Model-driven Walks for Resource Discovery in Peer-to-Peer Networks

M. Bakhouya and J. Gaber
Universite de Technologie de Belfort-Montbeliard (UTBM)
Rue Thierry Mieg 90010 Belfort Cedex, France

Abstract

In this paper, a distributed and adaptive approach for resource discovery in peer-to-peer networks is presented. This approach is based on the mobile agent paradigm and the random walk technique with reinforcement learning to allow for dynamic and self-adaptive resource discovery. More precisely, this approach augments random walks with a reinforcement learning technique where mobile agents are backtracked over the walked path in the network. A metric recording an affinity value that incorporates knowledge from past and present searches is maintained between nodes. The affinity value is used during a search to influence the selection of the next hop. This approach is evaluated with the network simulator ns2.

1. Introduction

Distributed computing such as peer-to-peer networks promise a distributed computing infrastructure that can provide globally available network resources. Their size and complexity continue to increase and allow an almost ubiquitous availability of resources. Resources are bound to specific hosts, representing hardware devises (e.g., disk), logical system objects (e.g., socket), files or software entities that are managed by an application.

In peer-to-peer networks, a resource discovery system to locate specific resources is required. More precisely, the resource discovery is a basic functionality that enhances the accessibility of resources in the distributed context: given a user request, a resource discovery mechanism should locate and return a set of peers' addresses that match the description of requested resources.

In a large-scale peer-to-peer network, resource discovery system should be decentralized and scalable in order to avoid potential computation bottlenecks such as the need for knowledge about a centralized location, and the need for updates to that location whenever resources characteristics or locations may change. Several systems are proposed in order to implement a resource discovery services. These systems can be classified as either structured or unstructured [1].

Structured architectures can be classified in indexation-based architectures and hashing-based architectures. In indexation-based architectures, typical resource discovery architectures, such as Napster [3], Kaaza [8], consists of three entities: resource providers that create and publish resources, resources brokers that maintain repositories of published resources to support their discovery, and resource requesters that search the resource broker's repositories. Repositories in resource discovery systems have traditionally a hierarchical architecture [8,17,18,19]. Indexation based architectures cannot meet the requirements of both scalability and adaptability simultaneously. The way in which they have typically been constructed is often very inflexible due to the risk of bottlenecks and the difficulty of repositories updating. Hashing-based architectures [9, 10] like Chord [11], Pastry [12], and Tapestry [13] proposed primarily to file sharing and use Distributed Hash Tables (DHTs) to assign files to specific nodes. Hashing-based architectures permit to implement a direct routing search algorithm to efficiently locate resources. However, the overlay network (i.e., DHT) between peers is generally hard to maintain. More precisely, peer join/leave operations could incur huge overheads [2].

In contrast, unstructured P2P networks, like Gnutella [14], have no precise control over resource emplacements and use flooding search protocols where the request is broadcast to all peers within a certain radius with TTL mechanism (TTL for Time To Live) [1, 2]. More precisely, in order to find a resource, the request is flooded to peer nodes in the network. Each request has an attached time to live to control the number of hops that a query can be propagated across the network. Each node that passed a request to its neighboring peers would decrement the TTL. When the TTL reached 0, the request was no longer forwarded. However, it is not possible to guarantee the success or failure of a request. In other words, a

resource may not be found even though it does in fact exist in the network. To overcome this disadvantage, the dynamic TTL using the expanding ring technique is proposed in [1]. The principle of expanding ring is as follows: a peer starts a flood with small TTL, and waits to see if the search is successful. If it is, then the peer stops the flooding. Otherwise, the peer increases the TTL and starts another flood. The process is repeated until finding required resource or covering the entire network. According to [1,2], the expanding ring technique guaranties that if the resource is presents in the network it will be found compared with regular flooding with a fixed TTL. However, if the expanding ring mechanism solves the TTL selection problem, it does not address the message duplication problem inherent in flooding that can generate large loads on the network [1,2].

Random walk is a well-known technique [7,16], which forwards a query message (walker) to a randomly chosen neighbor at each step until the service is found, can avoid the message duplication problem inherent in the flooding mechanism. Using one walker, it cuts down the message overhead significantly. However, it increases delay of successful searches [1,2]. To decrease the delay, a requesting peer sends k query messages, and each query message takes its own random walk. However, it is difficult to determine a priori the number of walks and when this number is big enough, the message traffic increases significantly [1, 2]. The replication mechanisms, such as cache some objects in the reverse path of queries, proposed in [2] can reduce the lookup length and decrease the message traffic. However, in a dynamic and distributed setting, it is difficult ever impossible to maintaining the coherence of duplicated objects.

Recently, an alternative approach to replication mechanisms, proposed in [15], uses both random walks and a cloning mobile agent-based technique for resource discovery in peer-to-peer networks. More precisely, in this paper, three scenarios with mobile agents for resource discovery are analyzed. To locate a service, a mobile agent starts, at its first step, on its requester peer. At each hop, a mobile agent determines the IP address of randomly chosen neighboring peers, creates clones, and pass tasks to these clones that move to these peers. This scenario with cloning allows mobiles agents to cover a much wide area of network peers in a reasonable amount of time compared to a single mobile agent scenario and multiple mobile agents' scenario [15].

In this paper, to decrease delay of request resolution proposed in [15], a reinforcement learning mechanism for resource discovery in peer-to-peer networks is proposed. This mechanism of reinforcement learning allows peers to learn from their satisfaction, and how coordinate to select and forward requests to the required peers in the most efficient ways. Unlike a random walk technique that allows peers to forward incoming queries to randomly chosen neighbors, in learning mechanism each peer selects a neighbor that has the highest probability of having query results.

Several works propose to use learning mechanisms for resource discovery in a peer-to-peer network. For example, Iamnitchi and al. [4] have proposed a learning-based strategy for request forwarding. Peers learn from experiences by recording the requests answered by other peers. By this learning strategy, each peer maintains information about other peers in the network. The learning mechanism however is not exposed in this paper. Wang in [5] has proposed another learning based strategy approach inspired by ant colony and by using mobile agents. In this approach, the learning strategy is based on the use of a migration policy to discover routes between peers with available resources. These routes are then used to resolve user requests. As pointed out by the author from the experiments [5], the main difficulty of this approach is how to design and set up the migration policy parameters to handle different user requests. Tsoumakos and Roussopoulos have proposed in [20] an adaptive probabilistic search approach, but no learning mechanism was given.

In this paper, a biased random walk based on mobiles agents and a reinforcement learning mechanism for resource discovery in peer-to-peer networks is presented. This reinforcement learning mechanism constitutes an organizational memory similar that the adaptive memory of immune system [21,23,24]. This organizational memory summarizes histories on how requests are performed in the past in order to adapt to current and future requests.

Recall that a mobile agent is an autonomous program that can move between sites of the network and perform computations at these sites on behalf of a user or an application [22]. Several applications have shown clear evidence of benefiting from the use of mobile agents such as electronic trading, distributed information retrieval and information dissemination [21].

Recently, Gaber in [23,24] has proposed two new paradigms alternative to the traditional client/server paradigm (CSP) to design and implement Ubiquitous and Pervasive Computing applications: the adaptive Servers/Client Paradigm (SCP) and the Spontaneous Service Emergence Paradigm (SEP). As pointed in [23,24], these paradigms could be implemented via a self-adaptive and reactive middleware inspired by a biological system like the natural immune system that

exhibits self-organizing and emergence capabilities. More precisely, unlike the classical Client/Server approach, each user request is considered as an attack launched against the global network. An immune networking middleware reacts like the natural immune system against pathogens that have entered the body. It detects the infection (i.e., user request) and delivers a response to eliminate it (i.e., satisfy the user request). Recall that in ubiquitous computing, the main objective is to provide users the ability to access services and resources all the time and irrespective to their location, while in pervasive computing, the main objective is to provide spontaneous emergent services created on the fly by mobiles that interact by ad hoc connections [23,24]. These new paradigms use mobile agents with cloning, self-organizing, and self-regulation capabilities.

2. Request resolution approach

The request resolution is the process by which the user will be provided by the required service. In the proposed approach, this process is made in two stages: request forwarding stage and result backtracking stage.

2.1. Request forwarding

In P2P networks, each peer contains links to other peers called neighboring peers, as typical Gnutella-like networks. Each peer has a bounded number of neighbors and provides at least one resource. One or a set of resources can represent a service $S=(R_1,R_2,...,R_n)$ and we called it a service path.

The peer willing to locate a service creates a mobile agent, called request agent. This agent initiates a random walk in the network until it discovers appropriate peers that can resolves the request *r* or it terminates its random walk. At each hop, the mobile agent can clone itself. When the all required resources are discovered, a request agent stops cloning itself, send results to the requester, and starts the backtracking phase. In this phase, mobile agent travels back to its initial peer following back the founded path. The role of this backtracking phase is to perform a reinforcement learning mechanism on links between peers. More precisely, the objective of this backtracking phase is to permit for peers to learn from mobile agents satisfactions on past requests to carry out biased random walk in order to improve performance of future requests.

It is worth noting that, in the agent-cloning scenario, the increasing of the agent population size with cloning operation will increase resource demands in the network, which will affect the overall performance. The self-adaptive and distributed regulation algorithm proposed in [21,25] is used to regulate dynamically the agents' population size in a network. In this algorithm, each mobile agent selects locally an appropriate behavior to its environment state from the following ones: death, moving or cloning without using any global information.

It should be noted that, at each step, a mobile agent adds the peer visited to its traveled path π_r. Also, the required resources discovered are added to an agent's set of founded resources denoted by F_r. The mobile agent moves back to the requester peer when all resources are found (i.e., $S_r=F_r$) or terminate its walk.

Since each peer could have several neighboring peers, a mobile agent needs a local decision mechanism to select the most next suitable peer to visit. The selection mechanism is based on link affinity values between peers that incorporate knowledge from past and present searches. More precisely, a mobile agent chooses a peer that has the highest affinity value. An affinity value is a real variable that is adjusted or reinforced by mobile agents satisfactions during a result backtracking phase (presented in the next section). The satisfaction value is calculated for each request *r* as follows:

$$Sat(\pi_r)=\frac{\#F_r}{\#S_r} \quad (1)$$

In addition, we consider that a TTL value is associated with each mobile agent. This TTL value decreases by one at each visited peer. When the TTL reaches the value 0, the agent checks with its initiator, the original requester. If any other mobile agent has already found the required resources, it terminates its random walk and starts the result backtracking phase with the reinforcement learning process. Otherwise, if the initiator renews its TTL value, the agent continues it walks and moves randomly to a next peer, if not, it starts the backtracking phase to adjust the affinity values along the computed path.

2.2. Result backtracking

Upon mobile agent termination (i.e. success or fail) it starts the backtracking phase. During this phase, the path computed between an end point of walk (i.e., residing peer) and the requester point will be reinforced by affinity adjustments. This phase makes the path more adapted to future requests. Therefore, the resource discovery system will be able to process the most frequent requests more efficiently. During the

result backtracking phase, the mobile agent goes back from the end point, via the intermediate peers on the founded service path, to the initial point (i.e., requester peer) and reinforce the service path based on its satisfaction calculated by the equation (1). The affinity variation for a particular request r between a peer P_i and a peer P_j in the path π_r is determined as follows:

$$\Delta m_{ij}(r)=\mu(Sat(\pi_r)-f(m_{ij})) \quad (2)$$

More precisely, when the mobile agent is on the peer P_j, it moves back to its predecessor server P_i and adjusts the affinity value m_{ij} using equation (2), wherein $Sat(\pi_r)$ is the satisfaction degree and μ is a constant value chosen between 0 and 1. For example, if the mobile agent found 80% of the resources for a particular request r, the satisfaction degree $Sat(\pi_r)$ is set to 0,8. If the mobile agent terminates with success, $Sat(\pi_r)$ is set to 1 since all the required resources are found (i.e., $S_r=F_r$). Also, when a mobile agent terminates its walk, it calculates the value of its satisfaction using the equation 1, and starts the result backtracking phase along the reverse path π_r towards the requester peer. The value of affinity is mapped to a value between 0 and 1 by using the following logistic equation:

$$f(m_{ij})=1/1+\exp(-m_{ij}) \quad (3)$$

Within this equation, the affinity value m_{ij} increases quickly when it is near 0 and satisfaction $Sat(\pi_r)$ is equal to 1. Also, the affinity value m_{ij} decreases quickly when the satisfaction is equal to 0.

To illustrate the forwarding and backtracking phases, we consider the following example. To locate a service $S=(R_1,R_2,R_3,R_4)$, a peer P_1 creates a request r and initiates two mobile agents A_1 and A_2 (since he has two neighbors) with the list S_r of the required resources, the IP address and an initial TTL. Figure 1 shows how a forwarding phase works. The two agents A_1 and A_2 add the peer P_1 to their visited peers list (i.e., a service path) and move to the peer P_2 and P_6 respectively. These agents repeat the same processes until they find the required service or their TTLs are expired. The mobile agents paths during the request-forwarding phase are shown with arrows in the figure 1. For example, since, the peer P_2 provides the required resource R_1, the agent A_1 adds it to its set of founded resources F_r and then moves to the P_2's single neighbor P_3 that possess the required resource R_4. Since, the peer P_3 has two neighbor peers P_4 and P_9, the agent A_1 is cloned to create another agent A_{11} that moves to peer P_9. A_1 moves to the other peer P_4 which holds R_3, creates another clone, the agent A_{12}, that walks to the peer P_8, and moves itself to the peer P_7 with the last required resource R_2. A_1 starts then the backtracking phase towards its initiator, the requester peer P_1.

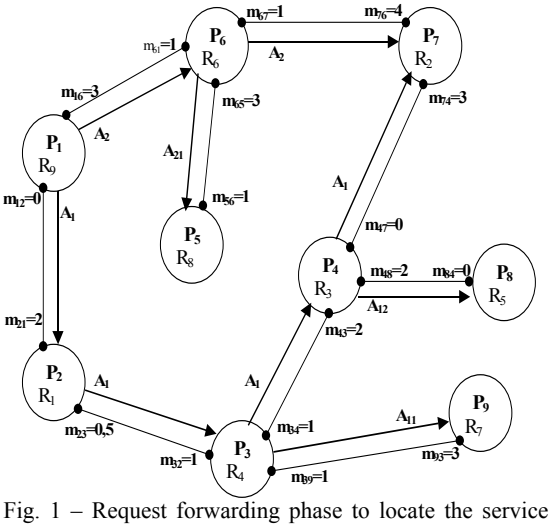

Fig. 1 – Request forwarding phase to locate the service $S=(R_1,R_2,R_3,R_4)$. Arrows illustrate mobile agents paths. A_p denotes the agent p and A_{pq} the q^{th} clone of A_p, m_{ij} denotes the affinity value between the peer P_i and the peer P_j that incorporates knowledge from past and present searches.

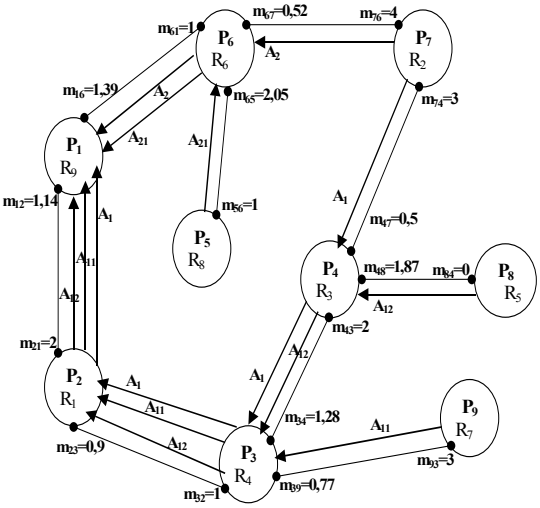

Fig. 2 – Request backtracking phase. Mobile agents, during backtracks illustrated by arrows, adjust the affinity values.

After forwarding phase either with success or fails, agents walk back as shown with arrows in figure 2. More precisely, during the backtracking phase, each

mobile agent goes back from the last visited peer, via the intermediate peers on the founded service path, to the initial peer. Agents use a reinforcement learning mechanism to adjust and reinforce dynamically link affinity values between peers according to their satisfaction deduced from delivered responses. More precisely, the new value of the affinity m_{ij} is obtained by adding its previous value with the variation determined by equation (2) (i.e., $m_{ij}=m_{ij}+\Delta m_{ij}(r)$). As illustrated by figure 2, the mobile agent A_{11} with path $\pi_r=(P_1,P_2,P_3,P_9)$ fails, while the mobile agent A_1 with path $\pi_r=(P_1,P_2,P_3,P_4,P_7)$ finds the whole service. Therefore, its satisfaction is set to 1 since 100% of the required resources are found. The mobile agent moves back from the peer P_7, via intermediate peers on the founded service path, to P_1 and reinforces affinities values of the path $\pi_r=(P_1,P_2,P_3,P_4,P_7)$. In this case, the affinity value of m_{47}, m_{34}, m_{23} and m_{12} are reinforced using equation (2), where $Sat(\pi_r)$ is set to 1. For example, m_{47} has increased from the value *0* to *0,5* (i.e., 0,5=0+(1-f(0)), with μ=1) and m_{34} has been adjusted from the value *1* to *1,28* by both the agents A_1 and A_{12}. The mobile agent A_{11} has failed and its satisfaction is set to 0,5 since only 50% of the required resources are found (i.e., R_1 and R_3). This agent moves back from the peer P_9 to a peer P_1 and adjusts the affinity values m_{39}, m_{23} and m_{12} along the path $\pi_r=(P_1,P_2,P_3,P_9)$ using equation (2) and $Sat(\pi_r)$ equals to 0,5. For example, m_{39} has decreased from the value *1* to *0,77* (i.e., 0,77=1+(0,5-f(1)), with μ=1)).

This proposed learning mechanism permits to peers to learn from past and present requests to improve performance of the future requests. It do not require any additional overhead to adapt to dynamic conditions changes in the network such as peer arrivals or departures, or when resources are removed or new ones are inserted since affinity values are adjusted dynamically by mobile agents during the requests resolution process.

3. Simulation results

The proposed discovery approach is evaluated by simulations implemented with NS2 [6]. A network of 100 peers is generated randomly. Each peer provides one resource of ten kinds of resources. The simulation abstracts any considerations about networking issues such as bandwidth constraints and time processing.

The objective of this simulation is to compare two strategies as shown in figure 3. In the first strategy, a reinforcement learning mechanism is not performed. In this case, at each simulation step, 10 mobile agents created at some peers selected randomly are asking for different kinds of resources generated randomly between 1 and 10. Mobile agents walk randomly in the network until to meet peers with the required resources and resolve their requests [15]. In the second strategy, at the beginning of the simulation, requester peers create mobile agents that initiate random walks in the network to resolve requests. In other words, at each simulation step, 10 mobile agents created at some peers selected randomly are asking for different kinds of resources generated randomly between 1 and 10. Mobile agents walk randomly to seek peers with one of the required resources. As the simulation progress and using the affinity adjustments during the request backtracking phase, for a particular request, mobile agent moves to a peer that has the highest affinity value and a selected service path will emerge as a response to that request.

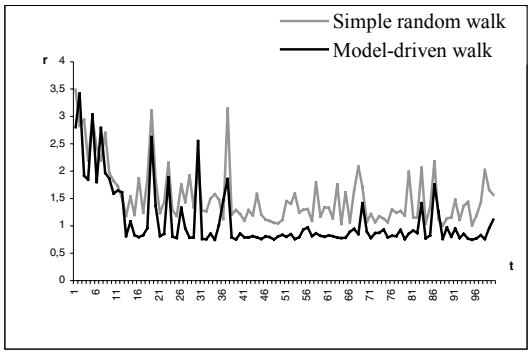

Fig. 3 – comparison of the average requests resolution time with a simple random walk without learning mechanism and the proposed model-driven walk.

This result shows that with reinforcement learning, at the beginning of the simulation, request resolution performs poorly like the simple random walk without learning. As more simulator time elapses, peers learn from delivered responses, which improve the performance of the resource discovery process.

4. Conclusion

In this paper, the use of mobile agents with a reinforcement learning mechanism is shown to be the appropriate approach to provide a distributed, scalable and adaptive resource discovery in peer-to-peer

networks. This reinforcement learning mechanism that incorporates knowledge from past and present requests improve the performance of the request resolution process. Future works address additional simulations with ns2 to evaluate the approach performance when storage and bandwidth communication are considered.

5. References

[1] C. Wang, and B. Li, "Peer-to-peer overlay networks : A survey", http ://comp.uark.edu/ cgwang/Papers/TR-P2P.pdf, 2003.

[2] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks", in Proceedings of 16th ACM International Conference on Supercomputing (ICS'02), New York, USA, pp. 84-95, June 2002.

[3] Napster Inc. The Napster homepage. In http://www.napster.com/, 2001.

[4] A. Iamnitchi, I. Foster, and D. Nurmi, "A peer-to-peer approach to resource discovery in grid environments". In HPDC'02, citeseer.ist.psu.edu/iamnitchi02peertopeer.html.

[5] D. Wang, "A resource discovery model based on multi-agent technology in P2P system", Intelligent Agent Technology (IAT'04), IEEE/WIC/ACM International Conference, pp. 548-551, 2004.

[6] Network Simulator, http://www.isi.edu/nsnam/ns/.

[7] Andrei Z. Broder, Anna R. Karlin, P. Raghavann and E. Upfal, "Trading Space for Time in Undirected s-t Connectivity", ACM Symposium on Theory of Computing, pp. 543-549, 1989.

[8] Kazaa media desktop. http://www.kazaa.com/.

[9] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-Peer Computing", Research Report No HPL-2002-57, HP Labs, March 2002.

[10] P. Gauron, "Topologies dynamiques pour les systèmes pair-à-pair", Rapport de stage de DEA Informatique distribuée, Université Paris-Sud-Orsay, 2002.

[11] I. Stoicay, R. Morrisz, D. Liben-Nowellz, D. R. Kargerz, M. Frans Kaashoekz, F. Dabekz, and H. Balakrishnanz, "Chord : A scalable peer-to-peer lookup protocol for internet applications ",
http ://www.pdos.csail.mit.edu/papers/, 2001.

[12] A. Rowstron, and P. Druschel, "Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems", Lecture Notes in Computer Science, N_ 2218, 2001.

[13] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph, "Tapestry : An infrastructure for fault-tolerant wide-area location and routing", UC Berkeley, UCB/CSD-01-1141, 2001.

[14] Gnutella: http://www.gnutella.com/

[15] J. Gaber, and M. Bakhouya, "Mobile agent-based approach for resource discovery in peer-to-peer networks", In Fifth International Workshop on Agents and Peer-to-Peer Computing (AP2PC) at AAMAS, Mai 2006.

[16] A. Broder, and A. Karlin," Bounds on the Cover Time", Journal of Theoretical Probability, 2(1):101-120, January 1989.

[17] W. Zhao, H. Schulzrinne, and E. Guttman, "mSLP-Mesh-enhanced Service Location Protocol", ICCCN 2000, Internet Draft draft-zhao-slp-da-interaction-07.txt.

[18] D. Xu, K. Nahrstedt, and D. Wichadakul, "Qos-aware discovery of wide-area distributed services", In First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid), 2001.

[19] C. Perkins, "Service Location Protocol", ACTS Mobile Networking Summit/ MMITS Software Radio Workshop, Rhodes, Greece, June 1998.

[20] D. Tsoumakos, and N. Roussopoulos, "Adaptive Probabilistic Search for Peer-to-Peer Networks", In Proceedings of the 3rd IEEE International Conference on P2P Computing, Sept 1-3 2003, Linkoping, Sweden.

[21] M. Bakhouya, "Self-adaptive approach based on mobile agent and inspired by human immune system for service discovery in large scale networks", PhD Thesis, No 34, Universite de Technologies de Belfort-Montbeliard, 2005.

[22] D. Chess, C. Harrison, and A. Kershenbaum, "Mobile agents: Are they a good idea?", IBM T. J. Watson Research Center, 1994.

[23] J. Gaber, "New paradigms for ubiquitous and pervasive computing", Research Report RR-09, Universite de Technologies de Belfort-Montbeliard (UTBM), France, 2000.

[24] J. Gaber, "New paradigms for ubiquitous and pervasive applications", Proceeding of First Workshop on Software Engineering Challenges for Ubiquitous Computing, Lancaster, UK, 2006.

[25] M. Bakhouya, and J. Gaber, "Adaptive approach for the regulation of a mobile agent population in a distributed network", In 5th International Symposium on Parallel and Distributed Computing (ISPDC'06). IEEE Press. Timisoara, Romania, 2006.