Toward Self-Organizing, Self-Repairing and Resilient Distributed 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

1 Introduction

As access to networked applications become omnipresent through PC's, handheld and wireless devices, an increasing number of interactions in our day-to-day life with significant economical, social and cultural consequences depend on the reliability, availability and security of distributed systems. Not surprisingly, the increased demand that is being placed by users on networked services can only be met by distributed infrastructures with greater complexity and larger scale. And the extreme dynamism that is often present in different forms in modern systems further aggravates our ability to reason formally about their behavior.

Recent examples of these trends can be found in the *peer-to-peer* (P2P) and *ad-hoc networks* (AHN) application areas. P2P systems are distributed systems based on the concept of resource sharing by direct exchange between nodes that are considered *peers* in the sense that they all have equal role and responsibility [7]. Exchanged resources may include content, as in popular P2P document sharing applications, and CPU cycles or storage capacity, as in computational and storage grid systems. P2P systems exclude any form of centralized structure, requiring control to be completely decentralized. The P2P architecture enables true distributed computing, creating networks of resources that can potentially exhibit very high availability and fault-tolerance. In AHN, heterogeneous populations of mobile, wireless devices cooperate on specific tasks, exchanging information or simply interacting informally to relay information between themselves and the fixed network [10]. Communication in AHN is based on multihop routing among mobile nodes. Multihop routing offers numerous benefits: it extends the range of a base station; it allows power saving; and it allows wireless communication, without the use of base stations, between users located within a limited distance of one another.

Both P2P and AHN may be seen as instances of *dynamic networks* that are driven by large numbers of input sources (users/nodes) and that exhibit extreme variability in their structure and load. The topology of these systems typically changes rapidly due to nodes voluntarily joining or leaving the network, due to involuntary events such as crashes and network partitions, or due to frequent changes in interaction patterns. The load in the system may also shift rapidly

from one region to another; for example, when certain documents become "hot" in a document sharing system.

Traditional techniques for building distributed applications are proving to be inadequate in dealing with the aforementioned complexity. For example, centralized and top-down techniques for distributed applications based on client-server architectures may simplify system management, but often result in systems that are inflexible and unable to adapt to changing environmental conditions. In these systems, it is not uncommon for some minor perturbation (e.g., a software update or a failure) in some remote corner of the system to have unforeseen, and at times catastrophic, global repercussions. In addition to being fragile, many situations (e.g., adding/removing components, topology changes) arising from the extreme dynamism of these systems require manual intervention to keep distributed applications functioning correctly.

In our opinion, we are at a threshold moment in the evolution of distributed computing: the complexity of distributed applications has reached a level that puts them beyond our ability to design, deploy, manage and keep functioning correctly through traditional techniques. What is required is a paradigm shift, capable of confronting this complexity explosion and enabling the construction of resilient, scalable, self-organizing and self-repairing distributed systems.

2 Drawing Inspiration from Nature

The centralized, top-down paradigms that are the basis for current distributed systems are in strong contrast with the organization of many phenomena in the natural world where decentralization rules. While the behavior of natural systems may appear unpredictable and imprecise at a local level, many organisms and the ecosystems in which they live exhibit remarkable degrees of resilience and coordination at a global level. Social insect colonies, mammalian nervous and immune systems are a few examples of highly-resilient natural systems.

We argue that ideas and techniques from *complex adaptive systems* (CAS), which have been applied successfully to explain certain aspects of biological, social and economical phenomena, can also be the basis of a programming paradigm for distributed applications. The natural systems cited in the previous paragraph all happen to be instances of CAS which are characterized by complete lack of centralized coordination. In the CAS framework, a system consists of a large number of autonomous entities called *agents*, that individually have very simple behavior and that interact with each other in simple ways. Despite this simplicity, a system composed of large numbers of such agents often exhibits what is called *emergent behavior* [3] that is surprisingly complex and hard to predict. Furthermore, the emergent behavior of CAS is highly adaptive to changing environmental conditions and unforeseen scenarios, is resilient to deviant behavior (failures) and is self-organizing toward desirable global configurations.

Parallels between CAS and modern distributed systems are immediate. In this paper, we suggest using ideas and techniques derived from CAS to enable the

construction of resilient, scalable, self-organizing and self-repairing distributed systems as ensembles of autonomous agents that mimic the behavior of some natural or biological process. It is our belief that this approach offers a chance for application developers to meet the increasing challenges arising in dynamic network settings and obtain desirable global properties such as resilience, scalability and adaptability, without explicitly programming them into the individual agents.

As an instance of CAS drawn from nature, consider an ant colony. Several species of ants are known to group objects (e.g., dead corpses) into piles so as to clean up their nests. Observing this global behavior, one could be mislead into thinking that the cleanup operation is somehow being coordinated by a "leader" ant. Resnick [8] shows that this is not the case by describing an artificial ant colony exhibiting this very same behavior in a simulated environment. Resnick's artificial ants follows three simple rules: (i) wander around randomly, until they encounter an object; (ii) if they were carrying an object, they drop it and continue to wander randomly; (iii) if they were not carrying an object, they pick it up and continue to wander. Despite their simplicity, a colony of these "unintelligent" ants is able to group objects into large clusters, independent of their initial distribution in the environment. This simple algorithm could be used to design distributed data analysis and search algorithms, by enabling artificial ants to travel through a network and cluster "similar" information items. It is also possible to consider a simple variant (the inverse) of the above artificial ant that drops an object that it may be carrying only after having wandered about randomly "for a while" without encountering other objects. Colonies of such ants try to disperse objects uniformly over their environment rather than clustering them into piles. As such, they could form the basis for a distributed load balancing algorithm.

What renders CAS particularly attractive from a distributed systems perspective is the fact that global properties such as adaptation, self-organization and resilience are exactly those that are desirable to distributed applications, and we may obtain these properties without having to explicitly program them into the individual agents. In the above example, there are no rules for ant behavior specific to initial conditions, unforeseen scenarios, variations in the environment or presence of deviant ants (those that are "faulty" and do not follow the rules). Yet, given large enough colonies, the global behavior is surprisingly adaptive and resilient.

Instances of CAS drawn from nature have already been applied to numerous problems with notable success [2, 5]. In particular, artificial ant colonies have been used for solving complex optimization problems, including those arising in communication networks. For instance, numerous simulation studies have shown that packets in networks can be routed by artificial ants that mimic real ants that are able to locate the shortest path to a food source using only trails of chemical substances called *pheromones* deposited by other ants [2].

3 Current Work and Future Directions

In order to pursue these ideas further, we have initiated the *Anthill* project [1], whose aim is to design a novel framework for the development of peer-to-peer applications based on ideas borrowed from CAS and multi-agent systems. The goals of Anthill are to provide an environment that simplifies the design and deployment of novel P2P applications based on swarms of agents, and provide a "testbed" for studying and experimenting with CAS-based P2P systems in order to understand their properties and evaluate their performance.

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 reading or modifying their environment, through information stored in the visited nodes. For example, an ant-based distributed lookup service could leave routing information to guide subsequent ants toward 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]. The benefits of basing our implementation on JXTA are several, including the reuse of various transport layers for communication, and dealing 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 easily specified using XML.

After having developed a prototype of Anthill, we are now in the process of testing the viability of our ideas by developing CAS-based P2P applications, including a load-balancing algorithm for grid computing based on the "dispersing" ant described in the previous section [6] and a document-sharing application based on keyword pheromone trails left by exploring ants [1]. Preliminary simulation results appear to confirm our intuition that applying CAS-based techniques to solving challenging problems in distributed systems is a promising approach. The performance of our preliminary results are comparable to those obtained through traditional techniques. Yet, the approach followed to obtain

our algorithms is completely different from previous approaches, as we mimick the behavior of natural systems, thus inheriting their strong resilience and self-organizational properties.

Current efforts in applying CAS techniques to information systems can be characterized as *harvesting* — combing through nature, looking for a biological process having some interesting properties, and applying it to a technological problem by adapting it through an enlightened trial-and-error process. The result is a CAS that has been empirically obtained and that appears to solve a technological problem, but without any scientific explanation of why. In the future, we seek to develop a rigorous understanding of why a given CAS does or does not perform well for a given problem. A systematic study of the rules governing fitness of CAS offers a bottom-up opportunity to build more general understanding of the rules for CAS behavior. This study should not be limited to biological systems; other areas such as economics and game theory [9] can be rich sources of inspiration. The ultimate goal is the ability to synthesize a CAS that will perform well in solving a given task based on the accumulated understanding of its regularities when applied to different tasks. The achievement of this goal would enable the systematic exploitation of the potential of CAS, freeing technologists from having to comb through nature to find the desired behavior.

We conclude with a brief discussion of the inherent limitations of the CAS-based approach to building distributed systems. While natural systems exhibit desirable properties at a global level, their local behavior can be unpredictable and imprecise. Thus, we expect CAS-based techniques to be appropriate for distributed systems where probabilistic guarantees on desired global system behavior are sufficient. The technique is probably not appropriate for guaranteeing strong properties such as consistency, synchronization, and QoS.

References

1. Ö. 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*, Vienna, Austria, July 2002.
2. E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems.* Oxford University Press, 1999.
3. J. Holland. *Emergence: from Chaos to Order.* Oxford University Press, 1998.
4. Project JXTA. http:// www. jxta. org.
5. E. Klarreich. Inspired by Immunity. *Nature*, 415:468–470, Jan. 2002.
6. A. Montresor, H. Meling, and O. Babaoğlu. Messor: Load-Balancing through a Swarm of Autonomous Agents. In *Proc. of the 1st Workshop on Agent and Peer-to-Peer Systems*, Bologna, Italy, July 2002.
7. A. Oram, editor. *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology.* O'Reilly, Mar. 2001.
8. M. Resnick. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds.* MIT Press, 1994.
9. K. Ritzberger. *Foundations of Non-Cooperative Game Theory.* Oxford University Press, Jan. 2002.
10. C. Toh. *Ad Hoc Mobile Wireless Networks.* Prentice Hall, 2002.