Aggregation Dynamics in Overlay Networks and Their Implications for Self-Organized Distributed Applications

FABRICE SAFFRE 1,* , RICHARD TATESON 1, JOSE´ HALLOY 2, MARK SHACKLETON 1 AND JEAN LOUIS DENEUBOURG 2

1 BT Group CTO, Pervasive ICT Research Centre, Adastral Park, Orion 1st floor pp 12, Martlesham Heath IPS 3RE, UK
2 Université libre de Bruxelles, Service d’Ecologie Sociale and Centre for Nonlinear Phenomena and Complex Systems, Campus de la Plaine CP 231, Blvd. du Triomphe, 1050 Bruxelles, Belgium
*Corresponding author: fabrice.saffre@bt.com

In this paper, we investigate the global self-aggregation dynamics arising from local decision-based rewiring of an overlay network, used as an abstraction for an autonomic service-oriented architecture. We measure the ability of a selected set of local rules to foster self-organization of what is originally a random graph into a structured network. Scalability issues with respect to the key parameters of system size and diversity are extensively discussed. Conflicting goals are introduced, in the form of a population of nodes actively seeking to acquire neighbours of a type different from their own, resulting in decreased local homogeneity. We show that a ‘secondary’ self-organization process ensues, whereby nodes spontaneously cluster according to their implicit objective. Finally, we introduce dynamic goals by making the preferred neighbour type a function of the local characteristics of a simulated workload. We demonstrate that in this context, an overlay rewiring process based purely on local decisions and interactions can result in efficient load-balancing without central planning. We conclude by discussing the implications of our findings for the design of future distributed applications, the likely influence of other factors and of extreme parameter values on the ability of the system to self-organize and the potential improvements to our framework.

Keywords: self-organization; autonomic systems; overlay networks; distributed applications; component aggregation

Received 13 October 2006; revised 25 February 2008

1. INTRODUCTION

1.1. Autonomic computing and service-oriented architectures

The service overlay mechanisms presented in this paper are motivated by the need for autonomic (self-managing) solutions that support future systems composed of distributed service components. Web Services management and configuration is one area where such autonomic solutions will be required. Web Services [1, 2] encapsulate both a technology and an industry trend towards distributed, component-based business solutions, where an application is realized by linking many individual services together into complicated workflows and higher-level composite services.

Related to the Web Services domain is the Open Grid Services Architecture (OGSA) that is being developed by the Global Grid Forum (GGF). The aspiration of this work is to support the distributed interaction and interoperability of large numbers of component services in order to meet the needs of users, especially in the eCommerce and eScience domains [3]. Some of the technical challenges arising in realizing such solutions are highlighted in Ian Foster’s seminal
paper ‘The Physiology of the Grid’ [4]. Similar challenges arise in the requirement for autonomic solutions for Pervasive ICT [5].

Large-scale deployments of web services and associated technologies provide major opportunities, but they also exacerbate problems that already plague ICT systems, namely the complexity of such systems arising from huge numbers of interacting component parts and the cost of deploying and managing the behaviour of such systems. IBM have probably captured the challenges most clearly and succinctly by launching the Autonomic Computing initiative and associated challenges [6]. A recent survey of progress and outstanding challenges in autonomic communications [7] highlights the need to design and control distributed systems by arranging the circumstances for cooperation, rather than relying on enforced, or explicitly coordinated, cooperative behaviour by all the system elements.

Given that future applications and services will almost certainly be realized by coordinating the actions of many constituent services, the question arises as to how this might be achieved. Clearly the component services need to be encouraged to cooperate together to provide higher-level services, and given the envisaged scale and complexity, much of this cooperation needs to be inherent, or ‘engineered into’, the underlying system as self-organizing principles. This paper presents a principled analysis of a self-organizing overlay that is a step along that path.

The need for self-managing approaches for Web Services has been acknowledged by the OGSA in their stated objectives for ‘self-management services’ and ‘service level managers’ but it is recognized that this work is at an early stage and is currently more aspirational than actual [3]. Currently the most popular approach is that advocated by IBM, motivated by an ‘autonomic management’ control loop [8]—in essence this relies on a ‘generic control loop pattern’ comprising monitoring, analysis, projection and action phases [3]. While this pattern will clearly form an important element in many autonomic systems, in this paper we advocate autonomic approaches that are more ‘inherent’ in the design and behaviour of the system as a whole. This has the additional benefit of producing a more lightweight architecture.

As already noted, our focus in this paper is on service provision via a peer-to-peer (P2P) overlay network that helps coordinate the cooperative behaviour of many individual service components. In this sense it is related to the Chameleon system for self-organized and decentralized P2P web services [9]. Our approach is closely allied with the techniques and algorithms arising from a community whose research is often labelled as ‘self*’ and draws upon highly interdisciplinary concepts and models, such as biologically inspired autonomic solutions [10, 11]. For example, we are interested in approaches that cause systems to reconfigure themselves ‘from within’, in response to changes in perceived demand in a fashion that gives rise to desirable system-level behaviours. In the context of a service overlay, the collaborative structure that emerges should ultimately reflect the overall distribution of workload.

Finally, we should stress that ultimate aspirations of both the Self* and Web Service research communities are in fact quite closely aligned. In practice, we expect the real benefit will be most effectively realized when solutions such as those presented in this paper begin to be adopted and combined with those arising from initiatives such as the OGSA in a truly interdisciplinary fashion.

1.2. Topological characteristics of service overlays and related issues

The development of P2P content-sharing [12] and other distributed applications [10] exploiting self-maintained overlay networks has emphasized the need for algorithms capable of promoting the emergence of a given global topology whilst running locally [13]. A notable example of this kind of algorithm is the ‘preferential attachment’ rule popularized by Barabasi et al., which gives rise to the so-called ‘scale-free’ topology characterized by a power-law distribution of node degree [14]. A node’s ‘degree’ is the number of its links (or ‘first neighbours’); so, for example, a node directly connected to four other nodes has a degree of four.

However, as we have pointed out in our previous work [15, 16], the preferential-attachment rule makes at least two strong assumptions on system dynamics, namely:

- The network grows through the sequential addition of new vertices
- Complete and accurate global information on the current topology is available to the algorithm used to select which pre-existing vertex the ‘newcomer’ will be connected to.

This effectively limits applicability of abstract models like the ‘preferential-attachment’ rule to the management of many real overlay networks, which tend to be characterized by the fact that decisions must not only be made locally, but also be based on locally available information (i.e. the decision-maker has limited visibility of the system) and in a very dynamic environment (i.e. growth is neither monotonic, nor, a fortiori, sequential).

Other local decision-based overlay management techniques have therefore been proposed involving, e.g., gossiping techniques and diffusion-like propagation of information [17]. But, at least in the particular example given in [17], the target configuration is expressed in terms of the relative location of nodes in the network and does not involve complex topological characteristics (i.e. at steady state, all vertices have a similar degree and are organized into a 2-dimensional mesh).

In an alternative approach to generating desirable global characteristics through local rewiring and gossiping, Ganesh
1.3. Positioning with respect to the wider P2P field

Peer-to-peer overlay networks required for cooperative service provision fulfil a very different role from those typically associated with the more familiar case of content-sharing networks [12, 20]. The former are typically characterized by lower diversity (there are fewer different service components than there are unique files in a typical file-sharing community) and more subtle interaction patterns (peer activity isn’t limited to propagating queries and uploading/downloading content).

In P2P service provision, components are distributed across a collection of processing nodes, each of them hosting only a subset of all services locally and relying on different nodes to satisfy other needs, via remote access. This has fundamental implications for the design and performance evaluation of the supporting overlay. Indeed, the key objective is no longer to locate and retrieve content as efficiently as possible, for instance by propagating a query via a subpopulation of super-peers [11, 21]. Instead, in cooperative service provision, success and stability are measured by the system’s ability to reach a configuration in which interactions are effectively limited to first neighbours. It eliminates the need for multihop propagation of service requests, and so of the complex, potentially error-prone, resource-consuming and/or delay-inducing routing or flooding procedures that would otherwise be required throughout the system’s useful life. In fact, one of our main goals was to identify and evaluate methods to attain such a configuration strictly through short-range gossiping (the longest range interaction involves turning ‘second neighbours’ into ‘first neighbours’, a procedure mediated by the ‘matchmaker’, a peer maintaining a link to both the others when rewiring is initiated).

This difference of objectives also influenced our choice of average node degree. Whereas most P2P overlays favour high connectivity (which tends to reduce network diameter and improve search efficiency), we deliberately kept average node degree relatively low (typically around four). Note that this is a mean value and doesn’t prevent some peers from acquiring large numbers of first neighbours. Our intention though was to have an average node degree of the same order as the number of services supported by the collaborative overlay (also called ‘diversity’). The main justification for this decision, apart from a design choice in favour of maximum parsimony, is that this region of the parameter space is where the most complex system behaviour can be expected and where our proposed solution will be most severely tested. For instance, when the objective for some peers is to have at least one neighbour belonging to every type (or service) different from their own, as in the load-balancing scenario of Section 4, having an average node degree similar to the diversity parameter implies that although the overlay rewiring algorithm has enough links to weave a useful web of interactions, it cannot rely on numerous ‘spares’.

1.4. Structure

The remainder of this paper is structured as follows.

In Section 2, we present a very simple algorithm using exclusively communication between first neighbours (without diffusion of information) and show how applying carefully selected local rules can foster self-organization of a ‘community’ of typed elements according to a global criterion (clustering or ‘reverse-clustering’ of elements belonging to the same type) and provide quantitative results as to how performance is affected by system size and diversity.

From Section 3, we bring our abstract network somewhat closer to a real situation by introducing the concept of heterogeneous behaviour in the network. That is to say that in addition to having various arbitrary ‘types’ the nodes in the network will now interpret rules according to their own individual circumstanes. In the first instance (Section 3) there are just the two contrary modes of clustering and ‘reverse clustering’. Each node is randomly assigned clustering or reverse-clustering mode. Even in this mixed network where half the nodes have an opposite goal from the other half, self-organization still occurs, both by type and, although it is not itself a goal for the nodes, by mode. By type: clustering nodes succeed in surrounding themselves with nodes of the same type; reverse-clustering nodes surround themselves with nodes of different types. By mode: clustering nodes tend to have clustering neighbours; reverse-clustering nodes have reverse-clustering neighbours.

In Section 4, nodes dynamically alter behaviour according to their ‘workload’. This workload is intended to represent local demand in a simple service-oriented architecture. The type of a node is no longer an arbitrary label, but reflects the service which that node is able to provide. Overloaded nodes inspect their neighbours in search of potential helpers to whom some load can be shifted. In addition they use a modified form of the clustering behaviour introduced in Section 2 where the type of the new neighbour requested by the node is a function of the node’s current workload. By applying these local rules the load is spread across the
network, improving the efficiency with which the global demand is satisfied. Overloaded nodes acquire more neighbours than idle nodes, giving them more potential helpers to deal with the local high demand.

Section 5 contains a general discussion of our results and of their implications for distributed applications, as well as suggestions for future work.

2. THE ‘ON-DEMAND’ CLUSTERING ALGORITHM

In contrast to the work reported in [18], the self-aggregation process described above maintains a homogeneous node-degree without making explicit use of this node-degree variable. This is achieved simply by distinguishing between the initiator of a rewiring procedure and the matchmaker. Basically, upon ‘waking-up’, the initiator requests a new link from one of its existing neighbours chosen at random, which will then act as the matchmaker. With this logic, the probability for a node to be appointed matchmaker is obviously a direct function of its own degree. Since the matchmaker ends up losing one neighbour (the ‘candidate’) in the process of a successful rewiring operation, there is a negative, ‘rich-becomes-poorer’ feedback.

The detailed algorithm governing key node behaviour in the three roles of ‘initiator’, ‘matchmaker’ and ‘candidate’ involved in a rewiring operation following the ‘on-demand’ clustering procedure is shown in Fig. 1. It involves exchanging five types of messages (plus the link termination message which isn’t discussed here). The ‘neighbour request’ (NRQ) message is sent by the initiator to the chosen matchmaker and specifies the type of node desired. The ‘neighbour reply’ (NRP) message is sent by the matchmaker to the initiator to inform it of a potential candidate. The ‘link’ (LNK) message is sent by the initiator to the candidate to ask for the establishment of a new link, which will only be effectively created if it

![FIGURE 1. The three ‘reasoning’ loops governing node behaviour in the ‘on-demand’ clustering scenario: (A) Initiator role, (B) matchmaker role and (C) candidate role (see text for details).](Image)
is compatible with the goals of the candidate, as evidenced by the receipt of an ‘acknowledgement’ (ACK) message by the initiator. Note that, for most of the results presented in this paper, this will always be the case, as all nodes in the system share the same objective, i.e. they are all assumed to be simultaneously in clustering (or reverse-clustering) mode. Finally, after a successful handshake between the initiator and the candidate, the matchmaker is informed via the ‘success’ (SCC) message, indicating that its own connection to the candidate has to be terminated in order to conserve the total number of links. Note that it is always the matchmaker’s link to the candidate (not to the initiator) that is severed following a successful rewiring operation.

For the sake of clarity, it should be emphasized that throughout this paper, we make the assumption that rewiring attempts are independent events. This is an approximation commonly used in physics for the situation in which the events under consideration are very short and/or infrequent compared to other phenomena affecting the system. In a distributed real-life operation environment, however, independence may have to be enforced through a ‘locking’ mechanism preventing a node from becoming involved in two concurrent rewiring procedures.

Note that these variables were chosen specifically to quantify aggregation dynamics as an explicit way of measuring global characteristics, indicative of the level of clustering achieved by rewiring. At this stage, we make no assumption as to whether higher homogeneity, fewer or larger domains etc. are useful features. The application of the algorithm to a concrete problem (namely decentralized load-balancing) comes under consideration in Section 4.

We typically present mean values of the chosen variables, averaged over 100 independent numerical experiments per combination of parameter values—population size and number of types—which ranged from 100 to 1000 peers (by increments of 100) and from two to ten node types, respectively.

Figure 2 shows the evolution of link homogeneity (an indirect measurement of aggregation success) over simulation time. The first immediately noticeable property is scalability, with the 1000-strong population converging to similar or higher homogeneity values than the 100-node network. Though this is not a surprising result, a higher ‘surface/perimeter’ ratio being in principle achievable in larger ensembles, it confirms that the chosen local rules are capable of supporting the self-organization process leading to such ordered state independently of system size.

The ‘slowing down’ effect for larger populations is attributable to the fact that a fixed number of rewiring attempts are made per time unit, independently of the number of peers. Had the probability of initiating such an attempt been fixed and identical for all nodes (assuming that possible inconsistencies resulting from concurrent modifications of the overlay can be prevented), leading to the number of rewiring attempts per time unit being statistically proportional to population size, indications are that convergence would actually be faster in larger systems.

For instance, if defining the plateau as the region in which the increase in overall homogeneity is $<0.1\%$, it is reached in $\sim 120$ time-steps in the 100 peers/2 types scenario and in $\sim 510$ time-steps in the 1000 peers/2 types scenario, i.e. multiplying the population size by a factor 10 only translates into increasing the required number of rewiring attempts by a factor $\sim 4$. Scalability is again confirmed by the fact that the values are very similar in the case when 10 node types are involved.

Note that performance can only be measured relatively to the conditions in the initial random graph in which, statistically, the homogeneity variable is simply an inverse linear function of diversity (number of types). So for instance, reaching a situation in which $\sim 35\%$ of links connect peers belonging to the same type in the 1000 nodes/10 types scenario effectively means that homogeneity has increased by a factor $\sim 3.5$ compared to the initial conditions.

Figure 3 shows the effect of the two key parameters (population size and number of types) on the characteristics of the domains, after 20,000 time-steps. Basically, the number of
domains appears to be a power law of the diversity (number of types), the slope of which is a function of system size (number of peers). The mean size of the largest domain of each type (i.e. averaged over all 100 simulations and over all types) seems to be an inverse exponential of the diversity but it is also almost totally unaffected by population size (in relative terms). So in effect, the expected ‘headcount’ of the largest domain is a constant function of system size.

In this section, we have established that a mixed population of types of node could successfully form clusters providing all nodes obey a set of simple local rules. In many real world scenarios however, we would expect that some nodes would follow one set of rules and some another. A node may alter its rule set from time to time to reflect its internal state and we might even imagine that the rule set would be chosen to secure some selfish advantage [22]. Some clear effects of local rule variability are presented in Section 3.

3. SPONTANEOUS SEPARATION OF A MIXTURE OF BEHAVIOURS

3.1. Mode separation

In this section, we begin to explore the dynamics of a population which is a mixture of ‘types’ as in Section 2 above, but where there is also a mixture of behaviours. In the work reported in this section, we are concerned with only two behaviours: each node is in either clustering or reverse-clustering mode. In the flow diagram of Fig. 1, Initiator role (A), a node in clustering mode will answer ‘yes’ to the query ‘in

![Figure 2](image1.png)

**FIGURE 2.** Evolution of homogeneity over simulation time, for: (A) two node types and (B) ten node types (dashed curves indicate extreme values observed in the 100 independent realizations per combination of parameter values; see text for details).

![Figure 3](image2.png)

**FIGURE 3.** Influence of the two key parameters (population size and number of types) on the (A) number of domains and (B) size of the domains (dashed curves in A are power-law fittings; dashed curves in B are exponential fittings).
clustering mode?’ whereas a node in reverse-clustering mode will answer ‘no’. In other words, clustering nodes will request links to new neighbours of the same type as themselves whereas reverse-clustering nodes will request links to new neighbours of a type different from their own.

We were interested to see whether a population of nodes forming new links by type according to these modes would also tend to separate along mode lines, even though the modes are not themselves directly visible from one node to another. Would we tend to see highly interconnected subpopulations in the same mode?

3.2. Mixed mode simulations

In all simulations, the population was formed by assigning each node the clustering or reverse-clustering mode with equal probability. The mode of a node remained unchanged throughout the simulation run.

To assess the dynamics of the system we introduce two new measures, closely related to the global homogeneity as described in Section 2 above. Just as with the homogeneity measure, these are variables of the system, not parameters. Furthermore, they, just like homogeneity, are used only by an external observer to assess system performance and are not used directly by the nodes in their rewiring logic.

First we have global separation, which is the fraction of links connecting nodes in the same mode. High values for this indicate that the initially mixed population (with a global separation value 0.5 on average) has indeed separated such that nodes have more neighbours in the same mode (regardless of what type those nodes may be).

Secondly, we have global satisfaction, which is the fraction of links that are acceptable to the nodes according to the type of the neighbour. In other words, a node in clustering mode is ‘satisfied’ with a link that connects it to a node of the same type. A reverse-clustering node is satisfied with a link that connects it to a node of a different type. If only one of the nodes connected by the link is satisfied with its neighbour, the global satisfaction score for that link is halved.

The scoring system is given in Table 1. To summarize, we have three scores: homogeneity measures the extent to which nodes of the same type are clustered; separation measures the extent to which nodes in the same mode are clustered; satisfaction measures the extent to which the nodes’ preferences for neighbours of a particular type are met.

In the same way as in Section 2 we saw global homogeneity rise during a simulation run, we expected to see global satisfaction rise in the mixed population because the local rules cause nodes to request nodes of a type that will ‘satisfy’ them. We were interested to see how high satisfaction would rise and whether separation, which is not explicitly sought by the local rules, would occur.

Note that although we have called all three measures ‘scores’ we do not wish to imply that high scores for homogeneity or separation are necessarily desirable for individual nodes or for the system as a whole. As an extreme example, a system composed entirely of nodes in the reverse-clustering mode would be performing well if it could reduce the global homogeneity score over the course of a simulation run (since each node is seeking neighbours of a type other than its own). For this reason homogeneity becomes a less useful measure when dealing with mixed mode systems, and hence we introduced the satisfaction score that does indeed reflect the extent to which nodes have achieved their aims. The separation score is unrelated to individual nodes’ aims or system goals since it deals with mode (which is not directly visible to the nodes) and it is purely arbitrary whether we score 1 or 0 for each link between nodes in the same mode. By choosing this score to be 1 we do not wish to imply that individual nodes are explicitly seeking neighbours of the same mode, nor that we expect that a system in which the global separation score rises during a simulation run is performing better (or worse) than one in which the global separation score falls.

| TABLE 1. Scoring system for the three measures of system organization: global homogeneity, global separation and global satisfaction scores are simply the mean values of these scores for all links in the system. |
|---|---|---|---|---|
| Node A | Node B |Homogeneity Score | Satisfaction Score | Separation Score |
| Type | Mode | Type | Mode | | | |
| 1 | Cluster | 1 | Cluster | 1 | 1 | |
| 1 | Cluster | 1 | Reverse | 1 | 0.5 | 0 |
| 1 | Cluster | 2 | Cluster | 0 | 0 | 1 |
| 1 | Cluster | 2 | Reverse | 0 | 0.5 | 0 |
| 1 | Reverse | 1 | Cluster | 1 | 0.5 | 0 |
| 1 | Reverse | 1 | Reverse | 1 | 0 | 1 |
| 1 | Reverse | 2 | Cluster | 0 | 0.5 | 0 |
| 1 | Reverse | 2 | Reverse | 0 | 1 | 1 |
3.3. **Mixed mode results**

Global separation increased over the course of each simulation run, resulting in values after 20,000 time-steps of 0.75 (±0.05). The mean final separation value was little affected by the number of nodes and the number of types, although the variance was larger in small networks (data not shown).

The number of types in the network did have a significant effect, however, on the number of time-steps taken to reach the final separation value. Figure 4 shows the separation of the mixed population over time. In all cases the population was 1000 nodes, with average degree four. Each node had an equal probability of permanently assuming the clustering or reverse-clustering mode.

The separation we see taking place is driven by the local rules through which nodes strive for ‘satisfactory’ links. Each new link will only form if it is satisfactory to both requestor and candidate (see Fig. 1) and the effect of that new link on the global satisfaction measure will depend on the score for the link dropped between the matchmaker and the candidate. If the dropped link was already mutually satisfactory (scoring 1) the global satisfaction measure will not change. If the dropped link was mutually unsatisfactory (scoring 0) or only satisfactory to one of the linked nodes (scoring 0.5) the global satisfaction measure will rise by 1 or 0.5, respectively (and hence the satisfaction measure can never fall over the course of a simulation run).

This ratchet effect also applies to the separation measure. Since a candidate node will not accept a new unsatisfactory link, it is impossible for a clustering node to acquire a reverse-clustering node as a new neighbour, and vice versa. We were concerned therefore that the separation we see is no more than rising satisfaction by another name. We therefore examined the global homogeneity, satisfaction and separation measures for individual simulation runs to see what the relationship was.

![Figure 4](image)

**FIGURE 4.** Evolution of network measures over simulation time: (A) Each curve shows mean separation results from 10 independent simulation runs; (B–D) three examples of individual simulation runs showing evolution over time of global homogeneity, global satisfaction and global separation—(B) 4 node types, (C) 7 node types, (D) 10 node types.
Figure 4B, C and D show three examples for four, seven and ten types of node. In each case there are 500 nodes in the network, permanently assigned to clustering or reverse-clustering mode at time 0 with equal probability.

The plots of Fig. 4B, C and D are typical of individual simulation runs and show that the observed increase in separation over time is indeed closely associated with, but not identical to, the rise in satisfaction. Both measures cannot decrease during a simulation run, as explained above. Both can only rise if there is a successful new link established. Any increase in global satisfaction or global separation score resulting from that new link will depend on the satisfaction or separation score, respectively, of the link dropped by the matchmaker to make way for the new link.

4. COLLABORATIVE COMPUTING AND LOADBALANCING

4.1. Rationale

The purpose of developing the type of self-organizing aggregation framework described in this paper is to identify and document suitable rule sets capable of supporting autonomic operation of distributed peer-to-peer applications in the absence of central control. In the current trend toward SOA, such an application would likely be designed as a collection of self-contained but mutually dependent service modules, providing elaborate ‘meta-services’ via transparent composition and workflow management [1, 9].

Engineering such an advanced self-managing distributed service provision framework is the focus of much attention, particularly from industry [6]. However, in most cases, it relies on centralized monitoring and implementation. We take the opposite view that effective system agility requires individual components to be fitted with the basic ‘reasoning’ and decision-making abilities required to build, maintain and recycle collaborative relationships ‘on-the-fly’ in a dynamic environment featuring a wide variety of perturbation factors (from variable resource availability to largely unpredictable demand patterns) [23, 24]. The IST Project ‘CASCADAS’ [25, 26] provides the background for our research effort toward achieving this goal.

In this section, we demonstrate how the local ‘on-demand’ clustering rules described in Section 2 can be used to generate and maintain a collaborative overlay connecting elements with complementary abilities, improving the performance of the system as a whole. Depending on individual circumstances, this can mean teaming up with processing nodes hosting the same service but facing different workloads (i.e. the number of jobs or service requests in the queue vary between task specialists), or, on the contrary, with elements performing completely different functions. In the first case aggregation can be used for load-balancing purposes, i.e. nodes can ‘share the burden’ with their overloaded neighbour(s). In the second, it allows an element to act as an access point to a service that it cannot offer by itself, possibly as part of providing a composite ‘meta-service’, by transparently forwarding the corresponding jobs to the appropriate neighbour(s).

The overall objective is to dispense from centralized orchestration, in a manner somewhat similar to the one we proposed in previous work [27, 28], but with the important difference that the topology of the overlay is not fixed. On the contrary it is used to connect pre-specialized service components and doesn’t involve (re-)allocation of local resources.

4.2. Generalized model formulation

For reference purposes, we propose a dynamical description of the system, including some simple topological constraints in the form of the adjacency matrix $M$ ($M_{ij} = 1$ denotes the existence of a link between $i$ and $j$ and $M_{ij} = 0$ the absence of such direct connection), to describe the evolution in time of $x_{ik}$, the number of jobs of type $k$ queuing at processing unit $i$. This formalism allows us to compute the final steady state reached asymptotically by the system in simple cases. Such limit cases can then be used as a benchmark for developing more realistic algorithms. We define the binary matrix $T$ that determines the type of jobs that a node is able to process ($T_{ik} = 1$ if unit $i$ is of type $k$, 0 otherwise). In a first approximation, the time evolution of the numbers of jobs queuing at every unit can be obtained using a set of differential equations of the form:

$$\frac{dx_{ik}}{dt} = \gamma_{ik} + \beta_{ij} \sum_{j \neq i} M_{ij} \left( \frac{T_{ik} x_{jk}^n}{x_{ik}^n + \theta_j^m} - \frac{T_{jk} x_{ik}^n}{x_{jk}^n + \theta_j^m} \right) - \frac{\alpha_i T_{ik} x_{ik}^{m^*}}{x_{ik}^{m^*} + \lambda_i^m}$$

(1)

where $\alpha_i$ is the job processing capability of unit $i$, $\beta_{ij}$ is the job transfer rate between nodes $i$ and $j$ and $\gamma_{ik}$ is the arrival rate of new jobs of type $k$ in $i$. $\theta_j$, $\lambda_i$, $m$ and $n$ are parameters used to tune the system’s response when the length of the queue approaches the processing capability ($\alpha_i$) or transfer capacity ($\beta_{ij}$), with typically $0 < \lambda_i < \alpha_i$, $0 < \theta_j < \beta_{ij}$, $m > 1$ and $n > 1$. When $x_{ik} >> \lambda_i$, processing rate tends toward $\alpha_i$ (the unit is fully loaded), while when $x_{ik} < \lambda_i$, it tends toward zero (the unit is idling). Note that this equation implies that units only forward jobs to neighbours capable of processing them (cf. presence of $T_{jk}$ in the sum), which in turn requires some basic reasoning capabilities. At present, $\alpha_i$, $\beta_{ij}$, $\lambda_i$ and $\theta_j$ are assumed identical for all units and all links and so can be replaced by $\alpha$, $\beta$, $\lambda$ and $\theta$.

This general equation only confirms the intuition that, for a closed system ($\gamma_{ik} = 0$) containing at least one node of each type and for a complete graph topology, the system will reach a steady state where all jobs have been processed...
\( x_{ik} = 0 \) for all \( i \) and \( k \). This property is independent of initial conditions (workload), system size (number of processing units) and diversity (number of types). The time evolution however strongly depends on the type matrix \( T \) and, in any network different from the complete graph, on the adjacency matrix \( M \), though the single steady-state is maintained for every topology in which every node has at least one neighbour of every type different from its own.

A particularly simple example of the latter is a ring-like topology in which each node is connected to its \((X - 1)/2\) first neighbours on each side (where \( X \) is the diversity) and types are distributed periodically (see Fig. 5A). In such a system, if all \( X \) queues start equal at all nodes, they will all shrink at the same rate if \( \beta = \alpha/X \). If \( \beta < \alpha/X \), jobs are ‘exported’ too slowly, whilst on the contrary, if \( \beta > \alpha/X \), they are ‘imported’ too fast, which results in a temporary build-up of the queue corresponding to jobs that can be processed locally (see Fig. 5B).

However, the modelling approach quickly becomes intractable when used to describe larger and/or more realistic systems, especially when the topology is dynamic, which is the very essence of self-aggregation. This is why, for the remainder of this section, we rely on numerical experiments rather than analysis to investigate and quantify emergent load-balancing properties.

### 4.3. Simulated implementation

In order to experiment with the collaborative computing and load-balancing abilities of our overlay construction and maintenance framework, we devised a typical distributed processing scenario, as well as a modified set of local rules to inform the decision of requesting a neighbour of a particular type.

At initialization, we compute a global workload distributed across the system according to a number of criteria (tuneable parameters). We then measure the global effect of rewiring on co-operative processing in terms of the evolution of that workload over time (no additional requests are made at run-time). This is of course meant as a demonstrator, featuring an easily measurable variable, i.e. the monotonically decreasing fraction of the initial workload still waiting to be processed. In practice, the same decision rules would take as their input the local structure of the demand in terms of service requests (type and arrival rate), not the properties of a static workload.

The two key parameters basically determine whether the demand is homogeneously distributed or not and what fraction of individual components receive requests for services that they do not provide directly. Defining heterogeneity of the workload is done by arbitrarily designating a specified fraction of nodes as being ‘overloaded’. This simply means that, in initial conditions, the number of requests queuing at these nodes is multiplied by a factor 10. A second parameter determines what fraction of the nodes are designated ‘specialized’ entry points. A generic entry point starts with all service types equally represented in the local load, whilst all requests queuing at a specialized entry point correspond to its particular type. Figure 6 shows an example of each of the four possible initial states (‘normal/generic’, ‘normal/specialized’, ‘overloaded/generic’ and ‘overloaded/specialized’).

The load-balancing logic is as follows. At every time-step, every node goes through the entire list of its first neighbours. For every one of them, it checks whether the queue maintained by that neighbour for the service that it provides is shorter than the corresponding local queue. If it is, one request is

![FIGURE 5. (A) Ring topology with periodic distribution of types and (B) evolution of the local queue lengths for variable \( \alpha/\beta \) ratio.](image-url)
transferred (one step toward evening the load). This is done independently of whether both nodes are of the same type or not. Obviously, if the node initiating the queue length comparison is a specialized entry point for a service that is different from the one provided by that particular neighbour, no transfer can occur (as the local queue will be empty).

The rewiring process is as follows. At every time-step, every node has a fixed probability of initiating a neighbour request. If it does, it follows the same procedure as the one described in Sections 2 and 3, apart from two modifications:

- The requested type is not based on the initiator being arbitrarily set in ‘clustering’ (same type) or ‘reverse-clustering’ (any other type) mode. Instead, it is designed to reflect the current workload. Basically, one specific request (not one specific queue or request type) is selected at random among all those queuing locally and determines the desired type. This ensures that the probability of requesting a neighbour providing a given service is linearly proportional to the relative length of the corresponding queue (which mimics a rational decision by the initiator to try to establish the most useful collaborative link). It should be noted that this logic also implies that a node without any requests queuing never initiates a rewiring attempt.
- To keep things relatively simple at this stage, the candidate proposed by the matchmaker does not perform any compatibility test: accepting the rewiring is automatic and mandatory (on the candidate’s side).

At initialization, all nodes are embedded in a random graph so as to reach a specified target average node degree (cf. Section 2). In all the simulations presented here, the population size is 1000 and the average node degree is 4. The diversity (number of service types) is set to 10.

4.4. Results

In order to present results in the clearest possible way, we introduce the notion of a ‘cumulative penalty’ that is incremented by one unit at every time-step for every request still queuing in the entire system. Obviously, this is an abstraction: in practice, a real penalty should only be incurred if queuing time exceeds an acceptable limit, probably defined as part of

FIGURE 6. The four possible initial node states, with three different service types.
a service level agreement (SLA). However, it is a simple and convenient variable to use for the purpose of comparing global performance between rules and scenarios (i.e. ‘the lower, the better’).

Figure 7 shows individual simulation traces for multiple combinations of the most relevant parameter values, namely the fraction of ‘specialized’ access points, the fraction of overloaded elements and the probability (per time-step) that a node initiates a ‘neighbour request’ procedure ($P$).

The overall trend is that the advantage of allowing the overlay to self-organize (reduced penalty) is maximized when all nodes act as generic access points (i.e. must handle requests for all services). This is to be expected in the sense that, with an average degree of four, finding suitable neighbours to process all ten request types effectively requires rewiring during the course of the simulation. Conversely, if all nodes are specialized access points (i.e. only receive requests for the service they can perform in isolation), the effect of rewiring is improved load-balancing (i.e. overloaded elements can more efficiently delegate processing if their neighbours belong in majority to the same type, which is highly unlikely in the initial random graph conditions).

Indeed, in this scenario (100% specialized), the entire workload can in principle be processed locally (without request transfers) at the cost of substantially increased delays (which is the reason why all three curves, including $P = 0$, eventually saturate in Fig. 7B, indicating that there are no requests left queuing in the system).

The intermediate case (50% specialized) is probably the most interesting and realistic, as it implies the presence of nodes with different and potentially conflicting objectives (e.g. a generic access point could be teamed up with an overloaded specialist of a different type, resulting in neither of them being able to use each other’s service). The expectation...
here is that the continuous rewiring process can implicitly deal with such a situation, statistically resulting in the establishment of more efficient partnerships in the long term. The evolution of the cumulative penalty (a measure of global success) suggests that the higher the fraction of overloaded nodes, the more beneficial the self-organization of the overlay, which confirms that load-balancing is the clearest advantage of rewiring (compare Fig. 7C and D). However, this macroscopic trend doesn’t convey any useful information about the local, microscopic dynamics.

These are best explained by tracking the evolution of node degree over time, decomposing the signal based on node characteristics (normal versus overloaded, specialized versus generic). Results show that overloaded elements effectively succeed in surrounding themselves with ‘helpers’ (independently of whether they are specialized or not), resulting in higher average degree (see Fig. 8).

The fact that the separation appears earlier in the 0% specialized scenario ($t \sim 100$) than in the 100% specialized scenario ($t \sim 1000$) reveals that the reason why overloaded nodes gradually acquire a higher degree is because they continue requesting new neighbours after their ‘normal load’ counterparts have finished processing all their local requests and so have effectively stopped doing so (resulting in the asymmetry). Indeed, when specialized, a normal node has 1000 requests in its single populated queue at $t = 0$, versus 100 in every one of its ten queues for a generic access point, resulting in a 1000 versus 100 time-steps delay before normal nodes start idling and effectively become available as ‘helpers’ (1 request processed per time-step). Note

![Figure 8](https://example.com/figure8.png)

**FIGURE 8.** Evolution of average node degree, split by node category (1000 nodes, 10 types, 10% overloaded, 100 independent realizations): (A) 0% specialized, (B) 100% specialized, (C and D) 50% specialized (see text for details).
however that a generic access point can be idling and still have requests queuing (and so initiate rewiring attempts), as long as the local queue corresponding to its own type is empty. So even in the 0% specialized scenario, normal nodes are only gradually stopping to act as initiators after \( t = 100, \) hence the ‘slow start’ noticeable on Fig. 8A.

Figure 8D is shown to indicate that, for an identical load, generic access points are statistically more successful at acquiring new neighbours than specialized ones. This effect is explained by the fact that, in a scenario featuring ten different types, the probability of a specialized node repeatedly asking for a single kind of neighbour resulting in a successful handshake diminishes rapidly as the local supply for the requested type gets depleted.

The attentive reader may have noticed that in the ‘50% specialized, 10% overloaded’ scenario (Fig. 8C), the system in which there is no rewiring (\( P = 0 \)) briefly outperforms that in which \( P = 0.01. \) We currently have no satisfactory explanation for this phenomenon, although it seems possible that, during the course of the self-organization process, the overlay may go through an inefficient transitory phase that is simply too short to be observed for higher rewiring rates (e.g. \( P = 0.1 \)). Verifying this hypothesis and measuring the adverse effect of such transient sub-optimality will be the subject of future work.

5. DISCUSSION

The most obvious high-level implication of the findings presented in this paper is that it is possible to foster the emergence of usable macroscopic patterns in overlay networks by using partly random and information-poor local interactions between first neighbours. Indeed, if pattern formation in the absence of any global template is ubiquitous in nature (see e.g. [29]), it typically involves the propagation of motifs through excitation and/or cross-inhibition between first neighbours. This leads to differentiation in the context of a fixed, pre-existing set of links that determines \( a \ priori \) which system element(s) can directly influence another’s behaviour or characteristics. Here, on the contrary, order appears through the progressive alteration of the topology of the same network over which communication and interaction take place, without any change to the local properties defining the type of the many units involved. In effect, it is the indirect ‘migration’ of elements through the overlay in which they are embedded that presides to the self-organization process, hence the use of terms such as ‘aggregation’ and ‘reverse-clustering’.

Although there are many cases in which the reallocation of resources can amount to a form of differentiation in artificial systems (e.g. loading of selected software modules, reservation of memory and/or CPU cycles for specific processes. . .) there are others in which changing the local characteristics of individual units is impractical (e.g. when there are hardware constraints and/or strict limitations on resource availability) or undesirable (e.g. when the benefits of altering local properties are not guaranteed to outweigh the cost of doing so). In that context, a technique to discover and exploit collaboration opportunities by progressively turning an initially random web of interactions between specialized units into a reflection of mutually compatible objectives is bound to be of interest for the autonomic self-management of distributed applications.

Throughout this paper, we have deliberately held the view that the simplicity of the decision rules and the uncompromising locality of information were the most important characteristics of the kind of self-organization framework that we wanted to develop. This decision was based on two separate criteria. First, we wanted to be able to explore the influence of key factors in a meaningful way, which meant that we had to restrict the number of independent parameters. Second, we were committed to making the logic so simple and the information transfer so limited that (1) no realistic constraints in terms of the processing capabilities of individual nodes would prevent usage of our algorithm and (2) the necessarily wasteful ‘trial-and-error’ rewiring process would not translate into a massive communication overhead (thanks to the amount of data circulating in every attempt being so limited).

We are fully aware however that the success of our framework is dependent on some important conditions being met, the most obvious being that the number of types represented in the system has to be of the same order of magnitude as the average node degree. Indeed if there are many more types than a node has first neighbours, the probability that the required one is available within two hops of the initiator of a rewiring attempt (an implicit assumption underlying ‘on-demand’ clustering) can become too small for the self-organization process to take place (at least at a sufficient pace). In practice, if diversity becomes so high that it cannot be compensated by increasing connectivity (and hence the average degree), then the constraint on locality of information has to be relaxed.

This can be done in at least two different ways. One is by maintaining data repositories where nodes can ‘advertise’ their type and find the necessary information to locate the desired kind of neighbour. The obvious drawback to this variant of ‘publish-subscribe’ is that it requires to explicitly designate and maintain such repositories, which can severely undermine adaptability and robustness. The other is by propagating non-local information via some form of gossiping mechanism. The disadvantage of this option is of course that it can generate a lot of network traffic (e.g. if using flooding) and also requires individual nodes to be able to collect and store large amounts of information (each one effectively acting as its own data repository). But the important thing is that, even though they would negatively impact on agility and generality, such hybrid solutions that combine self-organizing properties with more conventional approaches to
distributed computing do exist and can be called upon if needed.

Our future work will involve continuing to document the influence of key factors beyond system size and diversity, such as connectivity. In particular, we will explore the opportunity to dispense with the total link conservation law, which has been identified as ‘artificial’ in the sense that it doesn’t reflect a realistic constraint for overlay networks (although a variant may need to be kept in place in order to limit the proliferation of new connections and the associated overhead traffic). We will seek to explicitly take into account aspects such as selfishness and trust, as well as the ‘price of cooperation’ (in terms of the difference between the cost of remotely accessing services and that of taking the necessary steps to be able to execute them locally). Also, we intend to combine the rewiring logic described in this paper with the type of differentiation mechanisms involving co-ordinated resource (re-)allocation that we investigated in previous work [28] in an attempt to diversify the portfolio of local rules available to designers of autonomic distributed applications.

Finally, throughout this paper, we have chosen to ignore the role of the underlying network infrastructure and the negative impact that a mismatch between the configuration of the physical layer and the topology of the overlay might have on the efficiency of our framework (e.g. if a ‘one-hop’ collaborative link in the overlay involves using a congested route). Clearly, this issue must be investigated and techniques for preventing such problems need to be identified and tested before aggregation-based resource management can be practically implemented.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the European Commission for funding the Integrated Project CASCADAS ‘Component-ware for Autonomic, Situation-aware Communications, And Dynamically Adaptable Services’ (FET Proactive Initiative, IST-2004-2.3.4 Situated and Autonomic Communications) within the 6th IST Framework Program. The paper represents the work and contribution of individual parties involved in the project. The authors also wish to thank other project partners for excellent and fruitful discussions.

REFERENCES


