A Framework for Evolutionary Peer-to-Peer Overlay Schemes

Michele Amoretti

Dipartimento di Ingegneria dell'Informazione, University of Parma,
Via Usberti 181/a, 43039 Parma, Italy
michele.amoretti@unipr.it

Abstract. In the last half-decade, many considerable peer-to-peer protocols have been proposed. They can be grouped in few architectural models, taking into account basically two dimensions: the dispersion degree of information about shared resources (centralized, decentralized, hybrid), and the logical organization (unstructured, structured). On the other side, there is a lack of common understanding about adaptive peer-to-peer sytems. In our view, peers' internal structure may change in order to adapt to the environment, according to an adaptation plan. To formalize this approach, we propose the Adaptive Evolutionary Framework (AEF). Moreover, we apply it to the problem of sharing consumable resources, such as CPU, RAM, and disk space.

1 Introduction

Recently, the peer-to-peer (P2P) paradigm has emerged as a highly appealing solution for scalable and high-throughput resource sharing among decentralized computational entities, by using appropriate information and communication systems without the necessity for central coordination [16]. In distributed systems based on the peer-to-peer paradigm all participating processes have the same importance. 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 network is a complex system, because it is composed of several interconnected parts (the peers) that as a whole exhibit one or more properties (*i.e.* behavior) which are not easily inferred from the properties of the individual parts [14]. The reaction of a peer to direct or indirect inputs from the environment is defined by its internal structure, which can be based either on static rules shared by every peer (protocols), or based on an adaptive plan τ which determines successive structural modifications in response to the environment, and turns the P2P network in a complex adaptive system (CAS) [9].

In the last half-decade, many considerable peer-to-peer protocols have been proposed. They can be grouped in few architectural models, taking into account basically two dimensions: the dispersion degree of information about shared resources (centralized, decentralized, hybrid), and the logical organization (unstructured, structured). The behavior of a peer-to-peer system based on protocols follows a pre-established pattern.

On the other side, there is a lack of common understanding about adaptiveness. In our view, illustrated by fig. 1, peers' internal structure may change in order to adapt to the environment. For example, consider a search algorithm whose parameters' values change over time in a different way for each peer depending on local performance indicators. The evolution of a structure can be *phylogenetic*, for which memoryless transformations are applied to the structure to modify it, or *cultural or epigenetic*, which assumes learning and knowledge transmission. In general, adaptive peer-to-peer networks emulate the ability of biological systems to cope with unforeseen scenarios, variations in the environment or presence of deviant peers.

Adaptation mechanisms running in background introduce additional costs in terms of local resources consumed and message traffic in the network. However, also the most advanced protocols of "traditional" peer-to-peer systems, *e.g.* Chord and Kademlia, require costly periodic self-maintenance, such as stabilization of routing tables, etc.

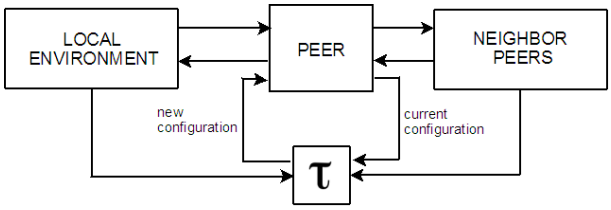

Fig. 1. Interactions between a peer, local environment and neighbors in an adaptive P2P architecture.

In fig. 1 we make a distinction between local environment and neighbor peers. The local environment is what the peer perceives with its sensors: direct inputs 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. 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.

In this paper we formalize our view by proposing the **Adaptive Evolutionary Framework (AEF)** for peer-to-peer architectures. In section 2 we define the theoretical foundation of AEF, with particular emphasis on adaptation based on genetic algorithms. To provide a concrete example, in section 3 we introduce the AEF strategy in a Gnutella-like overlay scheme for sharing consumable resources such as disk space, CPU cycles, etc. Related work on large-scale resource sharing architectures (*e.g.* Grids) is provided in section 4. Finally, section 5 concludes the paper summarizing its main results and proposing future work.

2 Adaptive Evolutionary Framework

The environment in which P2P networks operate is the set of users, robots, agents, etc. that directly or indirectly act on peers and on resources that each node can use and share with other nodes. Early P2P systems, both unstructured and structured, do not account any of the dynamics and heterogeneity of their environment. Systems like KaZaA and JXTA divide peers in supernodes and regular nodes, depending on the characteristics of their hosts, and provide strategies for dynamic promotions and demotions. Such kind of adaptation strategy may be affected by too much control overhead if the processing power available to peers, which depends not only on their hardware resources but also on usage load, quickly changes over time. Other systems incorporate IP network topology awareness into their design. An example is Pastry's proximity metric, for which each node key match in the routing table is tagged with the distance, in terms of IP hops, of the current node from the given IP address [15]. The drawback of this approach is its limited scope. A more comprehensive adaptation strategy has been proposed by Kalyvianki *et al.* [11], where each node analyzes log traces in order to find node capacity as perceived by the P2P application, by processing raw events such as frequency of schedule changes, number of running processes and duration of file system operations. Resulting information is periodically published in the overlay network, in order to be used by P2P applications to select nodes with best overall performance. The limit of this approach is the high bandwidth consumption due to control messaging, which could be excessive in case of high dynamism in the availability of each node's resources.

We propose the Adaptive Evolutionary Framework (AEF) as a response to these issues. According to the AEF, the internal structure of the peer is based on an adaptive plan τ which determines successive structural modifications in response to the environment. The adaptive plan, in the AEF framework, is based on an evolutionary algorithm, which utilizes a *population* of individuals (structures), where each individual represents a candidate solution to the considered problem.

Each structure $\boldsymbol{C}$ consists of p specified 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 represents a point in the search space $\mathcal{A}$, 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 structures, *i.e.* a trajectory through $\mathcal{A}$, by emulating the process of natural evolution. Before the evolutionary process can start, an initial population (being it a subset of $\mathcal{A}$) must be generated. The size of the initial population has consequences for performance in terms of accuracy and the time to converge.

If all elements of a structure were binary, there would be 2^p possible structures. Considering the whole complex system, a network of N peers, we could

state that the number of possible configurations of the system is 2^{pN}. Fortunately, the environment usually affects a limited set of peers at a time. For example, a resource request is usually triggered by a user interacting with a peer, or by an application running on the same host (or in the same process space) of a peer which receives the request. We say that the peer is directly affected by the environment's input. The peer may own the requested resource, or may propagate the request to other peers. We say that these peers are indirectly affected by the environment's input. When the system receives an input, not all its nodes must be considered for evolution, but only those which are directly or indirectly affected by the environment's input. Thus, a reasonable tradeoff is to design τ plans at the level of single peer. Each node can evolve either autonomously or merging its structure with those of other nodes, *e.g.* of its k neighbors.

The total information received by τ up to generation g is given by the sequence $< I(0), I(1), .., I(g-1) >$, with $I \in \mathcal{I}$. The part of this information which is retained is the *memory*, and its domain is $\mathcal{M}$. Then the adaptive plan can be represented by

$$\tau : \mathcal{I} \times \mathcal{A}^+ \rightarrow \mathcal{A}^+ \tag{2}$$

where $\mathcal{A}^+ = \mathcal{A} \times \mathcal{M}$.

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

$$F : \mathcal{A}^+ \rightarrow \mathbb{R} \tag{3}$$

The fitness function quantifies the quality of a structure, *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.

In addition to optimization criteria, the fitness function can also reflect the constraints of the problem through penalization of those structures that violate constraints. Constraints can be encapsulated within the fitness function, but also incorporated in the adaptation process.

Each run of an evolutionary algorithm produces a new generation of structures, representing a set of new potential configurations for the peer. The best structure of the new generation $\boldsymbol{C}_{g+1}$ is then selected from the offspring. Each new generation is formed through the application of a set of operators Ω to selected individuals of the current population. Selections are influenced by information obtained from the environment, so that the plan τ typically generates different trajectories in different environments.

To implement the AEF, different evolutionary strategies may be applied. The most obvious is using genetic algorithms (GAs) [10], for their relative simplic-

ity, but other solutions may be considered (for example, the *SLAC* algorithm proposed by Hales [8] can be placed under the AEF umbrella).

Until here we have used the term "structure" to indicate what the adactive plan modifies in an evolutionary peer. If the adative plan is a genetic algorithm, we can say that the structure is the code that maps the way the peer looks and/or acts (*phenotype*). In other words, the structure is the *genotype*, which is also called *chromosome*, in evolutionary computing (in genetics this simplification is not correct) [10], [5].

Each structure $\boldsymbol{C}$ consists of p specified *genes* (in the general framework we used the generic term "elements"). 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 $A_i = \{a_{i1}, .., a_{il_i}\}$, assuming finite values in limited specific ranges. Each structure 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, assuming a set of W initial search points.

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 [4], [5].

Algorithm 1 General GA scheme

1: Let g = 0
2: Define initial population $C_g = \{\boldsymbol{C}_{g,w} | w = 1, ..., W\}$
3: **while** termination criteria not fulfilled **do**
4: Evaluate the fitness of each individual $\boldsymbol{C}_{g,w} \in C_g$
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 the context of GAs, the fitness function is something like

$$F(\boldsymbol{C}) = a_1 F_1(\boldsymbol{C}) + a_2 F_2(\boldsymbol{C}) + .. \tag{4}$$

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 An AEF-based overlay scheme for resource sharing

In this section we introduce a GA-based AEF system 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 [17]. 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 easily interpreted as the evolutionary version of Gnutella [7], applied to consumable resource sharing rather than file sharing.

Algorithm 2 is executed by each peer involved in a resource discovery process, which is based on *epidemic* propagation of queries [6]. 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].

Algorithm 2 Resource discovery

```
1: if resource locally available then
2:    Notify interested peer
3:    if interested peer still interested then
4:       Allocate resource for interested peer
5:       Add interested peer to neighbor list
6:    end if
7: else
8:    if resource advertisement in cache then
9:       Propagate query (TTL − 1) to cached peer
10:   else
11:      if TTL > 0 then
12:         Propagate query (TTL − 1) to some neighbors
13:      end if
14:   end if
15: end if
```

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
- 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. 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

$$\langle QHR \rangle = \frac{1}{k+1} \sum_{j=0}^{k} QHR_j \tag{5}$$

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

$$\Phi = \phi_0 f_k + \phi_1 TTL_{max} + \phi_2 D_{max} \tag{6}$$

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 $\boldsymbol{C}$ 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 pseudocode in Algorithm 3 describes the adaptation process which is periodically executed by each peer.

The best neighbor is chosen using proportional selection, *i.e.* the chance of individuals (neighbors' chromosomes) of being selected is inversely proportional to

Algorithm 3 REVOL adaptation scheme

1: Let g = 0
2: **while** not converged **do**
3: Evaluate the fitness of own chromosome
4: $g = g + 1$
5: Select neighbor with best fitting chromosome
6: Perform cross-over with best neighbor to generate offspring $O_g = \{\boldsymbol{O}_{g1}, \boldsymbol{O}_{g2}\}$
7: Mutate offspring in O_g with probability $1 - \langle QHR \rangle$
8: Select the new generation C_g from the previous generation C_{g-1} and the offspring O_g
9: **end while**

their fitness values. Cross-over is always performed, with a randomly-generated crosspoint. Mutation depends on $\langle QHR \rangle$, being highly improbable while 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.

We simulated this example scheme with the Discrete Event Universal Simulator (DEUS) [3]. This tool provides a simple Java API for the implementation of nodes, events and processes, and a straightforward but powerful XML schema for configuring simulations.

Among others, we considered a network with fixed size and nodes performing periodic adaptation based on the following fitness function:

$$F1(\Phi, \langle QHR \rangle) = \begin{cases} 1/\Phi & \text{if } \langle QHR \rangle < 0.99 \\ \Phi & \text{if } \langle QHR \rangle > 0.99 \end{cases}$$

The idea behind F1 is that when resource usage is too low (below the minimum needed to provide a good service, *i.e.* a good query hit ratio), the system adapts itself in order to increase it until the query hit ratio is optimum; aftar that, it tries to minimize resource usage.

If QHR decreases, the system reacts in a way that chromosomes with high gene values survive and proliferate. At the beginning of the simulation, all nodes have the default QHR value which is 0.5. After a while, almost all nodes have performed at least one search, thus they have a QHR value that reflects the results of their searches. As clearly illustrated by fig. 2, when QHR is low, the gene values tend to increase. This lead to successful searches, for which QHR increases and the gene values decrease to a stable low value.

4 Related Work

A constructive criticism to both epidemic algorithms and mechanisms based on spanning trees is provided by Merz and Gorunova [13]. Being epidemic algorithms easy to implement, fault-tolerant, scalable, but slow, and spanning trees much more efficient but unresilient, the authors propose an hybrid approach.

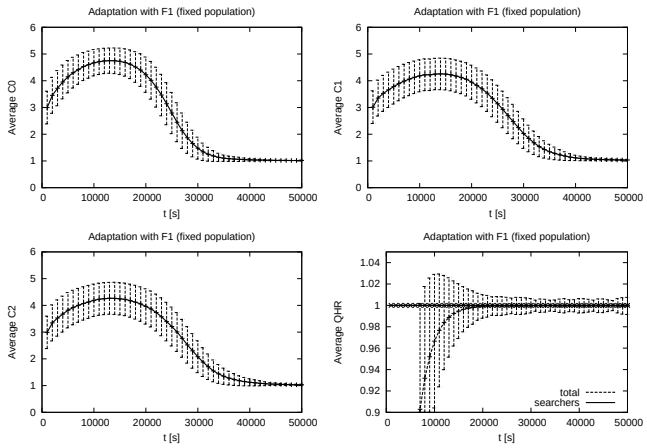

Fig. 2. Evolution over time of genes C_0, C_1, C_2 for a network with fixed population, fitness function F1, little load and adaptation taking place every 17 min.

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 10^5 peers is also shown. The authors defer to future work a detailed evaluation considering failures, high churn and presence of malicious peers.

Kessler *et al.* [12] 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. With respect to the GA-based AEF, for which evolution is an online process, this approach has many drawbacks.

A scheme that is suitable to be included in the AEF framework is illustrated in a recent work of Wickramasinghe *et al.* [18]. 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.

5 Conclusion

In this paper we formalized the Adaptive Evolutionary Framework (AEF), which is a framework for the design of self-configuring peer-to-peer overlay schemes. To show the potential of AEF, we used it to define a resource sharing scheme in which the evolutionary aspect is driven by a genetic algorithm.

In prospect, we are interested in finding other possible AEF implementations than genetic algorithms, considering also more complicated environments and applications, for which adaptiveness is not a plus but a fundamental requirement.

References

1. E. Ahi, M. Caglar, O. Ozkasap. Stepwise Fair-Share Buffering underneath Bio-inspired P2P Data Dissemination. In: 6th International Symposium on Parallel and Distributed Computing (ISPDC'07), Hagenberg, Austria (2007)
2. S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, E. Bonabeau. Self-Organization in Biological Systems. Princeton University Press (2001)
3. M. Agosti, M. Amoretti. Discrete Event Universal Simulator (DEUS). http://code.google.com/p/deus/
4. A. E. Eiben. Evolutionary Computing and Autonomic Computing: Shared Problems, Shared Solutions? In: Self-Star Properties in Complex Information Systems. LNCS, vol. 3460, pp. 36-48. Springer, Heidelberg (2005)
5. A. Engelbrecht. Computational Intelligence: An Introduction. 2nd edition, Wiley & Sons (2007)
6. P.T. Eugster, R. Guerraoui, A.-M. Kermarrec, L. Massoulie. From Epidemics to Distributed Computing. IEEE Computer, Vol. 37, No. 5 (2004)
7. Gnutella RFC homepage, *http://rfc-gnutella.sourceforge.net*
8. 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.
9. F. Heylighen. The Science of Self-organization and Adaptivity. In: The Encyclopedia of Life Support Systems (EOLSS), L. D. Kiel (ed.). Eolss Publishers, Oxford (2001)
10. J. Holland. Adaptation in Natural and Artificial Systems. The MIT Press, Cambridge (1992)
11. E. Kalyvianki, I. Pratt. Building Adaptive Peer-to-Peer Systems. In: 4th IEEE International Conference on Peer-to-Peer Computing (P2P'04), Zurich, Switzerland (2004)
12. 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)
13. P. Merz, K. Gorunova. Fault-tolerant Resource Discovery in Peer-to-peer Grids. Journal of Grid Computing, Vol. 5, No. 3 (2007)
14. J. M. Ottino. Engineering complex systems. Nature, vol. 427, p. 399 (2004)
15. A. Rowstron, P. Druschel. Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems. In: 3rd IFIP/ACM International Conference on Middleware, Heidelberg, Germany (2001)
16. D. Schoder, K. Fischbach. The Peer-to-Peer Paradigm. In: 38th Annual Hawaii International Conference on System Sciences (HICSS'05), Hawaii, USA (2005)
17. J. Tang, W. Zhang, W. Xiao, D. Tang, and J. Song. Self-Organizing Service-Oriented Peer Communities. In: International Conference on Internet and Web Applications and Services (ICIW 2006), Guadeloupe, French Caribbean (2006)
18. 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)