Intelligent Roaming for Nomadic Computing

Jun Zhang
EE Dept, Queen Mary University of London,
327 Mile End Road, London E1 4NS, UK
Jun.Zhang@elec.qmul.ac.uk

Chris Phillips
EE Dept, Queen Mary University of London,
327 Mile End Road, London E1 4NS, UK
Chris.Phillips@elec.qmul.ac.uk

***Abstract*—This paper provides details of a new system architecture enabling users / applications to roam seamlessly while they are actively processing executable code and / or communicating with remote entities. This architecture effectively decouples the running of applications from the underlying environment, facilitating both security and opportunistic processing on transitory resources. This decoupling also promotes the ability to create multiple clones of an application for performance gains. To improve the performance of the system, Fuzzy logic [2][16][17][18] and Case Based Reasoning (CBR) [3][19][20] have been applied to the system to better regulate activities such as job migration between resources or client connection handover from one Proxy to another. The paper provides a full description of the system and some simulation results to illustrate its functioning.**

Keywords-Distributed Computing, Transparent Application Migration, Fuzzy Logic, Case Based Reasoning

I. INTRODUCTION

In previous research [1], we outlined a new resource management architecture enabling users / applications to roam seamlessly while they are actively processing executable code and / or communicating with remote entities. The motivation for proposing such a system was observing that within an organisation there exists a plethora of cheap, networked computing power that is readily accessible, although transitory in nature. Although grid computing aims to take advantage of many networked computers to model a virtual computer architecture that is able to distribute process execution across a parallel infrastructure, the favoured approach is to provide a feature-rich middleware layer that is tailored to solve large-scale computation problems. However, we contend that a leaner, Java-based processing communication infrastructure provides an economical and more portable alternative.

The proposed system architecture makes use of a proxy device located somewhere between the client and server end-systems. In addition, either or both the clients and servers have a shim control layer within their protocol stack that is capable of monitoring the performance of the associated host and intercepting flows between the network / transport layers and the higher-level applications running on the host. We refer to this control function as the Interface Controller (IC). Its presence implies that both a traditional data path exists between communicating clients and servers, relayed via the external proxy, and a separate control channel from the IC in each host to the proxy.

The control channel from the IC to the proxy is used for a variety of purposes. It allows candidate hosts to register their availability with the proxy. They can then act as transient execution engines for either servers or clients. In addition, the proxy provides a relay point for all data connections / flows between the clients and the servers. The proxy employs a form of spoofing, where a TCP connection is actually the splicing together of a connection from the client to the proxy and another from the proxy to the server. This allows either the client or servers to migrate without the remote entity being aware of the change as it continues to use the IP address of the proxy. In addition, if the Java socket of the moving application is to remain bound to a particular local IP address and port number then the IC function can tunnel packets with a virtual IP address and port number inside that of the current transient host. The architecture also permits the use of cooperative handover of applications between hosts in separate pools administered by different proxies. In this case it is assumed that both the client and the server application entities are hosted on machines with IC functionality. Unlike previous schemes, the transparent migration of applications can be achieved either proactively or reactively to maintain efficient operation or in anticipation of a host shutdown.

Recently, we have applied fuzzy logic and CBR to the proxy device to enhance its performance. By using these technologies, the proxy cannot only make reasonable decisions but also has the ability to evaluate and improve its decisions. The use of fuzzy logic enables a wealth of current and historical data to be considered when evaluating the performance of an application. It can also be used to determine whether a migration should be triggered. Furthermore, once completed, the performance of the application on the new transient host can be used to refine the CBR decision information in anticipation of similar conditions arising again. In the following sections details of how the technologies are applied will be described and evaluations are also provided.

II. RELATED WORK

In this section, work we consider particularly relevant to our research will be introduced. We briefly describe their contribution and, in addition, the benefits and limitations of their approach. Points of novelty not addressed in research to-date will also be discussed.

The first relevant research is the work of Disco lab [4][5][6]. They focus on providing a framework for migrating server

whilst keeping the TCP connection alive. They propose a model called "Cooperative Service Model". In this model, they consider a "pool", which comprises some distributed and similar servers. These servers cooperate to sustain a service by migrating client connections within the pool. In order to cooperate successfully, they designed a new transport layer protocol called "Migratory TCP (M-TCP)" to support efficient migration of live connections. This work provides more reliable services and resilient to network congestion as they can migrate the connection to a suitable server, but the migration is restricted within the server pool and no server load-balancing scheme has been presented.

Another interesting approach is that of Zap. In this work[7][8], researchers have developed a novel system for transparent migration of legacy and networked application. They designed a *pod* abstraction, which provides a collection of processes with a host-independent virtualised view of the operating system. This decouples processes located within the pods from dependencies on the host operating system and other processes. By integrating Zap virtualisation with a checkpoint-restart mechanism, Zap can migrate a pod of processes as a unit among machines running independent operating systems without leaving behind any residual state after the migration. Though transparent application migration without modification to the operating system kernel or application can be achieved, no intelligence control scheme has been raised so that the migration still lacks intelligence.

In the vOS work [9][10][11], they designed a system that consists of a group of computers. This system decouples the application process from its physical environment by using "virtual" technology, which means that the application uses virtual files, virtual network connections and other virtual resources. This "virtualisation" is achieved by using API interception, which means intercepting calls made from the application to the underlying runtime system and reinterpreting the call. Similar to the work of Zap, transparent application migration without modification to the operating system kernel or application can also be achieved, but the migration is restricted within a group of machines and lack of an intelligent control plane.

In VMware[12][20], a software system Virtual Machine Monitor(VMM) was introduced to run between host operating system and guest application or guest operating system. It enables multiple operating systems (e.g. Windows, Linux and Mac) to run on a single machine at the same time. Each operating system will be isolated in a secure virtual machine and managed by the VMM. Physical resources like CPU, memory, hard disks and I/O devices will be mapped from the physical hardware to each virtual machine so that each operating system running seems to its own physical resources. It enables multiple operating systems to run on the same physical computer, but it does not provide any intelligent mechanism for migrating such a virtual machine.

III. SYSTEM ARCHITECTURE

Figure 1 shows the proposed system architecture [1].

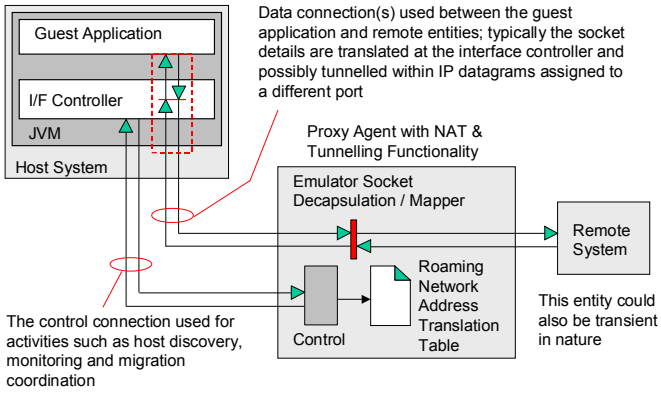

Figure 1. Proposed System Architecture

The system architecture consists of three main entities: the Host System(s), the Proxy Agent(s) and the Remote System(s). In figure 1 a client application and its associated operating system are running on a transient system and this client application communicates with a remote endpoint through an Interface Controller and a Proxy Agent.

The Host System is composed of three components: the Guest Application, the Interface Controller (IC) and the Java Virtual Machine that encapsulates them. The IC main functions include intercepting all the system calls sent from the Guest Application and redirecting them according to requirements, monitoring the execution performance of the JVM on the Host System, setting up the control connection between the Host System and the Proxy, tunnelling the data packets sent from the Guest Application, check-pointing and freezing the execution of the Guest Application, including the information held in any dynamic memory such as the state of variables.

The Proxy Agent has several functions including providing a roaming Network Address Translation table, being a point of presence for the Guest Application and so on. It acts as a bridge between the Host System and outside world. If the Host System uses TCP to communicate with the remote entities, the communication pathway between them is actually comprised of multiple connections spliced together to enable migration without closing the end system sockets. Besides data connection(s), there is also a control connection between the Host System and the Proxy Agent.

The most important functions of the Proxy Agent are to assign jobs to different resources, decide job / guest application migration and to cooperate with other Proxy Agents by using fuzzy logic and CBR. The proxy also maintains a registry of available Host Systems and their characteristics.

The Remote System can either be a traditional communication endpoint or has the same structure as the Host System. It can be a server or a client. If the Remote System is a client, it always considers the Proxy as the server and just knows the IP address of the Proxy. Service requests to the Proxy will then redirected to a suitable Host System by the

Proxy transparently to the Remote System. When a Remote System sends a job to the service which it believes is hosted by the Proxy, the Proxy Agent will try to find a suitable resource and assign the job to the resource. The Proxy Agent monitors the performance of each resource on the Host System and can make a migration decision if the performance of a resource is lower than a standard and there is "better" Host System available. A Proxy Agent is also able to perform cooperative handover of applications between Host Systems administered by a different Proxy Agent. This process may be triggered by a worsening round-trip time between a Host system and a roaming Remote System. Such a situation may cause to proxy to move the application to a new host system which is available via a Proxy that is closer to the Remote System's new location.

IV. INTELLIGENCE IN THE PROXY AGENT

In this section, details about how the fuzzy logic and Case-Base Reasoning are implemented within the Proxy Agent are given.

Intelligence plays an important role in the system architecture. The Proxy needs to decide which resource should be assigned to handle the job when the Proxy Agent receives a request from a guest application. The Proxy Agent also needs to make reasonable decisions about when and where to migrate an application and / or the connections between the resource and the guest application. By using intelligence, the load between different resources can be balanced and speed up the job completion time.

Fuzzy logic and Case Based Reasoning have been introduced to fulfil the requirement of intelligence in the system. Though there are many Artificial Intelligence (AI) technologies available, fuzzy logic mixed with Case Based Reasoning is considered suitable.

Using fuzzy logic, the Proxy Agent can make decision quickly by. In a large network, the number of requests will be numerous. Therefore, the Proxy Agent should find a suitable resource and make decision in an acceptable period of time. To make such a decision, the Proxy Agent may collect a lot of information from the Host Systems. However, to judge whether a Host System fulfils the requirement is not straightforward. As Fuzzy logic is an imprecise logic, the overall capability of a certain Host can be classified quickly so that the Proxy Agent can make decisions quickly. Nevertheless, "fuzzy logic is not any less precise than any other form of logic: it is an organized and mathematical method of handling inherently imprecise concepts" [15]. Fuzzy logic uses approximate to present things that cannot be judged precisely. At the same time, CBR is a method of solving problem based on the past experience. In our design, fuzzy logic is used to propose solutions and CBR is used to judge whether the solution is reasonable based on its past experience. Therefore, the Proxy Agent can make decisions reasonably. The Proxy Agent can adapt to changes in the environment by using CBR.

Figure 2 shows the structure of the Proxy Agent. There are three components in the Proxy Agent, Interface, AI component and Execution.

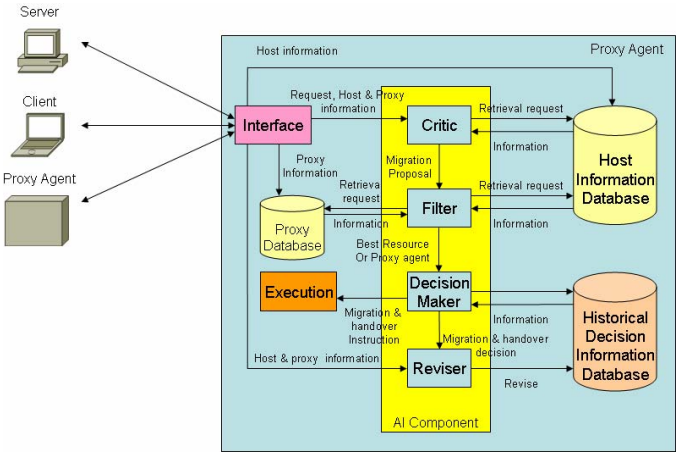

Figure 2. The Structure of the Proxy Agent

The Interface component is connected with the monitored hosts. Monitored hosts send their system performance information to the Proxy Agent at regular intervals. After the Information Receiver receives the information, it sends it to Critic and Feedback components for further use. In addition, the Interface component also sends the Proxy Agent's information to other proxies and receives information from them.

The AI Component is the intelligent part of the Proxy Agent. It is composed of four subcomponents, the Critic, the Filter, the Decision Maker and the Reviser. The Critic subcomponent evaluates the performance of the host. If the performance is lower than a certain standard, it will raise a migration proposal. It also compares the propagation delays between different proxies and different hosts. If necessary, it will raise a Proxy handover proposal. The Filter is used to retrieve best available resource or Proxy Agent for job migration or Proxy handover. The Decision Maker uses historical decisions to decide whether to carry out a migration or handover. If yes, it gives a migration/handover instruction to the execution component. The Reviser is then used for evaluating decisions. For example, if the performance of the host is higher than a standard in a certain period of time after the migration, the migration is considered to be successful. If not, the migration is considered a poor choice. The Reviser then updates the historical decision information database accordingly. Finally the Execution entity carries out the host migration and Proxy handover according to the migration/handover instruction sent from the AI Component.

In addition to the components, there are three databases in the Proxy Agent: the Host Information Database, the Historical decision Information Database and the Proxy Database. The Host Information Database records each host's system information. The information includes Host IP address, CPU speed, physical memory capacity, available physical memory, network speed, network utilisation, propagation delay between the Proxy Agent and the host, and host

performance evaluation marks (Evaluated by the Critic). The database is updated by the Interface component at regular intervals. The Historical decision Information Database records the host information with the Proxy Agent's decisions. The information includes details of both hosts (the host which needs migration and the host which should be migrated to) and the decision made by the Decision Maker. Lastly the Proxy Database records other proxies' information. The information includes the proxies' IP addresses.

A. Job Allocation Decision Making

Fuzzy logic and CBR are used when a Proxy Agent has to make a decision. For example when a Proxy Agent receives a job from a guest application, it has to find a suitable Host System (a.k.a. resource) for the guest application. To make such a decision, the Proxy Agent will look into its database and choose the best available resource for the job. The procedure for making such a decision is as follows:

1) The Interface receives a job from a client. Then it sends the job request to the Critic.
2) As it is a new job request, so when the Critic sub-component receives it, it redirects to the Filter.
3) The Filter chooses a suitable resource for the job according to the job requirements. For example, some jobs prefer to be finished as soon as possible but some other jobs prefer to be finished successfully. As there are many kinds of resources in the network, some may be powerful but not reliable as they may not be available all the time and some others may be reliable but not so powerful. Here the fuzzy logic technology is used. The power level and the reliability level are represented by fuzzy input variables. For example, if a resource which has CPU with frequency 3GHz, the power of the resource may be represented as "80% Powerful". Another resource which has CPU with frequency 1GHz may be represented as "30% powerful". As well as the power level, the reliability level is also represented by some descriptions. If a job prefers to be finished successfully rather than quickly, the filter will choose the more reliable for the job. The magnitude of participation of each input can be expressed as a membership function. It associates a weighting with each of the inputs that are processed, define functional overlap between inputs, and ultimately determines an output response. The rules use the input membership values as weighting factors to determine their influence on the fuzzy output sets of the final output conclusion.
4) After the Filter selects a resource, it sends the proposal to the Decision Maker. As it is not a proposal for resource migration or Proxy handover, the Decision Maker will approve the proposal and send it to the Execution engine. This will carry out the job of setting up the connection to the resource and redirect the client job accordingly.

B. Application Migration Decision Making

After a Proxy Agent assigns a job to a resource, it monitors the performance of the resource. If the performance of the resource is lower than a certain level, or the performance of the resource is lower than the requirement of the job, the Proxy Agent will start the procedure of looking for a better resource to take over the current job. Migrating the job may take place for different reasons. One is to accelerate the speed of processing the job and another is to balance the load between different resources. The migration decision procedure is as follows:

1) The Interface receives messages sent from each resource at regular intervals. The message contains performance information that the Interface then sends to the Critic.
2) When a Critic receives the message, it assesses the current performance of the resource. Then the Critic retrieves the information stored in the Host Information Database. If the current performance of the resource cannot fulfil the requirements of job(s), then the Critic will start to look for other resources.
3) If the Critic decides to propose a migration proposal, it will send the job's requirement to the Filter. Then the filter looks for a suitable resource in the Host Information Database. This is the same procedure as used for determining a suitable resource for a new job. If the Filter can find a suitable resource, it will send the migration proposal to the Decision Maker.
4) When the Decision Maker receives the migration proposal from the Filter, it uses CBR technology to determine what action should be taken. Firstly, it tries to retrieve a similar case from Historical Decision Information Database. If there is are similar migration cases in the past and most migration cases show that migrations were successful, the Decision Maker will elect to migrate the job. Conversely, if most cases show that the migration is not viable, the Decision Maker will probably decide not to proceed with migration. By default, if the Decision Maker cannot find any similar case in the database, it will opt to migrate the job.
5) If the Decision Maker elects to migrate the job it sends a migration instruction to the Execution engine. The Decision Maker will also sends its decision to the Reviser for further evaluation.
6) After the migration, the Proxy Agent will evaluate the results of the migration. This step also uses CBR technology. The evaluation is based on the monitoring messages sent from the new resource. The Interface will pass these to the Reviser. The Reviser evaluates the status of the new resource and the jobs assigned to it. If the new resource fulfils the job requirements, then the migration is considered to be successful and the Reviser will record the migration as being successful in the Historical Decision Information Database. Otherwise, the migration will be considered a failure and the Reviser will record the migration result accordingly. This may in turn trigger a further migration.

C. Proxy Handover Decision Making

Not only can a Proxy Agent migrate jobs between different resources under its control, it is also able to handover the job to

another Proxy. This type of handover may be due increasing propagation delay between the client and the resource. For example, if the client is a mobile client and it roams between different domains, the propagation delay between the client and the Proxy Agent might become unacceptably high. If there is another Proxy closer to the client, the original Proxy Agent might consider handing over the job to the closer Proxy. Another motivation might be as a result of a Proxy Denial of Service attack. In this case the job migration can be used to move essential services away from the threatened Proxy. The procedure of making a handover decision is as follows:

1) The Interface monitors the propagation delay to each resource and each client at regular intervals. At the same time, the Interface also receives message from other Proxy Agents. The message includes the information about the resources and the clients registered at that Proxy Agent, the propagation delay between each resource and that Proxy Agent, propagation delay between each client and that Proxy Agent.

2) The Interface sends the monitoring data and the information received from other Proxy Agents to the Critic. If a propagation delay to a certain client or resource is over a threshold, the Critic might propose a Proxy handover and send it to the Filter.

3) When the Filter receives this handover proposal, it will look into the Proxy Database to find out whether there is any Proxy that might be closer to the client's new location. If no Proxy Agent is available, the Filter will not send a handover proposal to the Decision Maker.

4) When the Decision Maker receives the handover proposal from the Filter, it uses CBR technology to make a decision. Firstly, it tries to retrieve similar Proxy Agent handover case from Historical Decision Information Database to assist it. However, if the Decision Maker cannot find any similar case in the database, it normally elects to proceed as this allows it to speculatively learn from the experience.

5) After making the decision, the Decision Maker will send the Proxy Agent handover instruction to the Execution engine to carry out the handover procedure. The Decision Maker will also send the decision to the Reviser for further evaluation.

6) After the Proxy Agent handover, the Reviser will evaluate the migration in order to update the Historical Decision Information Database uses CBR technology.

V. PERFORMANCE EVALUATION

In order to evaluate the intelligence mechanisms introduced in the previous section, a simulation environment was created and a number of tests carried out. The first one was to determine the effect of migrating a job between Host System resources. In this scenario, two sources keep sending jobs to the Proxy Agent. The Proxy Agent redirects the jobs to two resources in turn. The job size is defined as 10 units. The first resource's job processing capacity is set to 10000 units and the second resource's capacity varies from 1000 to 10000 (due to unrelated demands on the system). Their residual job processing capability changes along with the job being processed, which means their residual capacity become lower when they receive a new job and increases when they finish a job. Their reliability is set to 10000 as they are considered always available. The Proxy observes the residual capacity of each resource. If a resource's residual capacity is lower than 7000, the Proxy's fuzzy logic inference engine will consider the resource not sufficiently capable and it will try to find whether the other resource has a greater residual capacity. If it finds a more capable resource, the Proxy Agent migrates the job that has the largest job size to that new Proxy. The length of the simulation is set to 2000 units. The results shown in figure 3 indicate that the total number of jobs processed changes along with the capacity of resource 2.

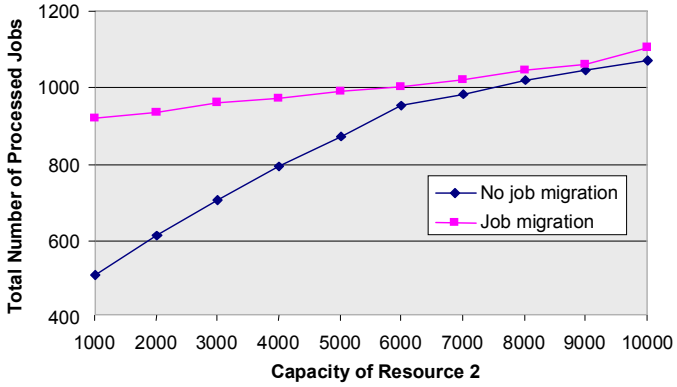

Figure 3. Example Job Migration Performance Relative to Fixed Assignment"

The x-axis shows different capacity setting of resource 2 (from 1000 to 10000) and the y-axis shows the total number of processed jobs during the simulation time. The result shows that the "job migration" improves the overall performance of the system.

A second test was setup to evaluate the CBR mechanism. In this scenario, one source keeps sending jobs to a Proxy Agent and the Proxy Agent redirects the jobs to two resources in turn. The job size is also set to 10 units. The first resource's processing capacity is 10000 and the second resource's capacity is 6000. Their reliabilities are both set to 10000. In this simulation, both resources are available from time zero to time 500. Then resource 1 becomes unavailable between time 500 and 1000. The jobs running on this resource will fail to finish. Later, resource 1 will become available again between time 1000 and 1300. Then resource 1 becomes unavailable between 1300 and 2000 again. The jobs running on resource 1 fail to finish as well.

To determine the benefit of CBR we implement fuzzy logic without CBR, which does not learn from the change of the environment, and fuzzy logic with CBR, which can learn from previous experience. Firstly, the proxy migrates jobs from resource 2 to resource 1, as resource 1 is more powerful. Later, as resource 1 becomes unavailable and some jobs which migrated from resource 2 to resource 1 will fail to finish. In the mechanism of fuzzy logic without CBR, the Proxy Agent does not take any measure to adapt this change. But in the fuzzy

logic with CBR case, the Proxy Agent learns from the fact that some jobs migrated to resource 1 are not finishing. Resource 1 is not very reliable. Therefore, the Proxy will change the reliability rating of resource 1 to lessen the likelihood of migrating jobs to it. The results are shown in figure 4.

The x-axis shows the time from 0 to 2000 and the y-axis show the number of processed jobs over every 100 time units. From the figure we can see that both mechanisms perform the same from time 0 to time 1000, as expected. However, the fuzzy logic with CBR mechanism performs better than fuzzy logic alone after time 1000. This is due to the Proxy Agent learning from the change in the environment. As resource 1 is not available from time 500 to 1000, the fuzzy logic with CBR mechanism learns that resource 1 is not so reliable. In consequence the Proxy Agent reduces the reliability of resource 1. The reduction of reliability of resource 1 results in the proxy migrating little or no jobs from resource 2 to resource 1 during the time 1000 and 1300. However, if the reliability of resource 1 improves then the Proxy may choose to migrate more jobs to it again, whilst monitoring their completion success rate.

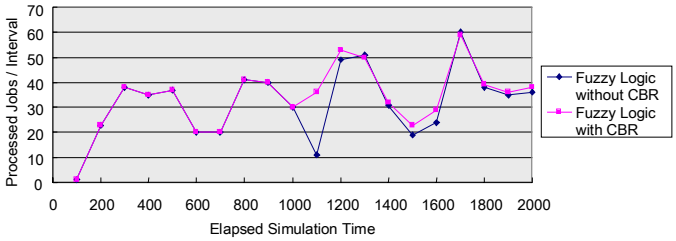

Figure 4. Comparative Migration Performance between "Fuzzy Logic with CBR" and "Fuzzy Logic without CBR"

VI. CONCLUSIONS

In this paper, we firstly introduced a new architecture for enabling users / applications to roam seamlessly while they are actively processing executable code and / or communicating with remote entities. We then focused on introducing the intelligence functionality of the system architecture in order to better regulate the migration of events. This benefits from being able to adapt its migration behaviour to changing circumstances. We focus on a combination of fuzzy logic and CBR as these are able to make reasonably good decisions within a short time, which is essential for our dynamic environment.

REFERENCES

[1] Zhang, J., Phillips, C., "Ubiquitous, Flexible and Distributed Computing", PGNet 2007, http://www.cms.livjm.ac.uk/pgnet2007/Proceedings/Papers/2007-060.pdf

[2] Cox, E, "Fuzzy fundamentals", Spectrum, IEEE Volume 29, Issue 10, October 1992, pp: 58 – 61G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[3] Aamodt, Agnar, and Enric Plaza. "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches", Artificial Intelligence Communications 7, no. 1, 1994, pp: 39-52

[4] Florin Sultan, Kiran Srinivasan, Deepa Iyer, and Liviu Iftode, "MigratoryTCP: Connection Migration for Service Continuity in the Internet", http://discolab.rutgers.edu/mtcp/icdcs02.ps

[5] Florin Sultan, Kiran Srinivasan, Deepa Iyer, and Liviu Iftode, "Migratory TCP: Highly Available Internet Services Using Connection Migration", http://discolab.rutgers.edu/mtcp/dcs-tr-462.ps

[6] Florin Sultan, Kiran Srinivasan, Deepa Iyer, and Liviu Iftode, "Transport Layer Support for Highly-Available Network Services", http://discolab.rutgers.edu/mtcp/hotos01.ps

[7] Steven Osman, Dinesh Subhraveti, Gong Su, and Jason Nieh, "The Design and Implementation of Zap: A System for Migrating Computing Environment", http://www.ncl.cs.columbia.edu/publications/osdi2002_**zap**.pdf

[8] Shaya Potter, Jason Nieh, Dinesh Subhraveti, "Secure Isolation and Migration of Untrusted Legacy Application", http://www.ncl.cs.columbia.edu/publications/cucs-005-04.pdf

[9] Tom Boyd and Partha Dasgupta, "Process Migration: A Generalized Approach Using a Virtualizing Operating System", Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on 2-5 July 2002 Page(s):385 – 392 Digital Object Identifier 10.1109/ICDCS.2002.1022276,

[10] Ravikanth Nasika and Partha Dasgupta, "Transparent Migration of Distributed Communicating Processes", http://cactus.eas.asu.edu/Partha/Papers-PDF/1900-2001/PDCS-ISCA2000.pdf

[11] Tom Boyd, Partha Dasgupta, "Injecting Distributed Capabilities into Legacy Applications Through Cloning and Virtualization", http://www.dvo.ru/bbc/pdpta/vol3/p251.pdf

[12] http://www.vmware.com/vinfrastructure/

[13] About the Java Technology, http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html

[14] James Gosling, Henry McGilton, A White Paper: The Java Language Environment, http://java.sun.com/docs/white/langenv/index.html

[15] http://en.wikipedia.org/wiki/Fuzzy_logic

[16] Biacino, L.; Gerla, G. (2002). "Fuzzy logic, continuity and effectiveness". Archive for Mathematical Logic 41 (7): 643-667. doi:10.1007/s001530100128. ISSN 0933-5846.

[17] Vasudevau, C. Smith, S.M. Ganesan, K., "Fuzzy Logic in Case Based Reasoning", Dept. of Ocean Eng., Florida Atlantic Univ., Boca Raton, FL, USA;This paper appears in: NAFIPS/IFIS/NASA '94. Proceedings of the First International Joint Conference of the North American Fuzzy Information Processing Society Biannual Conference. The Industrial Fuzzy Control and Intelligent Systems Conference, and the NASA Joint Technology

[18] Dick, S.; "Toward complex fuzzy logic", Fuzzy Systems, IEEE Transactions onVolume 13, Issue 3, June 2005 Page(s):405 - 414 Digital Object Identifier 10.1109/TFUZZ.2004.839669

[19] Leake, David. "CBR in Context: The Present and Future", In Leake, D., editor, Case-Based Reasoning: Experiences, Lessons, and Future Directions. AAAI Press/MIT Press, 1996, 1-30.

[20] Zhi-Wei Ni; Shan-Lin Yang; Long-Shu Li; Rui-Yu Jia; "Integrated case-based reasoning", Machine Learning and Cybernetics, 2003 International Conference onVolume 3, 2-5 Nov. 2003 Page(s):1845 - 1849 Vol.3 Digital Object Identifier 10.1109/ICMLC.2003.1259797

[21] http://www.vmware.com/virtualization