Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 775–780
DOI: 10.15439/2014F454
ACSIS, Vol. 2

Implementation of a Network Based Cloud Load Balancer

Sasko Ristov, Marjan Gusev, Kiril Cvetkov
University Ss Cyril and Methodius, FCSE,
Rugjer Boshkovic 16,
Skopje, Macedonia
Email:{sashko.ristov, marjan.gushev}@finki.ukim.mk, kiril_cvetkov@yahoo.com

Goran Velkoski
Innovation LLC,
Vostanichka 118,
Skopje, Macedonia
goran.velkoski@innovation.com.mk

***Abstract*—Cloud service providers offer their customers to rent or release hardware resources (CPU, RAM, HDD), which are isolated in virtual machine instances, on demand. Increased load on customer applications or web services require more resources than a physical server can supply, which enforces the cloud provider to implement some load balancing technique in order to scatter the load among several virtual or physical servers. Many load balancers exist, both centralized and distributed, with various techniques. In this paper we present a new solution for a low level load balancer (L3B), working on a network level of OSI model. When a network packet arrives, its header is altered in order to forward to some end-point server. After the server replies, the packet's header is also changed using the previously stored mapping and forwarded to the client. Unfortunately, the results of the experiments showed that this implementation did not provide the expected results, i.e., to achieve linear speedup when more server nodes are added.**

***Index Terms*—Distributed Computing; HPC; Performance; Web Services.**

I. Introduction

THE dynamic way of living enforces the companies to be more adaptive to changes. A lot of new companies are emerging and growing fast, thus taking the market share ruled of the leading companies. Therefore, services in each company must be prepared for dynamic changes and cope with increasing or decreasing the load [1]. If a company wants to save costs and buy a smaller amount of resources, then those resources cannot handle the load peaks. Nevertheless, buying huge amount of resources will increase the costs and the resources will be underutilized most of the time [2].

A possible solution is to migrate the services in commercial clouds. This process requires a dynamic strategy for a given company, to enable efficient acquiring or releasing the cloud resources. Additionally, increasing the resources requires a sophisticated load balancing strategy to maximize the effective and efficient usage and utilization of the rented resources. The final effect is to maximize the cost, performance and utilization ratio.

Recently, we have proposed an architecture for a Low Level Load Balancer (L3B) [3]. This architecture provides a scalable cloud environment, which can distribute server load among several active virtual machines that are integrated over the communication link [4].

In this paper, we present a new load balancer, realized on a network level. It dynamically balances the incoming network packets among all active virtual machines and returns the responses to the clients that send the requests. This balancer adds a small latency, which does not impact the total response time. The balancer also increases the security of the services since the client cannot see the internal cloud network.

The rest of the paper is organized as follows. Section II presents the related work in the area of load balancing. In Section III, we briefly describe the architecture of L3B. The development and the performance of the L3B balancer implementing the proposed architecture is presented in Section IV. Finally, Section V concludes the work and presents future work.

II. Related Work

Load balancing is an inherited feature from grids onto clouds. It may have various uses in cloud computing, such as:

- *Failover*, as continuation of a service after the failure of a cloud resource;
- *Energy conservation and resource consumption* kept to a minimum; and
- *Scalability*, as a feature of cloud computing requirements.

Rimal et al. [5] give an overview of load balancing techniques used by various cloud providers and solutions, including Amazon, Google, SalesForce, Azure, Eucalyptus, OpenNebula, etc. Most of these techniques are implementations of a conventional Round Robin schema, weighted selection mechanisms, HAProxy, Sticky session, SSL Least Connect, Source address, cluster server load equalization, high performance protocols over EC2, hardware load balancing, cloud controllers, etc.

A lot of load balancing techniques have existed long time before the introduction the cloud computing paradigm and the virtualization technique, which can be grouped in three groups [6]:

- Session switching at the application layer;
- Packet-switching mode at the network layer; or
- Processor load balancing mode.

978-83-60810-58-3/$25.00

Heinzl and Metz [7] classified the load balancers in two groups: Hardware and software, and commercial and open source.

In this paper we propose a network based load balancer, realized by packet-switching mode at the network layer of OSI model.

However, load balancing is not the only sufficient essential part for realization of the cloud. Resource brokering is also important, which is known as elastic load balancer (ELB), such as Amazon's ELB. Hu-Sheng et al. [8] proposed TeraScaler ELB, which is able to dynamically adjust the processing capacity of back-end server cluster with the applied load.

In our previous research, we proposed an architecture of a L3B balancer [3]. We have developed the load balancer and in this paper we present the architecture and design. A set of experiments is also conducted in order to prove the performance of our solution.

Load balancing in the cloud was analyzed by several authors, with corresponding surveys about performance of various load balancing algorithms [9], [10].

Load balancing techniques can be either centralized on a specific server or distributed by content aware policy. The centralized load balancing techniques are controlled by a single central node and all the other nodes communicate with this node, such as the Central Load Balancing Policy for Virtual Machines (CLBVM) solution proposed by Bhadani and Chaudhary [11], or the CLBDM solution proposed by Radojevic and Zagar [12], which takes into consideration other parameters as server load and application performance on top of the Round Robin Algorithm.

Centralized load balancing approaches in cloud computing do not offer full scalability features due to design limitations and communication overhead [13].

III. Proposed Architecture

This section describes the architectural design of the new proposed L3B balancer.

A. Overall model

The overall system architecture is presented in Figure 1, presenting the position of all items and their external interconnections. L3B is placed in front of a pool of virtual machine instances on the cloud, communicating with the Internet clients from one side and with physical or virtualized servers from the other side. The main function is to realize the load balancing of the clients' requests for services on various servers.

The latency that the L3B adds to the client requests should be compensated with the reduced response time by the servers due to reduced number of requests.

The internal architecture of the L3B balancer [3] consists of two modules: *Resource Management Module (RMM)* and *Packet Management Module (PMM)*, as presented in Figure 2 [3]. The purpose of these modules is to realize accounting functions, which is essential for load balancing.

The main objective of the RMM module is to manage the provision of cloud resources with dynamic accounting of the current load and utilization of active virtual machine instances. Whenever the accounting shows a need for more resources, the RMM module communicates with the cloud controller, by sending a command to initiate creation of a new virtual machine instance. If there is an information of underutilization, a corresponding command is sent by the RMM module to shut an existing virtual machine instance.

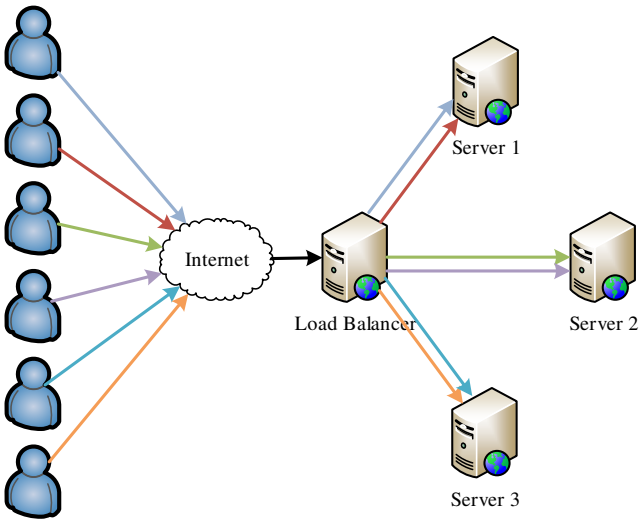

Fig. 1. L3B System architecture

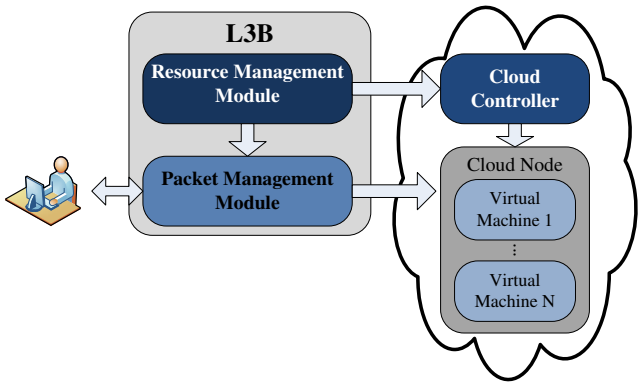

Fig. 2. Modules in the internal L3B architecture

Besides the connection to the cloud controller, the RMM also communicates with the PMM module. Particularly, the RMM module sends an information to the PMM module about active virtual machine instances to enable quality information about the work balance and to enable conditions for the PMM to realize the load balacing. The PMM module's main task is to redirect the input packets to some of the active virtual machine instances and then to forward the responses to the client that has sent the request.

This paper presents an extension of this idea, with details on the development of the PMM module, since we are interested

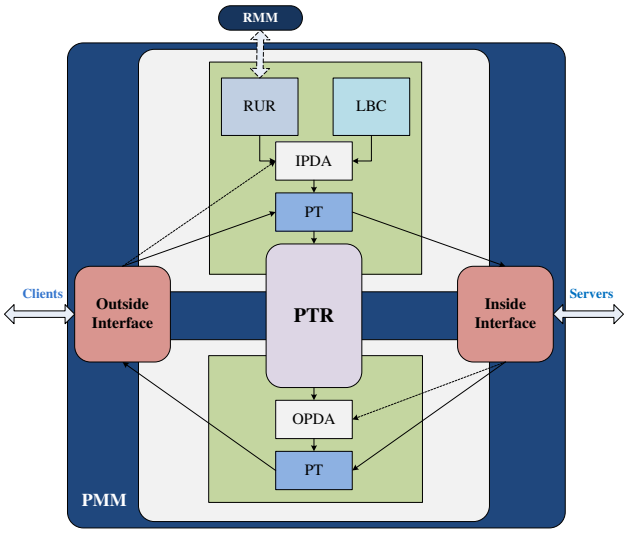

Fig. 3. Design of the PMM module

in presenting a proof that it will be efficient for load balancing of client requests among several active machines (or virtual machine instances in the cloud). The development of the RMM module is out of scope of this paper since it directly depends on the cloud controller and its APIs and interfaces.

B. Design of the PMM module

The PMM module is the core of the L3B balancer. The main functionality is to enable intelligent management of all the inconsistent traffic coming from the clients.

The design of PMM module is presented in Figure 3. It is used to manage the client's requests, i.e. to forward particular incoming packet to a particular virtual machine instance, enabling an environment to balance the load of all virtual machine instances. When the target virtual machine responds back, PMM forwards the response packet to the client that has sent the corresponding request.

The PMM module consists of two interfaces to establish a communication to:

- the *clients*, presented by the client machines that sends requests and
- *virtual machine instances*, presented by the Cloud Nodes.

Figure 3 shows the two internal modules for communication with the external parts:

- *L3B Outside Interface (OI)* that communicates with the clients, and
- *L3B Inside Interface (II)*, responsible to communicate with the virtual machine instances.

The main part of the L3B, realized as a core processing unit is the

- *Packet Translation Repository (PTR)* responsible for receiving the packets in both directions, their processing and forwarding to the destination.

The other internal PMM modules, which are designed to establish communication in direction from outside LAN to inside LAN, (direction from OI to II) and information exchange with the RMM module are the following:

- *Input Packet Decision Agent (IPDA)*, a part that realizes the most important agent, which implements the intelligent algorithm to derive smart decisions in assigning the requests to appropriate resources,
- *Resource Utilization Repository (RUR)* is a part that stores information about utilization of the physical resources, by sensing the active virtual machines and communicating with the RMM module;
- *Load Balancing Configuration (LBC)* is dedicated to store the information about configuration for the load balancing;
- *Packet Translation (PT)* agent is the part that realizes the low level networking on IP level in direction from OI to II.

The PMM module contains also parts responsible for low level network communication in direction from inside LAN to outside LAN (direction from II to OI), as follows:

- *Output Packet Decision Agent (OPDA)* is the agent responsible to make decisions and assign responses to the requestors, realized by appropriate communication with PT and PTR.
- Inverted *Packet Translation (PT)* agent realizes the low level networking on IP level in direction from II to OI.

The OI arbitrates among the clients and the PMM inner agents. Its function is based on a decision making and triggering some actions.

A typical scenario, when a new packet arrives, the OI instantly sends information to the IPDA. The IPDA agent uses a sophisticated intelligent algorithm to realize smart decisions needed to assign the requests to appropriate resources. The decision can be made only by using relevant information by the RUR and LBC about current utilization of the physical resources and load balancing configuration. Finally, after IPDA has analyzed the real time information of the hardware utilization, it determines which virtual machine can handle the request in order to preserve sustainable performance. The realized decision is sent to the PT agent, which receives the packet from OI and uses it to proceed with the IP header translation using the NAT/PAT (Network Address Translation / Port Address Translation). This functionality enables an environment to translate internal and external network addresses for each packet. These translations are then stored in the PTR and finally the corresponding transformed packet is forwarded to the II part, as inside interface to forward it to the target virtual machine instance.

As soon as the packet reaches the II part, it is forwarded to the corresponding cloud node and the target virtual machine instance. The packet has modified header to be transmitted in the network. The information stored in the header can be efficiently used for packet flow in the opposite direction. In this case the II receives the packet with responses from the

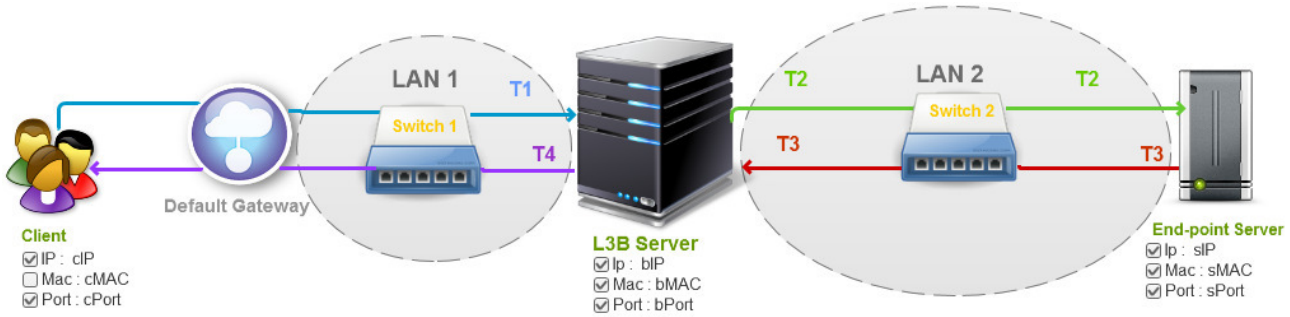

Fig. 4. The L3B implementation

virtual machine. The packet encapsulated in data link frame contains the corresponding MAC addresses of the source and target. Now the packet is sent to OPDA, which communicates to the PTR to receive detailed PT instructions. Note that there is an inverted PT in this section, which repeats the inverse NAT/PAT procedure to translate the external IP address into the original destination IP address. Then the packet is sent to the OI to forward the response to the client.

IV. The Implementation of L3B

This section presents the implementation of L3B according to the architecture described in Section III. The results of the experiments to determine its impact to the end-service's performance are also presented.

A. L3B functional requirements

The efficient and effective load balancer should comply with the following functional requirements:

- To balance all incoming packets that are sent by the clients to the active end-point servers;
- Must have a configuration file where a specific port can be configured where the clients should sent their requests;
- Must have a configuration file to specify the IP addresses of active end-point servers where the incoming packets will be balanced; and
- Introducing L3B between the client and more end-point servers must reduce the overall response time of the client requests.

B. The Architecture

The L3B is implemented in JAVA to be platform independent. This feature compensates the small JAVA deficiency, manifested as a reduced performance. Recently, this is not emphasized a lot, since modern Java virtual machines' performance do not fall behind other solutions such as C or C++ [14].

The architecture of the L3B implementation is depicted in Figure 4. Only one end-point server is presented in order to simplify the explanation. The participants in this communication scenario are the client, L3B server and end-point server (virtual machine hosted on the cloud). There are two LANs identified in the figure, LAN1 as a network segment between the default gateway and the L3B server, and LAN2, as a network segment between the L3B server and the end-point server (virtual machine hosted on the cloud). We assume that these LANs are supported by corresponding fast switches. In addition, this organization allows the end-point servers and L3B to be in the same subnet to decrease network latency. Considering the L3B modules, actually OI communicates with default gateway in LAN1 and II communicates with virtual machines hosted in LAN2. The participant' parameters (IP address, Mac address, Port) are presented below each participant, as presented in Figure 4.

TABLE I
Header values of a request in different states in the L3B implementation

Header Value	T_1	T_2	T_3	T_4
Source IP	cIP	bIP	sIP	bIP
Destination IP	bIP	sIP	bIP	cIP
Source port	cPort	obPort	sPort	bPort
Destination Port	bPort	sPort	obPort	cPort
Source MAC	Def. Gateway	bMac	sMac	bMac
Destination MAC	bMac	sMac	bMac	Def. Gateway

Lets discuss a typical scenario in this L3B architecture and implementation. When a client sends a request, it is routed through Internet until the L3B default gateway, identified as T_1 state in LAN1. Now, the incoming packet is received by the OI module in the L3B balancer. This packet is processed, as discussed in Section III.

The packet header processing, transforms its Ethernet, IP and TCP headers. More details for the scenario presented in Figure 4 are specified in Table I. Besides transformation, the old and new headers are stored in corresponding repository (PTR). As explained in the previous Section, IPDA is the agent that makes the decision to select the end-point server as a destination where the packet should be sent. In the figure, this is labeled as T_2 in the LAN2.

When the end-point server processing is finished, a response packet starts to be sent in the opposite direction in the analyzed architecture. The end-point server replies back to the L3B, by sending the packet in the LAN2, labeled with T_3 in the figure. Similarly to the previous explanation, II unit from the L3B balancer receives the packet, and OPDA uses the

information stored in PTR repository to make a decision about the header translation. Then the inverse PT agent transforms the corresponding Ethernet, MAC and IP addresses, according to the decision made by OPDA, and updates the configuration in PTR. After header translation, PT sends the packet with changed headers to the OI unit, which forwards it to the relevant client, which originated the counterpart request.

C. Implementation challenges

Our implementation of the architecture uses jNetPcap library [15] to sniff the network traffic and alter the incoming packets. Using the jNetPcap library made some problems during the packet forwarding.

The packet sniffer Wireshark [16] showed that a regular packet arrived at the server T_2, with correct values for appropriate fields in Ethernet and IP header. However, the server did not respond back to the L3B balancer and T_3 was not initiated.

The solution to this problem, acquired more experiments and programming. We redeveloped the L3B to open a new socket to the end point server for each packet. However, although this change solved the problem and the packet was sent back to the L3B balancer, and then from L3B to the client, the results were not promising. The following two sections present the testing methodology and the results of the experiment to determine the L3B performance.

D. Testing methodology

Four workstations with the same hardware resources and platform are used as a testing environment. Each workstation has Intel(R) Xeon(R) CPU X5647 @ 2.93GHz with 8GB RAM memory, installed with Ubuntu 12.04. The client is on the same LAN as the L3B workstation and two end-point workstations to reduce the network latency.

A client requests a packet with constant web page content, which size is $56KB$. The different number of concurrent requests are then initiated to create various loads. We choose this page size in order to be smaller than the IP packet limit of 64KB. Usually the web requests and responses are smaller than $64KB$.

The concurrent requests are simulated with SOAPUI. Each test case consists of sending N concurrent requests per second, such that N varies in each test case starting from $N = 5$ until $N = 100$ requests, by increasing N with step 5. Each test case lasts 60 seconds.

E. Performance analysis of the L3B implementation

The defined test cases were executed in order to check the impact of the L3B balancer to the end-point web server. Both experiment scenarios are examined, with and without L3B.

Figure 5 presents the results of the experiments. We observe that the last functional requirement is not satisfied for neither test case, meaning that the most important feature is not realized. In reality it shows that introducing the L3B balancer made the response time even worse, compared to the scenario without L3B.

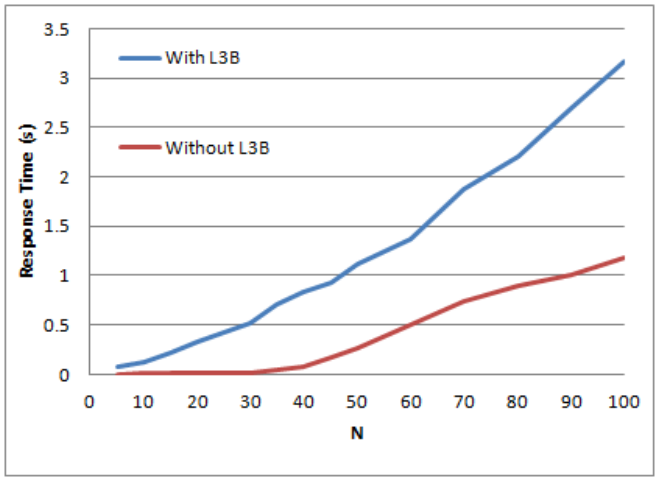

Fig. 5. Results of the experiments for the L3B implementation

The results show that although opening a new socket for each incoming packet was a solution to reply the server, it is performs worse and overall it is a bad solution.

This motivated us to analyze various proposals how to make the L3B more efficient. The main idea was to try to open one or several sockets from the L3B server to the servers in order to reduce the overall response time.

V. Conclusion and Future Work

In this paper we have presented a new approach for a load balancer, that is, the low level load balancer that works on a network layer of OSI model. The balancer is developed according to the L3B architecture [3], but its implementation have even reduced the performance when two end-point servers are used compared to the case when only one end-point server is used without the L3B balancer.

Since this implementation of the L3B balancer did not yield the expected results, we will proceed to improve the L3B architecture and implementation. Creating a new socket for each arrival packet reduced the performance of the L3B balancer and the architecture will be improved by creating a virtual client.

References

[1] A. Murua, I. Gonzalez, and E. Gomez-Martinez, "Cloud-based assistive technology services," in *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, Sept 2011, pp. 985–989.

[2] S. Ristov, M. Gusev, G. Armenski, K. Bozinoski, and G. Velkoski, "Architecture and organization of e-assessment cloud solution," in *Global Engineering Education Conference (EDUCON), 2013 IEEE*, March 2013. doi: 10.1109/EduCon.2013.6530189. ISSN 2165-9559 pp. 736–743, best paper award. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6530189&isnumber=6530074

[3] M. Simjanoska, S. Ristov, G. Velkoski, and M. Gusev, "L3b: Low level load balancer in the cloud," in *EUROCON, 2013 IEEE*, Zagreb, Croatia, 2013. doi: 10.1109/EUROCON.2013.6624994 pp. 250–257. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6624994

[4] L. Schubert, M. Assel, and S. Wesner, "Resource fabrics: The next level of grids and clouds," in *Computer Science and Information Technology (IMCSIT), Proceedings of the 2010 International Multiconference on*, Oct 2010. ISSN 2157-5525 pp. 677–684.

[5] B. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, Aug 2009. doi: 10.1109/NCM.2009.218 pp. 44–51. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5331755&isnumber=5331299

[6] B. Radojevic and M. Zagar, "Analysis of issues with load balancing algorithms in hosted (cloud) environments," in *MIPRO, 2011 Proceedings of the 34th International Convention*, May 2011, pp. 416–420. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5967092&isnumber=5967009

[7] S. Heinzl and C. Metz, "Toward a cloud-ready dynamic load balancer based on the apache web server," in *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2013 IEEE 22nd International Workshop on*, June 2013. doi: 10.1109/WETICE.2013.63. ISSN 1524-4547 pp. 342–345. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6570639&isnumber=6570561

[8] H.-S. Wu, C.-J. Wang, and J.-Y. Xie, "Terascaler elb-an algorithm of prediction-based elastic load balancing resource management in cloud computing," in *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, March 2013. doi: 10.1109/WAINA.2013.79 pp. 649–654. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6550470&isnumber=6550285

[9] K. Nuaimi, N. Mohamed, M. Nuaimi, and J. Al-Jaroodi, "A survey of load balancing in cloud computing: Challenges and algorithms," in *Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on*, Dec 2012. doi: 10.1109/NCCA.2012.29 pp. 137–142. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6472470&isnumber=6472451

[10] N. J. Kansal and I. Chana, "Cloud load balancing techniques: A step towards green computing," *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 1, pp. 238–246, 2012.

[11] A. Bhadani and S. Chaudhary, "Performance evaluation of web servers using central load balancing policy over virtual machines on cloud," in *Proceedings of the Third Annual ACM Bangalore Conference*, ser. COMPUTE '10. ACM, 2010. doi: 10.1145/1754288.1754304. ISBN 978-1-4503-0001-8 pp. 16:1–16:4. [Online]. Available: http://doi.acm.org/10.1145/1754288.1754304

[12] B. Radojevic and M. Zagar, "Analysis of issues with load balancing algorithms in hosted (cloud) environments," in *MIPRO, 2011 Proceedings of the 34th International Convention*, May 2011, pp. 416–420. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5967092&isnumber=5967009

[13] D. Ardagna, S. Casolari, and B. Panicucci, "Flexible distributed capacity allocation and load redirect algorithms for cloud systems," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, July 2011. doi: 10.1109/CLOUD.2011.32. ISSN 2159-6182 pp. 163–170. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6008706&isnumber=6008659

[14] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design, Fourth Edition: The Hardware/Software Interface*. Morgan Kaufmann, 2009. ISBN 978-0-12-374493-7

[15] S. Technologies, "jnetpcap," 2014. [Online]. Available: http://jnetpcap.com/

[16] "Wireshark," 2014. [Online]. Available: http://www.wireshark.org/