Journal of Information Systems and Communication
ISSN: 0976-8742 & E-ISSN: 0976-8750, Volume 3, Issue 1, 2012, pp.-327-331
Available online at http://www.bioinfo.in/contents.php?id=45

APLOMB: AGENT PEDESTAL FOR LOAD BALANCING IN PEER-TO-PEER NETWORKS

PATEL R.B.[1] AND VISHAL GARG[2]

[1]Department of Computer Science and Engineering, Deenbandhu Chhotu Ram University of Science & Technology, Murthal-1039, India
[2]Dept. of Computer Engineering, M.M. Engineering College, Mullana-133203, Haryana, India
*Corresponding Author: Email- patel_r_b@yahoo.com, vishalgarg_vg@yahoo.com

Received: January 12, 2012; Accepted: February 15, 2012

Abstract- This article presents a common pedestal to Peer-to-Peer (P2P) networks and distributed computing environment. An Agent pedestal for Load Balancing (APLOMB) for P2P systems is main focus in which mobile agents are used to manage the network. It provides common solution to P2P system's fault tolerance and load balancing problems and gives true distributed computing environment with the help of mobile agents. APLOMB is component of adaptation manager of the NADSE which supports code mobility over the mobile/fixed peer device. We also present a comparative study of the NADSE, Gnutella, and Freenet. Results show that APLOMB improves the performance of NADSE when number of nodes in network is very large.
Keywords- Cluster Head (CH), P2P, NADSE, Mobile Agent.

Citation: Patel R.B. and Vishal Garg (2012) APLOMB: Agent Pedestal for Load Balancing in Peer-to-Peer Networks. Journal of Information Systems and Communication, ISSN: 0976-8742 & E-ISSN: 0976-8750, Volume 3, Issue 1, pp.-327-331.

Introduction

Peer-to-peer (P2P) is a computing model in which peer nodes collaboratively perform a computing task [8]. These peers can serve as both clients and servers and eliminate the need for a centralized node. More simply, a P2P network links the resources of all the nodes on that network and allows the resources to be shared in a manner that eliminates the need for a central host.

The claim for P2P architecture is that enables true distributed computing [14, 15], creating networks of computing resources. Hosts that have traditionally been used as clients can act as both clients and servers. P2P allows systems to have temporary associations with one other for a while, and then separate. Besides, nodes in P2P systems are autonomous in the sense that: (i) they can join the system anytime, (ii) they can leave without any prior warning, and (iii) they can take routing decision locally in an ad hoc manner. P2P Unlike the conventional centralized systems, P2P systems offer scalability [16] and fault-tolerance [2]. It is a feasible approach to implement global-scale systems such as the Grid [3].

But the existing P2P systems take more computing time for finalizing the task. For the limitation of processing time of a request we need a system which must support device and computation mobility as per need of the application.

In this article an Agent pedestal for Load Balancing (APLOMB) for P2P systems is main focus in which mobile agents[12, 17, 18, 19] are used to manage the network. APLOMB provides common solution to P2P system's fault tolerance and load balancing problems and gives true distributed computing environment with the help of mobile agents. APLOMB is component of adaptation manager of the NADSE which supports code mobility over the mobile/fixed peer device. We also present a comparative study of the NADSE, Gnutella, and Freenet. Results show that APLOMB improves the performance of NADSE when number of nodes in network is very large. We also present a comparative study of the NADSE, Gnutella[4], and Freenet[6].

Rest of the article is organized as follows. Related work is explored in Section II. Section III highlights on challenges. System model is presented in Section IV. Section V gives the brief architecture of APLOMB. Section VI presents performance study of APLOMB based NADSE network and Freenet and Gnutella. Finally article is concluded in Section VII.

Related Works

Napster [7] adopts central database approach for song titles, but it has inherent reliability and scalability problems that make it vulnerable when there are attacks on the database. Another approach, at the other end of the spectrum, is for the consumer S to broadcast a message to all its neighbors with a request for F. When a node receives such a request, it checks its local database. If it has F, it responds with the item. Otherwise, it forwards the request to its neighbors, which execute the same protocol. Proceeding in this manner will ensure that a requested resource is always be found when it exists. However, this solution has some critical limitations such as large overhead produced and the looping problem.

Gnutella [4] is a system approach for distributed data management that is based on this idea with some mechanisms to avoid request loops. It uses the time-to-live (TTL) flags in the request message to limit the broadcast scope of the message. However, this scoped broadcast approach does not scale either because of the bandwidth consumed by broadcast messages and the computing cycles consumed by the many nodes that must handle these messages. In fact, the day after Napster was shutdown, reports indicated that the Gnutella network collapsed under its own load, created when a large number of users migrated to Gnutella for sharing MP3 music files. To reduce the cost of broadcast messages, several other studies have been proposed in the literature to support intelligently forwarding and directed bread first search. Many variants of the well-known depth first search (DFS) have been also proposed. Many of the current popular systems, such as KaZaA [9], which are all based on the FastTrack [10] platform, adopt DFS concept. However, the disadvantage of these approaches, which are considered as hierarchical, is that the nodes higher in the tree take a larger fraction of the load than the leaf nodes, and therefore require more expensive hardware and more careful management. The failure or removal of the tree root or a node sufficiently high in the hierarchy can be catastrophical for the stability of the system.

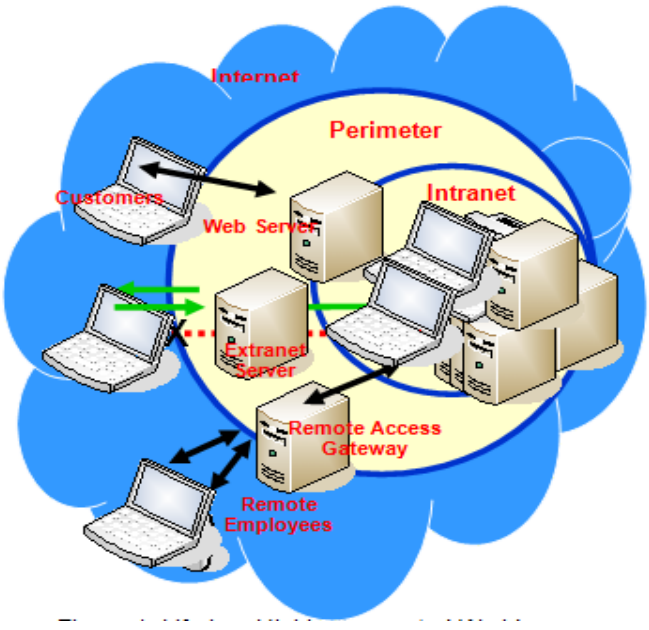

Fig. 1- Life in a Highly-Connected World

What are challenges in the life of a common person? It is well depicted in Figure 1, which shows that whole network is a setup of P2P network. There are several unsolved questions in the mind of a common user of this network. Few are as follows. How Interconnected networks will be used to- manage the network traffic, Distributed data, Mobile workers, Business extranets, Remote access, Web services, Wireless network, and Mobile smart devices, etc?

The current available solutions for structured and unstructured P2P have advantages and severe limitations with regard to performance issues. A better P2P performance can be achieved with a more flexible and intuitive architecture which should be well-researched. Thus, we need to design an adequate P2P system which should meet requirements/challenges for fulfilling need of a successful resource management solution.

System Model

We are required to develop a computing/communication P2P systems that fulfills most of the above the challenges. The developed system should enable the fast and cost-efficient deployment of self -managed computing/communication devices with high overall management cost, but with low management cost at each peer. With developed system one should be able to deploy large scale computing/ communication systems without the need of cost-intensive supercomputing infrastructure in which management is highly complex and requires high-skilled administrators for their maintenance. The approach should be evolutionary in the sense that it should give a new step towards the application of P2P into real-time services scenarios. This system should facilitate to improve the performance and incorporate new ideas. And also it should implement a structured P2P concept which must enable efficient resource management in P2P systems even during high rate of network whips. When NADSE bridges two networks in that situation it maintains services available in the network along with route information to the node maintaining available services.

APLOMB Architecture

A "Neighbor Assisted Distributed and Scalable Environment (NADSE)" based network both device and code mobility. A mobile device will be the member of a cluster. A mobile node in a cluster will work like cluster head (CH) and maintains information about other members of the cluster in the form of database [1]. When a mobile node wants to search some information it requests to CH for members information (viz. IP address, identification certificate, etc.). If the CH is not aware about availability of the type of services a mobile node is interested and presence of the same in the cluster then it guides the same to the mobile node. Then mobile node uses Agent pedestal for Load Balancing (APLOMB) for P2P systems [13] and creates a mobile agent to perform its desired task in the present cluster.

It provides common solution to P2P system's fault tolerance and load balancing problems and gives true distributed computing environment with the help of mobile agents. NADSE ("Neighbor Assisted Distributed and Scalable Environment") on support of APLOMB provides both device and code mobility. A mobile device will be the member of a cluster. A mobile node in a cluster will work like cluster head (CH) and maintains information about other members of the cluster in the form of database [5,6]. When a mo-

bile node wants to search some information it requests to CH for members information (viz. IP address, identification certificate, etc.). If the CH is not aware about availability of the type of services a mobile node is interested and presence of the same in the cluster then it guides the same to the mobile node. APLOMB permits a mobile node to create a mobile code for performing its desired task in the present cluster. Further, if the requested task is not completed with members of the present cluster mobile node may move to next cluster or it may take help of APLOMB running at the CH for multicasting the mobile code to the CHs in the network. After completion of the task final result reaches to the mobile node which was its launching station.

At present APLOMB contains mainly five agents- Mapping Agent (MAPA), Route Estimating Agent (REA), Migration Planning Agent (MPA), Code Container Agent (CCA) and Result Container agent (RCA)in future number of agents may be increased as per need of the applications. For balancing the load over the network these agents work together as a multiagent system. RCA and CCA facilitates distributed environment for adapting the nature of the network bandwidth. These agents are components of the adaptation manager (AM) [11]. Network manager (NM)[11] identify the topology of the network with assistantship of MAPA. APLOMB supports code mobility over the mobile/fixed peer device. Architecture of the functioning of the multiagent system is given in Figure 3. Brief introductions of these agents are as follows.

Mapping Agent (MAPA)- It is used by a mobile agent to locate services and to access information on network connection qualities. Connection qualities are especially important for the REA or and the MPA to achieve optimizations. In addition to throughput, latency and other network status information, this agent collects and distributes information on application-level services provided by the CH in the network. MAPA cares for precise and up-to-date knowledge (maps) within its local CH and provides a rough summarized view of the linked remote CHs. Utilizing the service descriptions in those maps, a mobile agent is able to locate points of interest within the network and see changes in the network structure. Once a list of interesting CH has been determined, another system component - the REA as shown in Figure 2 - can be used by the agent to plan an itinerary. REA calculates the shortest trip through the net based on the map data.

Route Estimating Agent (REA)-After getting the list of interesting CH that has been determined by the MAPA, another system component - the Route Estimating Agent (REA) as shown in Figure 2 - can be used by the agent to plan an itinerary. REA calculates the shortest trip through the net based on the map data. It uses the classic local optimization algorithms. If necessary, an itinerary can be recalculated and amended, for example, in the case of changes in the network or when the agent moves into new CHs and thus shifts its focus with regards to the fisheye paradigm. It facilitates to the mobile agents for optimizing the sequence of CH to visit, i.e., the itinerary. If an agent chooses a random path through a network, i.e., if an agent is visiting few nodes in cluster then visiting next cluster nodes and in future looking into the previously visited clusters once again then the sequence may lead to a non-optimal total migration time. The route estimating process itself is basically the Traveling Salesman Problem, which is a NP-complete type of problem. But dividing the problems into set of small subsets of problems eases the complex problems. For optimal migration time agent creates a clone for a cluster and number of clones being equal to the number of clusters of interest where the required services may be available. As a consequence, getting an optimal solution in practical application is ruled out. But there are heuristic algorithms (such as local search, genetic, simulated annealing, neural network algorithms, etc.) that have been supporting this working style of agents applied extensively for solving such problems.

The computation of an itinerary is based on the map data. We calculate a kind of distance matrix simply by using the reciprocal values of measured bandwidth. This matrix has to be updated at regular time intervals to fit the environment's dynamic behavior. Then, a pathfinder algorithm is applied in order to get a distance matrix with shortest paths between two places. In some experiments, we figured out that our distance matrix is not symmetrically in general. This is caused by variation in the bandwidth values and non-symmetrical connections measured by the MAPA.

The variation in network throughput influences the result and success of the route estimating, especially short time variations. The REA generates an itinerary with a fast path through the net on the basis of distance matrix. Thereby, some of the best paths may be blocked by short-time traffic. At the point in time, when an agent uses the optimized itinerary, the generated path may not be the best one any more or, in the worst case, is by now the slowest one. The probability that this happens is lower in networks with clearly differing connection qualities. The route estimating is especially useful in networks with different connection qualities and in networks with connections which have different loads over a longer time period. In networks with nearly identical connection qualities, the use of Route estimating algorithms makes no sense - just choose a random path instead of spending time to calculate the random path.

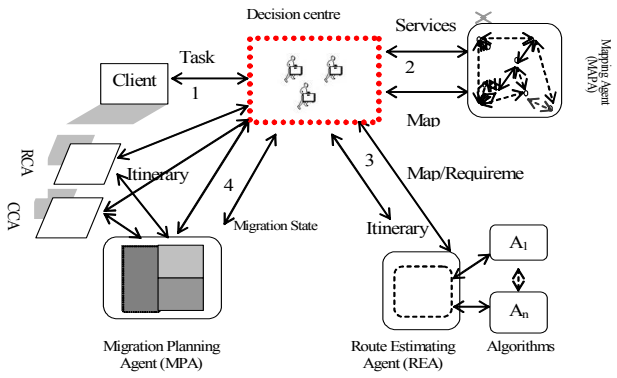

Fig. 2- Architecture of Agent pedestal for Load Balancing (APLOMB)

Migration Planning Agent (MPA)-At any point in time, as long as we have an itinerary, a mobile agent may also use a so called Migration Planning Agent (MPA) as shown in Figure 2 to optimize each single migration included in the itinerary. MPA is mainly designed to reduce network load by selecting and transmitting only those code and data portions of the agent that are needed at the upcoming remote CH. This is, if necessary, done by a concept called slicing or designated code. Other options are to place code in advance in the network, to send data home to carry fewer lug-

gages's, to change the transmission protocol, etc. An agent may contain one or more task code to be executed at different nodes in the network. The point of time when an agent's tasks are transmitted depends on the migration strategy, i.e., how a mobile agent is transmitted over the network? There are so called push strategies which transmit an agent's tasks along with the agent's state and data before the agent is started at a remote CH. Using a pull strategy, an agent's tasks are downloaded dynamically while the agent is executed at a remote CH from its home site/Code Container Agent (CCA). The agent's home platform is the Agent Submitter (AS) [7] where the agent was started first time, i.e., a client equipped with AS. Furthermore, strategies can be distinguished by which tasks are transmitted: all tasks code at once or only some tasks. For example, the pull-all strategy means: transmit an agent, start it at the remote site and in case that at least one task is required; download all tasks of the agent from its home/CCA. Using a push strategy, agent's tasks can be transmitted to the next CH of the agent's route or even to all CH visited by the agent. For example, the push-tasks-to-all strategy transmits first some of agent's tasks (those tasks which are needed potentially at remote CHs) to all CH which are visited by the agent. Missed tasks will be downloaded dynamically. Then the agent is migrated to the first CH of its itinerary. For the next hops, only the agent is transmitted. The MPA is used to optimize time and network load caused by a transmission. This is done by calculating the expected transmission times for different migration strategies. The results are compared to select a best fit migration strategy. It allows us to calculate network load and transmission time for migration of a mobile agent from home network, between CH of its route and back home. For the computation, it takes in account an agent's size (state, data, and tasks), data which is collected on its itinerary (increases with a constant factor) and connection qualities (latency and bandwidth). Thereby, a task is used at a remote CH with a certain probability. The data collected by an agent increases by a non constant size and might be transmitted back home from a CH on the agent's route. There are some technical problems to determine the actual size of an agent at runtime. For the comparison of different migration strategies, this size is constant and needs not to be involved in the computation. The same holds for the collected data. Hence, the MPA compares the transmission time for the tasks of an agent. The number of tasks and the point in time of transmission differs for different strategies. Possible requests for task downloads have to be taken in account.

In more detail, a computation of the migration time for different migration strategies for a hop is done according to the following scheme: A agent wants to hop from CH C_i to C_{i+1}. The agent's home site is C_0 client's node. The latency between two CH is defined by the function δ. Function τ denotes the available bandwidth between two CH. The amount of bytes which will be transmitted is B_c (size of all tasks) for push-all-to-next is $T = \delta(C_i, C_{i+1}) + B_c / (\tau(C_i, C_{i+1}))$ and for pull-all is $T = \delta(C_0, C_{i+1}) + B_c / (\tau(C_0, C_{i+1}))$.

Furthermore, it is difficult to determine the probability for the usage of a certain task at a remote CH it is not designated if it is designated it can be very easily traced with the database mapping. Thus, we decided to use the worst case assumption that every task has to be downloaded as long as we do not have any other options. A time computation can be made by pull-tasks

$$T = \sum_{k=1}^{n} \delta(C_0, C_{i+1}) + (B_c^k + B_\tau) / \tau(C_0, C_{i+1})$$

,

where B_c^k is the size of the k[th] task code of the agent. B_τ denotes the size of a request for downloading a certain task code.

Code Container Agent (CCA)- It contains all tasks of an agent. The CH is sued to serve as CCA. Such a server can be used by an agent to download tasks instead of downloading from home site. An automatic CCA initialization might be useful in a case where it takes more time to download tasks from the home site than from a near CCA with a fast connection. Such a CCA is the code base for further migrations as long as there is a good connectivity. This is useful only for pull strategies (downloading tasks code dynamically). The optimization is simply based on a comparison of migration times with and without a CCA initialization. With a low optimization degree, the module compares the migration time with the home site as a CCA and with a local CCA on the current CH. A medium optimization degree is reached, if all available CCAs are taken into account. As a variation of the low degree optimization, the migration times for further migrations with a dynamic CCA initialization are computed (high optimization degree).

Result Container Agent (RCA)- It is used by an agent to upload collected data instead transmitting data to the home site. The initialization of a RCA depends on whether an agent wants to transmit collected data to home site. Collected data loads the network again and again when the agent migrates. Where a mobile agent does not need this collected data for further computations, the data should be sent home site. MPA computes that whether it is cheaper to initialize a RCA to upload data instead of using home site to upload data. Automatic data upload variant calculates the migration time to the next CH, if all data is carried along with the agent. The result is compared with the time to upload collected data and to migrate without unnecessary data. An agent can initialize code and RCAs on its route. With this extended network model, the effort and the advantage of initializing and using code and RCAs can be computed. The introduced optimization variants are some approaches to reduce network load and migration time of a mobile agent. CH is also used as RCA.

Implementation and Performance Study

For testing NADSE we have used total 25 nodes (2.2 Core 2 Due processor, 1 GB RAM, 160 GB HDD, Windows-XP, Java SDK 1.5), one server (for Internet communication), 1 access point (2700 DLink), 6 routers (2 No. CISCO 2851, 2 No. CISCO 2811, and 2 No. CISCO 1841) dividing 24 nodes into six networks of categories (Class A, Class B and Class C). 25th node used to take the services of the global network.

Here in this setup this node (25th) used to move the mobile agent on the infrastructured network, i.e. wired network. Complete setup is wireless. Further it is also considered that same node may be considered for multiple times for increasing the number of nodes

in the system or mobile agent may visit infrastructure network depending on the availability of the services. This node also maintains list of available services and route to destination where services are available.

We have tested APLOMB the NADSE network for searching and downloading an information file over/from the network when NADSE services are active and not active. Figure 3 presents the time consumed in the completion of task (distributing an information file of size 512 KB). From the result it is clear that Freenet and Gnutella takes almost same time but NADSE is superior in term of overall performance. Reason behind the better performance is mobile agent which distributes the task in parallel by cloning itself and collects the outcome of the clones.

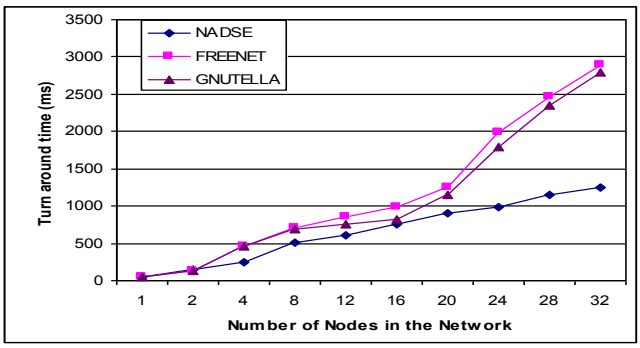

Fig. 3- Distributing an Information file which is distributed over several nodes when NADSE service node is active.

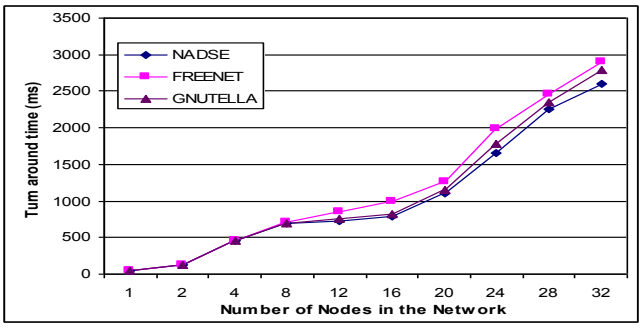

Fig. 4- Distributing an Information file which is distributed over several nodes when NADSE service node is not active.

Figure 4 presents the time consumed in the completion of task (distributing an information file of size 512 KB) when NADSE service is not active. From the result it is clear that Freenet and Gnutella takes almost same time but due to unavailability of NADSE service maintenance node NADSE network takes more time in comparison to NADSE service provider. It is found that still NADSE based network is superior in term of overall performance. Reason behind the better performance is mobile agent which distributes the task in parallel by cloning itself and collects the outcome of the clones.

Comparative results show that NADSE network performs better in comparison to Gnutella, and Freenet. From results it is also clear that APLOMB improves the performance of NADSE when number of nodes in network is very large.

Conclusion and Future work

In this article we have presented an Agent pedestal for Load Balancing (APLOMB) for P2P systems. In this system mainly mobile agents are used to manage the network. APLOMB provides common solution to P2P system's fault tolerance and load balancing problems and gives true distributed computing environment with the help of mobile agents. In future we will see more performance issues related to P2P networks.

References

[1] Kai Guo and Zhijng Liu (2008) *Fuzzy Systems and Knowledge Discovery*, 352-355.

[2] Nikta Dayhim, Amir Masoud Rahmani, Sepideh Nazemi Gelyan, Golbarg Zarrinzad (2008) *Third International Conference on Convergence and Hybrid Information Technology*, 166-171.

[3] Qian Zhang, Yu Sun, Zheng Liu, Xia Zhang and Xuezhi Wen (2005) 3*rd Annual Communication Networks and Services Research Conference,* 1-6.

[4] Ripeanu M. and Foster I.T. (2002) *First International Workshop on Peer-to-Peer Systems*, 85-93.

[5] Patel R.B. and Garg K. (2004) *A New Paradigm for Mobile Agent Computing,* 1(3), 57-64.

[6] Clarke I., Sandberg O. and Wiley B. (2000) *Workshop on Design Issues in Anonymity and Unobservability*, 25-26.

[7] Waldman M., Ad R. and Lf C. (2000) 9*th USENIX Security Symposium*.

[8] Theotokis S.A., Spinellis D. (2004) *ACM Computing Surveys*, 36(4), 335-371.

[9] Kazaa (2003) *http://www. kazaa.com*.

[10] FastTrack Accessed on-line (2003) *http://www.fasttrack.nu*.

[11] Patel R.B., Vishal Garg (2009) *International Conference on Innovative Computing, Information and Communication Technology*, 16-18, 323-329.

[12] Patel R.B. and Garg K. *Design and Implementation of a Secure Mobile Agent Platform for Distributed Computing*.

[13] Patel R.B. and Vishal Garg (2009) *International Conference on Recent Trends in Computing and Communications,* 18-19, 147-152.

[14] Bernhard Amann, Benedikt Elser, Yaser Houri, and Thomas Fuhrmann (2008) *Eighth International Conference on Peer-to-Peer Computing* 77-78.

[15] Hyuncheol Kim, Younghwa Kim and Kwangjoon Kim (2008) *Science and Security*, 58-61.

[16] Gareth Tyson, Andreas Mauthe, Sebastian Kaune, Mu Mu and Thomas Plagemann. (2009) 16*th ACM/SPIE Multimedia Computing and Networking Conference*.

[17] Gorodetsky V., Karsaev O., Samoylov V., Serebryakov S., Balandin S., Leppanen S. and Turunen M. (2008) *Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, 422-429.

[18] Kovacs E., Roehrle K. and Reich M. (1998) *Second International Workshop on Mobile Agents*, 124-135.

[19] Shih T.K. (2001) *Mobile agent evolution computing, Information Sciences,* 137, 53-73.