Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 1 (2010) 2633–2642
Procedia Computer Science
www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2010

Towards fully autonomic peer-to-peer systems

Michele Amoretti[1,*]

Information Engineering, University of Parma

Abstract

Large-scale distributed applications are becoming more and more demanding in terms of efficiency and flexibility of the technological infrastructure, for which traditional solutions based on the client/server paradigm are not suitable. The peer-to-peer paradigm provides an appealing solution to this problem, allowing to deploy robust networks of collectors, providers and consumers of resources. One step beyond, in an autonomic computing vision, the structure of each peer, or at least its configuration, may be dynamically adjusted by an adaptive plan which determines successive re-configurations in response to the environment, and turns the P2P network in a complex adaptive system (CAS). We analyze a Grid Computing architecture whose resource sharing mechanisms are based on this framework.

Keywords: autonomic, peer-to-peer, evolutionary, Grid

1. Introduction

A peer-to-peer (P2P) network is a complex system whose elements (peer nodes, or simply nodes or peers) are collectors, providers and consumers of resources. Resource sharing is based on the collaboration among peers, each one having a limited view of the system and contributing to the implementation of distributed control algorithms.

The P2P paradigm defines distributed systems where all participating processes have the same importance [17]. In contrast with the client/server approach, in which resource providers and resource consumers are clearly distinct, on the contrary peers usually play both roles. Furthermore, a peer-to-peer system is a complex system, because it is composed of several interconnected parts that as a whole exhibit one or more properties (*i.e.* behavior) which are not easily inferred from the properties of the individual parts [16].

The activities of a peer-to-peer system are driven by the environment and by internal feedbacks (fig.1). The environment in which P2P networks operate is the set of users, that directly or indirectly act on peers and on resources that each node can use (running environment) and share with other nodes. Each environmental input usually targets a subset of the whole set of peers in the network. Examples of environmental inputs are:

- connection commands
- disconnection commands

* *Email address:* michele.amoretti@unipr.it (Michele Amoretti)
[1] Corresponding author

1877-0509 © 2010 Published by Elsevier Ltd.
doi:10.1016/j.procs.2010.04.297

- resource discovery queries
- job submissions
- events detected by sensors

Internal feedbacks are communication messages among peers. They may be autonomously generated by peers themselves, or triggered by environmental inputs (*e.g.* a resource discovery query that propagates from one peer to another).

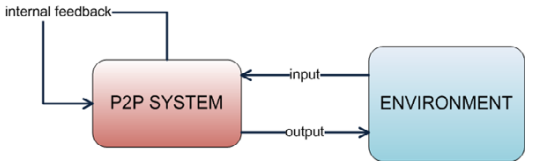

Figure 1: A P2P system and its environment.

When a peer receives an input (from the environment or from other peers), its internal structure maps the input to an output. The mapping process may require the peer to cooperate with other peers, exchanging messages in order to discover and eventually consume resources. For example, if a peer receives the request for a file, it firstly searches its resources; if the file is not found there, the peer propagates the request to its neighbors, or to a subset of them. It is important to point out that *localized reactions follow localized inputs*, which becomes more evident the more the system is large, *i.e.* composed by several thousands nodes.

Fig. 2 illustrates the distinction between local environment and neighbor peers. The local environment is perceived by the peer by means of its input interfaces and sensors: direct messages from users, but also contextual information (this is the typical case of ambient intelligence scenarios). The peer and its neighbors are part of the P2P system which is immersed in the environment (fig. 1). The response of the peer to the inputs that come from the local environment is usually contingent on interactions with neighbors (which in turn may involve their neighbors, etc.). The other way round, a peer can receive requests from its neighbors, and its response in general may depend and affect its local environment.

Figure 2: Interactions between a peer, its local environment and its neighbors in a traditional P2P architecture.

We adopt the following classification:

1. P2P systems in which each peer is owned by a user, which interacts with it by means of a user interface. This is the case of file sharing platforms, but also collaborative applications such as Skype [19]. These peer-to-peer systems are usually highly dynamic, since peers come and go with users;
2. P2P systems in which autonomous peers manage computational resources, sensors, actuators, etc. to provide services to users. These systems may be contextualized within many different fields, ranging from high-performance computing to ambient intelligence (AmI);
3. hybrid P2P systems in which some peers are owned and managed by users, while other peers are autonomous.

The behavior of a peer is determined by the configuration of its internal structure. Such configuration can be static, *i.e.* defined by static rules shared by every peer (protocols), or dynamic, *i.e.* adjusted by an adaptive plan which determines successive re-configurations in response to the environment, and turns the P2P network in a complex adaptive system (CAS) [11].

Many considerable peer-to-peer protocols have been proposed in the last decade. They can be grouped in few architectural models, taking into account basically two dimensions: the dispersion degree of information about shared

resources, and their logical organization [3]. The behavior of a peer-to-peer system based on protocols follows a pre-established pattern. *E.g.*, in Chord [20] messages are routed among peers according to a deterministic rule which considers the metric distance between message identifiers and peer identifiers.

On the other side, there is a lack of common understanding about adaptiveness. In [2] we introduced the Adaptive Evolutionary Framework (AEF), which defines autonomic behavior in terms of an adaptive plan (denoted as τ) that makes the peers' internal structure or configuration evolve, in order to adapt to the environment (fig. 3). In details, the adaptive plan τ is based on an evolutionary algorithm, which utilizes a *population* of individuals (*i.e.* a set of possible structures or configurations for a peer), where each individual represents a candidate solution to the considered problem. **In the general case, the structure of a peer is made of variables and procedures that may evolve, changing in number and semantics, based on interactions with other peers.** Such an evolution can be *phylogenetic*, for which memoryless transformations are applied to modify the structure or its configuration, or *cultural or epigenetic*, which assumes learning and knowledge transmission. With this approach, autonomic peer-to-peer networks emulate biological systems to cope with unforeseen scenarios, variations in the environment, or presence of deviant peers.

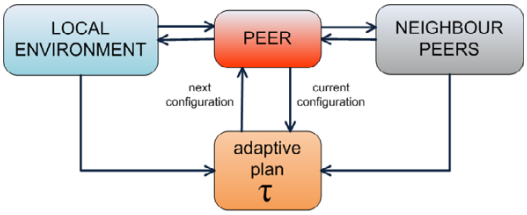

Figure 3: Interactions between a peer, local environment and neighbors in an adaptive P2P architecture.

Well-designed adaptation mechanisms, executed in background by each peer, should introduce little additional costs in terms of consumed local resources and additional message traffic in the network. After all, also the most advanced protocols of traditional peer-to-peer systems require costly periodic self-maintenance, such as stabilization of routing tables, etc.

In this paper we consider the case of fixed structures, each one consisting of the same p elements. The ith element ($i = 1, .., p$) of the structure has l_i possible values. The set of possible values for the ith element is defined as $A_i = \{a_{i1}, .., a_{il_i}\}$. Each structure configuration $\vec{C}$ represents a point in the search space $\mathcal{A}$ (*configuration space*), which is the domain of action of the adaptive plan. In other words, $\mathcal{A}$ is the set of all combinations of values for the elements of the structure:

$$\mathcal{A} = A_1 \times A_2 \times ... \times A_p \tag{1}$$

The adaptive plan τ produces a sequence of configurations, *i.e.* a trajectory through $\mathcal{A}$, following an evolutionary process. A very important step in the design of an evolutionary algorithm is to find an appropriate structure representation.

One case of particular importance is that in which the adaptive plan receives direct indication of the performance of each configuration it tries. That is, a part of the input I is the *fitness function*, which maps a structure representation into a scalar value:

$$F : \mathcal{A}^+ \to \mathbb{R} \tag{2}$$

The fitness function quantifies the quality of a configuration, *i.e.* how close it is to optimality, with respect to the environment. It is therefore extremely important that the fitness function accurately models the problem, including all criteria to be optimized. Frequently the fitness function does not directly evaluate the elements of a structure, but their expression as physical features of behaviors. If the fitness function represents a cost (as usual), it should return lower values for better structures.

The following section describes how AEF can be implemented with genetic algorithms. Other implementation strategies are shortly reported in section 3. An AEF-based architecture for sharing consumable resources such as disk space, CPU cycles, etc. in a Grid Computing environment is proposed in section 4. Its performance evaluation is presented in section 5. After a discussion about related work in section 6, the main achievements of this paper are summarized in section 7, together with a proposal for future work.

2. Implementing the AEF with genetic algorithms

Genetic algorithms (GA), introduced by John Holland in 1975, have been the first phylogenetic evolutionary computing paradigm to be developed and applied [12]. For their relative simplicity, in our opinion GAs are a very attractive evolutionary approach for the implementation of AEF-based autonomic peer-to-peer systems.

If the adaptive plan τ is a genetic algorithm, we say that the peer's configuration is the code that maps the way the peer looks and/or acts (*phenotype*). In other words, the configuration is the *genotype*, which is also called *chromosome*, in evolutionary computing (in genetics this simplification is not correct) [12], [7].

As previously stated, we assume each peer to be characterized by a fixed structure, consisting of p specified *genes* (in the general framework we used the generic term "elements" [2]). The ith gene ($i = 1, .., p$) of the structure has l_i possible *alleles* (previously called "values"); the set of possible alleles for the ith element is defined as $A_i = \{a_{i1}, .., a_{il_i}\}$, assuming finite values in limited specific ranges. Each configuration $\vec{C}$ represents a point in the search space $\mathcal{A}$, which is the set of all combinations of alleles (*i.e.* the *genome*). The pseudocode in Algorithm 1 describes the general GA scheme.

Algorithm 1 General GA scheme

```
1: Let g = 0
2: Define initial population C_g = {C⃗_{g,w} | w = 1, ..., W}
3: while not converged do
4:     Evaluate the fitness of each individual C⃗_{g,w} ∈ C_g
5:     g = g + 1
6:     Select parents from C_{g−1}
7:     Recombine selected parents through cross-over to form offspring O_g
8:     Mutate offspring in O_g
9:     Select the new generation C_g from the previous generation C_{g−1} and the offspring O_g
10: end while
```

In GAs, most frequently used operators are reproduction (cross-over, mutation) and elitism, which act on genotypes. Throughout the adaptation process, the selection operator is also used, in order to emphasize best configurations (chromosomes) in a population. The interplay of these operators leads to system optimization. Whenever the entities reproduce they create a surplus, variation amounts to innovation of novel entities (*i.e.* reproduction is not simply cloning), and finally selection takes care of promoting the right variants by discarding the poor ones [6], [7].

The original GAs developed by Holland had distinct features: (1) a bit string representation (in particular, Gray coding), (2) proportional selection, and (3) cross-over as the primary method to produce new individuals. GAs have also been developed that use integer or real-valued representations [18], and order-based representations where the order of variables in a chromosome plays an important role [21].

In the context of GAs, the fitness function is something like

$$F(\vec{C}) = a_1 F_1(\vec{C}) + a_2 F_2(\vec{C}) + .. \tag{3}$$

i.e. a function which rates the chromosomes according to a number of criteria, the influence of each criterion being controlled by the related constant $a \in \mathbb{R}$. Function F represents a cost and returns lower values for better chromosomes. In general, also the a_i factors may be set as evolving parameters, adapting the weight of each F_i to the environment. This aspect is out of the scope of this paper and will be considered in future work.

3. Other implementation strategies for the AEF

Hales proposes an algorithm named *SLAC*, which means selfish link and behavior adaptation to produce cooperation [10]. SLAC is based on the *copy and re-wire* approach, whose basic algorithm assumes that peer nodes have the freedom to change the way they handle and dispatch requests to and from other nodes, and drop and make links to nodes they know about. In addition, it is assumed nodes have the ability to discover other nodes randomly from the network, compare their performance against other nodes and copy the links and (some of) the behaviors of other nodes. In details, over time nodes engage in some activity and generate some measure of utility U (which might be the number of downloaded files or jobs processed, depending on the domain). It is important to note that the utility

function is much less complex than the fitness function. Periodically, each node i compares its performance against another node j, randomly selected from the network population. If $U_i \leq U_j$ node i drops all current links and copies all links of node j and adds a link to node j itself. Also, periodically and with low probability, each node adapts its behavior and links in some randomized way using a kind of mutation operator. Mutation of the links involve removing all existing links and replacing them with a single link to a randomly chosen node. It has been shown by simulation that, when applied in a suitably large population, the algorithm over time follows a kind of evolutionary process in which nodes with high utility tend to replace nodes with low utility (with nodes periodically changing behavior and moving in the network).

4. An AEF-based architecture for Grid Computing

In this section we introduce a Grid Computing system based on AEF (implemented with genetic algorithms), which enables efficient and scalable sharing of consumable resources, *i.e.* resources that cannot be acquired (by replication) once discovered, but may only be directly used upon contracting with their hosts [22]. An example of consumable resource is disk space, which can be partitioned and allocated to requestors for the duration of a task, or in general for an arranged time. We would like to emphasize that this overlay scheme is not the main contribution of this paper. It is a simple AEF example, which can be interpreted as the evolutionary version of Gnutella [9], applied to consumable resource sharing rather than file sharing.

The envisioned networks can be represented as undirected graphs whose arcs stand for mutual knowledge among peers. The *node degree* k is the number of neighbors of each node. Its statistical distribution depends on the history and on the dynamics of the network, and may affect the performance of the distributed algorithms which are executed.

The resource discovery algorithm is based on *epidemic* propagation of queries [8]. Query messages generated by a peer are matched with local resources and eventually propagated to neighbors. Each peer has a cache which contains advertisements of resources previously discovered in other peers. The cache is used as a preferred alternative to random (*i.e.* blind) query propagation. Each query message has a time-to-live (TTL) which is the remaining number of hops before the query message itself expires (*i.e.* it is no longer propagated). Moreover, each peer caches received queries, in order to drop subsequent duplicates. In general, bio-inspired epidemic protocols have considerable benefits as they are robust against network failures, scalable and provide probabilistic reliability guarantees [1].

The resource discovery process is affected by the following parameters, representing the phenotype of each peer: f_k (fraction of neighbors targeted for query propagation), TTL_{max} (max number of hops for messages), and D_{max} (max size of the cache).

Traditional epidemic algorithms use fixed values for parameters, set in advance according to the results of some tuning process which should guarantee good performance with high probability. Actually, if all nodes would be configured with $f_k = 1$ and the same TTL value, the resulting scheme would be exactly Gnutella [9]. In our scheme, parameters are functions of the chromosome, thus randomly initialized when a peer is created and joins the network. Moreover, adaptive tuning of parameters is based on GAs, with fitness function $F(\Phi, \langle QHR \rangle)$ to be minimized, where $\Phi = \Phi(f_k, TTL_{max}, D_{max})$ and

$$\langle QHR \rangle = \frac{1}{k+1} \sum_{j=0}^{k} QHR_j \tag{4}$$

QHR_j is the query hit ratio, *i.e.* the number of query hits QH versus the number of queries Q, assuming index $j = 0$ for current peer and $j = 1, .., k$ for its neighbors.

Being shared resources (*e.g.* CPU, RAM, disk space) consumable, the discovery process required by a job to start and complete its execution may be penalized by an increasing request rate. In general, the fitness function must be chosen in order to make each peer adapt to the rate of user requests which directly or indirectly affect its running environment.

If the values of the parameters which define the phenotype increase, the search process becomes more expensive in terms of time/space, but at the same time the discovery probability (which may be approximated by QHR) should result to be improved. In its most simple form

$$\Phi = \phi_0 f_k + \phi_1 TTL_{max} + \phi_2 D_{max} \tag{5}$$

where constant weights $\phi_0, \phi_1, \phi_2 \in \mathbb{R}$ control the influence of each parameter. In this case, the fitness function should reward peers with smaller Φ value, having the same *sufficiently high* $\langle QHR \rangle$ value.

The genotype $\vec{C}$ of each peer is given by three genes $\{C_0, C_1, C_2\}$ with values in a limited subset of $\mathbb{N}$. The relationship between the phenotype and the genotype is given by the following equations:

- $f_k = c_0 C_0$
- $TTL_{max} = c_1 C_1$
- $D_{max} = c_2 C_2$

where $c_i \in \mathbb{R}$, (with $i = 0, 1, 2$) are constants.

The pseudo-code in Algorithm 2 describes the adaptation process which is periodically executed by each peer.

Algorithm 2 Adaptation

```
Let g = 0
while not converged do
    Evaluate the fitness of own chromosome
    g = g + 1
    Select neighbor with best fitting chromosome
    Perform cross-over with best neighbor to generate offspring O_g = {O⃗_g1, O⃗_g2}
    Mutate offspring in O_g with probability 1 − ⟨QHR⟩
    Select the new generation C_g from the previous generation C_g−1 and the offspring O_g
end while
```

The best neighbor is chosen using proportional selection, *i.e.* the chance of individuals (neighbors' chromosomes) of being selected is inversely proportional to their fitness values. Cross-over is always performed, with a randomly-generated crosspoint. Mutation depends on $\langle QHR \rangle$, being highly improbable if the average query hit ratio of the peer and its neighbors tends to 1. The final selection between current peer's chromosome and the mutated offspring is random, with probability that is inversely proportional to their fitness values.

5. Simulation Experiments

The performance evaluation of the designed Grid Computing system has been carried out by means of the Discrete Event Universal Simulator (DEUS) [4]. Such a tool provides a Java API for the implementation of nodes, events and processes, and a straightforward but powerful visual tool for configuring simulations of complex systems.

Simulated peers have been characterized by three kinds of consumable resources: CPU, RAM, and disk space. Their values have been randomly generated (with uniform distribution) as multiple of, respectively, 512 MHz, 256 MB, and 10 GB. The maximum amount of resources per peer was 2 GHz, 1 GB for the RAM, and 100 GB for the disk space.

We have assumed $C_i \in [1, 6]$, $f_k = C_0/6$, $TTL_{max} = C_1$, $D_{max} = 2C_2$, and we have evaluated the performance of the adaptation scheme with respect to the following fitness functions:

$$F1(\Phi, \langle QHR \rangle) = \begin{cases} 1/\Phi & \text{if } \langle QHR \rangle < Th \\ \Phi & \text{if } \langle QHR \rangle > Th \end{cases}$$

$$F2(\Phi, \langle QHR \rangle) = \frac{1}{\delta^2}(1 - \langle QHR \rangle)\frac{1}{\Phi} + \langle QHR \rangle \Phi$$

In these fitness functions, Th is a threshold value, Φ is given by eq. 5, with $\phi_0 = 100$, $\phi_1 = 10$, and $\phi_2 = 5$, for which f_k has more weight than TTL_{max} and D_{max} in the computation of the fitness value. Of course $\langle QHR \rangle$ is given by eq. 4. We have assumed that each peer at the beginning had $QHR = 0.5$.

All the experiments have been carried out considering two different networks. The first one (called "fixed" in the following) is completely created before resource discovery starts, it is made of $N = 10000$ nodes and does not change during the simulation. This network is grown without preferential attachment, with N_0 completely connected

nodes, and each other joining peer choosing $m \in [1, N_0]$ existing peers, with even probability. All connections are bidirectional, *i.e.* if node n_a has node n_b in its peerview, then n_b has n_a in its peerview. The resulting degree distribution is exponential:

$$P(k) = (1 - e^{-\frac{1}{m}})e^{1-\frac{k}{m}} \forall k \geq m \tag{6}$$

In the experiments we have used $m = 3$. The second network (called "noisy" in the following) is given by the modification of the fixed network, throughout the simulation, by node departures and arrivals regulated by the same Poisson process with $\lambda = 10^{-1}s{-}1$. Since arrivals and departures have the same rate, the average network size is still $N = 10000$. As shown in figure 4, the node degree distribution is the same of the fixed network.

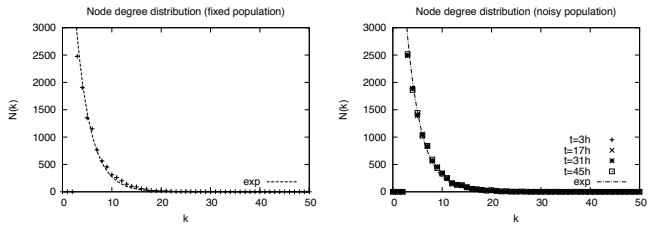

Figure 4: Node degree distribution for the fixed (left) and noisy (right) networks.

We have simulated about 56 h of the life of a network of peers, with resource queries scheduled according to a Poisson process with fixed rate $\lambda = 0.25$ s, *i.e.* one query every 4 seconds, associated to a randomly chosen node. Each query is a request for a randomly generated amount of resource (no more than 2 GHz of CPU, 1 GB of RAM, and 100 GB of disk space, respectively). Once a resource is found, it is consumed for a randomly distributed time interval (Poisson with 14 h average).

Firstly, we have studied the performance of the system without adaptation, considering peers having constant values for their genes. In details, we have adopted the three settings $C_i = 1, 3, 5$, and we have measured the evolution over time of the average *QHR*, considering peers which have sent at least one query (fig. 5). In this case, the performance depends on how parameters are pre-fixed. Setting $C_i = 1$ for all peers is not sufficient for covering the network in search for resources, while setting $C_i = 5$ is not so much better than setting $C_i = 3$. But we must take into account that for other networks, with different size, connection strategy and evolution, this result could not be true anymore. For the natural unpredictability of peer-to-peer networks, the autonomic approach is usually better than any prefixed and not flexible strategy, as we show in the following.

We have simulated periodic adaptation performed by all peers every 5000 s $\simeq$ 83 min. In this case the evolution over time of the average *QHR* (considering only those peers which have sent at least one query) is different, as shown in fig. 6 (fixed population) and 7 (noisy population).

Fig. 8 and 9 illustrate the dynamics of the genes, averaged *considering the whole network*, respectively with fixed and noisy peer population. Considering the whole network means that initially $\langle QHR \rangle = 0.5$ for all peers, and only after a while the average *QHR* of the whole network and the average *QHR* of the searchers converge, because all peers have performed at least one search.

In general, if *QHR* decreases for most peers, the system reacts in a way that chromosomes with high gene values survive and proliferate. When the network is fixed, both fitness function make the genes C_i increase over time, in order to maximize the *QHR*. It is interesting that, when the network is noisy, the values of the genes converge (to 4 for F1, and to 3 for F2). The reason of the quick stabilization is that joining and leaving nodes (*i.e.* the network noise) introduce random chromosomes that have the same effect of mutation.

6. Related Work

An almost complete survey on peer-to-peer resource sharing models for Grid systems is proposed in a recent work of Trunfio *et al.* [23]. Among others, the approach proposed by Iamnitchi and Foster [13] is interesting because it

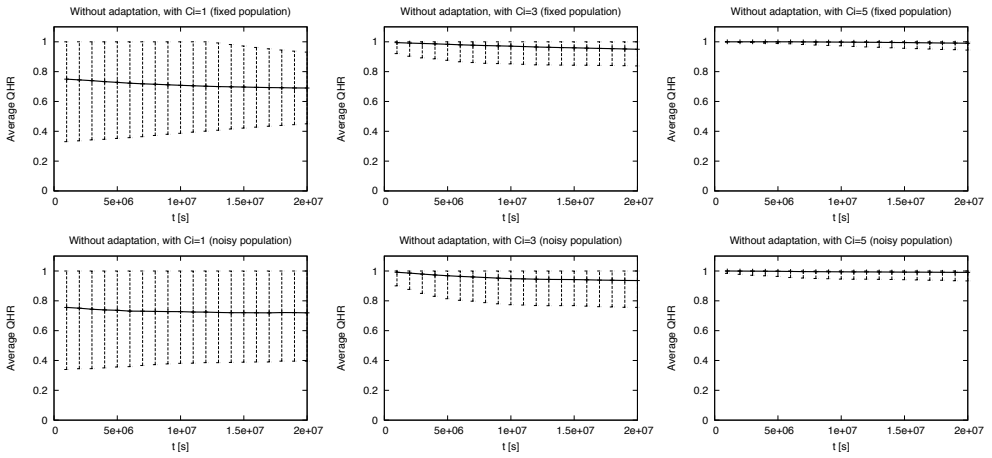

Figure 5: Average *QHR* over time for networks with fixed (top) and noisy (bottom) population, for different configurations of peers' genes.

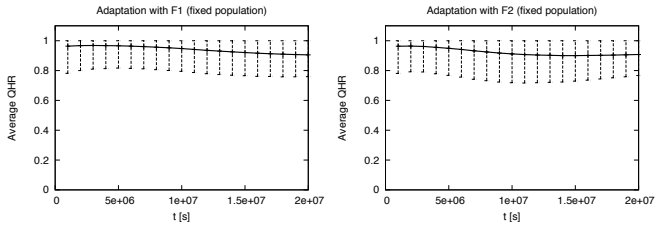

Figure 6: Average *QHR* over time for networks with fixed population, comparing fitness functions $F1$ and $F2$.

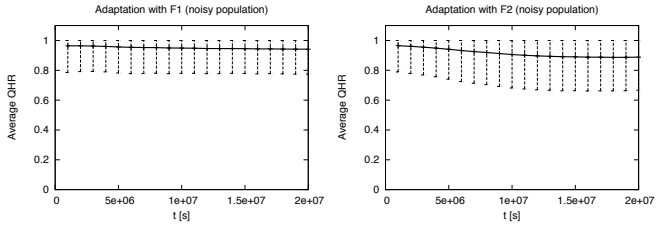

Figure 7: Average *QHR* over time for networks with noisy population, comparing fitness functions $F1$ and $F2$.

compares four strategies for epidemic request propagation, including one based on epigenetic (*i.e.* learning-based) evolution of the internal structure of peers.

A constructive criticism to both epidemic algorithms and mechanisms based on spanning trees (Chord-like) has been provided by Merz and Gorunova [15]. Being epidemic algorithms easy to implement, fault-tolerant, scalable, but slow, and spanning trees much more efficient but not resilient, the authors propose an hybrid approach. Epidemic dissemination is used to maintain the overlay, with an efficient mechanism for multicasting information. Simulation results show that the combination of the two methods is more efficient than the methods alone. Scalability up to 100000 peers is also shown. The authors defer to future work a detailed evaluation considering failures, high churn and presence of malicious peers.

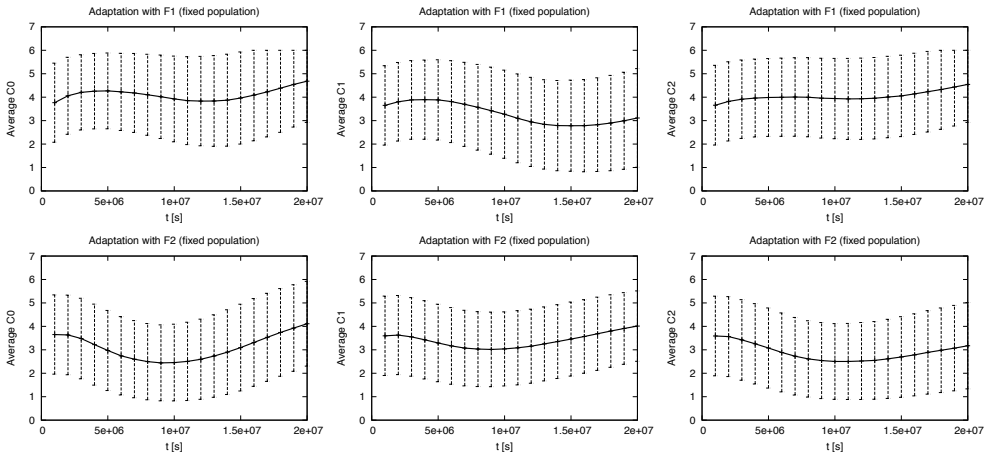

Figure 8: Evolution over time of genes C_0, C_1, C_2 for networks with fixed population, comparing fitness functions $F1$ and $F2$.

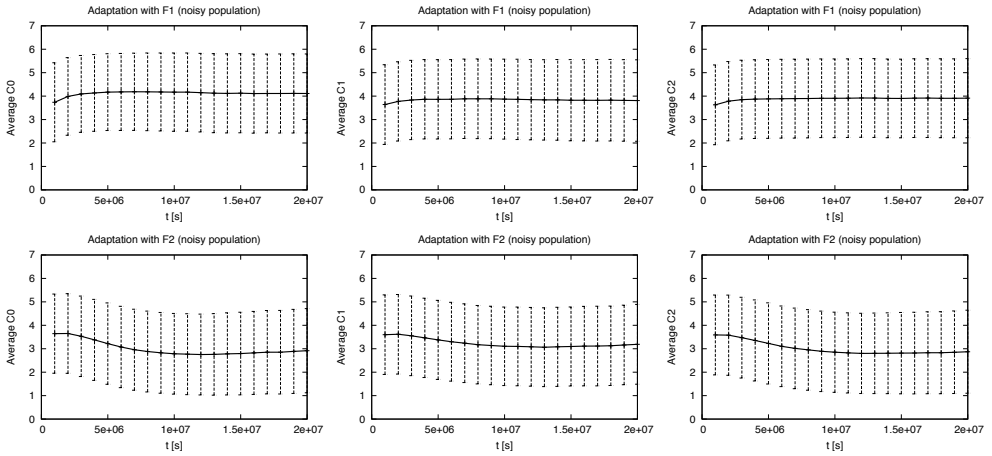

Figure 9: Evolution over time of genes C_0, C_1, C_2 for networks with noisy population, comparing fitness functions $F1$ and $F2$.

The feasibility of a fully decentralized evolutionary algorithm, which is the basic idea on which AEF is designed, is demonstrated in a recent work of Wickramasinghe *et al.* [24]. The authors describe and evaluate a fully distributed P2P evolutionary algorithm where decisions regarding survival and reproduction are taken by the individuals themselves independently, without any central control. This allows for a fully distributed evolutionary algorithm, where not only reproduction (crossover and mutation) but also selection is performed at local level. An unwanted consequence of adding and removing individuals in a non-synchronized manner is that the population size gets out of control too. This problem is solved by adding an adaptation mechanism allowing individuals to regulate their own selection pressure. The key to this is a gossiping algorithm that enables individuals to maintain estimates on the size and the fitness of the population.

Kessler *et al.* [14] propose to use genetic algorithms to find optimal network structures, by means of offline simulations. Each individual population member (chromosome) encodes a network design with connected superpeers and clusters of leaf nodes forming superpeer groups. The fitness of a population member is directly related to how

successful the particular network configuration is in answering resource queries. Our GA-based AEF architecture is more appealing, since re-configuration is an online process whose cost is distributed among peers.

Finally, Di Stefano and Santoro [5] propose a non-epidemic resource finding strategy based on the analogy between the 3D surface of available resources, where valleys correspond to nodes with a large quantity of available resources, and a potential field. Finding the node fulfilling a new job request entails navigating the surface trying to achieve the global minimum, which represents the node with the highest quantity of available resource. In the future we would like to quantitatively compare this approach with the scheme we illustrated in section 4.

7. Conclusion

We applied the Adaptive Evolutionary Framework (AEF), implemented with genetic algorithms, to a Grid Computing architecture where peer participants share their consumable resources. Performance evaluation by means of simulation has shown the effectiveness of the AEF approach. As future work, we plan to study the same Grid problem with other challenging environmental settings, *e.g.* highly dynamic workloads and churn rates. Moreover, we will apply the AEF to the design of other large-scale distributed systems. In particular, we will focus on techniques for evolving the internal structure of peers, rather than their configuration.

[1] E. Ahi, M. Caglar, O. Ozkasap, *Stepwise Fair-Share Buffering underneath Bio-inspired P2P Data Dissemination*, Proc. 6th International Symposium on Parallel and Distributed Computing (ISPDC'07), Hagenberg, Austria, 2007.

[2] M. Amoretti, *A Framework for Evolutionary Peer-to-Peer Overlay Schemes*, European Workshops on the Applications of Evolutionary Computation (EvoWorkshops 2009), Tubingen, Germany, April 2009.

[3] M. Amoretti, *A Survey of Peer-to-Peer Overlay Schemes: Effectiveness, Efficiency and Security*, Recent Patents on Computer Science, pp. 195-213, Vol. 2, Issue 3, ISSN 1874-4796, Ed. Bentham Science, November 2009.

[4] M. Amoretti, M. Agosti, F. Zanichelli, DEUS: a Discrete Event Universal Simulator, Proc. of the 2nd ICST/ACM International Conference on Simulation Tools and Techniques (SIMUTools 2009), Roma, Italy, March 2009.

[5] A. Di Stefano, C. Santoro, *A Decentralized Strategy for Resource Allocation*, Proc. Int'l Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05), pp. 295-300, Linkoping, Sweden, 2005.

[6] A. E. Eiben, *Evolutionary Computing and Autonomic Computing: Shared Problems, Shared Solutions?*, Self-Star Properties in Complex Information Systems, LNCS No. 3460, p.36-48, Springer, 2005.

[7] A. Engelbrecht, *Computational Intelligence: An Introduction*, Wiley & Sons, 2nd edition, 2007.

[8] P.T. Eugster, R. Guerraoui, A.-M. Kermarrec, L. Massoulie, *From Epidemics to Distributed Computing*, IEEE Computer, Vol. 37, No. 5, May 2004.

[9] The Gnutella Protocol Specification 0.4, *http://rfc-gnutella.sourceforge.net/developer/stable/index.html*

[10] D. Hales, *From Selfish Nodes to Cooperative networks - Emergent Link-based incentives in Peer-to-Peer Networks*, Proc. 4th IEEE Int'l Conference on Peer-to-Peer Computing (P2P'04), Zurich, Switzerland, 2004.

[11] F. Heylighen, *The Science of Self-organization and Adaptivity*, Knowledge Management, Organizational Intelligence and Learning, and Complexity, in The Encyclopedia of Life Support Systems (EOLSS), L. D. Kiel (ed.), Eolss Publishers, Oxford, 2001.

[12] J. Holland, *Adaptation in Natural and Artificial Systems*, The MIT Press, 1992.

[13] A. Iamnitchi, I. Foster, *A Peer-to-Peer Approach to Resource Location in Grid Environments*, In J. Weglarz, J. Nabrzyski, J. Schopf, and M. Stroinski (eds.), Grid Resource Management, Kluwer Publishing, 2003.

[14] J. Kessler, K. Rasheed, I. Budak Arpinar, *Using genetic algorithms to reorganize superpeer structure in peer to peer networks*, Applied Intelligence, Vol. 26, No. 1, pp. 35-52, 2007.

[15] P. Merz, K. Gorunova, *Fault-tolerant Resource Discovery in Peer-to-peer Grids*, Journal of Grid Computing, Vol. 5, No. 3, September 2007.

[16] J. M. Ottino, *Engineering complex systems*, Nature, 427, 399, 2004.

[17] D. Schoder, K. Fischbach, *Peer-to-Peer Paradigm*, Proc. 37th IEEE Int'l Conference on System Sciences, Hawaii, USA, 2004.

[18] S. Sevenster, A. P. Engelbrecht, *GARTNet: A Genetic Algorithm for Routing in Telecommunications Networks*, CESA96 IMACS/IEEE Multiconference on Computational Engineering in Systems Applications, Symposium on Control, Optimization and Supervision, pp. 1106-1111, 1996.

[19] Skype, *http://www.skype.com*

[20] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, H. Balakrishnan, *Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications*, IEEE/ACM Transactions on Networking, Vol.11, No.1, 2003.

[21] G. Syswerda, *Schedule Optimization using Genetic Algorithms*, L. Davis (ed.), Handbook of Genetic Algorithms, pp. 332-349, 1991.

[22] J. Tang, W. Zhang, W. Xiao, D. Tang, and J. Song, Self-Organizing Service-Oriented Peer Communities, *International Conference on Internet and Web Applications and Services (ICIW 2006)*, Guadeloupe, French Caribbean, February 2006, pages 99-103.

[23] P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, and S. Haridi: Peer-to-Peer resource discovery in Grids: Models and systems. Future Generation Computer Systems, vol. 23 (7), pp. 864-878 (2007)

[24] W. Wickramasinghe, M. van Steen, and A. E. Eiben. Peer-to-peer evolutionary algorithms with adaptive autonomous selection. In: 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 2007), pp. 1460-1467. ACMPress (2007)