Towards Adaptive, Resilient and Self-Organizing Peer-to-Peer Systems

Alberto Montresor[1], Hein Meling[2], and Özalp Babaoğlu[1]

[1] Department of Computer Science, University of Bologna, Mura Anteo Zamboni 7, 40127 Bologna, (Italy), E-mail:{montresor,babaoglu}@CS.UniBO.IT
[2] Department of Telematics, Norwegian University of Science and Technology, O.S. Bragstadsplass 2A, N-7491 Trondheim (Norway), E-mail: meling@item.ntnu.no

Abstract. Peer-to-peer (P2P) systems are characterized by decentralized control, large scale and extreme dynamism of their operating environment. Developing applications that can cope with these characteristics requires a paradigm shift, placing adaptation, resilience and self-organization as primary concerns. In this note, we argue that *complex adaptive systems* (CAS), which have been used to explain certain biological, social and economical phenomena, can be the basis of a programming paradigm for P2P applications. In order to pursue this idea, we are developing Anthill, a framework to support the *design*, *implementation* and *evaluation* of P2P applications based on ideas such as multi-agent and evolutionary programming borrowed from CAS.

1 Introduction

Informally, peer-to-peer systems are distributed systems where all nodes are *peers* in the sense that they have equal role and responsibility. In fact, distributed computing was intended to be synonymous with peer-to-peer computing long before the term was invented, but this initial desire was subverted by the advent of client-server computing popularized by the World Wide Web.

The modern use of the term peer-to-peer (P2P) and distributed computing as intended by its pioneers, however, differ in several important aspects. First, P2P applications reach out to harness the outer edges of the Internet and consequently involve scales that were previously unimaginable. Second, P2P by definition, excludes any form of centralized structure, requiring control to be completely decentralized. Finally, and most importantly, the environments in which P2P applications are deployed exhibit extreme dynamism in structure, content and load. The topology of the system typically changes rapidly due to nodes voluntarily coming and going or due to involuntary events such as crashes and partitions. The load in the system may also shift rapidly from one region to another, for example, as certain files become "hot" in a file sharing system; or the computing needs of a node suddenly increase in a grid computing system.

2 Contribution

In order to deal with the scale and dynamism that characterize P2P systems, a paradigm shift is required that includes self-organization, adaptation and resilience as intrinsic properties rather than as afterthought. In this note, we suggest that *complex adaptive systems* (CAS) commonly used to explain the behavior of certain biological and social systems can be the basis of a programming paradigm for P2P applications. In the CAS framework, a system consists of a large number of relatively simple autonomous computing units, or *agents*. CAS typically exhibit what is called *emergent behavior*: the behavior of the agents, taken individually, may be easily understood, while the behavior of the system as a whole defies simple explanation. In other words, the interactions among agents, in spite of their simplicity, can give rise to richer and more complex patterns than those generated by single agents viewed in isolation.

As an instance of CAS drawn from nature, consider an ant colony. Several species of ants are known to group objects in their environment (e.g., dead corpses) into piles so as to clean up their nests. Observing this behavior, one could be mislead into thinking that the cleanup operation is being coordinated by some "leader" ants. Resnick [7] describes an artificial ant colony exhibiting this very same behavior in a simulated environment. Resnick's artificial ant follows three simple rules: (i) wander around randomly, until it encounters an object; (ii) if it was carrying an object, it drops the object and continues to wander randomly; (iii) if it was not carrying an object, it picks the object up and continues to wander. Despite their simplicity, a colony of these "unintelligent" ants is able to group objects into large clusters, independent of their initial distribution.

What renders CAS particularly attractive from a P2P perspective is the fact that global properties like adaptation, self-organization and resilience are achieved without explicitly embedding them into the individual agents. In the above example, there are no rules specific to initial conditions, unforeseen scenarios, variations in the environment or presence of failures. Yet, given large enough colonies, the global behavior is surprisingly adaptive and resilient.

In order to pursue these ideas, we are developing *Anthill*, a novel framework for P2P application development, based on ideas such as multi-agent systems and evolutionary programming borrowed from CAS [10, 6]. The goals of Anthill are to provide an environment that simplifies the design and deployment of P2P systems based on these paradigms, and to provide a "testbed" for studying and experimenting with CAS-based P2P systems in order to understand their properties and evaluate their performance.

In the next sections, we provide an overview of Anthill and we present the first results obtained through it, by presenting a load-balancing algorithm called *Messor*. Messor is a simple variant of the above artificial ant algorithm: ants drop an object they may be carrying only after having wandered about randomly "for a while" without encountering other objects. Colonies of such ants try to disperse objects (in the case of Messor, computational tasks) uniformly over their environment rather than clustering them. As such, they form the basis for a completely decentralized load balancing algorithm.

3 Anthill

Anthill uses terminology derived from the ant colony metaphor. An Anthill distributed system is composed of a self-organizing overlay network of interconnected *nests*. Each nest is a peer entity sharing its computational and storage resources. The network is characterized by the absence of a fixed structure, as nests come and go and discover each other on top of a communication substrate. Nests handle requests originated by local users, by generating one or more *ants* – autonomous agents that travel across the nest network trying to satisfy the request. Ants communicate indirectly by observing and modifying their environment, through information stored in the visited nests. For example, an ant-based implementation of a distributed lookup service could store routing information to guide subsequent ants towards a region of the network where the searched key is more likely to be found.

The aim of Anthill is to simplify P2P application development and deployment by freeing the programmer of all low-level details including communication, security and ant scheduling. Developers wishing to experiment with new protocols need to focus on designing appropriate ant algorithms using the Anthill API and defining the structure of the P2P system. When writing their protocols, developers may exploit a set of library components and services provided by nests. Examples of such services include failure detection, document downloading and ant scheduling for distributed computing applications.

A Java prototype of the Anthill runtime environment has been developed. The runtime environment is based on JXTA [4], an open-source P2P project promoted by Sun Microsystems. JXTA is aimed at establishing a network programming platform for P2P systems by identifying a small set of basic facilities necessary to support P2P applications and providing them as building blocks for higher-level services. The benefits of basing our implementation on JXTA are several. For example, JXTA allows the use of different transport layers for communication, including TCP/IP and HTTP, and deals with issues related to firewalls and NAT.

In addition to the runtime environment, Anthill includes a simulation environment to help developers analyze and evaluate the behavior of P2P systems. All simulation parameters, such as the structure of the network, the ant algorithms to be deployed, characteristics of the workload presented to the system, and properties to be measured, are specified using XML. Unlike other toolkits for multi-agent simulation [5], Anthill uses a single ant implementation in both the simulation and actual run-time environments, thus avoiding the cost of re-implementing ant algorithms before deploying them. This important feature has been achieved by a careful design of the Anthill API and by providing two distinct implementations of it for simulation and deployment.

In Anthill, we further exploit the "nature" metaphor through the use of evolutionary techniques for improving various characteristics of a P2P system. In particular, we make use of genetic algorithms [6] in tuning the ant algorithms used by the P2P system, by specifying optimization criteria and constraints for the parameters of the operating environment and ant algorithms.

4 Messor

The aim of Messor is to support highly parallel computations, such as the one performed by the Seti@Home [9] project, in which the workload may be subdivided in a large number of independent tasks. Unlike Seti@Home [9], however, Messor is not based on the master-slave paradigm, in which a well-known centralized master is responsible for supplying slave machines with computational tasks. In Messor, every node of the network is enabled to produce new tasks and introduce them in the network for computation. A swarm of Messor ants is responsible for exploring the network and balancing the workload by dispersing the tasks among all reachable nodes. Once computed, task results are sent back to the originator node, that may use appropriate mechanisms based on lease techniques to keep track of task assignments, in order to re-insert tasks that have been lost because they were assigned to crashed or partitioned nodes.

The Messor algorithm is a variation of the artificial ant algorithm illustrated in Section 2. Each ant can assume three different states: SearchMax, SearchMin and Transfer. While in the SearchMax state, the ant wanders across the network, looking for overloaded nodes. When a sufficient number of nodes has been visited, the ant switch to SearchMin state, during which the ant wanders across the network looking for underloaded nodes. Again, after a sufficient number of steps, the ant switch to the Transfer state, during which it transfers tasks from the most overloaded node to the most underloaded one, selected among those visited during the SearchMax and SearchMin phases. When the transfer state is completed, the ants switch to SearchMax and the process repeats itself.

The SearchMax and SearchMin walks are not performed completely at random. When wandering, ants collect information about the load of the last visited nodes. This information is stored in the nodes themselves and is used by ants to drive the SearchMax and SearchMin phases: at each step, the ant randomly selects the next node to visit among those that are believed to be more overloaded (in SearchMax) or underloaded (in SearchMin). In this way, ants move faster towards those regions of the network in which they are more interested.

Figure 1 shows some preliminary results obtained by Messor. More details about the algorithm and its performance can be found in a companion paper [2]. These results were obtained in a network of 100 idle nodes, by generating 100,000 tasks from a single node. At each iteration, all ants perform a single step by executing their algorithm and moving to the next node. As shown in the figure, after only 40 iterations, the workload is evenly balanced among all nodes.

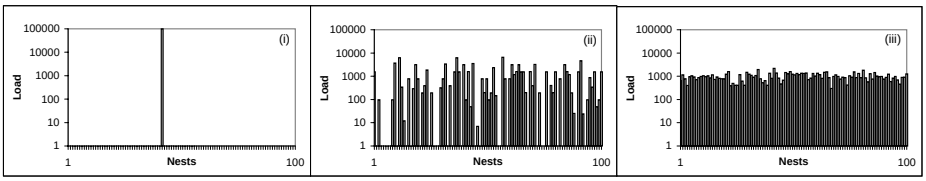

Fig. 1. Load distribution after (i) 0, (ii) 20, (iii) 40 iterations in the simulation.

5 Conclusions and Future Work

We have argued that ideas and techniques borrowed from CAS could form the basis for a new paradigm for building P2P systems that are adaptive, resilient and self-organizing. The approach we are advocating is quite different from those adopted in recent P2P routing algorithms [8, 11, 3] where complex protocols are required to reconfigure the routing tables in the event of nodes joining or leaving (voluntarily or due to crashes) the system. The adaptiveness and resilience of Anthill applications may be traced back to several sources. First, complex systems are composed of large number of entities, each of them interchangeable for another. Moreover, interconnections between entities are flexible, allowing transfer of tasks between entities, and communication throughout the system. Finally, the differences between entities enables a diversity of responses in a changing environment.

Algorithms developed in Anthill are often based on a probabilistic approach, and thus it is difficult to provide guarantees on their behavior. Nevertheless, our preliminary results are indeed interesting; using Anthill, we have implemented the load-balancing application briefly introduced in this paper, and we have realized a file sharing application called *Gnutant*, that again is inspired to the behavior of ants [1]. Gnutant ants builds a distributed index consisting of URLs to documents, by traversing the network looking for documents and leaving information trails to be followed in future searches.

References

1. O. Babaoğlu, H. Meling, and A. Montresor. Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems. In *Proc. of the 22th Int. Conf. on Distributed Computing Systems*, Wien, Austria, July 2002.
2. O. Babaoğlu, H. Meling, and A. Montresor. Implementing a Load-Balancing Algorithm in Anthill. Technical Report UBLCS-02-9, Dept. of Computer Science, University of Bologna, Apr. 2002.
3. S. R. et al. A Scalable Content-Addressable Network. In *Proc. of the ACM SIGCOMM'01*, San Diego, CA, 2001.
4. Project JXTA. http:// www. jxta. org.
5. N. Minar, R. Burkhart, C. Langton, and M. Askenazi. The Swarm Simulation System, A Toolkit for Building Multi-Agent Simulations. Technical report, Swarm Development Group, June 1996. http:// www. swarm. org.
6. M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Apr. 1998.
7. M. Resnick. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. MIT Press, 1994.
8. A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. of the 18th International Conference on Distributed Systems Platforms*, Heidelberg, Germany, Nov. 2001.
9. SETI@Home Home Page. http:// setiathome. ssl. berkeley. edu.
10. G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.
11. B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing. Technical Report UCB/CSD-01-1141, U.C. Berkeley, Apr. 2001.