METHODS AND ALGORITHMS OF LOAD BALANCING

Igor Ivanisenko

Abstract: *In this paper the classification of the most used load balancing methods in distributed systems (including cloud technology, cluster systems, grid systems) is described. Load balancing is represented on four levels of network model OSI: channel, network, transport, application. Features, advantages and shortcomings are presented for each level. Also strengths and weaknesses of network, transport and application levels are described. Basics of hardware based load balancing in Network Packet Broker and Application Delivery Controllers, that working on OSI layers 2-7, are described. In this work strengths and weaknesses of hardware based load balancing are shown. Basics of software based load balancing are carried out. Differences between software based load balancing and hardware based one are described too. In the work characterizations of the most used dynamic load balancing algorithms in distributed systems is described. Advantages and shortcomings of each algorithm are carried out. Load balancing uses a variety of methods and algorithms for balancing. In the work methods that are used on channel, network, transport, application levels of OSI model and available in balancers and/or can be configured on the servers are presented and analyzed. Employment, effectiveness, strengths and weaknesses of each type of the methods are described in accordance with analysis. Following methods are carried out: direct Routing, Network Address Translation, Source Network Address Translation, Transparent SNAT, SSL Termination or Acceleration, TCP/IP server load balancing, Hashing, Caching, DNS load balancing, Network Load Balancing, Proxy method, Load balancing by using redirection.*

Keywords: *Keywords— load balancing, distributed system, hardware and software load balancing cloud, DNS, network level, Network Address Translation, proxy.*

ACM Classification Keywords: *C.2.0 General – Open Systems Interconnection reference model (OSI), C.2.3 Network Operations - Network management, C.2.4 Distributed Systems - Client/server, Distributed applications*

Introduction

Evolutionary processes occurring in communication networks are inevitably reflected in amount and internal traffic structure. According to numerous studies the total amount of data transferred over the WAN, shows a steady exponential growth despite the fact that this trend will continue in the coming years [Kopparapu, 2008; Erl, 2013].

With increasing amount of data, the behavior of traffic in today's global network shows such a negative feature as the instability of the load, which is characterized by the possibility of the emergence of unpredictable surges of transmission intensity. There are many causes of such instability. A prime example can conduct multiple users on the network caused by the viral nature of the spread of the popular media. Avalanche-like process of attracting new users, new ways of collective communication, based on the widespread use of social services, mass online broadcast - all this makes talking about the new nature of emerging overloads.

The researchers note that today's networks suffer from a lack of bandwidth. According to research about 20-30% of WAN links are routed through congested areas. At the same time there is considerable nonuniformity of load distribution channel resources, indicating procedures are inefficient traffic management in the current environment [Erl, 2015].

The way out of this situation is the use of special methods of balancing traffic to effectively distribute the load in accordance with the existing untapped resources.

The issue of capacity planning should be addressed in the early stages of building a network or project. Initially, the problem of insufficient capacity nodes due to increased loads can be solved by increasing their capacities, or the optimization of the algorithms, software code, and so on [Erl, 2013; Кириченко, 2011]. But sooner or later the moment comes when these measures are insufficient. And then it is necessary to use load balancing methods.

Load balancing is implemented using hardware, software instruments, or a combination of both. Previously, it was clear delineation of hardware and software load balancing. Now, in connection with the development and improvement of both hardware and software load balancers, the boundaries between them are deleted [Roth, 2008; Natario, 2011]. Assume that if hub, switch, Application Delivery Controllers (ADC) are used it is hardware base balancing.

When using the server (computer), we assume that the software load balancing occurs. Hardware load balancing is often used at a channel, network and transport layers. In general, hardware load balancing, faster than software solutions, but its drawback is the cost.

Software based load balancing as opposed to hardware load balancer operates on a standard operating system and standard hardware components such as a PC. Software solutions operate in dedicated hardware load balancing node or directly in the appendix. The hardware load balancing devices are used called Network Packet Broker (NPB) (or Network Monitoring Switch), which have 100GbE interfaces and work on 2 and 3 levels of the network model OSI [Laviol, 2014].

NPB is embedded into a rack network device, receiving and aggregating network traffic from the ports, which it manipulates in the future. The primary and most important function of NPB is load balancing. For load balancing 3-7 levels used ADC [Gurevich 2010]. The procedure of load balancing is carried out using complex algorithms and methods that conform to following levels of the network model OSI: channel; network; transport; applied [Бажин, 2010].

The purpose of this paper is to review the existing basic algorithms and methods for hardware and software based load balancing at different levels of the network model OSI, analyze of their strengths and weaknesses.

In the first section of this paper client based load balancing and server based one are carried out. In the second section the load balancing is described on four levels of network model OSI, their features, advantages and shortcomings are analyzed. In the third section the basics of hardware load balancing and its differences from the software load balancing are described. In the fourth section of this work the survey of few existing load balancing algorithms are carried out. The basic methods of hardware and software load balancing, which are used at different levels of model OSI, are described in the fifth section, their advantages and shortcomings and range of application are described too.

1. Client and server based load balancing

Schematically load balancing in distributed systems can be represented as a structure (Figure 1).

Briefly describe the main components of the scheme. Client based load balancing usually is much worse than server based load balancing [Cardellini, 1999; Koteswaramma, 2012; Pavan Kumar, 2012]. The reason is that clients often can not monitor server availability or load rate. If the server is overloaded or offline, the client waits for a timeout before he tries to connect to another server. The dissemination of information to the client about servers load creates an additional network load and the delay of dissemination information is added to the total time of service. Availability of servers also is very dynamic, so the client can not use this information for a long time period. Also, the balancer can amortize the cost of querying server availability over many requests.

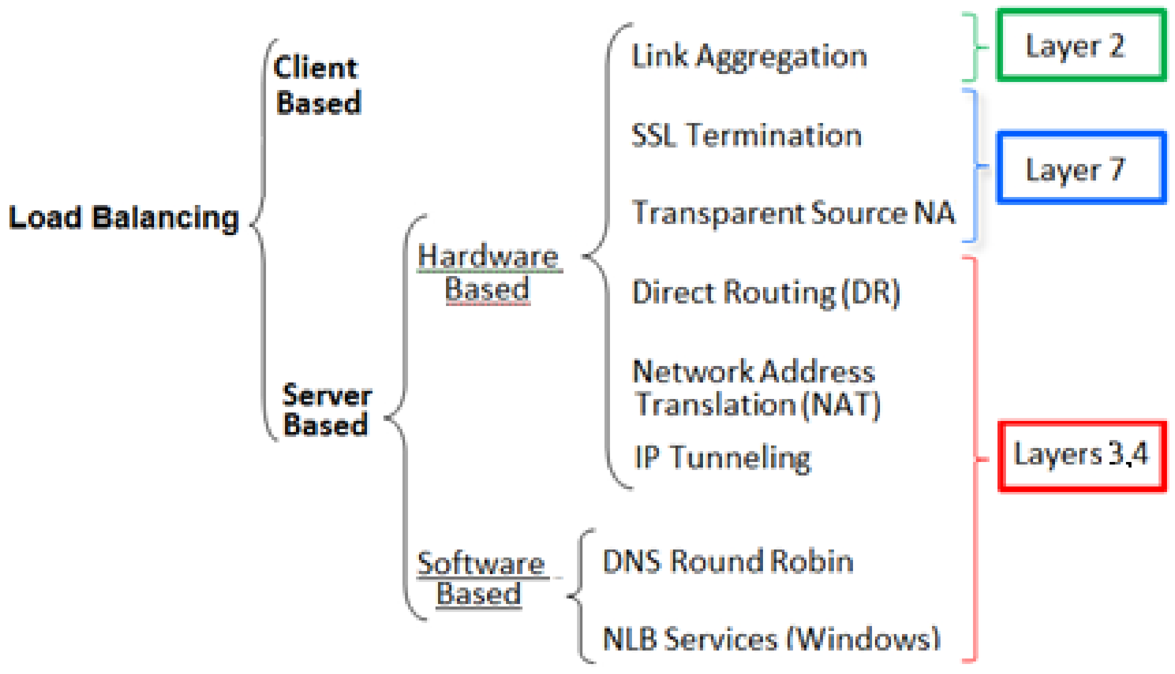


Figure 1. The methods of load balancing in OSI model

Server based load balancing, therefore, incurs no latency penalty to the client. Client based load balancing (Figure 2) typically uses more bandwidth than a server based load balancing. This is because the network path for each route client-server could potentially take different routes.

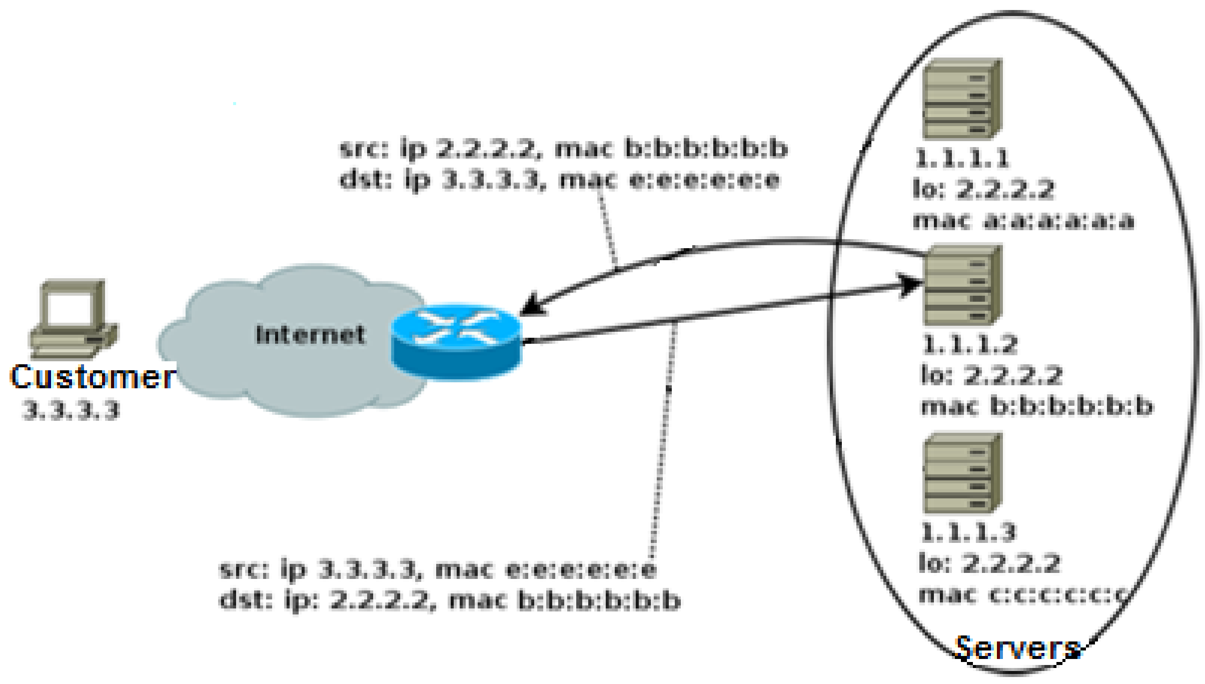

src: ip 2.2.2.2, mac b:b:b:b:b:b
dst: ip 3.3.3.3, mac e:e:e:e:e:e
src: ip 3.3.3.3, mac e:e:e:e:e:e
dst: ip: 2.2.2.2, mac b:b:b:b:b:b
Internet
Customer
3.3.3.3
1.1.1.1
lo: 2.2.2.2
mac a:a:a:a:a:a
1.1.1.2
lo: 2.2.2.2
mac b:b:b:b:b:b
1.1.1.3
lo: 2.2.2.2
mac c:c:c:c:c:c
Servers

Figure 2. Client based load balancing

In this example, the user directly connects to a server address 1.1.1.2 without any balancing. If the server does not respond, then the user can not get access to the necessary resources. Also, if a very large number of users access the server, many of them will be answered with a big time delay, and some have not it. Another variant of client based load balancing: a list of application servers can be inserted into client code. I.e. on the client side there is a list of available servers to which the client tries to communicate until he finds one that responds.

When server based load balancing is possible to do that multiple servers are reflected as a single server - single virtual service - transparent distribution of user requests between servers. The distribution of load to the server prevents the disabling of hardware and software services to end users, and can also provide disaster recovery services by redirecting service requests to a backup copy when the main resource is disable [Cardellini, 2001; Roth, 2008; Jiao, 2010; Tuncer, 2011; Zhihao, 2013]. There are two categories of implementation of server based load balancing:

• **software based load balancing** consists of a special software installed on the servers in the load balancing cluster. The software sends and receives requests from the client to the server based on various algorithms. For example, Microsoft Network Load Balancing is a software load balancing for Web farms, and Microsoft Component Load Balancing is a software load balancing for applications in the farm.

• **hardware based load balancing** consists of a special switch or router with software to provide load balancing functionality. This solution combines switching and load balancing into a single device, resulting in reduce the amount of additional equipment necessary for the realization of load balancing. Modern equipment load balancing devices are known as NPB.

2. Load balancing on levels of OSI model

2.1. Load balancing on the second (channel) level

Load balancing on the second level of the protocol stack there are two options: balancing using a separate dedicated balancer and without it (Figures 3, 4) [Бажин, 2010].

In both cases, some IP-address of the service is set for all servers or to other specialized interface. This is done to ensure that these servers can accept connections on this IP-address and respond from it, but do not respond to ARP-requests (Address Resolution Protocol - the protocol definition addresses) belonging to this address.

This balance is performed as follows: on the balancer which has IP-address and responds to ARP, first packet connection comes. Balancer determines that packet was first. This packet is sent to the server using needed algorithm, changing the MAC-address to destination, it is written in connection table. If this is not the first packet, it looks at which server processes this connection using connection table, and the packet is sent there. Considering that the headers of the third and higher levels are not modified, the response from the server can be sent past the balancer directly over the Internet to the client (to the necessary gateway).

The most common currently solution from software implementations of this method is called the Linux Virtual Server. In the URL-terminology this load balancing method is called direct routing.

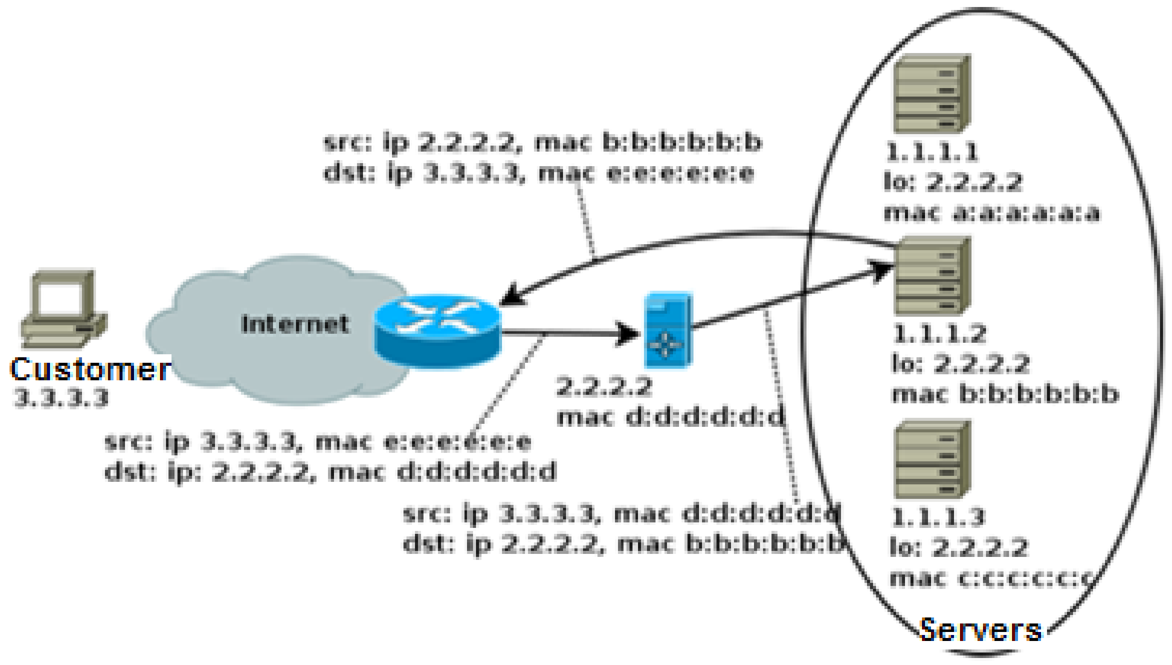

src: ip 2.2.2.2, mac b:b:b:b:b:b
dst: ip 3.3.3.3, mac e:e:e:e:e:e
1.1.1.1
lo: 2.2.2.2
mac a:a:a:a:a:a
Internet
Customer
3.3.3.3
1.1.1.2
lo: 2.2.2.2
mac b:b:b:b:b:b
2.2.2.2
mac d:d:d:d:d:d
src: ip 3.3.3.3, mac e:e:e:e:e:e
dst: ip: 2.2.2.2, mac d:d:d:d:d:d
src: ip 3.3.3.3, mac d:d:d:d:d:d
dst: ip 2.2.2.2, mac b:b:b:b:b:b
1.1.1.3
lo: 2.2.2.2
mac c:c:c:c:c:c
Servers

Figure 3. Load balancing on the channel level

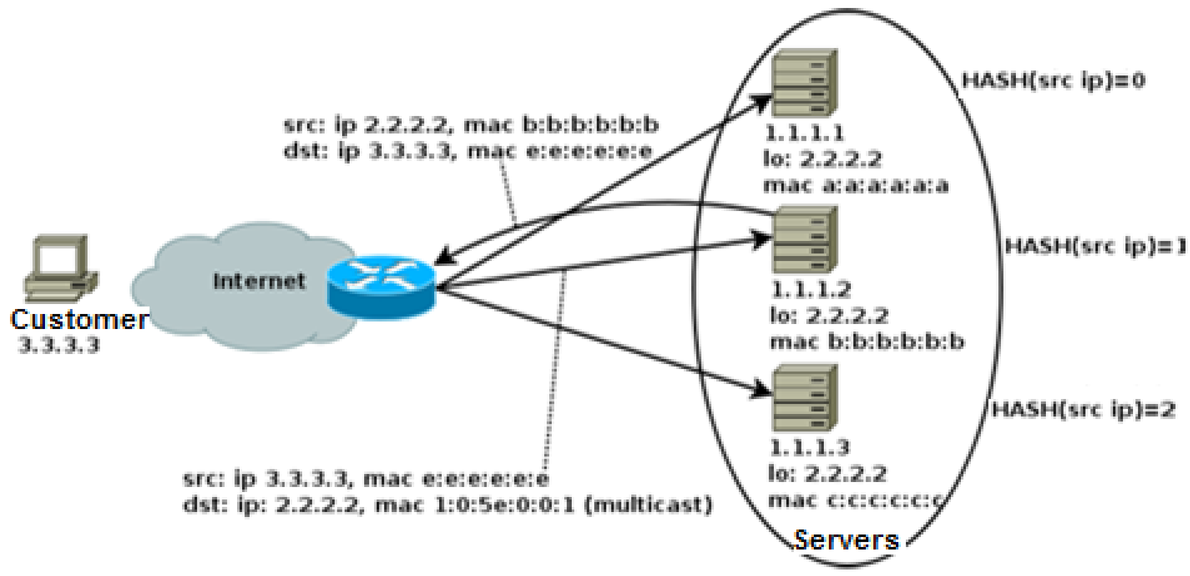


Figure 4. Load balancing on the channel level without a dedicated load balancer

Load balancing without a dedicated load balancer. In this case, the IP-address of the service is prescribed as a static ARP-record at the gateway to some of the multicast MAC-address. The switches are configured in a way that frames that coming in the MAC-address, delivered with all the necessary servers. The hash is calculated on some servers, for example the clients IP-address. According to the value server determines whether it should respond to these requests. If HASH = 0, the first server must answer. It responds. Other servers "know" that they do not have to answer.

Advantages of load balancing on the channel level.

• Independence from the high level protocol. Relatively low resource consumption. It is possible to balance the HTTP, FTP or SMTP - the difference will not be. There is a method of balancing without a dedicated load balancer. With low number of servers it can be actual. It is possible to send answers past balancer. Taking into account that, for example, in the HTTP protocol the size of response is typically larger than the size of the request, then the increase of resource economy is take place.

Shortcomings of load balancing on the channel level.

• All servers must be in the same network segment. Specific configuration of servers and network equipment are necessary. Therefore, this method is not always convenient and applied. As the implementation of this scheme the cluster IP on the firewall IP-tables for Linux can be used.

2.2. Load balancing on the third (network) level

Load balancing on the network layer (Figure 5) surmise the following tasks: to do so for one particular server IP-address corresponding to different physical machines [Hong, 2006; Roth, 2008; Бажин, 2010]. Balancer is assigned the same IP-address service. When a request comes to it, destination NAT is applied, i.e. a destination IP-address is substituted in the packet: IP-address of current server is changed to necessary IP-address of the server that will handle the request by selected algorithm. The difference between this method and the previous one is that headers of the third level are modified. Sender IP-address should be changed from the server IP-address that handles the request to the service IP-address, which is used in the balancer.

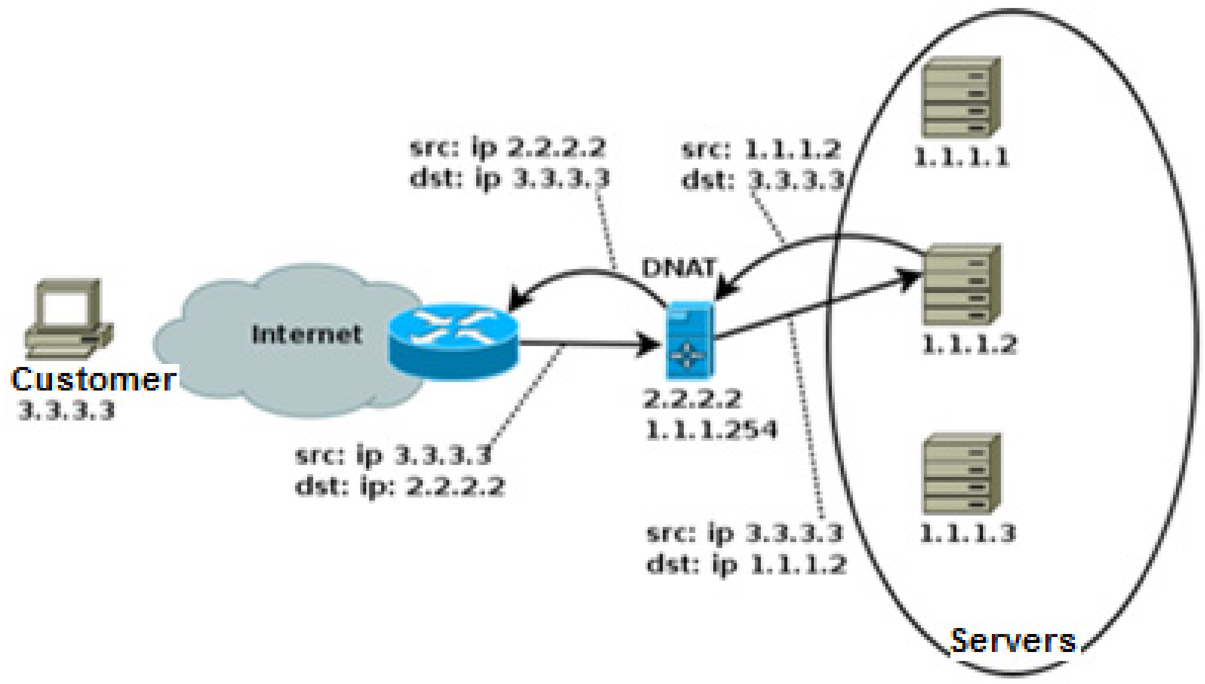


Figure 5. Load balancing on the network level

There are many realizations of this method.

- DNS-balancing. For the one domain name is allocated several IP-addresses. The server, on which a client request will be sent, is typically determined by a load balancing algorithm, e.g. Round Robin.

- Construction of NLB-cluster. By using this method, the servers combined into clusters, which consist of input and computing nodes. Load balancing is performed by using a special algorithm. It is used in Microsoft solutions.

- Load balancing IP network using additional router.

- Load balancing on a territorial basis is performed by placing the same services with the same address in geographically different regions of the Internet.

Advantages of load balancing on the network level.

• Independence from the high level protocol. Complete transparency for servers.

Shortcomings of load balancing on the network level:

• Reverse server traffic should pass through the balancer. Accordingly, the load on it will be higher than when using the balancer on the second level.

2.3. Load balancing on the fourth (transport) level

This type of balancing is the easiest: the client address to the balancer, that forwards the request to one of the servers, which will process it (Figure 6) [Hong, 2006; Roth, 2008; Бажин, 2010]. The choice of the server that will process the request can be carried out in accordance with the different algorithms: sorting by simple circular, by selecting the least loaded server from a pool, etc.

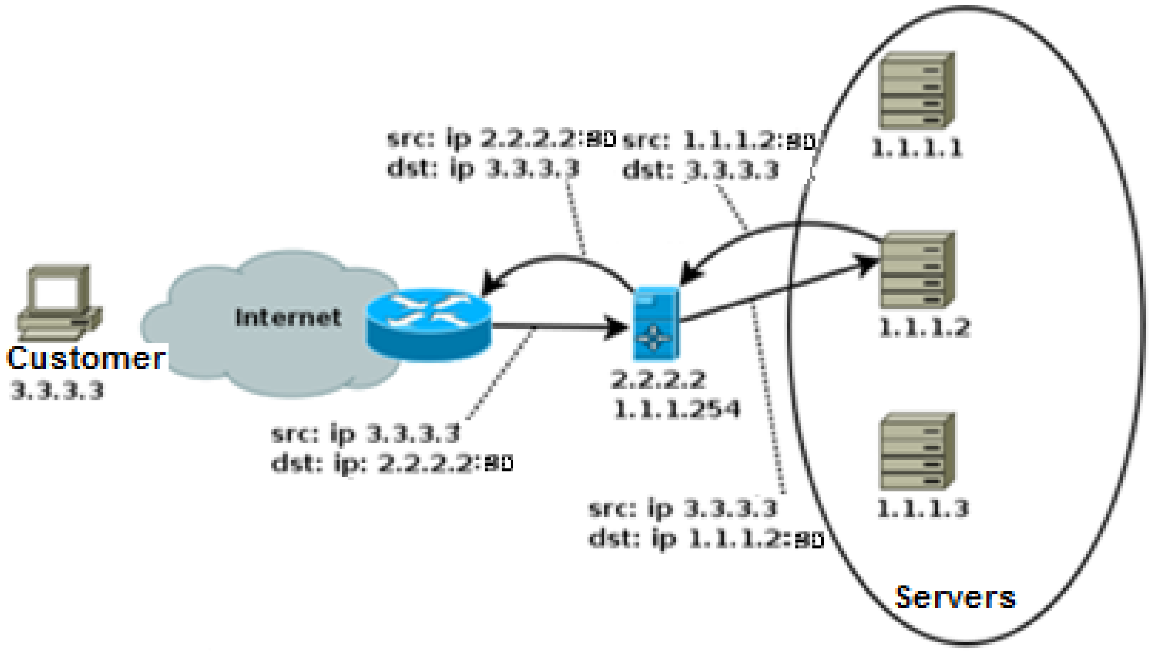


Figure 6. Load balancing on the transport level

Sometimes the balance on the transport layer is difficult to distinguish from the balance on the network level. When balancing of outgoing traffic at the network level takes place, specific port or specific communication protocol are indicated. The difference between the levels of balance can be explained as follows. Solutions of the network layer are not terminates the user session on themselves. They simply direct traffic and do not work in proxy mode. On the network level load balancer decides on which server to transmit packets. A server provides a client session. On the transport layer a communication with the client becomes isolated on balancer, which acts as a proxy. It communicates with the server on its own behalf by passing client information in additional data and headers. The popular software balancer HAProxy works in this way.

2.4. Load balancing on the seventh (application) level

When balancing on the application level load balancer works as "smart proxy" [Vlaeminck, 2004; Hong, 2006; Roth, 2008; Mendonca, 2014]. It analyzes client requests and forwards it to different servers depending on content of requests. Web server Nginx works by distributing requests between frontend and backend. In contrast to the low-level load balancing solutions, load balancing in the seventh level works with knowledge of application. One of the popular architectures load balancing is shown in Figure 7, it includes load-balancing on the application and transport layers.

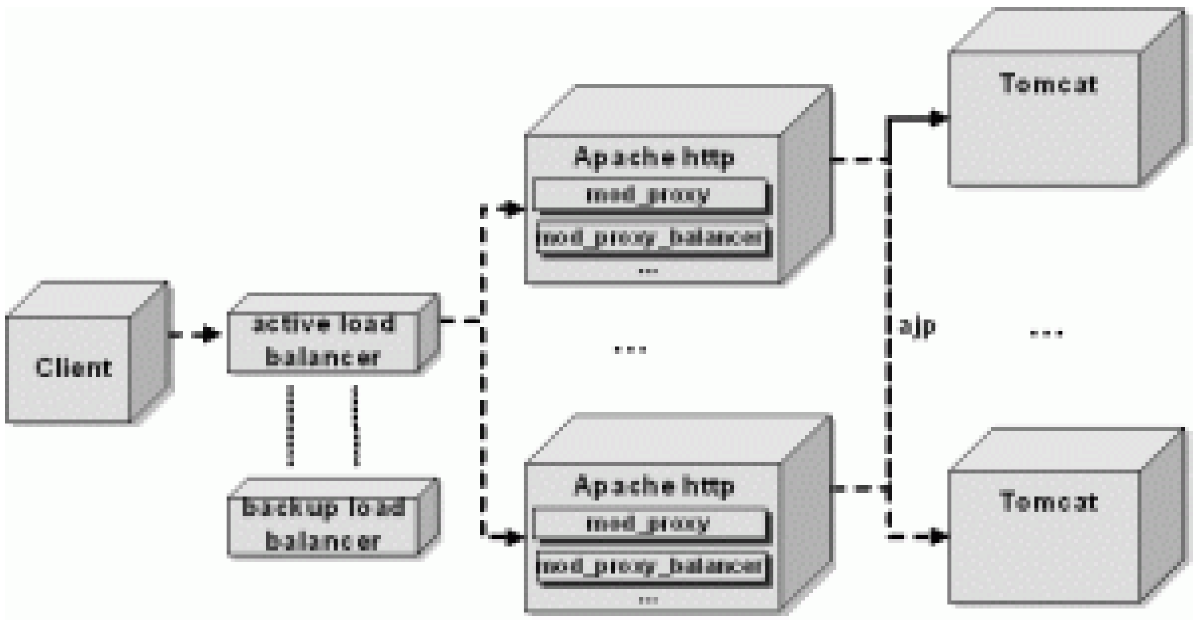


Figure 7. Load balancing on application and transport levels

Load balancing on application level serve as a normal server for load balancing on transport layer. Incoming TCP connection are directed to balancer at the application level. When it receives a request on the application level, it determines the destination server based on application layer data and forwards the request to that server.

In this example (figure 8) user visits a high-loaded site. During the session, the user can request a static content (images or videos), dynamic content (news channel, transactional information - the order status). Load balancing on 7-th level allows to route requests using information of the requested content. So now the request of image or video can be routed to the servers, which store it, and it is possible to optimize the service for multimedia content.

Request for transaction information, such as payment, can be directed to the application server, responsible for managing pricing.

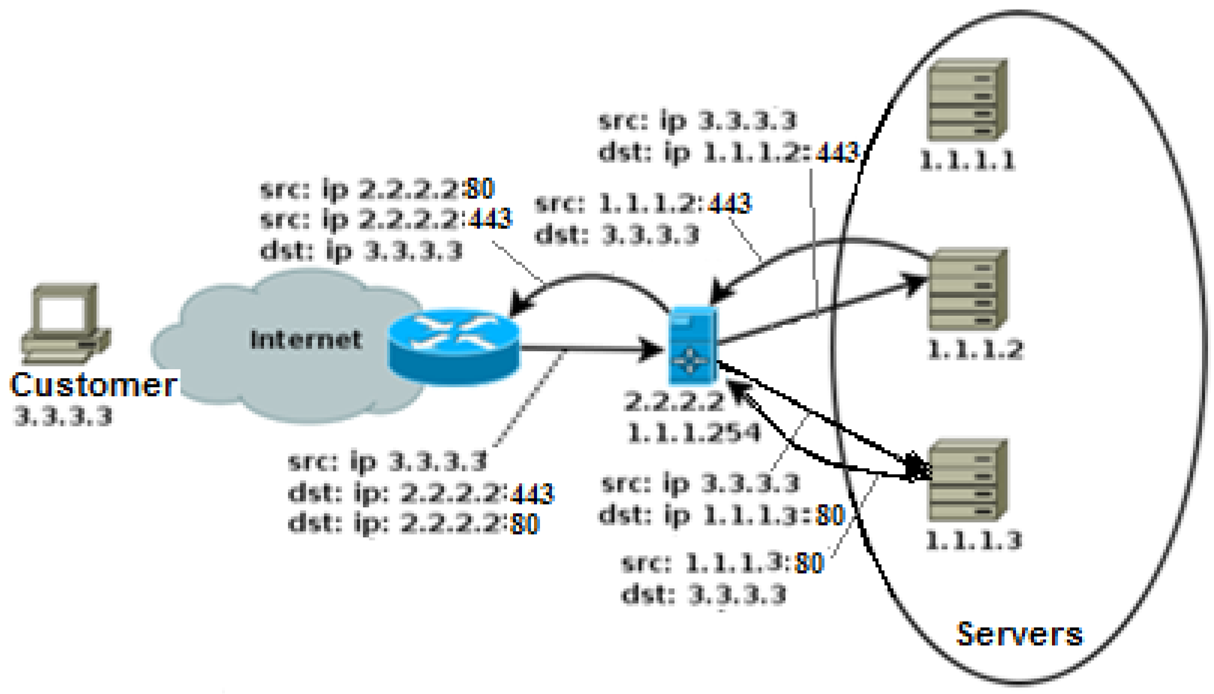


Figure 8. Load balancing on application level

Load Balancing Layer 7 also allows to increase the efficiency of the application infrastructure, because different types of content have different requirements in terms of CPU usage, bandwidth, etc. Thus it is possible to obtain higher efficiency servers classify them in such a way that some of them are treated with a transaction, while others simply act as a massive storage systems for serving static pages or optimized for downloading streaming video, for example.

Application delivery controllers perform load balancing on 7-th layer called (ADC), and they combine the features of traditional load balancing with advanced application switching of 7 level to provide design a scalable, optimized application delivery network.

Advantages of load balancing on the application level:

• rarely reduce performance in the modern server; make more informed decisions of load balancing; it is possible to apply the optimization and content changes (such as compression and encryption); buffering is used to unload the slow connections from superior server, that improves performance.

Shortcomings of load balancing on the application level:

• overhead on the analysis of requests is high; limited scalability as compared with load balancing on the other levels.

2.5. Differences between 4-th and 7-th levels load balancing

Load balancing level 4 deals with the message delivery, without regard to the content of the message. Transmission Control Protocol (TCP) is a protocol of level 4 for hypertext transfer protocol (HTTP) traffic on the Internet. The transport layer load balancing only transmits network packets to and from the superior server without checking packages contents. At this level, it can be making limited decisions to route by checking the first few packets in a stream of TCP.

Load balancing layer 7 works at the application level, which has to deal with the actual content of each message. HTTP is the dominant protocol level 7. Load balancing on the application-level directs network traffic using much more complex mechanisms than the choice of the path on the transport level load balancing.

Load Balancing layer 7 pauses network traffic for reading messages. Thus, the load balancing decision made using message content (URL or cookie, for example). Then new TCP connection is created to selected upstream server (or to recurring, via HTTP-support activity) and request is sent to server.

3. HARDWARE BASED LOAD BALANCING

Load balancing hardware devices working on OSI layers 2-7 and used for splitting the network load among multiple servers in terms of factors such as utilization of processor CPU, number of connections, total server performance [Natario, 2011; Laviol, 2014].

Use of this type of technology minimizes the probability that any particular server will be overloaded, and optimizes the throughput for each computer or terminal. In addition, by using balancer it can be minimized network downtime, to ease traffic prioritization, to monitor applications from end to end, to provide user authentication, and help defend against harmful activity such as denial of service (DoS) attacks.

Router LVS uses low-level filtering, that has advantages compared to redirect requests on the application level, because of the load balancing on the transport level does not cause significant computational costs and can be scaled [Red Hat, 2015].

The basic principle is that the network traffic is routed to a common IP called virtual IP (VIP), or listening IP, and this address is assigned to the balancer. After the load balancer receives a request on this VIP, and it will need to decide where to send request, and that decision is usually controlled by the load balancing algorithm and set of rules. Then the request is sent to corresponding server and server will generate a response which depending on the type of load balancing. Response will be sent back to either the balancer in the case of the device layer 7 or, as a rule, from the device level 4 directly back to the end user (typically using default gateway). In the case of proxy based load balancer, a request from a Web server can be returned to the balancer and processed before being sent back to the user. This processing can include replacement of content or compression or other scenarios. A more detailed look at the processes taking place inside the NPB at the level of software solutions and basic methods and algorithms that used for load balancing.

Load balancing in NPB is division process input stream from one or more interfaces to multiple output interfaces by certain rules or criteria. The following functions are used with a balancing:

- filtering - rules to identify flows for their subsequent balancing and reducing the amount of data in these flows;
- aggregation - merge of flows from multiple input interfaces into a united before the operation of balancing;
- connections - most NPB can work as a switch.

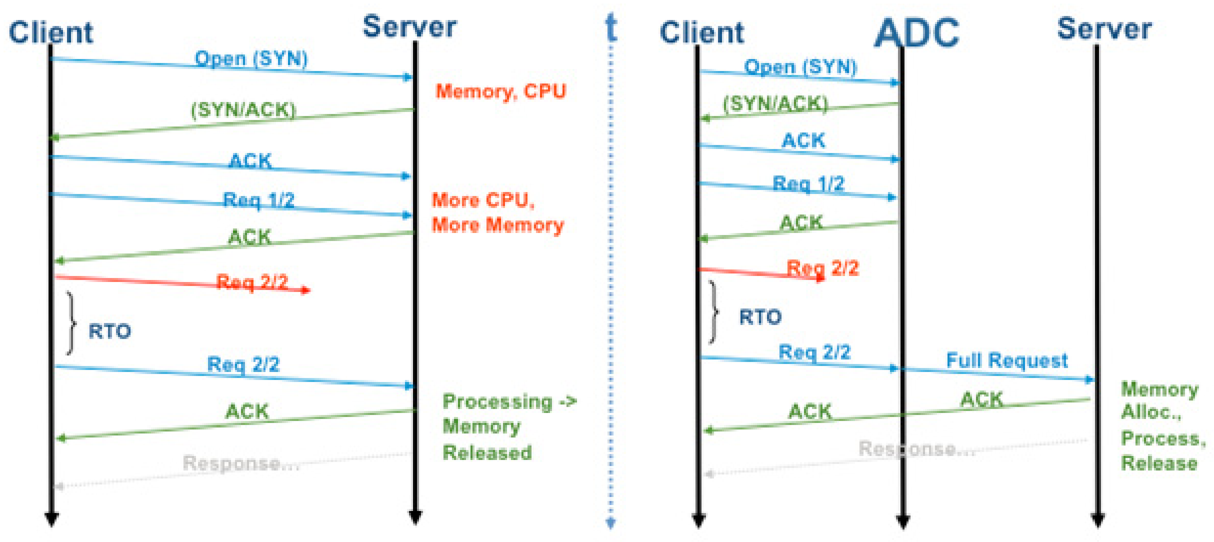
The main application of NPB is a selection of large necessary data flows and their division into smaller ones. This problem occurs quite often. Traffic coming from the several ports is aggregated and comes to the NPB. Accordingly the first condition, one of the main require is made if client equipment works with a flow on the session level, then NPB does not have these sessions to break, then packages of one session should always come to the same output interface. This property is called Flow Coherence. As discussed in [Laviol, 2014] in the NPB traffic are filtered, which allows to filter incoming packets using the defined rules by various network protocols, cut and analyze part of the package, insert ports labels, VLAN, MPLS into packets, mark and delete duplicate packets.

Usually flows coming in NPB are determined by header (src/dst IP, src/dst port, protocol). But the structure of flows may change, and NPB must be able to adapt to them. For example, if the IP-address is fixed, and the ports vary depending from the path of a package or vice versa, the ports are fixed and IP-addresses can vary.

The NPB has functions that allow to monitor the status of channels, to reserve and manage channels. NPB also uses different algorithms for traffic management, which will be discussed later in this paper.

3.1 Differences software and hardware load balancing

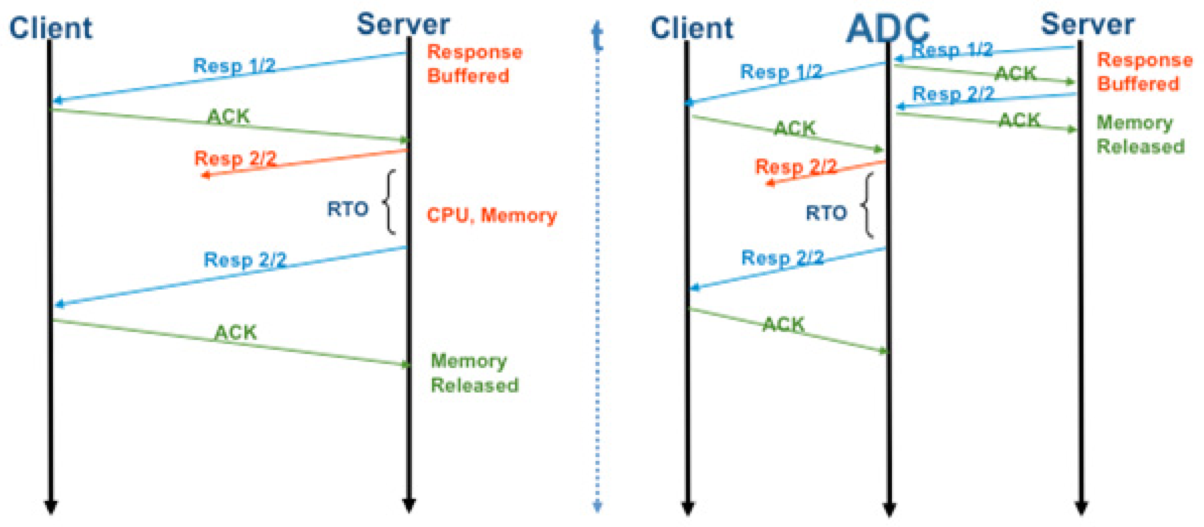
The preliminary program approval communication conditions (handshake) is made separately, multiplexes - separately. The server is not required to deal with such problems. It does not deal with inquiries, transactions, and sends HTTP-page. As a result, processor, bus and memory are unloaded (Figure 9) [Гуревич, 2010].



a) b)

Figure 9. Handshakes for software and hardware load balancing

Consider example of buffering requests (Figure 10).



a) b)

Figure 10. Example of buffering requests for software and hardware load balancing

Buffering looks as follows on the server: open (SYN), (SYN/ACK), etc. Processor and memory are unloaded. If the application delivery controller (ADC) is placed between the server and the client, all realized using ADC. The server does not know about it. Only at the end, it begins to engage to return pages.

An answer is buffering by other way. A server communicates directly with the client. It works with Crescendo or other application delivery controllers, it behaves as if the computer is in the local network. A server does not work in the global computer network and from it side, retransmission do not occur, and the process proceeds normally. Response time is much less. After using the application delivery controller a load on the processor decrease significantly. The processor is completely unloaded. Response time decreases only due to the fact that the server return deals exclusively with web-pages.

4. SURVEY OF FEW EXISTING LOAD BALANCING ALGORITHMS

Load balancer uses different algorithms for traffic management for the purpose of load balancing and/or the maximum use of all the servers in distributes systems. Higher bandwidth and improves response time in a distributed system are achieved because of algorithms. Each algorithm has advantages and shortcomings.

1. **Task Scheduling based on LB** [Singhal, 2011; Ghanbari, 2012; Ghuge, 2014; Rajwinder, 2014] is dynamic algorithm, based on load balancing, consists of a two-level scheduling mechanism. It provides a high efficiency of resources using. This algorithm balances the load by the primary tasks distribution in virtual machine, and then all virtual machines distribute on the host resources, in this way to improving the tasks response time, resource utilization and efficiency of cloud computing environment. This algorithm ensures the satisfaction of the dynamic users requirement and high ratio of resource utilization.

2. **Opportunistic Load Balancing (OLB)** [Singhal, 2011; Ghuge, 2014; Rajwinder, 2014] is static algorithm, trying to occupy each node, so it do not take into account the current load of each node or its suitability for the task. In other words, OLB sends unfulfilled tasks to currently available nodes in a random order, regardless of the current nodes load. The advantage is the simplicity, achieving load balancing, but its disadvantage is that it does not address the expected execution time for each task, resulting in a higher average completion time (total cycle time).

3. **Round Robin** [Keshav, 1997; Ghuge, 2014; Rajwinder, 2014] is a listing of the circular cycle – the first task is transferred to a node, then the next task is transferred to another and so on until it reaches

the last node, and then it all starts again. In this algorithm, all tasks are divided equally among all the processors, but various problems have a different run time, that is absolutely not taken into account load of nodes in the cluster. In Round Robin Scheduling (task management in systems with a time distribution), the algorithm identifies ring as a queue, and the fixed time slot. Each task can be done only in the time slot and queue. If the task can not be completed within one time slot, it will return to the queue for waiting a next round. However, it is difficult to determine the appropriate time slot. When the time slot is very high, the RR scheduling algorithm works the same as FCFS Scheduling. When a time slot is too small, the Round Robin Scheduling is known as Processor Sharing algorithm. Balancing Method Round Robin DNS does not require communication between servers, so it can be used for local and global balancing, and solutions, based on the Round Robin algorithm, are low cost.

4. **Weighted Round Robin** [Keshav, 1997; Gupta, 2013; Rajwinder, 2014] is improved version of Round Robin algorithm: each node is assigned a weight in accordance with its performance and capacity. This helps distribute the load more flexibly: nodes with more weight process more requests.

5. **Randomized** [Ray, 2012; Rajwinder, 2014] is static algorithm that randomly distributes the load across the available nodes, selecting one of them using a random number generator and sending a current task to it. This algorithm works well when all processes have the same load, but when the load is different computational complexity there are problems. This algorithm does not support the deterministic approach.

6. **Min-Min Algorithm** [Elzeki, 2012; Chen, 2013; Ghuge, 2014; Kashyap, 2014; Rajwinder, 2014] is a static load balancing algorithm, so that the parameters relating to the work are previously known. The algorithm as soon as possible provides resources for tasks that can be done as soon as possible. Minimum execution time of each task is found. The minimum value of time is searching among minimum execution time and the task with that value is sent for execution. Until all tasks have been assigned for execution a queued task will be updated and completed and executed tasks will be removed from the queue. Tasks with the maximum wait time should a non-specific period of time. The main problem with this algorithm is that it can lead to acute shortage. It works best when most of tasks have minimal execution time.

7. **Max-Min Algorithm** [Elzeki, 2012; Katyal, 2013; Ghuge, 2014; Kashyap, 2014; Rajwinder, 2014] is algorithm works almost the same way as in the algorithm min-min. The main difference is: in this

algorithm first finding out the minimum time perform tasks, select the maximum value that is the maximum time of all the tasks on all resources. Further task with founded maximum time is assigned for execution specifically on the selected node. Then execution time of all tasks is calculated on that node by adding the execution time of task to execution time of other tasks on that node. Then, set task is removed from the system.

8. **Honeybee Foraging Behavior** [Ghuge, 2014; Rajwinder, 2014] is decentralized algorithm that helps to achieve increased throughput and global load balancing by using of local actions server. Actual VM load calculate, after that a VM condition is solved: or it is overloaded or underloaded or balanced. VM are grouped in accordance with the current load. The priority of the task that awaits in the VM, is taken into account after removing it from the overloaded VM. Then, the task is assigned to underloaded VM. Previously taken task is useful for finding underloaded VM. These problems are known as bee intelligence agents in the next step. The algorithm reduces response time of VM and waiting time of task. System performance is enhanced with raise the heterogeneity of the system. The main problem is that the bandwidth does not increase with raising system size. Algorithm is most appropriate when is necessary diversity of species services.

9. **Active Clustering** [Gupta, 2013; Rajwinder, 2014]. In this algorithm the same components of the system are grouped together and they work in groups. It works as in the technique of self-assembled load balancing, where the network is rewired for load balancing system. The system is optimized using a similar assignments work by connecting these services. System performance is enhanced with improved resources. Bandwidth is improved by effectively utilizing all the resources.

10. **Compare and Balance** [Hu, 1998; Gupta, 2013; Rajwinder, 2014] is used to reaching the equilibrium state and managing of load balanced system. In this algorithm, based on the probability (the number of virtual machines running on the current host, and the whole cloud system), the current host randomly selects the host and compares their loads. If the load of current host is more than one of selected host, it sends an additional load on this particular node. Then, each node of the system carries out the same procedure. This load balancing algorithm is also designed and implemented to reduce the time migration of VM. Shared memory is used to reduce the time migration of VMs.

11. **Lock-free multiprocessing solution for LB** [Liu, 2013; Ghuge, 2014; Rajwinder, 2014] it offered unblocking multiprocessor load balancing solution that excludes the use of shared memory in contrast

to other multiprocessor load balancing solutions that use shared memory and locking to maintain the user's session. This is achieved by modifying the kernel. This solution helps to improve the overall performance load balancing in multicore environments by running multiple processes of load balancing in a single load balancer.

12. **Ant Colony Optimization** [Mishra, 2012; Dhinesh, 2013; Katyal, 2013; Rajwinder, 2014] is distribution algorithm. In this algorithm, resource information is dynamically updated with every move of ants. Multiple ant colonies are described so that a node sends colored colonies throughout the network. Painted ant colonies are used to prevent movement of ants from the same slot following by one route, and to ensure their distribution across all nodes in a system where each ant acts as a mobile agent, which carries an update load balancing information in the next node.

13. **Shortest Response Time First** [Singhal, 2011; Liu, 2013; Kashyap, 2014]. The idea of this algorithm is a direct forwarding. A priority is assigned to each process for run it. In processes with equal priorities planned FIFO order. SRTF algorithm is special case of the general priority scheduling algorithm. The priority is the reverse of the next burst of processor (CPU) in the SRTF algorithm. This means that if the burst processor increases, the priority is lowered. SRTF policy selects the task with the shortest processing time. In this algorithm, short tasks are done before long one. The SRTF is very important to know or estimate the processing time of each job and this is the main problem of SRTF.

14. **Based Random Sampling** [Singhal, 2011; Raghava, 2014] has approach of scalable and distributed load balancing, which uses a random sample of the system domain to achieve self-organization, in this way the load is balancing between all nodes in the system. System performance is improved by increasing the amount and similarity of resources, which leads to increased throughput by efficient using more scope of system resources. However, the algorithm gets worse with increasing variety resources.

15. **The two phase scheduling load balancing algorithm** [Singhal, 2011; Ghanbari, 2012; Katyal, 2013]. This combination OLB (Opportunistic Load Balancing) and LBMM (Load Balance Min-Min) scheduling algorithms, which uses a high performance implementation and system support load balancing. OLB keeps every node in operating condition to achieve the purpose of load balancing. LBMM scheduling algorithm is used to minimize the execution time of each task on the node, thereby

minimizing the total completion time. This algorithm is used to improve the efficiency of resource utilization and increases efficiency.

16. **Active Clustering load balancing Algorithm** [Hu, 1998; Singhal, 2011; Katyal, 2013]. The algorithm optimizes task by connecting similar services using the local rewiring. Work by grouping similar nodes. The process of grouping is based on the concept referee node. Referee node defines connection between the neighbors, which is similar to the initiating node. Then, a referee node breaks the connection between itself and a primary node. Next set of processes repeates again and again. System performance is increased by high availability of resources, because of this bandwidth also increases.

17. **ACCLB** [Singhal, 2011] is load balancing method based on ant colony and the theory of complex network (ACCLB) in open cloud computing. It uses low-level specifications and scaleless complex network to gain a better load distribution. This technique allows to overcome nonuniformity, is adaptive to a dynamic environment, is excellent in fault tolerance and has good scalability, consequently, helps to improve system performance.

18. **Decentralized content aware** [Randles, 2010; Singhal, 2011] is load balancing method, referred to as the workload and client notification policy (WCAP). This method uses a parameter called the USP for guidance unique and special properties of requests, as well as compute nodes. USP helps the planner to decide about the most appropriate node for processing tasks. This strategy is implemented on a decentralized basis with low costs. Using the content information to narrow the search, the search performance overall system is improved and downtime of computing nodes is reduce, thus improving their utilization.

19. **Server-based LB for Internet distributed services** [Randles, 2010; Singhal, 2011] is load balancing method for web services, distributed throughout the world. It helps in reducing the response time of the service by using protocol, which limits the forwarding of requests to the closest remote servers without overloading. For the implementation of this protocol is typically of middleware. It also uses heuristics to help web servers to withstand overload.

20. **Join-Idle-Queue** [Singhal, 2011, Gupta, 2013] is load balancing algorithm for dynamically scalable web services, provides a large-scale load balancing at distributed senders. At first it calculates the

availability of idle processors in each sender, and then assigns the task to processors to reduce the average length of the queue for each processor. When removing load balancing task from the critical path request process the algorithm effectively reduces the system load, it does not assume any communication load on the newly arrived task and does not increase the actual response time.

21. **Token Routing** [Ray, 2012] has the main goal to minimize the cost of the system by moving the markers within the system. But in a scalable a cloud system, agents can not have enough information to spread the workload due to communication bottlenecks. So load distribution between agents is not fixed. The disadvantage of this algorithm can be removed using a heuristic approach load balancing based marker. This algorithm provides a quick and effective solution to the routing. In this algorithm, agents do not need to have knowledge about the global state and workload neighbor. In order to make decisions on marker transfer agents actually build their own knowledge base. This knowledge base is derived from previously received tokens. Thus, in this approach there are no communications overhead.

22. **Central queuing** [Rajwinder, 2014] operates on the principle of dynamic allocation. Each new task comes to the queue manager and is queued. When a queue manager is received a request to perform a task, it removes the first task from the queue and sends it to the requester. If a queue does not have ready tasks, the request is buffered until the new task will not be available. But in the case of recording a new task in a queue until there are unanswered questions in the queue, the first such request is removed from the queue, and a new task put before it. When the CPU usage falls below the threshold value then the local boot manager sends a request for a new task to the central download manager. Then, the manager responds to the request, if finished task is found, otherwise complied with the order request before the new task.

23. **Connection mechanism** [Ray, 2012; Gupta, 2013; Liu, 2013]: load balancing algorithm may also be based on the mechanism of smallest amount connections, which is part of the dynamic scheduling algorithm. It is required for counting the number of connections for each server of dynamic load estimation. The load balancer writes the number of connections per server. Number of ports is increased when the new connection is sent to the server, and decreases when the connection is terminated or interrupted.

24. **Least connections** [Mishra, 2012; Kashyap, 2014] algorithm sends requests to the server, which is currently served by the smallest number of connections. The load balancer will monitor the number of connections of server and send the following request to the server with a minimum of connections.

5. LOAD BALANCING METHODS

In this section are described main operation load balancing methods that are available in modern balancers and/or can be configured on the servers.

5.1 Direct Routing

Direct routing mode (DR) is a high-performance solution with a slight modification in the existing infrastructure, and it works by changing the MAC-address of incoming packet on-the-fly, which works very quickly (Figure 11) [Roth, 2008; Бажин, 2010; Turnbull, 2015].

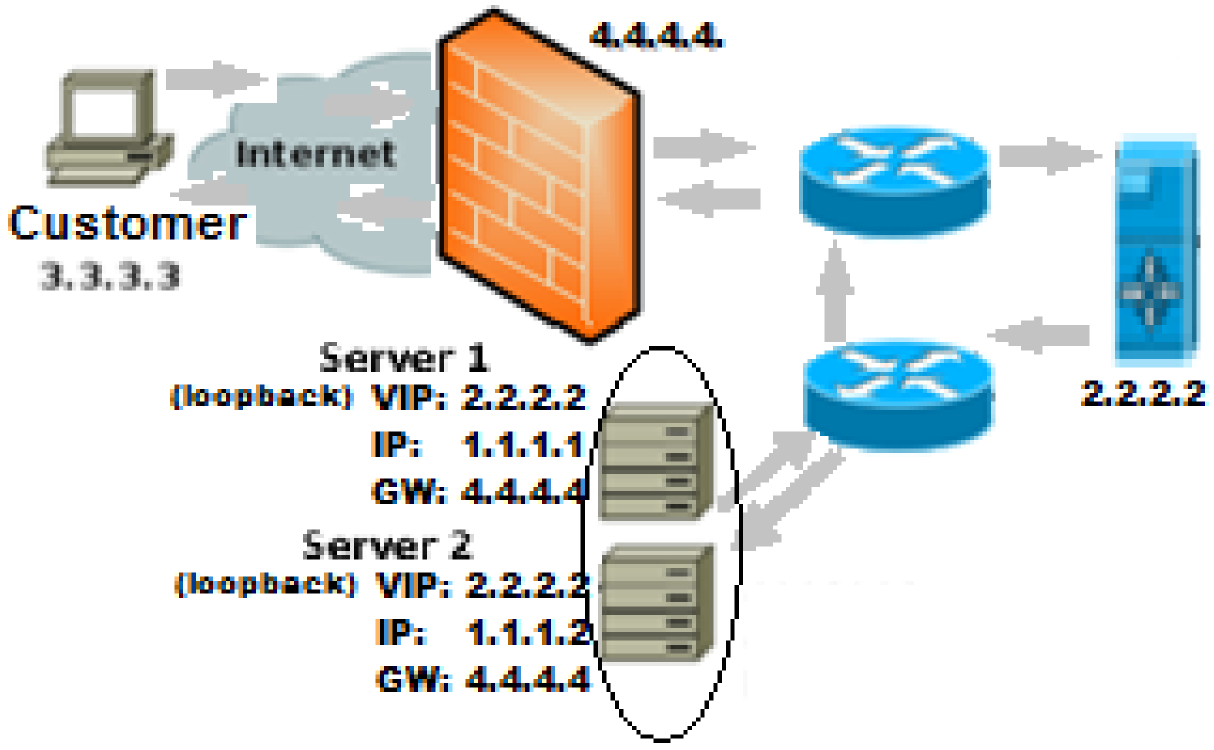


Figure 11. Example of direct routing load balancing method

I.e. it means that when the packet comes to the real server, it expects that server has a virtual IP, since it is necessary to assure the real server to respond to VIP, but do not respond to Address Resolution

Protocol (ARP) requests. Direct routing mode is enables for servers to access the network with VIP or real IP-address, without requiring any additional subnets or routes, but the real server must be configured to respond both VIP and it own IP-address.

5.2 Network Address Translation

Sometimes it is impossible to use the DR mode: either because the application can not communicate to RIP and VIP in the same time, or because the host operating system can not be modified to respond to ARP requests [Roth, 2008; Бажин, 2010; Turnbull, 2015]. In this case, it is possible to use the mode Network Address Translation (NAT) (Figure 12), which is also a high-performance, but requires infrastructure changes in internal and external subnet.

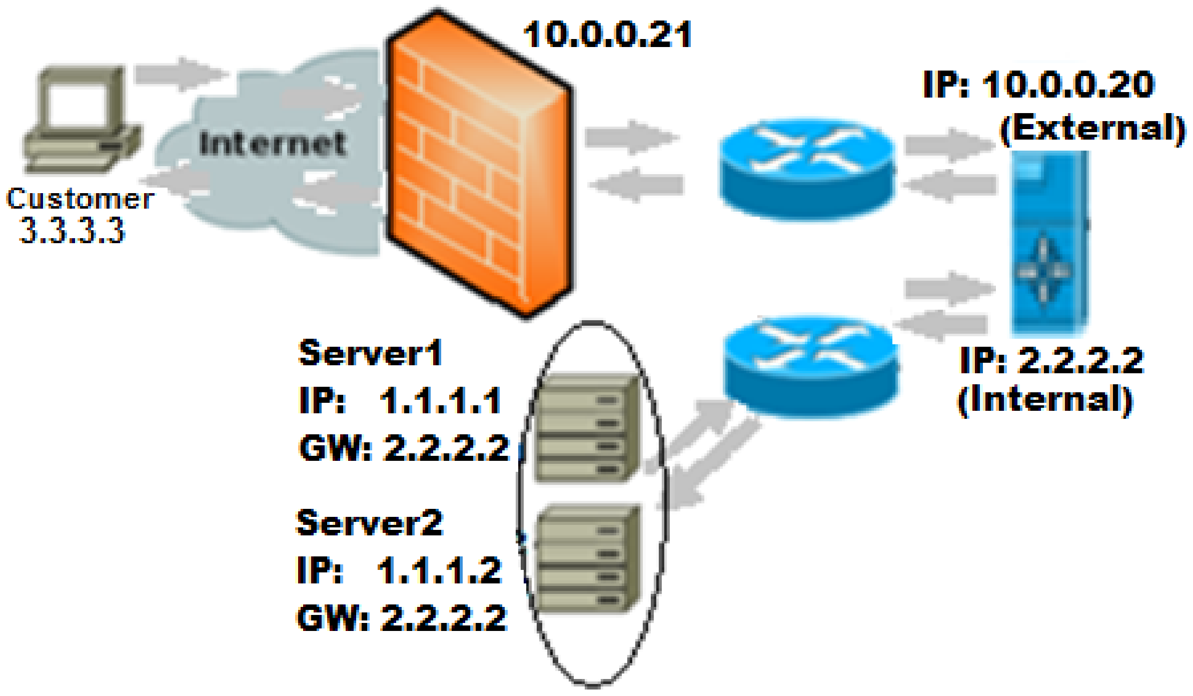


Figure 12. Example of Network address translation load balancing method

In this mode the load balancer translates all requests from external virtual server on internal real servers and real servers must have their default gateway, which are configured so that it points to load balancer. If real servers must be accessible by their own IP-address for a non-load balanced services, i.e. SMTP, it will be necessary to install individual firewall rules for each real server.

5.3 Source Network Address Translation

If the application requires the load balancer to handle cookie insertion, it is necessary to use Source Network Address Translation (SNAT) configuration, which does not require any changes to the application servers. However, since the load balancer acts as a full proxy, it does not have the same capacity as in the previous methods (Figure 13) [Roth, 2008; Turnbull, 2015].

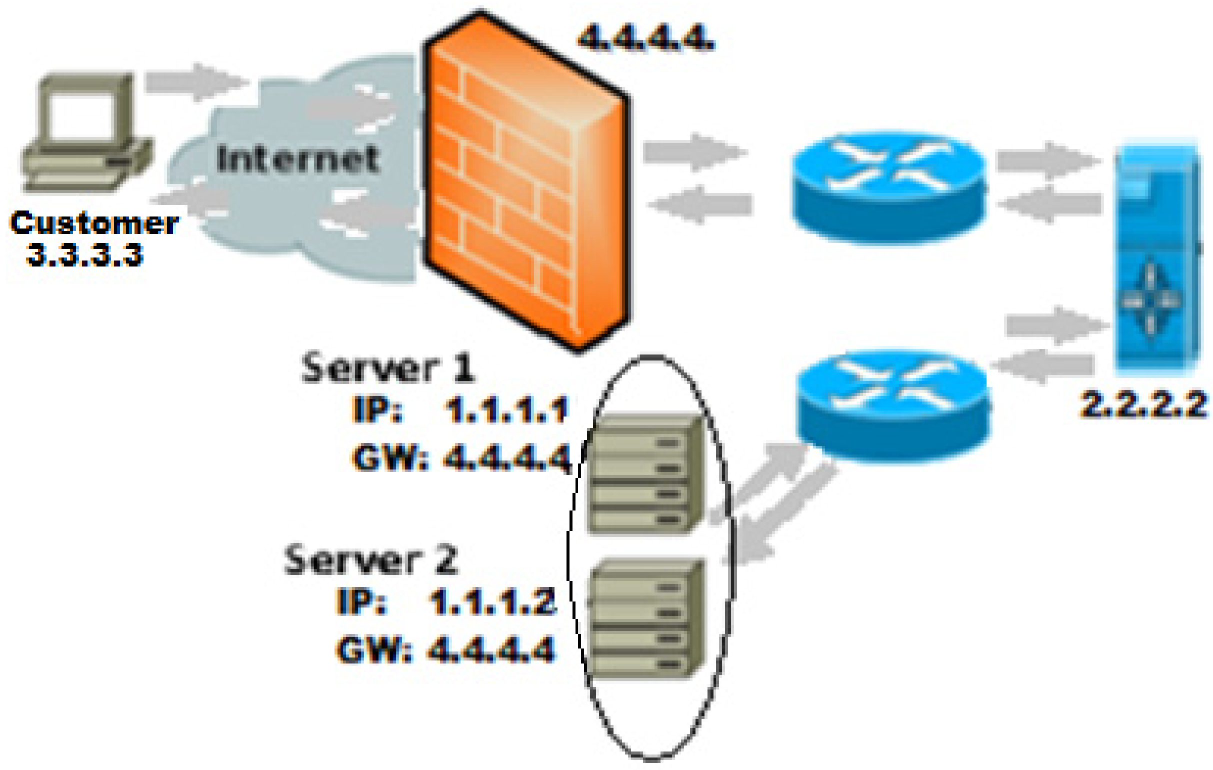


Figure 13. Example of Source Network address translation load balancing method

The load balancer proxies the application traffic to the servers so that the source of all traffic becomes the load balancer.

5.4 Transparent SNAT

If the source address of the client is a requirement then the balancer can be forced into transparent mode requiring that the real servers use the load balancer as the default gateway (as in NAT mode) and only operates for directly attached subnets (also as in NAT mode) [Roth, 2008; Turnbull, 2015].

5.5 SSL Termination or Acceleration

All load balancing methods of 4 and 7 levels can process SSL traffic on their level, i.e. internal servers do the decryption and encryption of traffic [Roth, 2008; Mendonca, 2014; Turnbull, 2015]. However, in order to check the HTTPS traffic, to read or insert cookies there is a necessity to decode (terminate) SSL traffic balancer, and this can be done by importing secure key and certificate to the load balancer, giving to it the right for decrypt traffic .

5.6 TCP/IP server load balancing

The tunnel mode looks like the Direct Server Return mode, except that traffic between the load-balancer and the server can be routed. The load-balancer encapsulates the request in IP tunnel to the server. The server recovers the client request from the load balancer, process it and forward the response directly to the client.

TCP/IP server load balancers operate on low-level layer switching [Roth, 2008]. The real servers appear to the outside world as a single "virtual" server. The incoming requests on a TCP connection are forwarded to the real servers by the load balancer, which runs a Linux kernel patched to include IP Virtual Server (IPVS) code.

To ensure high availability, a pair of load balancer nodes is set up, with one load balancer node in passive mode. If a load balancer fails, the heartbeat program that runs on both load balancers activates the passive load balancer node and initiates the takeover of the Virtual IP address (VIP). While the heartbeat is responsible for managing the failover between the load balancers, simple send/expect scripts are used to monitor the state of real servers.

Transparency to the client is achieved by using a VIP that is assigned to the load balancer. If the client issues a request, first the requested host name is translated into the VIP. When it receives the request packet, the load balancer decides which real server should handle the request packet. The target IP address of the request packet is rewritten into the Real IP of the real server.

5.7 Hashing

The load balancer on the 4-th level distributes a load across all available virtual machines by calculating the hash function of the traffic that has entered to this endpoint. This hash function is calculated such that all packets received within one connection (TCP or UDP) send to same server. Load balancer uses a set of 5 fields (IP address source, source port, IP destination address, destination port, protocol type) to calculate the hash used in the comparison of traffic and available server. Moreover, the hash function is selected in a way that distribution of servers connections are sufficiently random. However, depending on the type of traffic is acceptable that various connections are attached to the same server. Basic hash function allows get good distribution of requests at sufficiently large number of them from various sources [Roth, 2008; Mendonca, 2014; Turnbull, 2015].

Shortcoming of this algorithm is that it must be quite difficult to be able to distribute the load to other servers in the case of exclusion or inclusion of servers.

5.8 Caching

Caching is technology that introduces intelligence element to the concept of store and forward data to determine which information resources are copied from the core to the periphery of the network and how often they are updated. The caching technology is based on the fact that it is cheaper and more efficient to store data than to transmit. In fact, this idea is not new: in computers and other electronic computing devices, data is stored locally for reducing access time to them, the same principle is implemented in Web browsers, where the last viewed Web pages are cached on user hard drive [Meyer, 1998; Roth, 2008; Yucesan, 2011;Turnbull, 2015].

Caching removes some of the load from the overloaded Web sites and protects them from sharp traffic fluctuations.

Three configurations are most popular: the cache is placed near the router and processes traffic by Web cache control protocol (WCCP); cache combined with the switch Level 4 or above, controls the traffic; cache is embedded in a layer 2 switch or Layer 3, and traffic control is transferred to a separate load balancer.

Based on load-balancing scheduling algorithm user session requests are handled by different servers. If the cache is used on the server side, wandering requests will be a problem. In this situation, there is a method in which the cache is placed in a global space. Memcached is a popular solution of distributed caching, which provides a large cache on multiple machines. This is a distributed cache that uses sequential hashing to determine the cache server (daemon) for the cache entry. Based on the cache the hash code in the client library always displays the same hash code for the same address of the server cache. Then this address is used to store the cache entry. Figure 14 illustrates this approach cache.

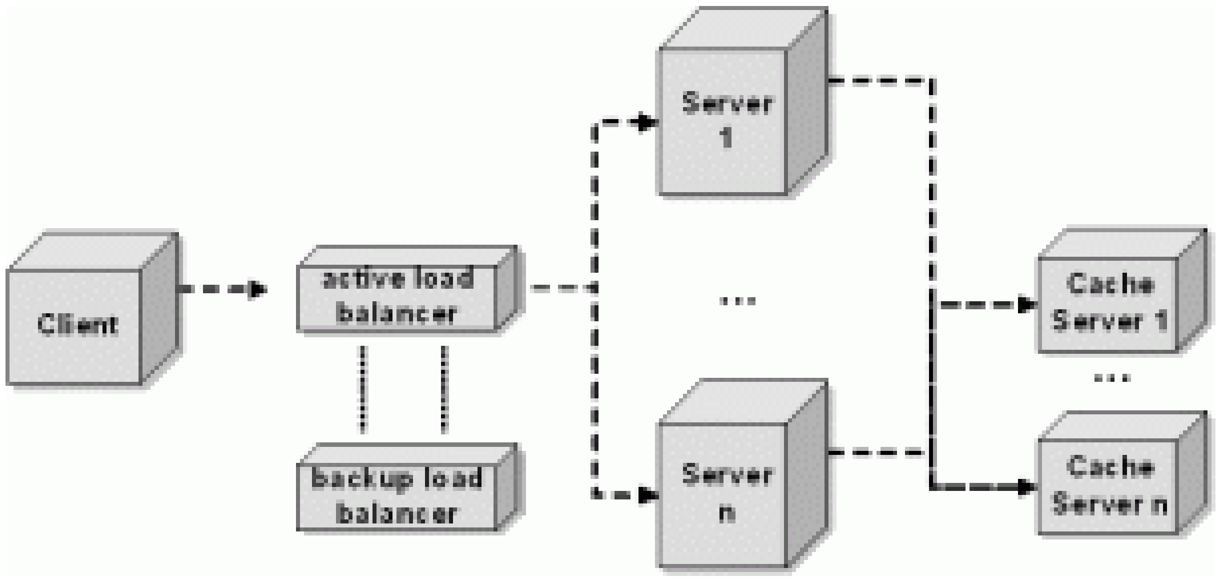


Figure 14. Load balancer architecture enhanced by a partitioned, distributed cache

Cache methods are divided into two types [Cao, 1996; Meyer, 1998; Forney, 2002; Angrish, 2011; Савчук, 2012]:

1) cache-unaware. This kind of strategy does not take into account the possible difference between the data that are cached locally on each server. Such strategies work well in two cases:

If the server does not cache anything locally (stateless servers). For example, when data load from a remote resource on the network requires less CPU resources and time of same data load from the local cache. Such situation is possible if the cache hit ratio for the local cache aspires to zero due to the large scope of data, which needs to be cached. It is also possible with the frequent updating data, so that the next request have out of date.

If local caches on all servers contain the same data. In this case, no matter where are sent the next requests. For cache-unaware strategies the best is distribution of load on the least loaded server at this moment. This strategy allows to achieve the lowest latency in the request queue.

2) cache-aware. This kind of strategy sends requests so as to maximize the amount of cache hits in local caches of servers.

The most famous of these strategies is sticky load balancing (SLB). It is based on fair assumption that the requests from the one user are needed of the common data which are amenable to local cache on the server. Such data may include user settings and data that have meaning only for a particular user. It is obvious that the direction of requests from the one user on the one server allows to maximize the cache hit ratio.

This strategy works properly with the following conditions:

• If the user performs more than one request to the server for a short time.

• When processing requests from one user a server needs the one data that are specific to that user, and these data require large computing resources or consume a lot of network traffic when pull them out of the remote services.

Comparing cache-aware and cache-unaware distribution strategies requests.

Advantages of the cache-aware strategy:

• Well-optimized design (i.e. actively using local caches to minimize the cost of CPU time and network traffic) working on a single server is much easier to migrate to multiple servers by using cache-aware load balancing.

• It reduces server load and increases the number of requests that can be processed by each server per unit time, as no need to waste CPU time to generate the data, if they are already present in the local cache. Also, as opposite to shared cache, the local cache may contain a prepared data that do not need to waste the network traffic and CPU time on serialization before writing and deserialization before reading.

• It reduces the load on external data sources and the network between servers and external data sources, as it does not need to pull the data if they are already present in the local cache.

• It reduces the time of request processing, as it does not need to wait for a response from the external data sources if they are already present in the local cache.

• It reduces the total amount of memory required on the local caches, as data cached on different servers almost are not doubled. On the other hand, it gives opportunity to cache more different data in a fixed amount of local cache, thereby increasing the effective cache size.

Shortcomings of a cache-aware strategy:

• Higher average waiting time in the queue requests compare to the round robin and least loaded at the same average server load. This shortcoming is offset by the fact that cache-aware strategy usually can process more requests per unit of time at a comparable server load compared to the cache-unaware strategies due to the above advantages.

• More complex synchronization of local caches in comparison with a shared cache. If data is cached only in the shared cache, the current request can be processed on an arbitrary server, as current data for the user is always possible to try to pull out of the total cache.

In cache-aware strategists a same data may be out of sync if the group of requests gets for a short time on the "foreign" server, and then extend to "own" server. This is possible in the case of short-term false "failure" of one of servers, which quickly returns to operation without loss of the local cache.

It should not rely on the safety of data in the cache, because at any moment they can be lost. This can happen in various ways - for example, the cache has grown to enormous size and its need to compress one by removing out any data. Or service responsible for caching is fail. Critical data of user sessions are not recommended to keep in the local cache without placing them in storage, guaranteeing their safety.

5.9 DNS Load Balancing

This is the easiest load balancing method, the essence of which is that it creates multiple DNS-record type A for the domain record on DNS-server [Roth, 2008; Бажин, 2010; Natario, 2011; Turnbull, 2015]. DNS server issues recording-type A in an alternating cyclic basis.

Usually clients' resolver is programmed in a way that a customer's caching DNS-server does not affect to balance. Also the client chooses random entries from received one (Figure 15). Accordingly, there is a connection to the appropriate server.

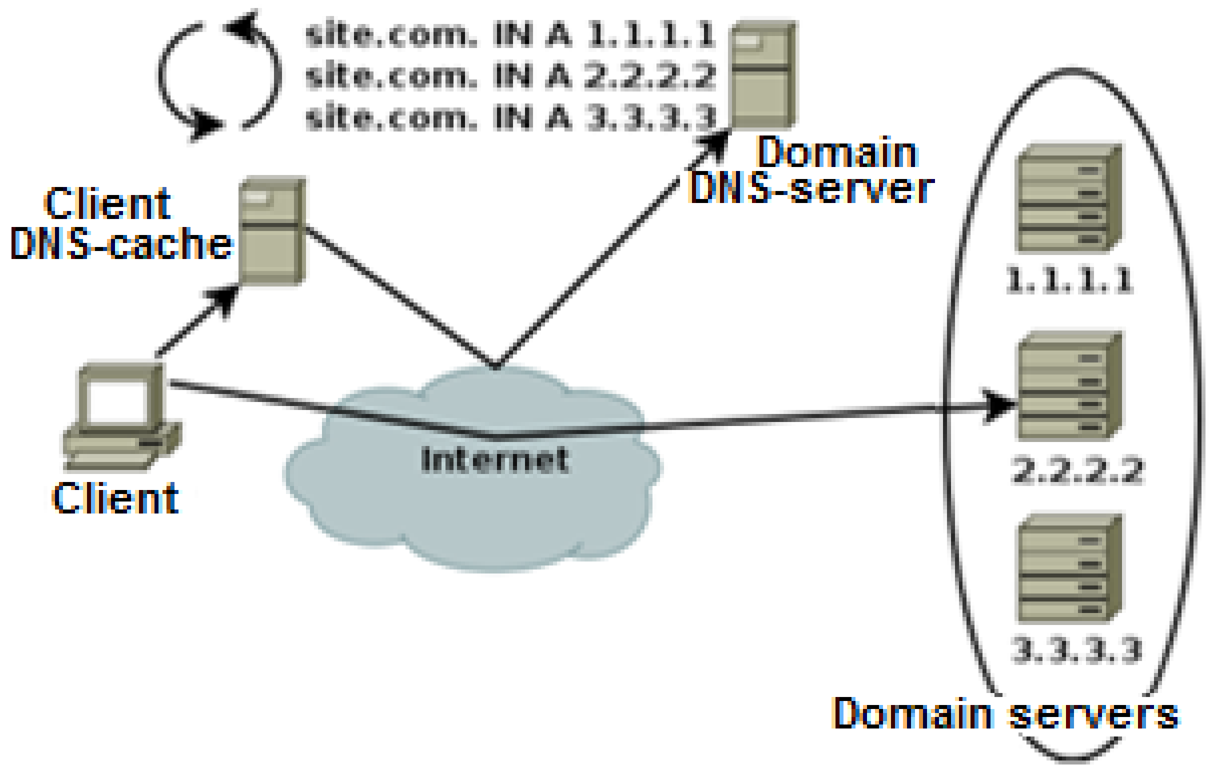


Figure 15. DNS load balancing

To implement this method any DNS-server is suites perfectly.

Advantages of the method:

• It does not depend on a high-level protocol, i.e. this method can be used by any protocol, which handle to the server by name.

• It does not depend on the server load: because there is caching DNS-server, it is no difference how many customers are: one or millions.

• No communication between servers, so it can be used for local balancing (this balancing of servers within the same data center), and for the global balancing when there are multiple data centers, where servers are not connected with each other.

• Low cost solutions.

Shortcomings of the method:

• It is difficult to disconnect the servers that do not respond or have failed: because of the cache entry is deleted only after the time that is specified by TTL (Time To Live), or when forced caching is more longer.

• Load balancing between servers in the correct proportions is difficult.

• Clients, which concentrated in certain areas, provide the load on one server. This can give a large nonuniformity in the distribution of the load among servers.

• Maximum numbers of IP-addresses that can be balanced is limited.

•Calls of TCP from clients are not open on all DNS-servers.

5.10 Network Load Balancing

Microsoft's Network Load Balancing (NLB) – implementation is based on software that runs on each node in the cluster using a hashing algorithm that takes IP-address, or IP-address and port from incoming request and determines which node (host) cluster will handle the request. Each node in the cluster receives each packet traffic and determines which node is responsible for the reaction perform by applying a filter to each packet, in this way, only one node will eventually have service the request [Natario, 2011; Turnbull, 2015].

NLB concept is quite simple: each server in the cluster of load balancing is configured with a virtual IP-address and this address is configured on all servers that take part in the cluster. Whenever a request is made to the virtual IP, the network driver on each of these machines intercepts the request for the IP-address and forwards the request to one of the machines in the cluster of load balancing on the basis of rules that can be configured for each server in the cluster.

NLB acts as a virtual network device with a private IP-address and a real device (the actual port Ethernet) associated with load balancing. Instead of using and the ports' IP addresses, the system will use the NLB software's IP address and this will ultimately result in the NLB software and its ports looks like a single device to the clients (Figure 16).

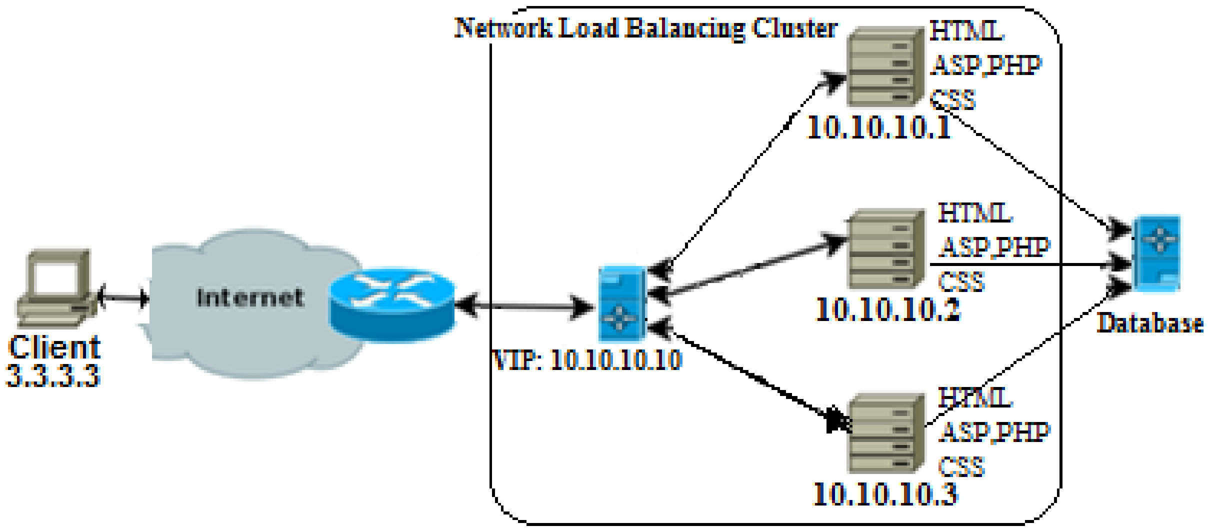


Figure 16. The usage of Network Load Balancing

5.11 Proxy method of load balancing

There is some proxy server that uses IP-address of our service, it receives a request, does something with it if it necessary and forwards it from his own to a necessary server (Figure 17) [Бажин, 2010; Red Hat, 2015]. This method is not transparent for the server, as it sees that it had been approached proxy [Roth, 2008; Red Hat, 2015]. To do this inside a high-level protocol we have to somehow send information about which client approaches us. For the HTTP protocol, we can add the header X-Real-IP with the client IP-address.

Advantages of proxy method:

• Because the proxy works at the protocol level, a proxy server can analyze and change our questions and answers. It allows to do bind a client to the server. For example, at value a specific setup cookie. This can be useful if we use local storage of the client session on the server.

• Proxy allows to distribute different types of requests to different servers. We can identify individual servers, which will "give" statically generated homepage, and do it with much less latency than the servers that handle "difficult" questions.

• It is possible to modify the response request. There can be done addition the title, for example, conversion. It is possible to cache the responses to this proxy.

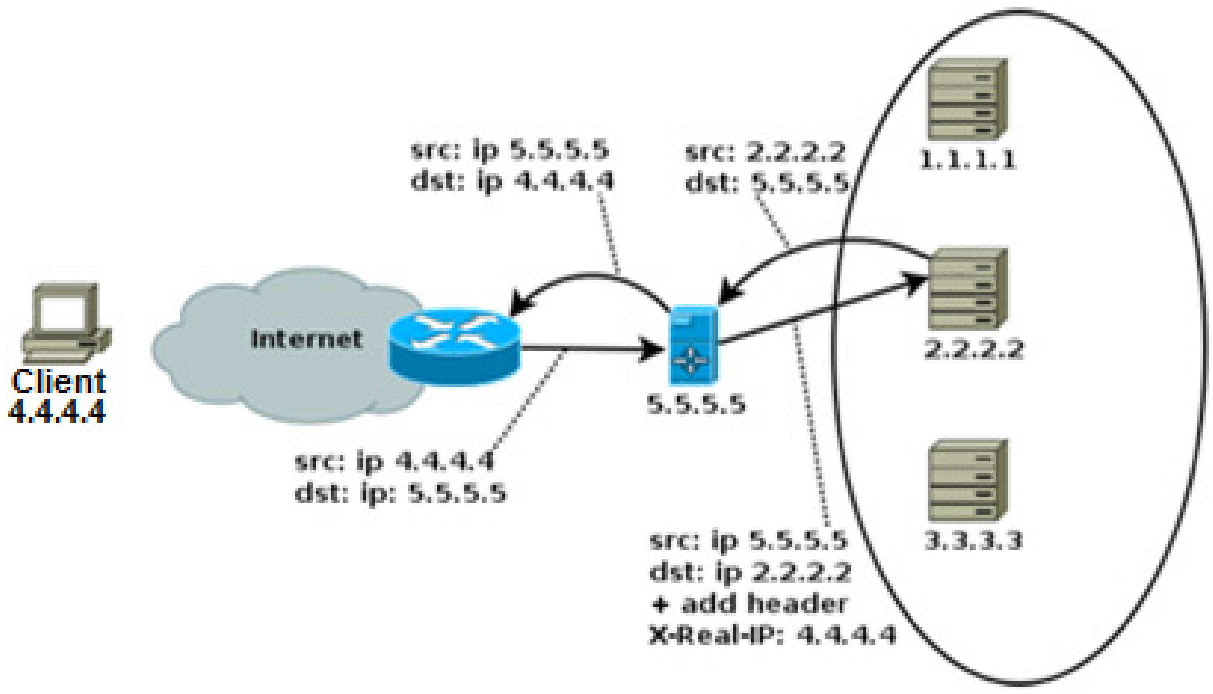


Figure 17. The usage of Proxy method

Shortcomings of the method:

• Proxy method has greatest resource consumption, as it involves higher-level protocols in comparison with other methods.

• Each protocol must have their own proxy type. Proxy cannot be implemented for everyone protocols in a way that it decides the required tasks.

5.12 Load balancing by using redirection

Another method of balancing is balancing redirects. Redirect is applicable to quite low number protocols [Red Hat, 2015]. Fortunately, it is applicable for HTTP.

There is some balancer that by referring to our service gives to client redirect to a specific server. In the case of HTTP this will look like a "HTTP redirect 302", the redirect code will appear as "Moved temporary".

Advantages of proxy method:

• If the request is enough "difficult", sometimes it makes sense to use a redirection even for global balancing.

• The method also allows to distribute various types of requests to different servers. Requests may be analyzed well.

Shortcomings of the method:

• It is applicable to a very low number of high-level protocols.

• On every client request two requests are made. One is to the redirector, one is to the server that handles the connection. This increases the time during which the client will receive the final answer to his request.

Conclusion

In this paper, a review of the main existing algorithms and methods for load balancing of distributed systems were carried out and the structure of load balancing methods for levels of OSI model are shown. The analysis of load balancing methods on different levels of the network model is carried out, the advantages and shortcomings of each method were given. The comparative analysis of hardware and software load balancing is dissected. Their differences, strengths and weaknesses are dissected. Also, description of the main features of load balancing algorithms, their advantages and shortcomings were presented.

Bibliography

[Angrish, 2011] R. Angrish, D. Garg. Efficient String Sorting Algorithms: Cache-aware and Cache-Oblivious. International Journal of Soft Computing and Engineering (IJSCE). – Vol1(2). – 2011. – P.12-16.

[Cao, 1996] P.Cao, E.W.Felten, A.R.Karlin, K.Li. Implementation and Performance of Integrated Application-Controlled File Caching, Prefetching, and Disk Scheduling. ACM Transactions on Computer Systems. – Vol.14(4). – 1996. – P.311-343.

[Cardellini, 1999] Valeria Cardellini, Michele Colajanni, Philip S. Yu. Dynamic Load Balancing on Web-server Systems. IEEE Internet Computing. - Vol.3. - No.3. – 1999.– P.28-39.

[Cardellini, 2001] V. Cardellini. A performance study of distributed architectures for the quality of web services. Proceedings of the 34th Conference on System Sciences. – Vol.10. – 2001. P.213-217.

[Chen, 2013] H. Chen, F. Wang, N. Helian, G. Akanmu. User-priority guided min-min scheduling algorithm for load balancing in cloud computing. National Conference Parallel Computing Technologies. - 2013. – P.1-8.

[Dhinesh, 2013] Dhinesh Babu L.D., P. Venkata Krishna, Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied Soft Computing. – Vol 13(5). - 2013. - P.2292–2303.

[Elzeki, 2012] O. Elzeki, M. Reshad, M. Elsoud. Improved max-min algorithm in cloud computing. International Journal of Computer Applications. - Vol. 50(12). – 2012. - P. 22–27.

[Erl, 2013] Thomas Erl, Ricardo Puttini, Zaigham Mahmood. Cloud Computing: Concepts, Technology & Architecture. Prentice Hall. Ed.1st. – 2013. – P.528.

[Erl, 2015] Thomas Erl, Robert Cope, Amin Naserpour. Cloud Computing Design Patterns. Prentice Hall. Ed.1st. – 2015. – P.592.

[Forney, 2002] Brian C. Forney, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau. Storage-Aware Caching: Revisiting Caching for Heterogeneous Storage Systems. Conference on file and storage technologies.– 2002. – P.61-74.

[Ghanbari, 2012] Shamsollah Ghanbari, Mohamed Othman. A Priority based Job Scheduling Algorithm in Cloud Computing. International Conference on Advances Science and Contemporary Engineering (ICASCE). –V. 50. - 2012. – P.778-785.

[Ghuge, 2014] Kalyani Ghuge, Minaxi Doorwar. A Survey of Various Load Balancing Techniques and Enhanced Load Balancing Approach in Cloud Computing. International Journal of Emerging Technology and Advanced Engineering. - Volume 4(10). 2014. – P.410-414.

[Gupta, 2013] Rohit O. Gupta, Tushar Champaneria. A Survey of Proposed Job Scheduling Algorithms in Cloud Computing Environment. International Journal of Advanced Research in Computer Science and Software Engineering. – Vol.3(11). – 2013. – P.782-790.

[Hong, 2006] Y.S. Hong, J.H. No, S.Y. Kim. DNS-based load-balancing in distributed web-server systems. Proceeding, in: Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (WCCIA 2006). – 2006. – P.251-254.

[Hu, 1998] Y. Hu, R. Blake, D. Emerson. An optimal migration algorithm for dynamic load balancing. Concurrency: Practice and Experience. – V.10(6). – 1998. P. 467–483.

[Jiao, 2010] Yang Jiao, Wei Wang. Design and Implementation of Load Balancing of Distributed-system-based Web Server. Electronic Commerce and security. – 2010. – P.337 – 342.

[Kashyap, 2014] Dharmesh Kashyap, Jaydeep Viradiya. A Survey Of Various Load Balancing Algorithms In Cloud Computing. International Journal Of Scientific & Technology Research. - Vol.3(11). – 2014. - P.115-119.

[Katyal, 2013] Mayanka Katyal, Atul Mishra. A Comparative Study of Load Balancing Algorithms in Cloud Computing Environment. International Journal of Distributed and Cloud Computing. – Vol.1(2). – 2013. – P.6-14.

[Keshav, 1997] S.Keshav. An Engineering Approach to Computer Networking. Addison-Wesley, Reading, MA. – 1997. - P. 215-217.

[Kopparapu, 2008] Kopparapu Chandra. Load balancing servers, firewalls, and caches. Published by John Wiley & Sons, Inc. – 2002. – P.208.

[Koteswaramma, 2012] Rudra Koteswaramma. Client-Side Load Balancing and Resource Monitoring in Cloud. International Journal of Engineering Research and Applications (IJERA). - Vol.2(6). – 2012. - P.167-171.

[Laviol, 2014] Vitalij Laviol. Приборы с балансировкой нагрузки в системах сетевого мониторинга или «что такое Network Packet Broker». . – 2014. - URL: http://m.habrahabr.ru/company/metrotek/blog/259633/

[Liu, 2013] Jing Liu, Xing-Guo Luo, Xing-Ming Zhang, Fan Zhang, Bai-Nan Li. Job Scheduling Model for Cloud Computing Based on Multi-Objective Genetic Algorithm. IJCSI International Journal of Computer Science. – V.10(1). - № 3. - 2013. – P.134-139.

[Mendonca, 2014] M. Mendonca, B.A.A. Nunes, X.-N. Nguyen, K.Obraczka, T. Turletti. A Survey of software-defined networking: past, present, and future of programmable networks. Communications Surveys & Tutorials, IEEE. – Vol.16(3). – 2013. – P.1617-1634.

[Meyer, 1998] Richard A. Meyer, Rajive Bagrodia Parsec. User Manual. Release 1.1. UCLA Parallel Computing Laboratory. - 1998. – URL: pcl.cs.ucla.edu/projects/parsec.

[Mishra, 2012] Ratan Mishra, Anant Jaiswal. Ant colony Optimization: A Solution of Load balancing in Cloud. International Journal of Web & Semantic Technology (IJWesT). - Vol.3. - No.2. - 2012. - P.335-338.

[Natario, 2011] Rui Natario. Load Balancing. – 2011. - URL: http://networksandservers.blogspot.com/2011/03/load-balancing-iv.html

[Pavan Kumar, 2012] Illa Pavan Kumar, Subrahmanyam Kodukula. A Generalized Framework for Building Scalable Load Balancing Architectures in the Cloud. International Journal of Computer Science and Information Technologies. - Vol.3(1). – 2012. – P.3015 – 3021.

[Raghava, 2014] N. S. Raghava, Deepti Singh. Comparative Study on Load Balancing Techniques in Cloud Computing. Open journal of mobile computing and cloud computing. – Vol.1. – No.1. – 2014. – P.18-25.

[Rajwinder, 2014] Rajwinder Kaur, Pawan Luthra. Load Balancing in Cloud Computing. Association of Computer Electronics and Electrical Engineers. – 2014. - P.374-381.

[Randles, 2010] Martin Randles, David Lamb, A. Taleb-Bendiab. A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing. IEEE 24th International Conference on Advanced Information Networking and Applications Workshops. – 2010. –P.551-556.

[Ray, 2012] Soumya Ray, Ajanta De Sarkar. Execution analysis of load balancing algorithms in cloud computing environment. International Journal on Cloud Computing: Services and Architecture (IJCCSA). - Vol.2. - No.5. – 2012. - P.2657-2664.

[Red Hat, 2015] Red Hat Enterprise Linux. Обзор планирования распределения нагрузки. – 2015. - URL: https://access.redhat.com/documentation/ru-RU/Red_Hat_Enterprise_Linux/6/html/Virtual_Server_Administration/s1-lvs-scheduling-VSA.html.

[Roth, 2008] Gregor Roth. Server load balancing architectures, Part 1: Transport-level load balancing. – 2008. - URL: http://www.javaworld.com/article/2077921/architecture-scalability/server-load-balancing-architectures--part-1--transport-level-load-balancing.html.

[Singhal, 2011] Priyank Singhal, Sumiran Shah. Load Balancing Algorithm over a Distributed Cloud Network. 3rd IEEE International Conference on Machine Learning and Computing. – Singapore. – 2011. - P.37-42.

[Tuncer, 2011] D. Tuncer, M. Charalambides, G. Pavlou, N. Wang. Towards decentralized and adaptive network resource management. Network and Service Management (CNSM), IEEE. – 2011. – P.1–6.

[Turnbull, 2015] Malcolm Turnbull. Load Balancing Methods. – 2015. - URL: http://www.loadbalancer.org/blog/load-balancing-methods

[Vlaeminck, 2004] Vlaeminck K., Van Hoecke S., De Turck F., Dhoedt B., Demeester P. Design and implementation of an application server load balancing architecture supporting the end-to-end provisioning of value-added services. Telecommunications Network Strategy and Planning Symposium. - 2004. - P.345 – 350.

[Yucesan, 2011] Enver Yucesan, Yah Chuyn Luo, Chun-Hung Chen, Insup Lee. Distributed Web-based Experiments for optimization. Simulation Practice and Theory. – Vol.9. – 2001. - P.73-90.

[Zhihao, 2013] Zhihao Shang, Wenbo Chen, Qiang Ma, Bin Wu. Design and implementation of server cluster dynamic load balancing based on OpenFlow. Awareness Science and Technology and Ubi-Media Computing (iCAST-UMEDIA). - 2013. – P.691 – 697.

[Бажин, 2010] Алексей Бажин. Принципы балансировки Компании Mail.ru. – 2010. - URL: http://profyclub.ru/docs/21.

[Гуревич, 2010] Григорий Гуревич. Crescendo Networks - эволюция в мире WEB балансировки. – 2010. - URL: http://profyclub.ru/docs/99.

[Кириченко, 2011] Л.О. Кириченко, Э. Кайали, Т.А. Радивилова. Анализ методов повышения QoS в сетях MPLS с учетом самоподобия трафика. Системні технології. – 2011. – Вип. 3. – С. 52–59.

[Савчук, 2012] Игорь Савчук. Балансировка нагрузки сервера по методу SLB. – 2012. – URL: http://blogerator.ru/page/high-load-balansirovka-nagruzki-servera-po-metodu-sticky-load-balancing

Authors' Information

Igor Ivanisenko *– post graduate student, department of Applied Mathematics, Kharkiv National University of Radioelectronics; 14 Lenin Ave., 61166 Kharkiv, Ukraine;*

e-mail: ivanisenko79@yahoo.com.

Major Fields of Scientific Research: Modeling of Network traffic behavior, Self-similar traffic properties in Cloud technologies, Modeling of Queuing Systems, Computer networks