Mobile Agent-based Approach for Resource Discovery in Peer-to-Peer Networks

J. Gaber and M. Bakhouya

Universite de Technologie de Belfort-Montbeliard
Laboratoire SeT
90010 Belfort cedex, France
{gaber,bakhouya}@utbm.fr

Abstract. Peer-to-peer networks are distributed computing infrastructures that can provide globally available network resources. Their size and complexity continue to increase and permit an almost ubiquitous availability of resources. Therefore, new discovery approaches are required and need to be highly flexible in order to cope with a dynamically changing environment. In this paper, a distributed agent-based approach for resource discovery in peer-to-peer network is proposed. This approach is based on the mobile agent paradigm and uses random walks to allow dynamic and adaptive resource discovery. We analyze this approach through three distributed resource discovery scenarios by NS2 simulator.

1 Introduction

Resource discovery is an important issue in peer-to-peer network; given a user request, a resource discovery mechanism should locate and return a set of peer addresses that match the description of the requested resources. Resources can be divided into two basic categories [1], [2]: system resources and application resources. System resources are bound to specific hosts, representing hardware devises (e.g. disk) or logical system objects. Application resources are software entities managed by an application. In this paper, a service is considered to be composed by a set of resources that users need to discover and select [3], [4], [5]. These resources could be interfaced by web services. Web services are applications that permit to describe software components; in particular they define identification and accessing methods that enable the discovery and the use of these components (i.e., the resources). For example, for a punctual need, a user peer who would like to open video files or create video CD might need the following resources: a video player, format transcoding software, the MPEG-4 codec (for his wireless laptop), video effect or edge detection algorithms, etc.

In peer-to-peer networks, centralized discovery architecture cannot meet the requirements of both scalability and adaptability simultaneously. Issues are that network resource discovery systems must be able to scale and able to adapt to dynamic conditions in the network.

In this paper, we will use both random walks and a cloning mobile agent-based approach for resource discovery in unstructured peer-to-peer networks.

The rest of the paper is organized as follows. Section 2 presents the related work. In section 3, the proposed approach is presented. Section 4 presents the simulation results. Conclusion is given in section 5.

2 Related Work

Resource discovery systems proposed in the literature can be classified into three categories as depicted in figure 1: structured systems, unstructured systems and self-organization systems [4], [6]. Structured systems can be classified also into indexation-based architectures and hashing-based architectures. In indexation-based architectures, there are two subcategories: centralized and decentralized systems. In centralized indexation-based systems, typical resource discovery architectures consists of three entities: resource providers that create and publish resources, resource brokers that maintain repositories of published resources to support their discovery, and resources requesters that search in the resource broker's repositories. Centralized approaches, such as Napster [7], scale poorly and have a single point of failure. To overcome the scalability problem, decentralized approaches adopt traditionally a hierarchical architecture consisting of multiple repositories that synchronize periodically [1], [8], [9]. In a large-scale P2P network, hierarchical architecture cannot meet the requirements of both scalability and adaptability simultaneously. More precisely, 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 [1], [10], [11]. Also, peer-to-peer network is a dynamic environment where the location and availability of resources are constantly changing; some resources could be disconnected from the network and new ones may join it at any time.

Therefore, a resource discovery system should be decentralized to avoid bottlenecks and guarantee scalability and adaptability. Most of structured systems that implement non hierarchical and decentralized infrastructures use Distributed Hash Tables (DHTs) to locate and assign resources to specific peers. Hashing-based architectures such as Chord [12], Pastry [13], and Tapestry [14] allow the implementation of direct routing search algorithms to efficiently locate files [15]. However, a Hashing-based architectures require overlay networks between peers that are generally hard to maintain.

In contrast, unstructured systems [7] have no precise control over resources emplacements. Therefore, the most typical localization method is the flooding technique, wherein the request is broadcasted to all neighbors within a certain radius with TTL mechanism (TTL for Time To Live) [15]. More precisely, in order to find resources on the network, search queries are flooded to neighbor peers for bounded number of hops. Each query has an attached TTL to control the number of hops that a query can be propagated to away in the network. On receiving a query, a peer decrements its TTL, and if the TTL is greater than 0, it forwards the query to neighbor peers. When a TTL reaches 0, the query was no longer forwarded. However, it is not possible to guarantee the success or failure of a query. In other words, a resource may not be found even though

it does exist in the network. To overcome this disadvantage, the mechanism of dynamic TTL based on the expanding ring technique is proposed in [16]. The principe of expanding ring is the following: 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 restarts another flood. This process is repeated until finding the required resource or covering all the network. According to [15], [16], unlike regular flooding with a fixed TTL, the expanding ring technique guaranties that if a required resource is present in the network it will be located. However, if expanding ring technique solves the TTL selection problem, it does not address the message duplication problem that could generate large loads on the network [15], [16].

Random walk-based search mechanism, which forwards a query message (i.e., walker) to a randomly chosen neighbor at each step until the service is found, is a well-known technique that can avoid the message duplication problem. Using one walker, it cuts down the message overhead significantly, but it could increase the delay of successful searches [15], [16]. To decrease the delay, a requesting peer could send k parallel query messages, and each query message takes its own random walk. However, it is difficult to determine a priori a suitable number k of walkers. More precisely, if this number is big enough, the message traffic could increase significantly [15]. Replication mechanisms, such as caching some objects along the reverse path of queries, are proposed in [16], in order to reduce the lookup length and decrease the message traffic. However, in dynamic and distributed setting, it is difficult to maintain the coherence of duplicated objects.

Both structured and unstructured systems present some drawbacks. Structured systems use repositories or DHT overlay networks between peers that are generally hard to maintain. In particular, peer join/leave operations could incur huge overheads [16]. In contrast, unstructured systems allow the peer to enter and leave the systems without overheads. However, mechanisms used in the request resolution can generate large traffic loads on the network.

Recently, new alternatives paradigms to the traditional Client/Server paradigm have been proposed in [3] and [17] for ubiquitous and pervasive computing. These new paradigms require that discovery systems should have self-organizing and self-adaptive capabilities. An approach inspired by the human immune system to carry out these alternatives paradigms has been presented in [3], [5], [4]. This approach operates as follows: unlike the classical Client/Server approach,, each user request is considered as an attack launched against the global network. The immune networking middleware reacts like an 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) [3], [17]. This approach can be classified in the third category of self-organization discovery systems (see Figure 1). More precisely, peers (i.e., servers) are organized into communities by the creation of affinity relationships, like the idiotypic network [18] created by human immune cells (i.e., peers) against foreign antigens (i.e., user requests). The establishment of relationship affinities between peers allows to solve, by collaboration, user requests. A reinforcement learning mechanism is

used to adjust and reinforce dynamically relationship affinity values according to delivered responses. This reinforcement mechanism permits to cope with dynamic changes in the network, the services availability and the user requests. In other words, new communities may be created or others may be modified according to dynamic environment changes. Peers may acquire new memberships to new communities or drop themselves from current ones through establishing or deleting affinity relationships [5], [4].

In this paper, a cloning approach based on mobile agent for resource discovery in unstructured peer-to-peer network together with random walks technique is presented, as depicted in the figure 1. Within this approach, peers might dynamically and unpredictably join/leave the network, such that any complete knowledge of these changes and modifications is difficult even impossible to obtain.

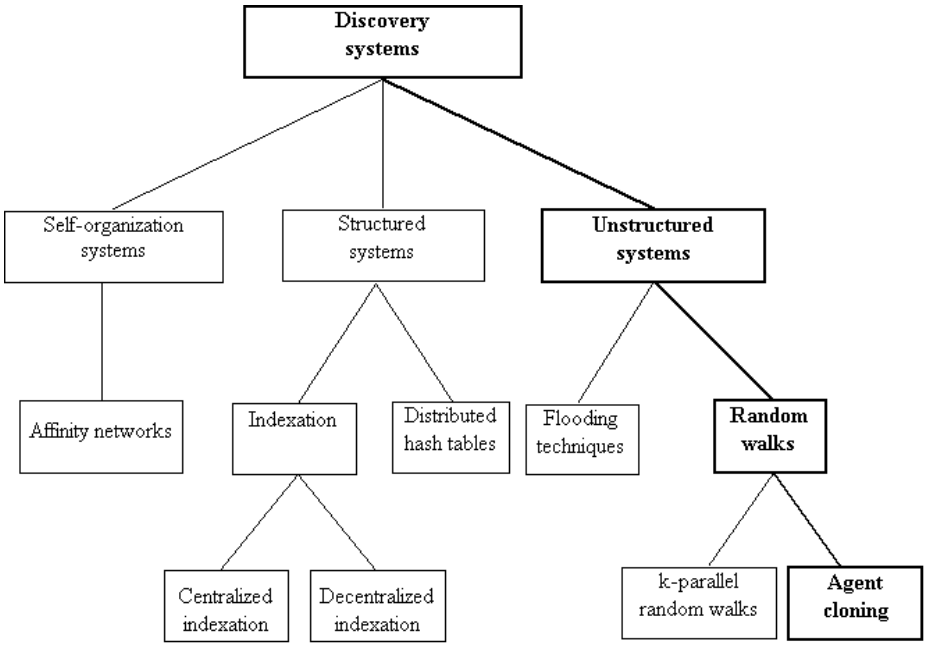

Fig. 1. Classification of service discovery systems according to their architectures and their operating modes. Cloning approach presented in this paper is indicated by bold lines.

3 The resource Discovery Approach

Unstructured peer-to-peer network can be viewed as an indirect connected graph $G = (S, V)$, where S is the set of peers ($|S| = n$) and V the set of links ($|V| = m$) connecting the peers. A peer p_i is considered to be connected to a peer p_j, if there is a link between p_i and p_j. In this section, the possibility of embedding the mobile agent paradigm and the random walk approach in designing distributed algorithm for resource discovery in unstructured peer-to-peer network is analyzed. A mobile agent is a software entity which may move from location to location to meet other agents or to access resources provided at each location [19], [20]. A random walk on a graph is a stochastic process that iteratively visits the vertices of the graph. From a given peer, the walk process moves at the next step to an adjacent peer chosen uniformly at random [21], [22].

To use mobile agents for resource discovery, let us consider the following three scenarios. The first scenario associates a unique agent to each peer request while the second scenario involves multiple agents for each request. The third scenario uses cloning operation to clone an agent during its random walk.

In the first scenario, to locate a service, the requester peer (origin of the request) creates a mobile agent, called request agent, and gives it the service to be located. A service can be composed of one or a set of resources. The mobile agent starts from a requester peer and then uses links within it to get access to other peers. The mobile agent chooses randomly between these peers, determines the IP address of the chosen one and moves to the corresponding peer. The mobile agent repeats this process with the new peers reached until it find the required resources. Upon mobile agent termination (i.e. success or fail) it starts a backtracking phase. During this phase, the agent comes back using the path computed between the peer holding the last remaining resource to collect (i.e., an end point of walk) and the requester peer. It should be noted that compared to client/server approach, in this scenario, the single mobile agent eliminates the transfer of intermediate results across the network and thus reduces the end-to-end latency [20]. However, the time to resolve the request could be unreasonable in large scale network where peers have no preexisting knowledge of where resources are located so searching for them could require steps. More precisely, steps are required for a mobile agent to cover a given graph with n nodes i.e., the mobile agent visits all nodes of the graph [23].

To reduce this latency problem, multiple mobile agents and mobile agent cloning scenarios could be more suitable scenarios for the resolution request process. In the multiple agents' scenario, an initial population of mobile agents is initially created and dispatched randomly for resource discovery. This scenario should allow the agents to resolve request in a reasonable amount of time compared to the single mobile agent scenario. However, it is difficult to determine the initial mobile agent population size. When this number is big enough, the agents traffic increases significantly, but the delay of successful searches is decreased. Also, the use of very small number decreases the agents traffic and increases delay of successful searches.

In the mobile agent cloning scenario, a mobile agent starts, at its first step, on its requester peer. At each hop, mobile agent determines the IP address of randomly chosen neighboring peers, creates replication (i.e. clone) to itself, passes tasks to this clones that move to further peers. This scenario should allow mobiles agents to cover a much wide area of network peers in a reasonable amount of time compared to the single mobile agent scenario and the multiple mobile agents' scenario. It's worth noting that in multiple mobile agents' scenario, initial population size does not change at each step but in mobile agent cloning scenario, it evolves during the random walks.

More precisely, the algorithm of mobile agent cloning scenario is as follows. 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 meets appropriate peers that can resolves the request or it terminates its random walk. At each hop, the mobile agent can clones itself. The request discovery process is made in two phase: forwarding phase and backtracking phase. During the forwarding phase, request agents seek peers that can provide the required resources. 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 [24]. 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.

During its random walk, a request agent stores the list of the visited peers. Based on this stored list, called service path, the agent chooses moving to peers that are not visited yet. However, mobile agents require a mechanism to terminate their walks. To this aim, a mobile agent starts with an initial TTL. If the required service is found, it stops the search and starts the backtracking phase. Otherwise, the agent checks if the requester has already get the service from another clone. If it is the case, the agent is killed. If not, the requester could assign a new initial TTL to the agent and initiates a new random walk.

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. Amin and Mikler have proposed in [25] a regulation algorithm to control the number of clones spawned in the network (AM algorithm). This algorithm is inspired by stigmergetic propriety of "ant colony" to facilitate coordination between mobile agents. More precisely, mobiles agents with minimum cognitive capabilities communicate with each other using pheromones that assist them to select an appropriate action. The intensity of pheromones deposited by agents at each node visited is determined by the equation $e^{-\lambda \Delta t}$, where Δt is the time since the deposition of pheromone and λ is a constant value fixed between 0 and 1. The controller of each agent contains the action selection algorithm that is defined as follows. An agent visiting a node at time t_b extracts the value of the pheromone that was

deposited at time t_a ($t_a \leq t_b$) using the equation $e^{-\lambda(t_b - t_a)}$. If this value is above a certain termination threshold Max, the agent kills itself. On the other hand, if the pheromone value reduces below a cloning threshold Min, the agent clones itself. But, if the pheromone is comprised between the termination and cloning threshold, the agent neither clones nor kills itself. In this case, it migrates to another peer node.

Bakhouya and Gaber have proposed in [26], [27] a self-adaptive and distributed regulation algorithm inspired by the "human immune system" to regulate dynamically the agents population size in a network, without using any global or constant threshold parameters (BG algorithm). The immune system has emergent properties to make self-regulating and self-adapting in dynamically changing environment [27], [5]. 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 [27].

It should be noted that with AM algorithm, the number of agents in the network converges to a constant number that depends only on the threshold parameters that should be determined a priori with empirical simulations for example. With the BG algorithm however, the number of agents in the network converges to a value that is adjusted dynamically according to the changes in the network. Therefore, BG algorithm is a distributed self-regulation algorithm.

The Figure 2 shows an example of how request resolution process works with cloning scenario. Peer P_1 possesses the resource R_1 and desires locate a service $S = (R_1, R_2, R_3, R_4)$ To locate this service, P_1 creates a request, initiates one mobile agent MA_1 and gives it the list of resources to be located, the IP address and the initial TTL. The agent add the peer P_1 to its visited peers list (i.e., service path) and moves to the peer P_2 chosen randomly. Since, this peer provides the required resource R_2, it is added to the visited peers list. At this step, MA_1 residing in peer P_2 creates another agent MA_2 that walk to peer P_6. These agents repeat the same process until they find the required service or their TTL is expired (i.e., TTL becomes equal to 0). The visited peers of request forwarding phase of the two mobile agents are shown with thicker arrows. The mobile agent (MA_2) with service path (P_1, P_2, P_6, P_5) fails, while the second mobile agent (MA_1) with service path (P_1, P_2, P_3, P_4) terminates with a successful research. During the backtracking phase, the mobile agents goes back from the last peer visited, via the intermediate peers on the founded service path, to the initial peer. The backtracking phase is started for this two mobile agents on the reverse path shown in figure 2 with dotted arrows. It should be noted that we can use a reinforcement learning mechanism to adjust and reinforce dynamically relationship affinity values between peers according to the delivered responses as pointed out in [24].

4 Simulation Results

Simulations are implemented by NS2 [28]. The objective of the simulations is to compare these three mobile agent-based scenarios. A network of 100 peers is

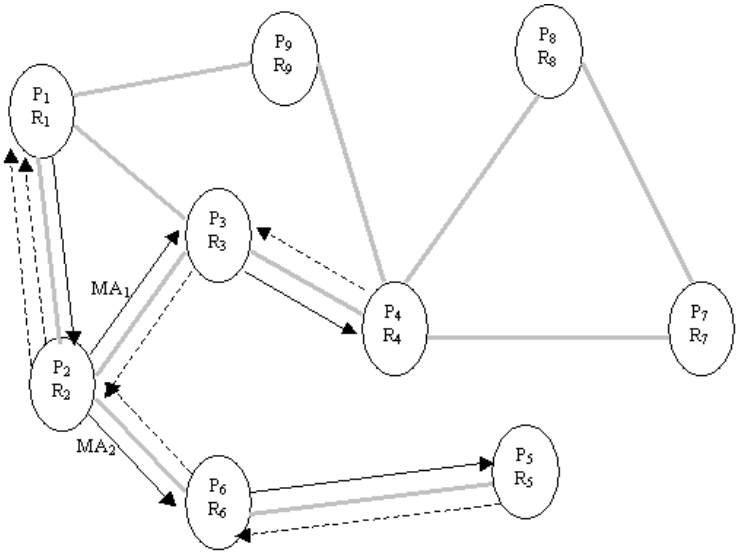

Fig. 2. Request forwarding and backtracking phases to locate the service (R_1, R_2, R_3, R_4) with the mobile agents cloning scenario.

generated randomly with BRITE generator [29]. Each peer provides one resource among ten kinds of resources. The simulation abstracts any considerations about networking issues such as bandwidth constraints and time processing. Recall that the objective of the resource discovery system is to discover and select peers that can resolve user requests.

In figure 3, the cloning mobile agent scenario shows a request resolution time better than that of single and multiple mobile agents scenarios independent of searched service (one, five or ten resources). This is due to the number of agents launched by cloning operation for the request discovery. More precisely, mobile agents may be cloned and dispatched in different directions. Therefore, this end allows mobiles agents to cover a much wide area of network peers and resolve requests in a shorter time. In multiple mobile agent scenario, a requester peer creates a fixed number of mobile agents (5 mobile agents in this simulation) and dispatch them in different directions. This scenario allows mobile agents to work in parallel, but shows a running time greater than that of a mobile agent cloning scenario as shown in figure 3.

5 Conclusion

This paper analyzes three scenarios for resource discovery in unstructured peer-to-peer network based on a mobile agent-based approach together with random walks technique. From the simulation, the agent-based approach with cloning scenario outperforms the single agent scenario and the multiple agent scenario.

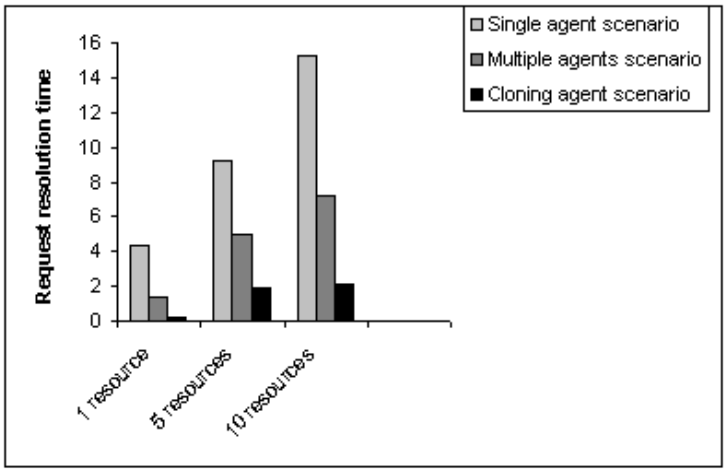

Fig. 3. Resolution request time for mobile agent scenarios.

Future work will address additional simulations with ns2 to evaluate the approach performance when storage and bandwidth communication are considered and compare it with other approaches proposed in the literature. In this paper, we have concentrated our effort first to demonstrate the viability of the proposed agent based approach with three scenarios together with ns2 simulations. Further investigation will address also the specification issue and the performance evaluation of the approach with various biased random walks schemes.

References

1. Krauter, K., Buyya, R., Maheswaran, M.: A taxonomy and survey of grid resource management systems for distributed computing. Software Practice and Experience **32** (2002) 135–164
2. Gidron, Y., Holder, O., Ben-Shaul, I., Aridor, Y.: A taxonomy and survey of grid resource manage-ment systems for distributed computing. Software Practice and Experience **13** (2001) 5–21
3. Gaber, J.: New paradigms for ubiquitous and pervasive computing. Technical Report RR-09-00, Universite de Technologie de Belfort-Montbeliard (2000)
4. Bakhouya, M., Gaber, J.: Adaptive approches for ubiquitous computing. Mobile networks and wireless sensor networks, ed. H. Lobiod, Hemes-Lavoisier, ISBN 2-7462-1292-7 (2006) 129–163
5. Bakhouya, M.: Self-adaptive approach based on mobile agent and inspired by human immune system for service discovery in large scale networks. Phd thesis, Universite de Technologie de Belfort-Montbeliard, Belfort CEDEX, 90010 France (2005)
6. Bakhouya, M., Gaber, J.: Self-adaptive and self-organizing approaches to design ubiquitous and pervasive applications. Encyclopedia in Mobile Computing and Commerce (EMCC) (2007) To appear

7. Saroiu, S., Gummadi, P.K., Gribble, S.D.: Measuring and analyzing the characteristics of napster and gnutella hosts. ACM Multimedia Systems Journal **9** (2003) 170–1840
8. Czerwinski, S., Zhao, B., Hodes, T., Joseph, A., Katz, R.: An architecture for a secure service discovery service. In: ACM MobiCom'99, Atlanta, USA, ACM (1999)
9. Xu, D., Nahrstedt, K., Wichadakul, D.: Qos-aware discovery of wide-area distributed services. In: First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid), IEEE/ACM (2001)
10. Iamnitchi, A., Foster, I., Nurmi, D.: A peer-to-peer approach to resource discovery in grid environments. In: High Performance Distributed Computing (HPDC'02), Edinbourgh, UK, IEEE (2002)
11. Talia, D., Trunfio, P.: Web services for peer-to-peer resource discovery on the grid. In: High Performance Distributed Computing (HPDC'02), www.grid.it/ (2002)
12. Stoicay, I., Morrisz, R., Liben-Nowellz, D., Kargerz, D., Kaashoekz, M.F., Dabekz, F., Balakrishnanz, H.: Chord : A scalable peer-to-peer lookup protocol for internet applicationss. In: Proceedings of the 2001 ACM SIGCOMM Conference, California, USA, ACM (2001) 149–160
13. Rowstron, A., Druschel, P.: Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, IFIP/ACM (2001) 329–350
14. Zhao, B., Kubiatowicz, J., Joseph, A.: Tapestry : An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley (2001)
15. Wang, C., Li, B.: Peer-to-peer overlay networks : A survey. Technical Report TR-P2P, Department of Computer Science, HKUST (2003)
16. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer network. In: 16th ACM International Conference on Supercomputing (ICS'02), New York, USA, ACM (2002) 329–350
17. Gaber, J.: New paradigms for ubiquitous and pervasive applications. In: Proc. of the First Workshop on Software Engineering Challenges for Ubiquitous Computing SEUC06, Lancaster, UK (2006)
18. Jerne, N.: Towards a network theory of the immune system. Ann. Immunol. (Inst. Pasteur) . **125** (1974) 373389
19. Carzaniga, A., Picco, G., Vigna, G.: Designing distributed applications with mobile code paradigms. In: 19th International Conference on Software Engineering, Boston, MA (1997)
20. Straber, M., Schwehm, M.: A performance model for mobile agent systems. In: Parallel and Distributed Processing Techniques and Application (PDPTA'97), Las Vegas, USA (1997) 1132–1140
21. Broder, A.Z., Karlin, A., Raghavan, P., Upfal, E.: Trading space for time in undirected s-t connectivity. In: ACM STOC'89). (1989) 543–549
22. Baala, H., Flauzac, O., Gaber, J., Buid, M., El-Ghazawi, T.: A self-stabilizing distributed algorithm for spanning tree construction in wireless ad hoc networks. Journal of Parallel and Distributed Computing (JPDC) **63** (2003) 97–104
23. Broder, A., Karlin, A.: Bounds on the cover time. Journal of Theoretical Probability **2** (1989) 101–120
24. Bakhouya, M., Gaber, J.: A reinforcement learning of link affinities and user requests for self-adaptive graph emergence from an arbitrary graph. Technical Report RR-03-07, Universite de Technologie de Belfort-Montbeliard, UTBM (2003)

25. Amin, K., Mikler, A.: Dynamic agent population in agent-based distance vector rout-ing. In: Second International Workshop on Intelligent Systems Design and Applications, Atlanta, USA (2002) 543–549
26. Bakhouya, M., Gaber, J.: Distributed autoregulation approach of a mobile agent population in a network. Technical Report RR-02-12, Universite de Technologie de Belfort-Montbeliard (2002)
27. Bakhouya, M., Gaber, J.: Adaptive approach for the regulation of a mobile agent population in a distributed network. In: International Symposium on Parallel and Distributed Computing (ISPDC'06), Timisoara, Romania, IEEE (1997) 1132–1140
28. Wittner, O.: Nework simulator patch. In: Faculty of Information Technology, Mathematics and Electrical Engineering, Department of Telematics. (2000)
29. Medina, A., Lakhina, A., Matta, I., Byers, J.: Brite: An approach to universal topology generation. In: Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS'01), Cincinnati, Ohio (2001)