# Towards autonomic overlay self-load balancing

**Conference Paper** · March 2013

DOI: 10.1109/SSD.2013.6564018

Ibrahim Al-oqily
Hashemite University
**25** PUBLICATIONS **79** CITATIONS
SEE PROFILE

Mowafaq Alzboon
Universiti Utara Malaysia
**2** PUBLICATIONS **0** CITATIONS
SEE PROFILE

Ayoub Alsarhan
Hashemite University
**29** PUBLICATIONS **59** CITATIONS
SEE PROFILE

2013 10th International Multi-Conference
on Systems, Signals & Devices (SSD)
Hammamet, Tunisia, March 18-21, 2013

SSD'13 1569694537

# Towards Autonomic Overlay Self-Load Balancing

Ibrahim Al-oqily
Department of Computer of Computer Science, Hashemite University, Jordan,
e-mail: izaloqily@hu.edu.jo
Mwafaq Alzboon, and Hussein Al-Shemery
Collage of Information Technology, Middle East University for Post Graduate Studies, Jordan
e-mail: mwafaqalzboon@gmail.com, and howaied@meu.edu.jo
Ayoub Alsarhan
Department of Computer Information Systems, Hashemite University, Jordan
e-mail: AyoubM@hu.edu.jo

***Abstract*— Overlay networks are virtual networks built on top of the physical computer networks. A special kind of these networks is built specifically to meet specific user's requirements. They are called Services Specific Overlay Networks (SSON). Managing and achieving load balancing in such environment is challenging. This challenge is always increasing as the current technology faces the challenge of increased complexity, cost, and heterogeneity. However, traditional strategies don't satisfy the needs of current technology trends. In this paper we thoroughly review the state of the art in load balancing techniques and propose a novel self-load balancing scheme for autonomic overlay networks. The proposed scheme employs the spatial index and partitions the overlay space to build a distributed quad tree.**

***Index Terms*— Load balancing; SSONs; Media Ports; Autonomic Computing; Overlay Networks**

## I. Introduction

Your distributed and parallel computer systems are two faces for the same coin. the former may be observed as a collection of computing and communication resources shared by users to achieve their goals, while the later can be viewed as a collection of processing devices that exchange messages and assist to resolve large computational troubles powerfully. To allow users to solve their big computational problems in distributed or parallel era, the computational problem should be partitioned into several tasks with different workloads after that they are allocated to diverse processing devices for computation.

Many solutions have been proposed to undertake the difficulty of load balancing in distributed and parallel computer systems. Nevertheless, all these solutions pay no attention to the heterogeneity environment of the system, or reassign loads among nodes without considering proximity relationships, or both. A large amount of research has been done on load balancing technology in distributed system. These load balancing technologies could be divided into static and dynamic schemes. Static load balancing algorithms distribute the tasks to processing elements at compile time, while dynamic algorithms attach tasks to processing elements at run time.

According to IBM Corporation current technology faces the challenge of increased complexity, cost, and heterogeneity [22]. Communications and software technologies are growing rapidly; though, the scale & complexity has grown as well. The growth in systems/applications development, configuration, and management has begun to overcome existing tools and methodologies which is rapidly making systems/applications fragile, unmanageable, and insecure. Therefore, there is a great demand to change the way in which these systems are managed [3].

Overlay networks is getting a great attention lately due to the services that they can provide without the need to modify equipments. A special kind of these networks is usually designed to meet the users' specific needs and requirements. They are called Service Specific Overlay Networks (SSON). SSONs have the advantage of incorporation service nodes in the overlay service path between the source and the destination. These service nodes usually provide value added functions such as media caching or adaptation. The management of SSONs faces many challenges. One of which is load balancing. With large number of SSONs coexisting, each satisfying a certain user requirements, a competition for their basic resources, the service nodes, could results in unbalanced load distribution between service nodes. This could degrade the overall network performance as well as the performance of individual services. With the increased complexity and heterogeneity of services and devices, traditional load balancing techniques are not adequate. IBM's autonomic concepts are a viable solution for this challenge and thus a self-load balancing scheme should be developed to achieve fair distribution of loads among service nodes.

This paper is organized as follows. In section 2, load and self-load balancing techniques are reviewed. In section 3, self-load balancing in overlay networks are thoroughly reviewed and analyzed. The proposed autonomic overlay self-load balancing is presented in section 4. Finally, a conclusion is given in section 5.

## II. State of the art in load Balancing

In this section, a thorough review of the state of the art in load balancing schemes is presented and analyzed.

### A. Load balancing definitions

Various definitions for load balancing have been introduced. For example, [2] define the load balancing "as the process of redistributing the work load among nodes of the distributed system to improve both resource utilization and job response time while also avoiding a situation where some nodes are heavily loaded while others are idle or doing little work". Again [2] define it as "the mechanism that enables jobs to move from one computer to another within the distributed system, this creates faster job service e.g., minimize job response time and enhances resource utilization".

There is more interesting definition proposed by [12] where they define it as the process of roughly equalizing the workload among all nodes included in the distributed system. It strives to produce a global improvement in system performance. In this manner, load balancing goes one-step further than load sharing which only avoids having some nodes idle in the distributed system when other nodes have too much work.

### B. benefits of load balancing

The load balancing of an application has a direct impact on the speedup to be achieved as well as on the performance of the system [1, 2]. The major objectives of load balancing are to minimize the total execution time and the delay encountered when accessing the data in memory. Redistribution of balanced workload by means of tasks and minimizing the inter process communication needs optimal resource utilization and job response time. Hence, improving the performance of parallel computers by equalizing the workloads of processing elements is the aim of load balancing. Some of the main goals of load-balancing algorithms, as pointed out by [20] are:

(1) *Performance Improvement*: Achieve a greater overall improvement in system performance at a reasonable cost.

(2) *Workload Equality*: process all jobs that exist in the system equally in spite of their foundation.

(3) *Fault Tolerance*: performance survival under partial failure.

(4) *Ability of modification*: modify harmony with any change in the distributed system configuration.

(5) *System stability*: have the ability to be aware of sudden emergency cases in case of a lot of workload arrived and prevent nodes from spending much time passing the jobs between them.

### C. Categories of load balancing:

To achieve advantages of load balancing Based on the availability of a priori information, static and dynamic scheduling approaches can be used to schedule tasks on a multiprocessor architecture. In a static scheduling approach, it is assumed that all the relevant information for scheduling the tasks (e.g. the actual task execution times and processor-task affinity information) is available at compile time. The scheduling algorithm can use this a priori information to compute an optimal schedule. In many applications, however, the relevant information required for optimal scheduling is not available a priori, which may lead to a poor schedule computed by the static algorithm. The dynamic scheduling approach is used in such applications to gather relevant and accurate information at runtime to compute a schedule that is capable of meeting, more closely, the previously-mentioned objectives of task scheduling on multiprocessor architectures. The drawback of dynamic scheduling approach is that the time to gather useful information, and the time to use that information to compute an optimal schedule, can slow down the total execution time of the task [11].

*C.1 Static Load balancing algorithms:* There are different kinds of load-balancing technology (LBT) that are used widely, Selecting an appropriate load balancing strategy permits Common Service Information System (CSIS) to offer balancing service according to the load capacity of each server [21],[20].

*Round Robin balancing*: In a round-robin algorithm, the IP sprayer assigns the requests to a record of the servers on a rotating basis. The earliest request is allocated to a server selected randomly from the cluster, so that if additional than one IP sprayer is concerned, not all the original requests depart to the same server. For the subsequent requests, the IP sprayer goes after the circular sort to forward the request. Formerly a server is allocating a request; the server is going out to the end of the record. This remains the servers uniformly assigned. In computing, "round-robin" illustrate a technique of selecting a resource for a task from a list. Round-robin selection has an optimistic attribute of avoiding starvation, since every resource will be sooner or later chosen by the scheduler, excluding may be not fitting for some applications where similarity is desirable, for instance when handing over a process to a CPU or in link aggregation. The service requests from consumer are distributed to each server in a cluster in turn. This kind of equilibrium algorithm is appropriate for all the servers in a server cluster having the similar software and hardware configuration; and the average resource practice of each request relative to the same efficiency. This algorithm clearly belongs to static load balancing.

### D. Main polices in load balancing algorithms

Load balancing algorithms can be defined by their realization or execution of the following policies [18], [31]:

- *Information policy*: identifies what, when and from where jobs information to be gathered.
- *Firing policy*: determines the proper phase to begin a load balancing procedure.
- *Server or receiver type policy*: categorizes a resource as server or receiver of responsibilities according to its accessibility status.
- *Locality policy*: employs the results of the server or receiver type policy to detect an appropriate co-worker for a server or receiver.
- *Choice policy*: classify the responsibilities that should be moved around from overloaded resources (source) to most idle resources.

### *E. Analytical of load balancing algorithm*

Performance Analysis of Load Balancing Algorithms has been proposed in [30]. The authors present the performance analysis of various load-balancing algorithms based on different parameters, considering two typical load-balancing approaches static and dynamic. The analysis shows that static and dynamic algorithms can both have advancements as well as weaknesses over each other [30]. Load balancing algorithms work on the belief that in which situation workload is assigned, during compile time or at runtime. The above comparison shows that static load balancing algorithms are more stable in contrast to dynamic and it is also has the simplicity to guess the behavior of static, but at the same time, dynamic distributed algorithms are always measured better than static algorithms.

In [8], the authors made a qualitative parametric comparison of load balancing algorithms in parallel and distributed computing environment. Decrease in hardware costs and advances in computer networking technologies have led to increased interest in the use of large-scale parallel and distributed computing systems. One of the biggest issues in such systems is the development of effective techniques/algorithms for the distribution of the processes/load of a parallel program on multiple hosts to achieve goal(s) such as [8]:

1. Minimizing execution time.
2. Minimizing communication delays.
3. Maximizing resource utilization.
4. Maximizing throughput.

Research using queuing analysis and assuming job arrivals following a Poisson pattern, have shown that in a multi-host system the probability of one of the hosts being idle while other host has multiple jobs queued up can be very high. Such imbalances in system load suggest that performance can be improved by either transferring jobs from the currently heavily loaded hosts to the lightly loaded ones or distributing load evenly/fairly among the hosts .The algorithms known as load balancing algorithms, helps to achieve the above said goal(s).

These algorithms come into two basic categories - static and dynamic. Whereas Static Load Balancing algorithms (SLB) take decisions regarding assignment of tasks to processors based on the average estimated values of process execution times and communication delays at compile time, Dynamic Load Balancing algorithms (DLB) are adaptive to changing situations and take decisions at run time.

## III. Self-load balancing in overlay networks

In recent years, with the fast increasing magnitude of network systems and the internet, in addition to user's requirements, overlay networks come into view to be one of the significant issues in research community and the optimal solution in certain cases. An overlay network is a virtual network, which implements the end-hosts of the underlying network connecting them by logical connections. Overlay networks do not need new equipments or modifications to the underlying network. They can improve the performance and the final throughput, increase routing strength, and provide supplementary services not existing in the underlying network. Therefore they have become extensively deployed and more accepted [3].

By means of the huge development in computer machinery in addition to the ease of use of many distributed systems, the difficulty of load balancing in distributed systems has increase a superior awareness and significance. As a result, a huge quantity plus diversity of investigate has been carried out in a challenge to resolve this trouble. Classification of load balancing algorithms in distributed systems are reported in [1], [34], and [33]. Solutions to the load balancing problems are divided into two main approaches depending on whether a load balancing algorithm bases its decisions on the current state of the system or not: static and dynamic. As we described load balancing algorithms can be classified into two categories: static or dynamic. In static algorithms, the decisions related to load balance are made at compile time when resource requirements are estimated. Multicomputer with dynamic load balancing allocate or reallocate resources at runtime based on no a priori task information, which may determine when and whose tasks can be migrated[5],[2]. In the present network nowadays all of them have common factors such as heterogeneity, scalability, and adaptability in addition to request load and resource administration. Several load balancing types or strategies and algorithms have been proposed to get better the global throughput of these software situations, workloads have to be equally planned between the accessible resources. Most strategies were developed in state of mind, assuming homogeneous set of sites linked with homogeneous and high-speed networks.

IBM Corporation in 2001, introduced the concept of (AC) to overcome this complexity. It is inspired from the human biological system. IBM envisioned a computing environment with the capability to manage itself and dynamically adapt to change in accordance with business policies and objective through a set of self-managing functions such as self-configuring, self-healing, self-optimizing, and self-protecting. To this end an architectural blueprint for AC has been proposed by IBM and has been revised in 2002[15], 2003[24], 2004[98], 2005[29], and finally in 2006[22]. The blueprint defines concepts and constructs for building self-managing abilities into modern computer systems as well as architectural building blocks of these abilities. The goal of AC is thus to manage complexity (technology manage technology), to reduce cost of ownership (automation reduces human involvement/error) and to enhance other software qualities [25]. AC systems are used to automate the management of a resource (Hardware or software) such as storage, server, network, etc. The resource is monitored for significant events and controlled accordingly. An interface is used for sensing a change in the monitored resources and another is used to enforce a behavior for managed resources to react autonomically and manage the targeted environment with minimal human intervention.

The authors in [31], provide a new relationship control technique standing ahead maintaining a local leader for each group. The local leader acquires information from the normal nodes as well as from other neighboring leaders. It then from time to time makes a decision whether to offer out servers to other services, or asks other local leaders for extra servers. The authors present their proposal for a self-adapting system that offers several services using a group of server nodes. We can summarize it as follow: each node in the network maintaining a table for its neighbors and from time to time update this table.

Capital expenditures (capex) stay the majority closely watched metric for determining the direction and altitude of different investment that telecommunications carriers are making in network devices, equipment and services [7].The authors in [19], proposed a novel self-organizing load-balancing framework called Self-Organizing Network (SON). The complexity management can be solved using the concept of Autonomic Computing (AC). AC has been proposed by IBM to allow systems to mange themselves instead of relying on IT professionals [22]. SON aims at development by improving network performance throughput and reducing final costs of configuration and management by make straight forwarding operational tasks through automated mechanisms such as self-configuration, self-optimization, and self-healing functionalities [28]. The structure meeting point on the novel Self-organizing Cooperative Partner Cluster (SCPC) and Comprehensive Load Balancing Policy Stack (CLBPS). Weigh against with the existing SON load balancing; this structure investigates the Fixed Relay Station (FRS) networks with novel signaling costs and user experience factors.

Balanced Overlay Networks (BON) [6] is a decentralized load-balancing scheme. BON is scalable, self-organized, and relies only on local information to build job assignment decisions. New jobs are allocated to a node by a random walk on the graph which not only samples the graph preferentially but also selects the highest-degree node that was visited on the walk. Each node's idle resources are relative to its degree, so this approach works very fine when a network is not loaded further than its clipping point. When a BON is clipped, the relationship between load and in-degree breaks down, but the balancing performance remains quite good due to the so-called "power of two choices" in ball-bin load balancing. Based on previous theoretical results and extensive simulation results, BON is seen to be efficient and practical. Added ongoing work on this difficulty includes geographical alertness extensions using more difficult walk objective functions. Lastly, it should be distinguished that this is only one probable way to code information about a network in its topology; other distributed algorithms may benefit from using graph state to favoritism node selection.

The authors in [17] presented a self-adaptive distributed balancing scheme for large-scale High Level Architecture HLA-based simulations. The adjustment method intends to decline the amount of unwanted balancing migrations although maintain simulation performance increases similar to or better than the distributed balancing system. The reworked copy generates modifications for the balancing system's parameters to control the system's sensitivity. Such alterations are based on the incoming and outgoing relocation moves executed.

A spatial index is a particular way in method used to retrieve records from within the record-store. Spatial indexes allow consumers to pleasure data within a data-store as existing within a two dimensional context. A spatial index is a grid divided into an amount of rectangles or cells. All cells have the equal width and height. All cells, jointly, map a single large rectangle. Two fields are necessary from any record to map the record to the grid effectively, the (X,Y) coordinate pair. More over spatial indexing depends on the physical grouping of entries that are indexed. For example, countries can be assembled by continent. With spatial indexing, it is yet probable to offer VR (virtual reality) fly through (or fly over, or fly under) to assist consumers carry on the context of their search as the users increase or decrease the scope of their explore. Spatial indexing has a prosperity that parallels indexing by subject or author. The idea of spatial indexing is extremely powerful for records management, records are established based on their association with a position. Many records are strongly coupled to a position like other forms of indexing; geographic indexing can be combined with other indices in [16,9,27].

Quad trees are the majority frequently used to division a two dimensional space by recursively subdividing it into four quadrants or regions. The regions may be square or rectangular, or may have random shapes. A quadtree is a tree data structure in which every interior node has up to four children. This data structure was named a quadtree by Raphael Finkel and J.L. Bentley in 1974. A related partitioning is also known as a Q-tree. Each and every one form of Quadtrees share some ordinary features, like: They decompose space into adjustable cells; each cell (or bucket) has a maximum capability. When maximum capability is achieved, the bucket splits, and the tree directory follows the spatial decomposition of the Quadtree [10, 13, 32, 35, 23, 14, 4].

Z-order is a space-filling curve which is frequently implemented in computer science (aka Morton-order, first proposed by G. M. Morton), in order to its good neighborhood caring for behavior it is used in data structures for mapping multidimensional data to one dimension. The z-value of a multidimensional point is basically calculated by bitwise interleaving its coordinate values. Every one-dimensional data structure can be used such as binary search trees, Btrees, Skip lists or (with low significant bits truncated) Hash tables [16, 4]

## IV. Proposed autonomic overlay self-load balancing

In our work, we propose a layered algorithm that achieves dynamic and self-load balancing. It is based on a quadtrees model. It consists of the following main features: (i) It is layered; (ii) it supports heterogeneity and scalability; (iii) it is independent from any physical architecture such as grid, (iv) it achieves self-load balancing depending only on local knowledge for each MPs.

The algorithm makes use of spatial indexing; it starts by partitioning the geographical area of the network into equal section. Each subsection is partitioned in the same way. It will continue partitioning until we reach a predefined goal or until each node resides in a single partition. Each partitioning stage results in a separate level. Then each individual partition is indexed using Morton code and is given a unique ID. These indexes are used to build a distributed quad-tree (DQT). However, these nodes IDs are logical and it has to be tied to a physical node address. To achieve this each node broadcast its logical and physical IDs to its neighbors that are 2 hops away. Once the physical and logical IDs are linked, each node can know from the Morton code provided by the logical IDs its parent. And thus the tree is formed.

To achieve load balancing, each node computes the *Nodepower*. The Nodepower is computed locally by each node and it is a weighted value for the computing resources of that node such as memory, CPU speed, available harddisk space, and bandwidth in addition to the bus speed. Each node then decides based on its power the number of incoming edges for it. The incoming edges are few for nodes that have low Nodepower and increases with the Nodepower. To guarantee connected resources the minimum incoming edges is restricted to 4 while the maximum depends on the Nodepower. It is important to note that the incoming edges to a certain node are computed for nodes that are expected to provide a service. i.e it is for MPs only. This results in another logical layer in the network. The first layer is the quad-tree layer and the second on is the incoming edges for resources.

Each incoming each to a certain MP has to be linked with another MP of the same type from the other end. This will results in having all MPs that provide the same service connected together using the incoming edges which will facilitates the process of distributing the load between them later. The problem of selecting the other end of each incoming edge is solved using the quad-tree. Each MP is a member in the quad-tree. Therefore, it will send a message to its parent in the tree requesting a certain number of MPs of the same type to be linked to itself. The parent starts assigning from its children, if it is not enough, it will send the same message to its parent until we have a full assignment.

After the construction of the second layer is complete, the network will be ready to achieve load balancing. Once a job enters the network, it will be analyzed to find out the type of service it is requesting. This will decide which type of MP will be used to serve the job. Using the quad-tree, the job will be forwarded to nearest MP of the requested type. Then the receiving MP will follow the incoming edges to direct the job to the least busy MP. Once the job is started processing, the available Nodepower of the selected MP is decreased based on the current job needs. Therefore, the node incoming edges will be recomputed and decreased to reflect the actual available Nodepower. The decrease of the incoming edges implies that some edges have to be deleted. The nodes that will be deleted will be selected using the quad-tree.

## V. Conclusion

Load balancing is a challenge due to the increased complexity, cost, and heterogeneity of current technologies. Autonomic computing can indeed be used to solve this increasing complexity. In this paper we reviewed the state of the art in load balancing techniques. Our study shows that traditional techniques are not adequate to face the ever increasing challenge and complexity of technology. To this end, a novel self-load balancing scheme for autonomic overlay networks is proposed. The scheme employs the spatial index and partitioning the network to build a distributed quad-tree. Another logical layer is built based on the vailable resources. This layer is connecting resources of the same type thus facilitating the process of load balancing. Local knowledge is exploited to achieve a better performance. We are currently implementing the proposed method to quantify its cost and efficiency.

## References

[1] A. Goscinski, “Distributed Operating Systems,” Addison-Wesley, Sydney, 1991.

[2] Alakeel A. A Guide to Dynamic Load Balancing in Distributed Computer Systems, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.

[3] Al-Oqily,I,Karmouch,A, and Glitho ,R. (2008).IEEE, 'An Architecture for Multimedia Delivery Over Service Specific Overlay Networks'.

[4] ASHRAF ABOULNAGA, WALID G. AREF,(2001),Citeseerx,'Window Query Processing in Linear Quadtrees'.

[5] Belabbas Yagoubi and Yahya Slimani "Dynamic Load Balancing Strategy for Grid Computing" 2006, citeseerx,2006.

[6] Bridgewater Jesse S.A., Boykin P. Oscar, , Member, IEEE, and Roychowdhury VwaniP. (2007).IEEE, 'Balanced Overlay Networks (BON): An Overlay Technology for Decentralized Load Balancing'.

[7] Celentano, J.M.; , "Carrier Capital Expenditures," *Communications Magazine, IEEE* , vol.46, no.7, pp.82-88, July 2008.

[8] Chhabra Amit, Singh Gurvinder, Sandeep Singh Waraich, Bhavneet Sidhu, and Gaurav Kumar, (2006).IEEE,'Qualitative Parametric Comparison of Load Balancing

[9] Demirbas, M.; Xuming Lu;(2007),IEEE,'Distributed Quadtree for Spatial Querying in Wireless Sensor Networks'.

[10] Dooley, R.; (2004),IEEE,'A Distributed Quadtree Dictionary Approach to Multi-Resolution Compression'.

[11] E. Badidi. *Architecture and services for load balancing in object distributed systems*. PhD thesis, Faculty of High Studies, University of Montreal, Mai 2000.

[12] Eager D.L., Lazowski E.D., and Zahorjan J.,(1986)'Adaptive Load Sharing in Homogeneous Distributed Systems'.

[13] Egemen Tanin, Aaron Harwood and Hanan Samet (2005),ACM,' A Distributed Quadtree Index for Peer-to-Peer Settings'.

[14] G M Schuster, A K Katsaggelos ,(1998),IEEE,'An Optimal Quadtree-Based Motion Estimation and Motion-Compensated Interpolation Scheme for Video Compression'.

[15] G. M. Lohman and S. S. Lightstone. Smart:Making db2 (more) autonomic. In VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases, pages 877–879. VLDB Endowment, 2002.

[16] Geographic information system,2011,access date 4/07/2011 at 13.40.22 <http://www.gis.com>

[17] Grande, R.E.D.; Boukerche, A.; , "Self-Adaptive Dynamic Load Balancing for Large-Scale HLA-Based Simulations," *Distributed Simulation and Real Time Applications (DS-RT), 2010 IEEE/ACM 14th International Symposium on* , vol., no., pp.14-21, 17-20 Oct. 2010

[18] H.D. Karatza. Job scheduling in heterogeneous distributed systems. *Journal. of Systems and Software*, 56:203–212, 1994.

[19] Heng Zhang; Xuesong Qiu; Luoming Meng; Xidong Zhang; , "Design of Distributed and Autonomic Load Balancing for Self-Organization LTE," *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd* , vol., no., pp.1-5, 6-9 Sept. 2010

[20] http://camel.apache.org/load-balancer.html

[21] http://fusesource.com/docs/ide/camel/1.0/eip/MsgRoutLoadBalancer.html#_IDU_LoadBalancer_HSH_Failover

[22] IBM Corporation,(2006),'An architectural blueprint for autonomic computing', White Paper.

[23] Jamel Tayeb , Özgür Ulusoy , Ouri Wolfson (1998),Citeseer,'A Quadtree-Based Dynamic Attribute Indexing Method'.

[24] Kephart, J.O.; Chess, D.M.; , "The vision of autonomic computing," *Computer* , vol.36, no.1, pp. 41- 50, Jan 2003 [98]M. Parashar and S. Hariri. Autonomic computing: An overview. In UPP, pages 257–269, 2004.

[25] Khalid, A.; Haye, M.A.; Khan, M.J.; Shamail, S.; , "Survey of Frameworks, Architectures and Techniques in Autonomic Computing," *Autonomic and Autonomous Systems, 2009. ICAS '09. Fifth International Conference on* , vol., no., pp.220-225, 20-25 April 2009

[26] Lexi Xu; Yue Chen; Yue Gao; , "Self-organizing Load Balancing for Relay Based Cellular Networks," *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on* , vol., no., pp.791-796, June 29 2010-July 1 2010

[27] Myoung-Ah Kang and Sylvie Servigne and Ki-Joune Li and Robert Laurini, (1999), doi.acm.org,' Indexing Field Vahes in Field Oriented Systems: Interval Quadtree'.

[28] Prehofer, C.; Bettstetter, C.; , "Self-organization in communication networks: principles and design paradigms," *Communications Magazine, IEEE* , vol.43, no.7, pp. 78- 85, July 2005

[29] Roblee, C.; Berk, V.; Cybenko, G.; , "Implementing Large-Scale Autonomic Server Monitoring Using Process Query Systems," *Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on* , vol., no., pp.123-133, 13-16 June 2005

[30] Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma,(2008),IEEE.'Performance Analysis of Load Balancing Algorithms'.

[31] Senbel, S.A.; , "Load-balancing in a self-organizing server cluster using local leaders," *Informatics and Systems (INFOS), 2010 The 7th International Conference on* , vol., no., pp.1-8, 28-30 March 2010

[32] Stéphane Popinet, Richard M Gorman, Graham J Rickard, Hendrik L Tolman, (2010),mendeley'A quadtree-adaptive spectral wave model'.

[33] T. L. Casavant, "A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems," IEEE Trans. Software Eng., vol 14, no. 2, pp 141-154, February 1988.

[34] Y. Wang and R. Morris, "Load Sharing in Distributed Systems," IEEE Trans. Comput., vol. C-34, no. 3, pp. 204-217, Mar. 1985.

[35] Zeeshan Hameed Mir, Young-Bae Ko.,(2007),Springerlink,'A Quadtree-Based Data Dissemination Protocol for Wireless Sensor Networks with Mobile Sinks'