International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:7, No:3, 2013

Towards a Load Balancing Framework for an SMS–Based Service Invocation Environment

Mandla T. Nene, Edgar.Jembere, Matthew O. Adigun, Themba Shezi, and Siyabonga S. Cebekhulu

Abstract**—**The drastic increase in the usage of SMS technology has led service providers to seek for a solution that enable users of mobile devices to access services through SMSs. This has resulted in the proposal of solutions towards SMS-based service invocation in service oriented environments. However, the dynamic nature of service-oriented environments coupled with sudden load peaks generated by service request, poses performance challenges to infrastructures for supporting SMS-based service invocation. To address this problem we adopt load balancing techniques. A load balancing model with adaptive load balancing and load monitoring mechanisms as its key constructs is proposed. The load balancing model then led to realization of Least Loaded Load Balancing Framework (LLLBF). Evaluation of LLLBF benchmarked with round robin (RR) scheme on the queuing approach showed LLLBF outperformed RR in terms of response time and throughput. However, LLLBF achieved better result in the cost of high processing power**.**

Keywords**—**SMS (Short Message Service), LLLBF (Least Loaded Load Balancing Framework), Service Oriented Computing (SOC).

I. INTRODUCTION

SMS has long been established as the de facto standard for sending and receiving text messages on mobile phones [1]. Its popularity has led to its use by service providers as an alternative means of technology to render services which allows service consumer (mobile user) to reach services from a service provider by requesting and retrieving content via SMS. This is known as SMS-based service invocation in SOC environments. The SOC environment makes SMS-based service invocation feasible by enabling mobile users to access the wealth of applications which were otherwise only accessible through personal computers. However, SOC environments are by nature dynamic and composed of autonomous entities, which makes them unpredictable and difficult to manage [2]. One of the entities in such environments is service consumers, which are known to behave unexpectedly in relation to generating traffic. This implies that service providers may receive very few requests from service consumers for a given service at one time and subsequently receive heavy number of requests other times thereby overloading service provider.

Load balancing in literature (e.g. [3], [4]) is regarded as one of the techniques for addressing the sudden load peaks in a dynamic environment such as SMS-based service invocation. Load balancing in this work is defined as a process that evenly distributes traffic amongst computers (i.e. servers) hence solving overloading so that no single computer is overwhelmed. Load balancing provides different schemes and there are traditionally implemented in a mechanism called load balancer that acts as front end of the service providers' servers as depicted in Fig. 1 (a). The load balancer is responsible for load (i.e. requests) scheduling. The load balancer in this work is the SMS broker which acts as front end for the service providers servers that provide the content and services as shown in the architecture by [6], [7]. This SMS broker deals with translating SMS requests sent by service consumers to HTTP requests and direct the requests to a service provider servers using some load balancing techniques as depicted in Fig. 1 (b).

Scholars like [5] proposed queuing approach which implemented store and forward mechanism. The forwarding mechanism is load distributor that implements simple load balancing schemes [5] such as the round robin (RR) scheme that distributes load around service provider servers iteratively. We argue that using RR which does not consider any system state information and this may lead to poor load distribution decisions. Thus, affecting the system performance. Based on such flaw, the RR then is not appropriate load balancing technique for such dynamic environments as SMS-based service invocation.

Mandla Nene is with the Department of Computer Science and Mobile e-service in the University of Zululand, Private Bag X1001, KwaDlangezwa 3886. South Africa (email:mandlatnene@gmail.com).
Edgar Jembere is with the Department of Computer Science and Mobile e-service in the University of Zululand, Private Bag X1001, KwaDlangezwa 3886. South Africa (email:ejembere@gmail.com).
Matthew Adigun is with the Department of Computer Science and Mobile e-service in the University of Zululand, Private Bag X1001, KwaDlangezwa 3886. South Africa (madigun@pan.uzulu.ac.za).

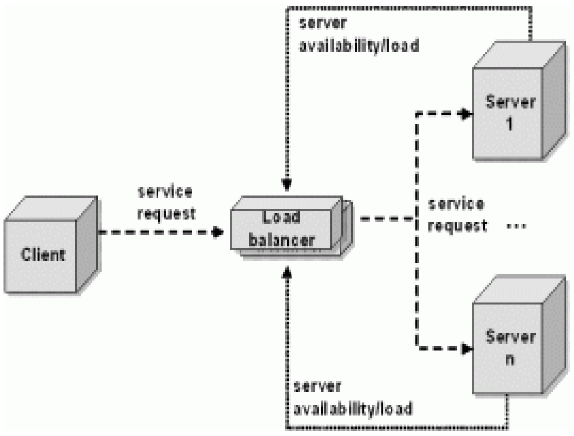

Fig. 1 (a) Conventional load balancing

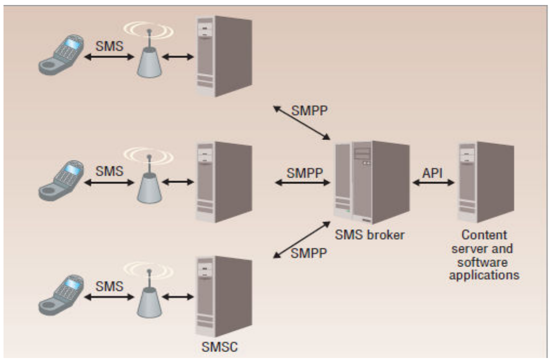

Fig. 1 (b) SMS based service invocation architecture (adopted from [6])

To address the aforementioned challenge, this paper proposed a load balancing model made up of adaptive load handling and load monitoring mechanisms .This model is able to adapt to change occurring SOC by relying on monitored system status. Based on specification of the model we crafted LLLBF that allowed these mechanisms introduced in load balancing model to work together to try the goal of this work.

The rest of the paper is organized as follows. Section II discusses literature reviewed. The design considerations of our load balancing Model that assist to achieve our goal, are discussed Section III. Section IV presents our proposed load balancing Model. In Section V presents a load balancing framework derived from proposed load balancing Model. Section VI discusses performance evaluation of the model. We conclude the paper with the discussion of the implications of our findings and our future work in section VII.

II. Related Work

A lot of work has been done in distributed environment related to load balancing. The major goal is to improve performance of such system. In order to achieve this goal various load balancing schemes have been introduced at different levels.

Currently, SMS-based service invocation implements queuing mechanism that distribute load using simple load balancing scheme such round robin scheme proposed by [5] .The RR [8], [9], [10] scheme assigns request load on a rotational basis between servers. However, the round robin scheme makes assumption when distributing load i.e. it take guesses when distributing load because it has less or no information about the system status hence RR is not appropriate scheme for dynamic environment. The other flaw is RR scheme can only serve homogeneous environment while most of distributed system today are heterogeneous. Based on RR scheme flaws we explored load balancing schemes that can better deal with the dynamic nature of distributed environments. This load balancing approaches are briefly described below.

More successful and accurate load balancing requires load balancing strategy to have some notion of the server load in order adapt the load balancing weight to the current load. This can be done by monitoring servers' behaviour .The following schemes tries to achieve latter requirement. The load sharing scheme [11], [12], [13] achieves that by balances the load between servers by checking which ones are heavily or lightly loaded using load metrics being monitored. The requests are moved from the heavily loaded server to the lightly loaded server (offloading) .This is a well-known approach but it incurs some delay when migrating load.

In [14], [15], [16] presented round-trip scheme assigns requests to the server that is responding the fastest based on monitored response times of servers .However, this scheme provides best effort service. The round-trip scheme is derived from least loaded approach that distributes requests to a group of servers, based on which server is currently has lowest load index [17], [18], [19]. Least loaded achieves this through monitoring some threshold value or load index metric. The least loaded scheme has been widely adopted and it has shown its robustness and flexibility by being applicable in environments such as networking for load balancing in routing level [20], [21]. Moreover, this scheme can be combined with other solutions to achieve a good load balancing solution [21], [27]. However, the least loaded scheme can have possible delay depending on the monitoring approach chosen to gather system current information.

From the foregoing; it is clear that distributed computing environment such as SMS based service invocation must support dynamic and adaptive load balancing in order to handle dynamic requests loads. Therefore in this paper, we proposed load balancing model that can achieve the latter objective.

III. Design Criteria for a Load Balancing Framework

As literature suggested that load balancing techniques can be used to alleviate performance challenges faced by infrastructure in distributed environment such as SMS based service invocation. Our work is aimed at modelling the load balancing solution for the SMS broker in order to enable to cope with sudden burst loads in a scalable manner. From our review of literature, we have identified the design criteria to take into consideration when designing a load balancing model for SMS invocation of service environment. The design criteria for crafting the load balancing model include:

A. Adaptive Load Handling Scheme

The environment we are dealing with is dynamic which implies that such an environment is capable of manifesting unpredictable behaviour. During SMS invocation of services in service oriented environment, an abnormal behaviour from mobile users can lead to unexpected traffic or sudden load peaks or congestion in the service provider's servers. In order to address this issue, the environment needs an adaptive load handling scheme which adapt to change of the environment and make decision based on the systems current status being monitored. This load handling scheme should be able to make load balancing decisions based on particular situation occurring in the system which is reflected through monitored system status information.

B. Provision of Load Monitoring Assistance

The adaptive load handling scheme must be complemented in terms of making fair load distribution decision by monitoring the system current state and reporting the load situation based on load index metrics being monitored [24].This load monitoring mechanism should supply information about system status before an adaptive load handling scheme make load balancing decision.

IV. Proposed Load balancing Model

From the design criteria outlined in Section III, a load balancing model for SMS based service invocation environments is proposed. The model proposed in this work is aimed at providing load balancing approach that can improve performance in dynamic environment such as SMS based service invocation environment even when there are overwhelmed with heavy traffic. The model derives from the idea that a successful and accurate load balancing solution should be aware of system status whenever distributing load. Therefore, our load balancing model consists of two parts: an adaptive load handling mechanism and a load monitoring mechanism.

A. Adaptive Load Handling Mechanism

Based on the need for an adaptive load handling requirement, the least loaded scheme was chosen [14], [17] [18]. This scheme was chosen because of its dynamicity, adaptiveness and performance when working with stress load [18]. Moreover, [17] showed that, it outperforms the Minimum, Threshold, Random, and Round robin approach in service-oriented environment where Enterprise Service bus was tested for the best routing scheme that can suit it. The least loaded load balancing scheme can be adapted to suit any load balancing environment owing to its flexibility [19], [20].The least loaded scheme has requests transfer policy which determines whether a particular service provider's server is suitable for receiving requests. This least loaded scheme transfer policy allows a server or service endpoint to continue receiving requests if the server currently has least load than the other servers based on monitored load index metric. The load index metric is used as a deciding factor to know which server is the most appropriate to receive requests at that period of time. In order for that to happen, the least loaded scheme transfer policy is supported by load information policy which disseminates information about the status of each service provider's servers. The load Information policy is a load monitoring mechanism that supports least loaded scheme to make appropriate load distributed decision. The following section discusses load monitoring mechanism for the purpose of load balancing.

B. Load Monitoring Mechanism

As literature suggested that promising load balancing schemes is required to have some notion of the system's current status before making load balancing decisions [9], [23]. This is enabled by having a monitoring mechanism that collects information about system status using a periodic or an on-demand approach. In this work an on-demand approach is preferred over periodic approach because ensures that collected information is never outdated [26]. The on-demand load monitoring mechanism collects system status information only when it is needed for load balancing. That is, the information is collected dynamically when there is load to be distributed otherwise if there is no load then there is no need for collecting the information. The load monitoring mechanism usually uses the following metrics to support load balancing decision making: CPU utilization, memory utilization, IO utilization and Bandwidth occupancy [24], [25]. These load metrics can be used individually or collectively depending on the type of load the application or service consumer requests exert on the system [25]. In our case, we select CPU utilization as our monitored load metric based on assumption that client requests are more demanding on computational resources. Moreover, it is a well-known load index with minimum overhead [28]. The other reason is that when monitoring load for any server machine, the calculation used to get load metric should not be computational expensive [22]. This has been proven by [28] that CPU load index incurs a minimum overhead. The CPU load index metric is given by division CPU utilization (CPUu), CPU capacity (CPUc) and it is supported by CPU idle time metric. The CPU load index metric is used for getting the status of service provider's servers which is collected by load monitoring mechanism and published to the adaptive load handling mechanism containing least loaded scheme so that load is distributed. From the above presented mechanisms there is need find way to connect them so that they form part of SMS-based service invocation system which is discussed in the next section.

V. Least loaded Load balancing Framework

Based on the specification of adaptive load handling and load monitoring mechanisms mentioned in previous section the proposed load balancing model has led to Least Loaded Load Balancing Framework (LLLBF). This LLLBF have components derived from adaptive load handling and load monitoring mechanisms .The components are load balancing decision maker and load monitor as shown in Fig. 2 as result of crafted LLLBF. The LLLBF employ event based communication style which contributes to its scalability [29] because it allows independencies between components thus, the components communicate with one another having little or no knowledge of each other. Event based communication style brings in loose coupling which facilitate scalability in such way that a system can grow without any effect on other components. Thus, the LLLBF take in producer/consumer paradigm which means load balancing decision maker components acts as consumer while load monitor component acts as producer. This LLLBF is incorporated in load dispatching mechanism such SMS broker.

The interactions between components are as follows: the load balancing decision maker component receives requests from service consumer. The requests are taken and distributed among available service provider's servers based on the least loaded scheme which implemented and it makes decisions rely on status information of service provider's servers published on-demand by load monitor component this means that load monitoring (producer) does information dissemination that assists the load balancing.

.

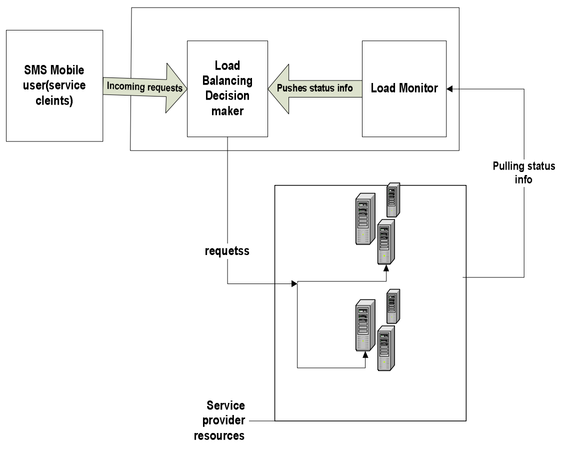
SMS Mobile user(service cleints)
Incoming requests
Load Balancing Decision maker
Pushes status info
Load Monitor
Pulling status info
requetss
Service provider resources

Fig. 2 Least Loaded Load Balancing Framework

A. Load Balancing Decision Maker Component

The load balancing decision maker component is the component for managing the distribution of requests. The actual destination of a request is decided using a load balancing strategy. This component receives load information on-demand from load monitor component, and uses the obtained data to check which service application has the least amount of load at that time.

The load balancing strategy implemented by load balancing decision making component is called *least loaded scheme* which works as follows: The least loaded scheme distributes requests to server with the lowest load index and in this case we used CPU utilization as a load index. The CPU load index information of service provider servers is pulled by load monitor on-demand and published in load balancing decision maker component for load balancing purposes.

B. Load Monitor Component

The Load monitor is responsible for connecting to service provider's servers to collect load information. It acts as a producer for the load balancing decision maker component by publishing monitored status information which is based CPU load index metric of service provider's servers. This status information is used by load balancing decision component to decide which service provider's server to send the load to.

The load monitor component works as follows: The *on-demand approach* is used to achieve gathering of system status information that means status information of each server is fetched and published to load balancing decision making component whenever it needed so that load balancing decision can be made. For the load monitor component to gather CPU load index for each of the servers it used an Application Programming Interface (API) which is called Hyper Sigar [30].

VI. Performance Evaluation

To evaluate the efficacy of the LLLBF, we investigated the performance of the LLLBF compares with RR which is currently used in SMS invocation of services. The evaluation of the LLLBF against RR in this paper presents one of two parameters which is scalability. Scalability is defined as the ability of a system to handle growing amounts of work in a graceful manner or its ability to be enlarged to accommodate that growth. This section entails the following subsection given below.

A. Testbed Specification

In developing our testbed the following assumptions were considered due to the duration of this project and considerations of the environment where the LLLBF will be used.

1. We assumed the web services exposed by the service provider servers are purely computational services. As a consequence, their execution time is directly proportional to the amount of service requests sent by service consumers.
2. We assume that the network delay is constant throughout the experimentation.

B. Testbed Setup and Environment

The testbed setup consisted of the LLLBF and the RR scheme that served as a benchmark. We chose RR as our benchmark because the argument revolved around it. These load balancing approaches were implemented and incorporated in a Synapse engine. The synapse engine served as an SMS broker responsible for scheduling load to service provider servers. The Synapse engine comes with a set of transport, mediator and standard brokering capabilities, such as round-robin, weighted round robin load balancing scheme and fail-over. Based on the capabilities of Synapse engine, it is used as a load balancer where we deployed our own load balancing approach using the existing pattern.

To mimic the SMS-based service invocation environment, we used 3 machines running Windows 7 OS, each serving their own purpose. The first machine was used to simulate clients via Apache HTTP load generator. For simulation of client's requests we chose a machine that is fast so that heavy loads can be imposed on other machine that acts as servers and we got inspiration from [18]. To further avoid potential resource constraints, the Synapse engine was deployed on the same machine were the client's simulator (load generator) was running.

The machine that was used for simulation of client HTTP requests and hosting the Synapse engine was running on an Intel Core i5 3.20 GHz PC with a RAM of 3GB and Hyper-Threading Technology. The second and third machines were homogeneous servers running on Intel Core2Duo 2.94GHz PC with RAM of 2GB. Each of these two machines was running Apache Axis web service engine coordinated by the Apache Synapse engine running on the first machine. The servers are used for serving requests coming from the client simulator machine that generated the web service requests containing. These requests were distributed among the servers using Apache Synapse engine containing our LLLBF and RR scheme for benchmarking purposes. The web service that the servers were running was a similar pure computationally web service replicated in both servers. This computational service finds permutations of 5 elements (i.e. given "abcde", what are the possible ways that these characters can be ordered?). All these machines were connected through our departmental wired LAN network.

TABLE I
PRESENTS CHARACTERIZATION OF LLLBF AND THE RR

	LLLBF	RR
Nature	Dynamic	Static
Adaptability	More adaptive	Less adaptive
Centralized	Yes	Yes
Load balancing policy	Least loaded scheme	Round Robin scheme
Overhead	More	Less
Simplicity	Not simple to implement	Simple to implement

C. Experimental Setup

We use response time and throughput to investigate how each load balancing scheme scales with increases in the number of client requests. Response time is measured as the time taken from when a request is send when a response is received. Throughput is defined as the maximum number of requests a load balancing approach can process within a unit time. For further analysis investigated how does the LLLBF consume resource such CPU as the workload increases compared to the RR. The CPU utilization is the amount CPU time taken to send a request and to receive the response. For each number of requests 10 runs were carried out and the above-mentioned metric were observed. The averages of each metric over the 10 runs for each number of the request were recorded against their corresponding number of requests. The metrics at the client side or front end were obtained as shown in the procedure below .Two overloading variants were considered for the experiments. For this paper we only presented which involved overloading server interchangeable throughout the process of sending requests. This means one server is overloaded at certain time while the other server is free and the next moment the free server is overloaded while the overloaded server is relieved. The experimental procedure goes as follows;

1. Deploy one computational service replicated in two Servers(service endpoint)
2. Pass *n* number of requests to the Synapse engine using Load generator
3. Record the response time, throughput, and CPU utilization at each group or bulk requests after processing completed

D. Experimental Result and Analysis

Fig. 3 shows that the response time is directly proportional to the number of requests sent. However, we noted LLLBF has better response time than RR. From this particular experiment we conclude that LLLBF was able cope increasing amount of load while providing better system responsiveness compare to RR. Moreover, Fig. 4 presents inverse of throughput graph and the reason behind having inverse of throughput graph is that original throughput graph does allow for asymptotic analysis. Fig.4 shows that the inverse of throughput increases as number of requests increase for both LLLBF and RR. Consequently; we observed that the LLLBF has better inverse throughput than RR i.e. LLLBF could process more request at certain period of time. From Fig. 3 and 4 showed the LLLBF and RR scheme there are both scalable .For furthermore analysis we investigated computation resource such as CPU to observe how each of the load balancing solution utilizes this type of resource, Fig. 5 shows that the CPU utilization increases as the number of client's requests increases on both the LLLBF and RR. From Fig. 6 we observed LLLBF has higher CPU utilization than RR. We concluded that this is due LLLBF having more capabilities than RR scheme.

Based on above result we conclude LLLBF provide better performance than RR in cases where SMS-based service invocation environment is dealing with traffic or sudden load peak. LLLBF achieves better performance in trade-off requiring bit of computational power.

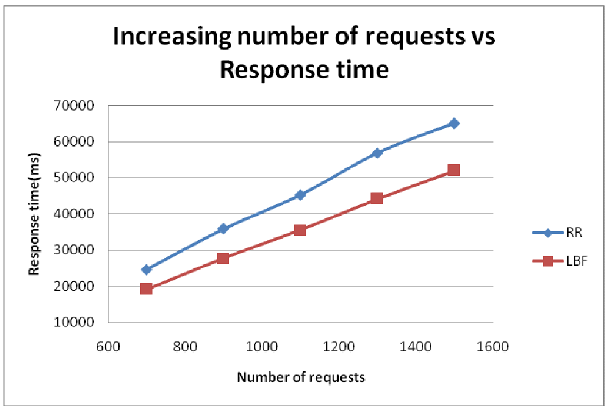

Fig. 3 Response Time vs. Number of requests for Scalability of LLLBF and RR scheme

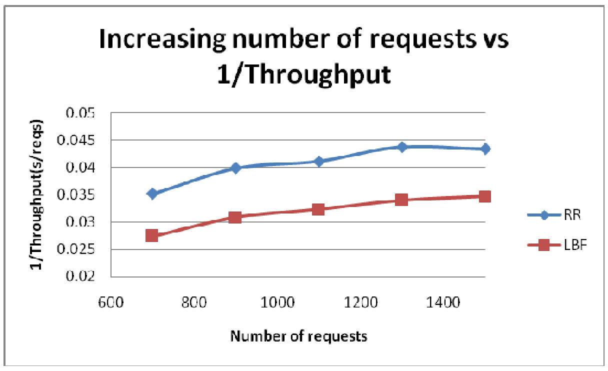

Fig. 4 Inverse Throughput vs. Number of requests for Scalability of LLLBF and RR scheme

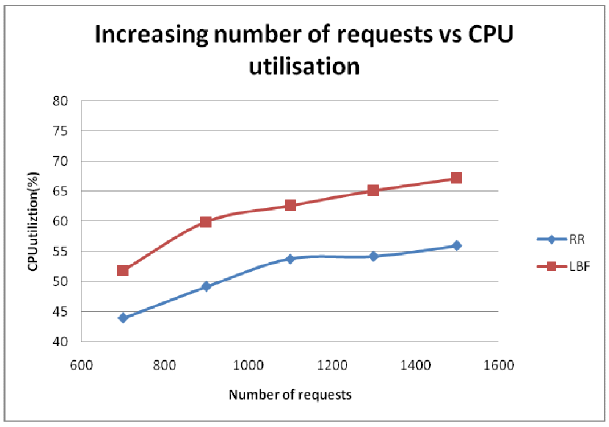

Fig. 5 CPU utilization vs. Number of requests for Scalability of LLLBF and RR scheme

VII. Conclusion and Future Work

In this work we have proposed a load balancing model consisting of adaptive load handling mechanism and load monitoring mechanism. This model led to realization of LLLBF which is our solution that tries to address the issue of sudden load peaks in the SMS based service invocation environment which imposes performance degradation to the service provider infrastructure. The key to this framework is the introduction of dynamic and adaptive load balancing scheme i.e. Least Loaded scheme. This load balancing scheme attempts to use runtime state information of service provider servers to more accurate decision in sharing the system load. We implemented our LLLBF and conducted experiments. Our main contribution, in this work is that our LLLBF is able to cope with increasing amount of load in scalable manner while providing better performance in such environments as SMS based service invocation. This is shown through scalability (load testing) experimentations by comparing LLLBF with RR where LLLBF outperformed RR even though LLLBF incurred some cost in resources such as CPU .As for future work; we plan on incorporating awareness mechanism in our LLLBF so that it can serve clients based on their categories if service providers has premium and regular client who requires different service quality.

References

[1] Risi, D. and Teófilo, M. 2009. MobileDeck: turning SMS into a rich user experience. In Proceedings of the 6th international Conference on Mobile Technology, Application & Systems (Nice, France,September 02 - 04, 2009). Mobility '09. ACM, New York, NY, pp. 1-4. DOI=http://doi.acm.org/10.1145/1710035.1710068
[2] Papazoglou M.P., *Web Services: Principles and Technology,*Prentice Hall, 2007.
[3] Kopparapu, Chandra." Load balancing servers, firewalls, and caches". Wiley Computer Publishing John Wiley & Sons, Inc. 2002
[4] Tony Bourke , Server Load Balancing O'Reilly & Associates, Inc., 2001
[5] Ramana Kumar K,Manesh V Ghatage Jataayu,"Load balancing of servicers with server initiated connections",2005.
[6] Brown, J., Shipman, B., and Vetter, R. SMS: The Short Message Service. Computer 40, 12 (Dec. 2007), 106-110. DOI= http://dx.doi.org/10.1109/MC.2007.440
[7] Mauricio Tia Ni Gong Lin, Thomaz Philippe Cavalcante Silva, Roberto Oliveira dos Santos, Andr´e Ferreira da Silva Neto ,"SMBots - An architecture to manage dynamic services based on SMS ",Nokia Institute of Technology,2009.
[8] B.A. Shirazi, A.R. Hurson and K.M. Kavi, Editors, "Scheduling and Load-Balancing in Parallel and Distributed Systems", IEE CS Press .1995
[9] Chhabra A. Chhabra, G. Singh, Qualitative Parametric Comparison of Load Balancing Algorithms in Distributed Computing Environment,14th International Conference on Advanced Computing and Communication, July 2006 IEEE, pp 58 – 61.
[10] Hendra Rahmawan, Yudi Satria Gondokaryono , The Simulation of Static Load Balancing Algorithms ,Electrical Engineering .2009
[11] Cortes A. Cortes, A. Ripoll, M. Senar, and E. Luque, "Performance Comparison of Dynamic Load-Balancing Strategies for Distributed Computing," Proc. 32nd Hawaii Conf. System Sciences, vol. 8,p. 8041, 1999.
[12] Raqabani A. Al-Raqabani, H. Barada, & R. Benlamri, Performance of probing and coordinated load sharing, *Proc. 17th IASTED Int. Conf. on Parallel and Distributed Computing and Systems*, Phoenix, Arizona, USA, 2005, 66--71.
[13] Alex King Cheung,Hans-Arno Jacobsen,"Dynamic Load balancing in Distributed Content-Based Publish/Subscribe",IFIP,2006.
[14] Cardellini CARDELLINI, V., CASALICCHIO, E., COLAJANNI, M., AND YU, P. 2002. The state of the art in locally distributed web-server systems. *ACM Comput. Surv. 34*, 2 (June), 263–311.
[15] CARDELLINI, V., COLAJANNI, M., AND YU, P. 1999. Dynamic load balancing on web-server systems.*IEEE Internet Comput. 3*, 3 (May), 28–39.
[16] C. Yan, M. Zhu, and Y. Shi, "A Response Time based Load Balancing Algorithm for Service Composition," in *Pervasive Computing and Applications, 2008. ICPCA 2008. Third International Conference on*, 2008, vol. 1, pp. 13 –16.

[17] Aimrudee Jongtaveesataporn ,Shingo Takada. “Enhancing enterprise service bus capability for load balancing”. Journal WSEAS Transactions on Computers archive Volume 9 Issue 3, March 2010

[18] J. Balasubramanian, D. C. Schmidt, L. W.Dowdy, O. Othman, “Evaluating the Performance of Middleware Load Balancing Strategies”, Proc. of 8th Intl.. Conf. on Enterprise Distributed Object Computing,2004, pp. 135-146.

[19] Karasum E. Karasam, E. Ayanoglu, Effects of wavelength routing and selection algorithms on wavelength conversion gain in WDM optical networks, IEEE/ACM Transactions on Networking 6 (2) (1998) 186±196

[20] Shan G. Shen, S. K. Bose, T. H. Cheng, C. Lu, and T. K. Chai, “Efficient wavelength assignments for light paths in WDM optical networks with/without wavelength conversion,” Photon. Netw.Commun. 2, 349–360 (2000).

[21] Ossama Othman, Jaiganesh Balasubramanian, and Douglas C.Schmidt.” The Design of an Adaptive Middleware Load Balancing and Monitoring Service”. In LNCS/LNAI: Proceedings of the Third International Workshop on Self-Adaptive Software, Heidelberg, June 2003. Springer-Verlag

[22] Grosu D. Grosu, A. T. Chronopoulos, and M. Y. Leung, “Load balancing in distributed systems: An approach using cooperative games,” in *Proc.IPDPS*, 2002, pp. 52–61.

[23] Goyal Sandip Kumar Goyal, Dr. R.B. Patel. Adaptive and Dynamic Load Balancing Methodologies for Distributed Environment, International Journal of Engineering Science and Technology (IJEST), 3(3), 1835 - 1840.2011

[24] Qin Xiao Qin, Hong Jiang, Yifeng Zhu, and David R. Swanson, “Dynamic Load Balancing for I/O-Intensive Tasks on Heterogeneous Clusters,” in the Proceedings of the 10th International Conference on High Performance Computing (HiPC 2003), December 17-20, 2003, Hyderabad, India.

[25] J. Wang, J. Chen, Y. Wang, and D. Zheng, “Intelligent Load Balancing Strategies for Complex Distributed Simulation Applications,” 2009, pp. 182–186.

[26] N. Arapé, J. A. Colmenares, and N. V. Queipo, “On the Development of an Enhanced Least Loaded Strategy for the CORBA Load Balancing and Monitoring Service,” pp. 205–211, 2003.

[27] F. J. L. Rosas and J. C. M. Romo, “Improving Dynamic Load Balancing Under CORBA with a Genetic Strategy in a Neural System of Off-line Signature Verification,” presented at the PDPTA, 2007, pp. 510–516.

[28] T. Kunz, “The influence of different workload descriptions on a heuristic load balancing scheme,” *Software Engineering, IEEE Transactions on*, vol. 17, no. 7, pp. 725 –730, Jul. 1991.

[29] G. Mühl, L. Fiege, and P. Pietzuch, *Distributed event-based systems*. Springer-Verlag, 2006.

[30] http://support.hyperic.com/display/SIGAR/Home