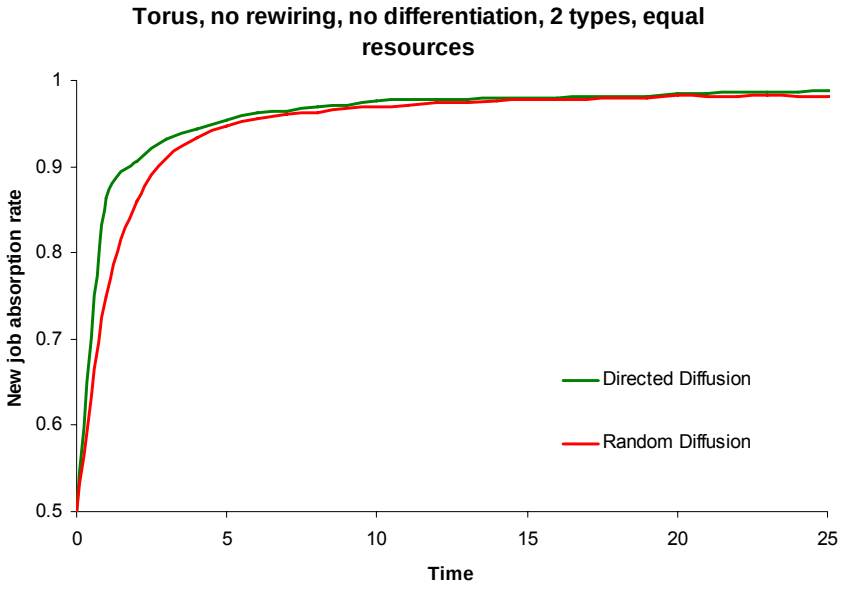

Fig 14 Comparison of the performance of the unchanging network in processing the load with random diffusion and directed diffusion

We expected that the introduction of directed diffusion would improve the ability of all networks to cope with demand, since it should allow nodes to exploit opportunities to pass a job directly to a node which can process that job, rather than relying on lucky encounters. Figures 14 and 15 bear this out, over rather different timescales.

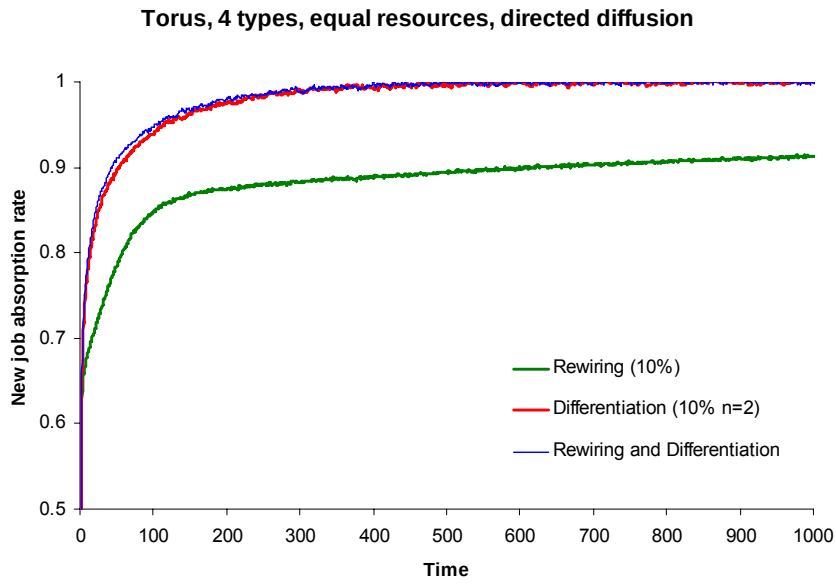

Fig 15 The absorption rate for the toroidal lattice topology with four equally prevalent job types and directed diffusion.

In Fig 14 we see that the directed diffusion allows system performance to rise much more rapidly, in other words the system is less congested with jobs when it approaches the optimal steady state absorption rate.

In Fig 15 it can be seen (compare with Fig 10) that the rewiring alone, although it is still much less effective than differentiation, rises to a much higher absorption rate than is the case for random diffusion.

In summary directed diffusion does improve baseline performance but, as with random diffusion, differentiation is more effective than rewiring in allowing the system to resculpt itself to improve on that baseline.

2.4.3 Heterogeneous Load, Random Diffusion

In this set of simulations we move away from homogeneous load and return to the random diffusion rule. Everything else remains the same as section 2.4.2

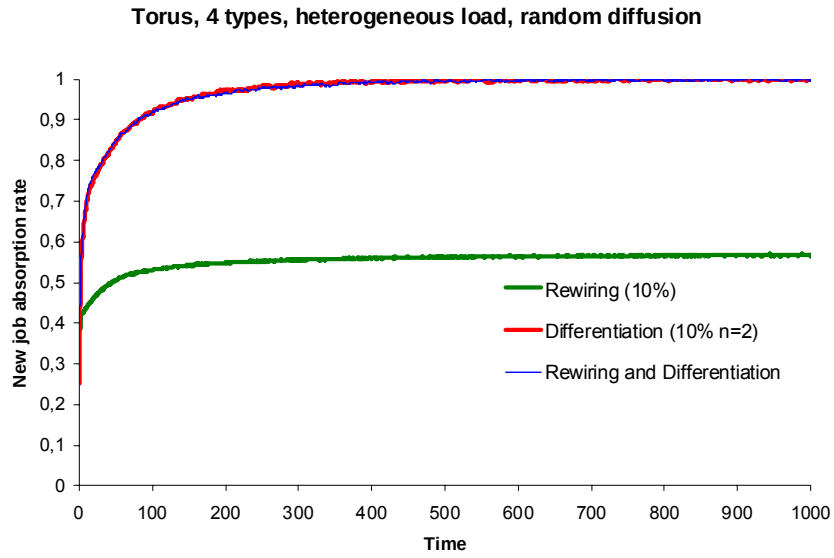

Fig 16 Absorption rates reached over the course of simulation runs in the face of heterogeneous load.

The load is now created as follows: in every timestep each node receives 4 jobs (this is the same as before). However, the probabilities of receiving jobs of different types are no longer the same. Type 0 jobs are twice as likely as type 1 jobs, which are twice as likely as type 2 jobs etc.

This means that, for the first time, the ratio of node specialisms in the initial network does not match the ratio of jobs arriving in the network. There are still equal numbers of all types of node, but not equal probabilities of the different job types. Because rewiring does not alter the ratio of node specialisms, we anticipated that this would be a difficult test case for

rewiring alone. Differentiation does alter the type of the node and hence can affect the ratio of node types in the system as a whole. So we expected that the advantages of differentiation already suggested in the homogeneous load simulations would be more marked. In fact, comparing Fig 16 with Fig 10 shows little change in the gap between performance when differentiation is enabled and when rewiring is enabled. The gap was already wide and has not significantly widened.

2.4.4 Heterogeneous Load, Targeted Diffusion

Despite being very difficult to model and interpret, this case is also probably the most interesting as it involves presenting the most advanced processing units (i.e. capable of targeted subcontracting as well as, potentially, rewiring and differentiation) with the most difficult problem to solve (i.e. heterogeneous workload, implying an initial mismatch between offer and demand).

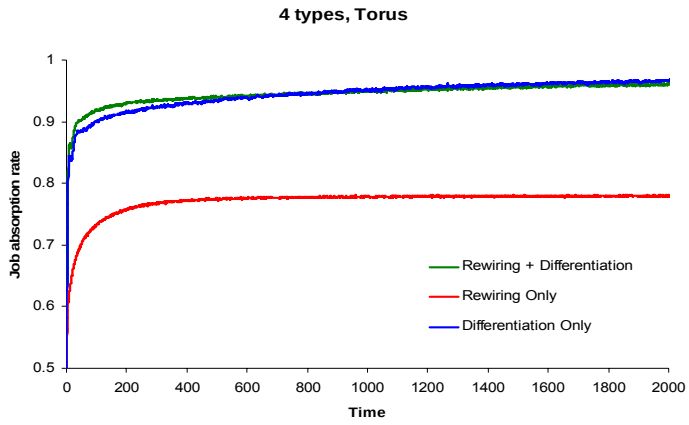

Fig. 17: Performance comparison between the 3 different forms of adaptive behaviour in the (initial) torus topology, *n* = 8

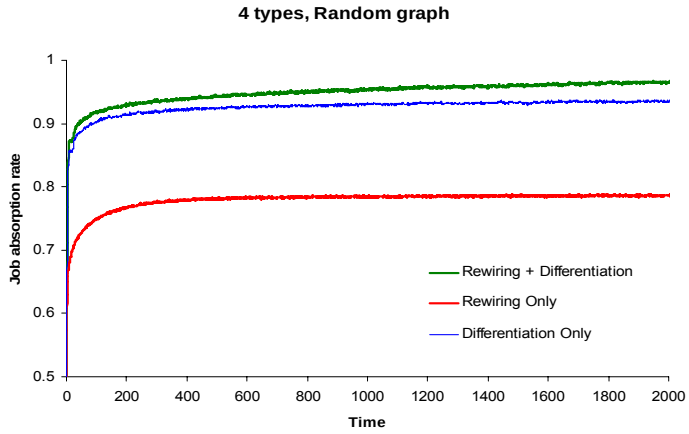

Fig. 18: Performance comparison between the 3 different forms of adaptive behaviour in the (initial) random graph topology, *n* = 8

As illustrated by figures 17 and 18, the introduction of differentiation results in a considerable improvement of the efficiency of the system as a whole (in terms of its global processing capability after self-organisation), compared to the "rewiring only" strategy. This is an intuitive result since only differentiation can lead to the adjustment of the offer to the demand. However, it should be noted that the system being over-provisioned by a factor 2 (4 job arrivals per node per time unit, with a nominal processing capability of 8 jobs per node per time unit), rewiring can, to a limited extent, contribute to the increase in the absorption rate (by increasing the degree, and therefore the incoming flow, for nodes belonging to the type that is most in demand), which is why the curve goes up over time.

2.4.5 Underlying Dynamics

It is essential to understand the dynamics underlying the increase and stabilization of the absorption rate. These can partly be inferred from Fig. 19 which shows the distribution of the agent population and of the various job types in the most complicated scenario (16 job types), after 2048 time-steps (i.e. approaching steady state). Basically, the nonlinearity in the decision function (*n*) causes the population to "overshoot" the ideal resource allocation for the most frequent type of job, at the expense of the other types. This causes job types of intermediate frequency to accumulate in the system (very rare job types are rightly ignored, and very common ones are adequately or over-provisioned), resulting in the emergence of a maximum.

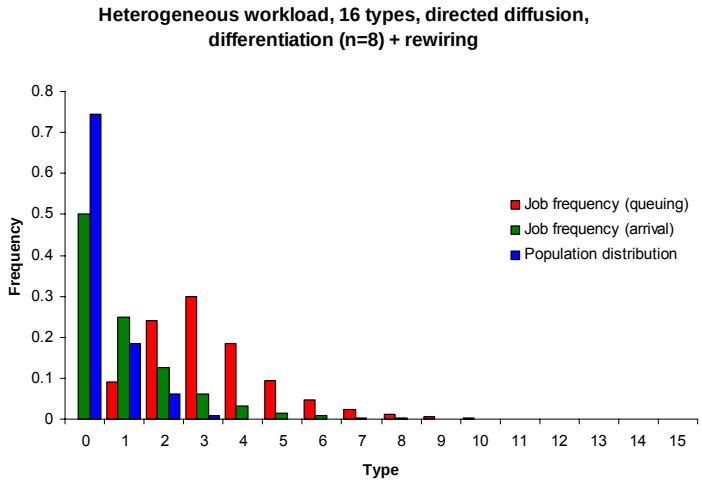

Fig. 19: Frequency distribution of jobs and agents as a function of type (*n* = 8)

Reducing the value of the nonlinearity parameter *n* can alleviate this effect, as shown by the better match between job arrival frequency and population distribution in figure 20
This even impacts on the actual absorption rate, which comes very close to one (fig. 21a) for a comparatively low accumulated workload (fig. 21b).

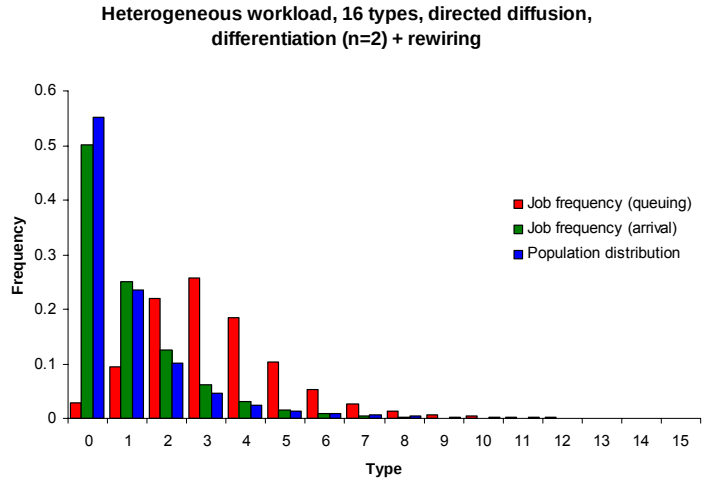

Fig. 20: Frequency distribution of jobs and agents as a function of type (*n* = 2)

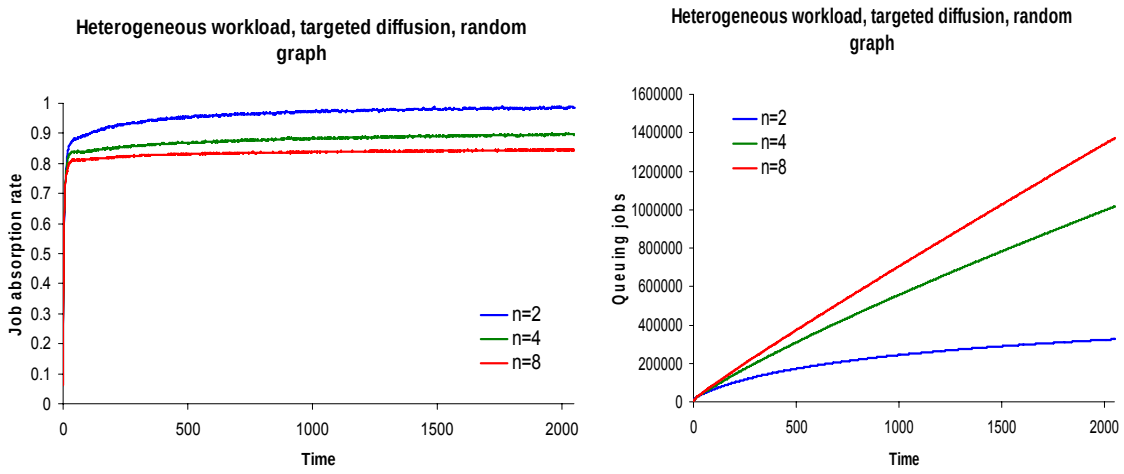

Fig. 21: Evolution of job absorption rate (a, left) and total accumulated workload (b, right) over simulation time

However, this comes at the price of greatly increased "residual activity" (in terms of rewiring and differentiation) even when nearing steady state (see fig 22). In the current simulation framework, such activity has zero cost and changing type is instantaneous, but this is obviously a simplification. In reality, changing activity would at the very least incur some downtime, due to the need to load new software components, "scrub" memory etc. Such aspects of load-balancing are actually well know to be of critical importance for optimal resource allocation.

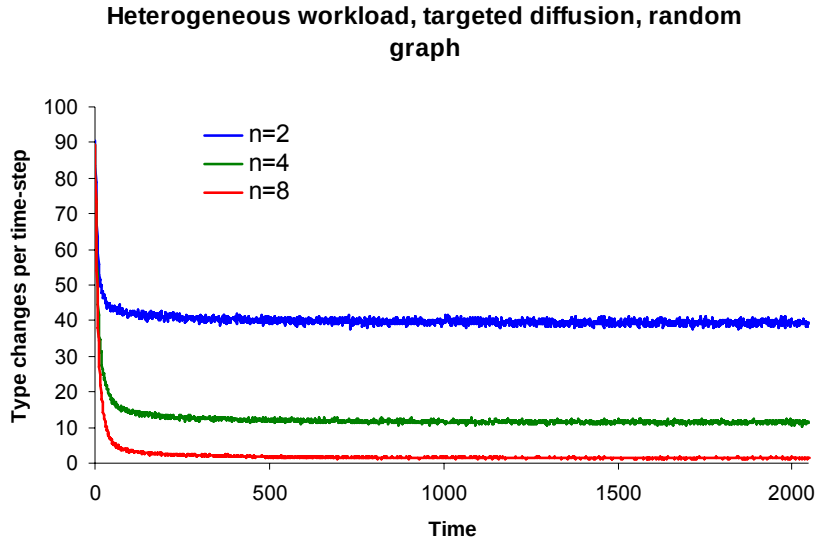

Fig. 22: Evolution of the residual activity (here, the number of type changes) over simulation time

In short, if we want to be able to rely successfully on self-organisation for load-balancing and adaptation purposes in a real world implementation, we will need to find ways of keeping "self-interest" under control so as to avoid such pathological system behaviour. Indeed, the source of the long-term instability is that individual nodes keep re-differentiating into what they perceive as the commonest type, in order to maximize a local utility function. Discouraging this behaviour could likely be done explicitly, via inhibitory signalling. Alternatively, increasing the ratio between rewiring and differentiation rates could lead to the overlay reorganization process giving specialists of infrequent job types a chance to find a viable niche (i.e. to become subcontractor for enough other nodes to compensate for the overall rarity of their job type).

2.5 Conclusion

Our results demonstrate that, even though differentiation or rewiring alone can to some extent help maximise the throughput of a distributed and decentralised processing infrastructure, combining them makes the system more responsive to heterogeneities in the workload. However, they also emphasise that even the simplest identifiable set of rules

tends to combine a large enough number of parameters to make interpretation difficult, therefore making selection of appropriate values a non-trivial problem. Finally, they indicate the presence of a substantial residual activity (both in rewiring and in self-reallocation) at steady state, which would potentially be very detrimental to performance in a real implementation (overhead) and would therefore need to be brought under control. This can potentially be achieved simply by taking explicitly into account the cost of changing type or establishing a new collaboration.

3 Minimising maintenance costs of networks where nodes can differentiate

3.1 Introduction

In addition to issues of load-balancing, minimizing the cost of maintaining links across a network is also a worthwhile aim for applications. Nodes of different types may place different demands upon the network. Ideally nodes of different types should be able to connect without imposing too great a cost. For simplicity we represent node type by colour.

We are seeking to minimize the costs of maintaining a network of nodes, while maintaining connections between nodes of different colours. At the same time we are using the capability of nodes to change colour (differentiation) to impart adaptability to a network. However it is not clear whether differentiation will make it easier to obtain adaptability, or it will act as a disruptive influence, so part of the research is about exploring the influence of differentiation.

Objectives:

- Minimize cost of maintenance of a network of self-organising nodes.
- Attain a stable equilibrium where each node is connected to one node of each other colour.

Note that in the case of the second objective, having each node connected to only one node of each other colour is the ideal solution, because this means that the first objective will have been satisfied. The second objective is a sub-set of solutions to the graph colouring problem, to be discussed further later, except that solutions to that usually allow "one or more" adjacent nodes/vertices (term used in graph theory) of each other colour.

Why are these objectives relevant?

Because we are aiming for self-organisation algorithms using differentiation to be applicable in real-world applications, we must assume that there will be some costs associated with resource use at nodes and with information transmission between nodes. Hence it would seem to make sense to reduce the number of nodes and the number of links between nodes to the minimum required to satisfy the global objectives, in order to reduce these costs and maximize the benefits from the application.

Because of the aim of applicability to real-world scenarios we must assume that nodes of different types will specialize in providing different services, as maintaining generality is likely to be costly and inefficient. Hence it will be beneficial if nodes of different types only

have neighbours of other types, so that requests for services other than those that they specialize in can follow minimal paths before reaching an node that can act upon them.

These global objectives are quite strict as a consideration of related work will show.

3.2 Related Work

3.2.1 Game theory

Initial consideration suggested that a game theoretic analysis would be the best way to investigate solutions to both these objectives. Game theory has proved useful in providing explanations for how cooperation can arise in groups through iteration even when self-interest might suggest otherwise (e.g. Axelrod & Hamilton 1981).

We wish to consider a situation where the costs/benefits of interactions between nodes can be defined based on their actions, types/colours, and whether they are linked. And we wish to move from an initial state which is either randomized or at least about which we have limited knowledge to one in which we reach a stable equilibrium.

Game theory provides a means of identifying equilibria in games between players, but it does depend upon being able to define the game in terms of a payoff matrix, where the benefits or costs to each player in terms of their actions can be defined, or in terms of the extensive form of the game, where the decision tree arising from all possible actions can evaluated.

The problem with the global objectives given above is that there are several different types of actions and axes of cost/benefit that need to be considered simultaneously. Given a location in an arbitrary network, a node in a network construction game may build a link with another node, break a link, or maintain an existing link, depending on the benefits of being linked to that other node, and the costs of building/breaking and maintaining links. Defining the complete payoff matrix or decision tree that includes all these issues is difficult.

3.2.2 Graph theory

Graph theory studies mathematical graphs, mathematical structures used to model pairwise relations between objects from a certain collection. Since CASCADAS is concerned with the behaviour of collections of ACEs, and WP3 of CASCADAS is concerned with the interaction between self-organising ACEs, graph theory seems particularly appropriate to investigate networks of ACEs where the network/graph has already been defined, and where the types of ACEs are defined simply in terms of colour.

Given the global objective above for minimizing costs in a network, we can, at least for relatively simple and finite graphs, identify graphs that will satisfy such objectives for a given set of vertices of different colours. This is the classic graph colouring problem, which has been the subject of much mathematical research. However, we may not be able to identify the route to which the optimal coloured graph is reached from an initial random state, which is why game theory was initially suggested as a means to finding this. We also cannot assume that all optimally coloured graphs are attainable.

The introduction of differentiation as described above makes this more complicated, because most solutions to graph colouring problems assume that vertices/nodes stay the same colour – but it is an aim of our research that they do not if it is not appropriate. Differentiation makes the problem dynamic: while there may be a lot to learn from the large body of work in graph theory, more applied work focusing on dynamic applications may also be relevant.

3.3 Example Problem

Analysis of the consequences of differentiation among ACEs or nodes has to start somewhere. Accordingly we define an example problem as follows:

- A finite graph
- Vertices of different types represented by different colours
- Vertices linked by undirected non-coloured edges
- Graph initially complete
- Vertices initially assigned colours at random.

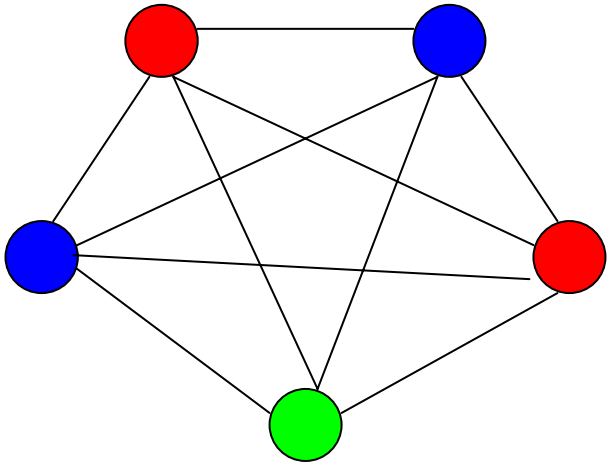

Figure 23. ***An example initial graph***

The problem is how to proceed from this initial state to the global objectives given above.

We assume that this takes place over a number of time-steps that may correspond to iterations of a simulation. Whether or not each time-step is identical in length may be defined by the model or simulation. During each time-step edges (links) can be broken or built, and vertices (nodes) can change colour.

3.4 Results

3.4.1 Solutions for Networks of Limited Size

Devising the payoff matrix for the iterated game between nodes proved difficult for the reasons given above, which is why we proceeded to assumptions about the outcome of the iterated game and then proceeded to the analysis of differentiation as a problem of graph theory.

Assuming that payoffs relating to the maintenance of links (edges) lead to a network of minimal cost, that is to say minimal numbers of links, and that this network corresponds to an equilibrium in a game between nodes (vertices), then we are in a position to separate the iterated game between nodes about maintenance of the network from the actions of nodes in differentiating in a minimal network. Whether these are reasonable assumptions needs further exploration.

Then it is possible to study the consequences of the potential for nodes (vertices) to differentiate, where differentiation implies:

- Change of colour at random with respect to initial colour and with respect to the colours of vertices to which the vertex is currently connected.
- Change of colour simultaneously and independently with respect to change of colour of other differentiating vertices.

Then we can study the results of "brute force" exploration of the outcome of differentiation in some simple graphs of finite size. In each case the exploration was carried out by considering all possible initial states given the convergence of the graph to a minimal graph satisfying global objective 1 above, and then exploring all possible outcomes of differentiation, where differentiation implies the conditions given.

In these results, n is number of vertices, m is number of colours, and l (ell) is number of edges (links), not to be confused with e – in contrast to some of the conventions in graph theory and other areas of mathematics.

$n = 1, m = 1$ only possibility is $l = 0$ (empty graph). Minimal cost graph is only state possible, vertex not connected to any other vertex, so *objectives cannot be achieved*.

$n = 2, m = 1, l = 1$ (bipartite graph). Minimal cost graph is only state possible, but with only one colour each vertex is not connected to a vertex of another colour, irrespective of whether differentiation can occur, so *objectives cannot be achieved*.

$n = 2, m = 2, l = 1$ (bipartite graph). Minimal cost graph is only state possible. Stable equilibrium of each vertex connected to vertex of different colour *only achieved if this is the initial state of the graph*, in which case differentiation of the vertices does not occur, and *the objectives are achieved*.

If the initial state has both vertices of the same colour, both will differentiate, but since they have only one other colour to choose, the vertices will proceed into an endless cycle of disruptive differentiation, and the *objectives cannot be achieved*.

$n = 2, m > 2$ overcomes the problem of continuous disruptive differentiation, but *the objectives cannot be achieved* because one vertex of each colour cannot be included in the graph.

n = 3, m = 2, l = 3 (triangle graph). Decomposes to *n = 2, m = 2, l = 1* with one isolated vertex. Ignoring this isolated vertex, the objectives *can be achieved* as in the situation with the bipartite graph above.

If decomposition of the graph is not allowed, there are not initial states in which each vertex is connected to one other vertex of each different colour, and differentiation will occur continuously and disruptively, and the *objectives cannot be achieved*.

n = 3, m = 3, l = 3. Minimal cost graph appears to be only state possible. Stable equilibrium of each vertex connected to one other vertex of each different colour appears to only occur if arises in *initial state of graph*, in which case differentiation does not occur, and *objectives can be achieved*.

If the initial state of the graph has adjacent vertices of the same colour, they will both differentiate, and while this may not result in both of them differentiating again in the next iteration, because there are only three colours, at least one vertex will differentiate and the vertices will proceed into an endless cycle of disruptive differentiation, and the *objectives cannot be achieved*.

n = 3, m > 3 has the same advantages and disadvantages as *n = 2, m > 2* above.

n = 4, m = 2, l = ? A fully connected graph will have *l = 6*. WeI hypothesize that it decomposes to two bipartite graphs, or possibly a bipartite graph and two isolated nodes, depending upon initial state. In either case the outcome depends on the initial states of the vertices and *the objectives are achieved* only if two vertices in the bipartite graph are *initially in the state required by the objectives*.

If decomposition of the graph is not allowed we hypothesize an endless cycle of disruptive decomposition as above and the *objectives cannot be achieved*.

n = 4, m = 3, l = ? We hypothesize that this will decompose to a triangle graph plus one isolated node, in which case the outcome will be as *n = 3, m = 3, l = 3* above.

If decomposition of the graph is not allowed we hypothesize an endless cycle of disruptive decomposition as above, and the *objectives cannot be achieved*.

n = 4, m = 4, l = 6. No edges will be lost from the initial fully connected graph in order to satisfy the objective of each vertex being connected to one vertex of each other colour.

However differentiation may occur depending upon the initial state of the vertices with respect to colour.

Clearly there is more to be done, but this becomes very laborious as the number of outcomes with or without differentiation increase exponentially, and there may already exist in the graph theory literature analyses of this type of problem. Alternatively the outcomes can be explored computationally, although only for finite networks.

3.4.2 General Observations

Principle of simultaneity: since vertices (nodes/ACEs) are assumed to be acting autonomously, it seems reasonable to consider differentiation within a network to be taking place simultaneously and independently with respect to other vertices in the same network. However this frequently seems to be disruptive because it causes vertices connected by an edge to differentiate simultaneously and this can lead into endless cycles of differentiation without attaining the objectives.

With respect to a simulation or a real-world application it is of course not appropriate to consider nodes/ACEs to be acting simultaneously, so when sequential differentiation is considered the disruptive effects identified above may not occur.

Principle of randomness: because the rule-sets/algorithms being studied here are intended to be as general as possible, each colour is equivalent, meaning that differentiation occurs at random to a different colour other than the one which the vertex is already. In a real-world application scenario, or a biological differentiation context, different types are not likely to be equivalent, and that will affect the outcomes making the differentiation occur in a non-random fashion, and making it less likely to be continuously disruptive as above.

3.5 Cost minimization in networks with differentiating nodes: discussion and on-going work

We have learned several things about how to understand the consequences of differentiation among self-organising nodes through considering different techniques that can be applied to it.

The first is that initial assumptions about the use of game theory to describe a problem that involves both the optimization of a network configuration (or mathematical graph) and the optimization of the colors of nodes in a network relative to other nodes (or vertices in a graph relative to other vertices) are quite ambitious, but not impossible, since we are dealing with two different optimization problems that may require different notions of utility.

The second is that the splitting of the problem into two sorts of optimization problems suggests that we may be able to deal with them separately.

If we assume that the problem of satisfying the first constraint of a graph of minimal cost has to be satisfied before nodes can achieve the capability of differentiation, then we can study this using an optimization heuristic that ignores node type or color. Which is most appropriate needs further exploration.

What we do have to draw upon is a large body of work about the conclusions to network optimization problems which can tell us about minimal configurations that are likely to arise from the application of some optimization heuristic to the initial network configuration. But we do not have the means to ensure that all minimal configurations will be attainable. For some small networks these issues can be explored directly (see above). But this is not preferable as a general solution. For that we may have to proceed to simulation or other aspects of graph theory.

If we can attain graphs that satisfy the first objective of minimal cost without having to consider the effects of differentiation, then we have gained a third insight. The representation of types of nodes in networks by colours is akin to colored vertices in graphs in the graph coloring problem, where there is a substantial body of achievement.

However, there is one significant difference. Vertices are usually assumed to stay the same color. By introducing differentiation we are making the problem more dynamic, and complicated. Further exploration of graph theoretic results, or use of simulation, is needed in order to explore the outcomes fully.

3.5.1 Applying Evolutionary Game Theory to Differentiation

Can we use evolutionary game theory to explore the evolution of the cost function for interaction between nodes where differentiation affects the cost function, and where the development of the minimal network between nodes is (at least initially) ignored?

We believe the answer is 'yes' and we address the problem in our current work. Evolutionary game theory can be used to study the fitness of populations of entities (we will refer to nodes taking part in an overlay network in the following) that may decide to adopt strategies that would increase their utility.

The relation to the differentiation problem is clear here: strategies represent again colours that characterize nodes adopting them, while a utility function defines the rules of the game in which participants are taking part. Leveraging the concepts of evolutionary game theory inspired by biological studies we are currently defining the arena in which our differentiation evolutionary game will take place. We are also currently studying the impact on the overall system of spontaneous mutations in the strategies adopted by a node, of strategic deviations from the original strategy assigned to each peer and the coexistence of aggressive versus gentle strategies.

To substantiate our work, we will focus again on a particular application that is well suited for the CASCADAS project: autonomic content distribution schemes. Related to what has been done in the first year of the project (clustering) and on its natural extension (differentiation as described in the preceding sections) we are studying the implications of differentiation in the arena of P2P content distribution wherein peers have to decide which kind of service policy to adopt. Based on a prominent example that exist today of a generous strategy (the one that is currently implemented, for example, in the BitTorrent protocol) we will analyze through modeling and simulations (if possible we will also determine the feasibility of coming up with a working prototype) the impact on application-level performance, as well as on the structural topology that defines the interactions that can take place among peers, of the adoption of a more aggressive strategy that would make use of the knowledge of other entities of the overlay being highly cooperative.

Several open questions will be addressed, for example related to what kind of countermeasure it is possible to design (and this work will be done in tight cooperation with WP4 of the project) to defend against such misbehavior.

4 Conclusion

The problem of scheduling the execution of diverse jobs across a distributed processing infrastructure is in itself a difficult problem, as evidenced by the huge body of work dealing with optimal resource allocation in computational grids or even individual data centres.

In this deliverable we have described several different ways to examine the consequences of network dynamics where nodes can change type in a manner inspired by biological differentiation. Analytical models and simulation give indications its role in load-balancing. We also report on on-going work on the minimization of the cost function in networks under interaction constraints, where both graph theoretic and game theoretic approaches can be useful.

In the present work, not only are we trying to find ways of dealing with a heterogeneous workload and maximise utilisation of available resources, we are also deliberately avoiding any solution that would require centralised accounting of these resources and/or hierarchical relationships whereby some elements would have authority to assign roles to others (like a data centre manager allocating servers to specific applications). This fully decentralised approach to scheduling is made necessary by the pledge to be able to operate in an open environment with a high churn rate, in which owners of useable resources are contributing them to the collaborative framework on a voluntary basis and only as long as they perceive their participation to be in their own best interest ("give and take").

Combining locally mediated rewiring of the collaborative overlay with self-induced differentiation of individual units (i.e. their ability to change their own purpose so as to maximise their local utility function) was the chosen approach because it intuitively seemed the best and perhaps only way to achieve load-balancing and adaptation to an unpredictable and fluctuating demand in the absence of central control and/or macroscopic information. However, past that point, a huge variety of decision-making algorithms and interaction mechanisms could have been considered. We aimed to select the simplest possible set of rules capable of promoting the desired behaviour at system level, yet we are fully aware that other models exist and are likely to outperform our basic framework. For instance, we didn't use any explicit excitatory or inhibitory signalling between units, which is known to play a key stabilising role in most biological systems (e.g. in development).

5 References

CASCADAS WP3 deliverable D3.1

Axelrod, R. & Hamilton, W.D. 1981 *The evolution of co-operation*. Science 211, 1390-1396.

Eisenfeld J, DeLisi C: *On conditions for qualitative instability of regulatory circuits with application to immunological control loops*; in Eisenfeld J, DeLisi C (eds):Mathematics and Computers in Biomedical Applications.Amsterdam, Elsevier, 1985, pp 39–53.

Cinquin O,Demongeot J: *Positive and negative feedback: Striking a balance between necessary antagonists*. J Theor Biol 2002;216:229–241.

Gouzé J-L: *Positive and negative circuits in dynamical systems*. J Biol Syst 1998;6:11–15.

Michiardi, P, Marrow, P., Tateson, R. and Saffre, F*. Aggregation Dynamics in Service Overlay Networks* in Proceedings of the first IEEE International Conference on Self-Adaptive and Self-Organizing Systems, Boston, MA, July 2007

Plahte E,Mestl T,Omholt WS: *Feedback circuits, stability and multistationarity in dynamical systems*. J Biol Syst 1995;3:409–413.

Saffre F., Halloy, J and Deneubourg J. L. *Steering and Evaluating Autonomic Deployment of Service Components in a P2P Network*. Int. Trans. Syst. Sc. App. 2006 2:3, 315-318

Saffre, F. and Inglesby, P. *A Principled Approach to Modeling Self-configuration in Distributed Systems* in proceedings of 2nd International Workshop on Engineering Emergence in Decentralised Autonomic Systems, Jacksonville, Florida 2007

Saffre, F., Tateson, R., Halloy, J. and Deneubourg, J-L. *Aggregation Dynamics in Overlay Networks and their Implications for Self-Organised Distributed Applications*. Computer Journal. In press.

Snoussi EH: *Necessary conditions for multistationarity and stable periodicity*. J Biol Syst 1998;6:3–9.

Soulé C. *Graphic requirements for multistationarity*. ComPlexUs 2003;1:123-133.

Thomas R.. *On the relation between the logical structure of systems and their ability to generate multiple steady states or sustained oscillations*. Springer Ser Synergetics.1981;9:180-193.