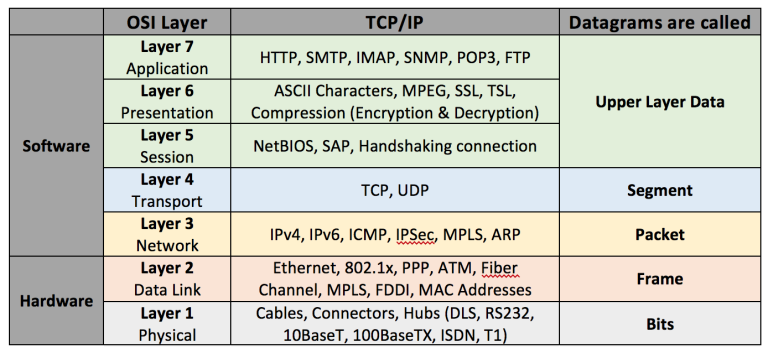
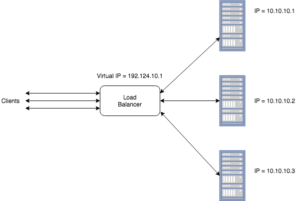
OSI Model:



* **Application Gateway** is an HTTP/HTTPS web traffic load balancer with URL-based routing, SSL termination, session persistence, and web application firewall. [Learn more about Application Gateway](https://go.microsoft.com/fwlink/?linkid=2032051)
* **Azure Load Balancer** supports all TCP/UDP network traffic, port-forwarding, and outbound flows

## **How load balancers work**

Load balancers distribute incoming client requests to computing resources such as application servers and databases. Load balancer is deployed in front of a server farm. All servers are connected to the load balancer. The load balancer appears as a virtual server to the clients. It has a virtual IP address and represents entire server farm.



Clients connects to the load balancer instead of real servers using virtual IP and load balancer distribute the request across the real servers.

## **Benefits**

* Preventing requests from going to unhealthy servers. The load balancer continuously monitors the health of real servers and the applications running on them. If a real server or application fails the health check, the load balancer avoids sending any client requests to that server.
* Preventing overloading resources. Load balancer evenly distribute the load across real servers.
* Helping eliminate single points of failure.
* SSL termination.  Decrypt incoming requests and encrypt server responses so backend servers do not have to perform these potentially expensive operations. It improve overall performance of the system.
* Session persistence. Issue cookies and route a specific client’s requests to same instance if the web applications do not keep track of sessions

## **Load balancing methods**

There are two ways to perform load balancing: **stateless** or **stateful**. If the balancer doesn’t keep track of any session information, it is called stateless load balancing. If the load balancer keeps track of state information for every information, it is called stateful.

## **Stateless load balancing**

The common implementation of a stateless balancing involves creating a **hash** of IP address of the client down to small number. Then balancer uses this number to decide which server should handle a request. Hashing on source IP doesn’t provide a good distribution since one client can create a lot of requests that will be send to one server. But since client generates each request using a different source port, we can use combination of IP and port to create a hash value.

Hashing is the most basic algorithm. **Hash buckets** is an advanced form of stateless load balancing algorithm. The algorithm uses a two tier distribution. Lets assume you have 5 servers connected to a single load balancer. First step is creating a N buckets, for example 255. Then we assign a server to every bucket:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 3 | 5 | 1 | 1 | 2 | 4 | 3 | … | 3 |

If server 1 goes down, load balancer reassigns hash buckets for this server:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 3 | 5 | 5 | 2 | 2 | 4 | 3 | … | 3 |

**Advantages**:

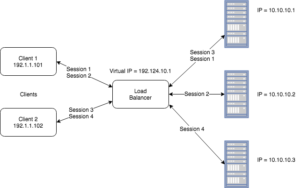
* Simple and fast
* Don’t need a lot of resources (processor and memory)

**Disadvantages**:

* Treats all clients equally, for example one client generates 100 requests per second and another generates 5 seconds per second. The load distribution will be poor in this case.
* Many clients can be hashed to a single server

## **Stateful load balancing**

Stateful load balancer keeps a session table to track each session. When session is initiated, the load balancer uses a load distribution algorithm to choose the destination server and sends all subsequent packets to this server until session is terminated.



**Session table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Source IP | Source port | Destination IP | Destination port | Server |
| 192.1.1.101 | 321 | 192.124.10.1 | 80 | 10.10.10.1 |
| 192.1.1.101 | 322 | 192.124.10.1 | 80 | 10.10.10.2 |
| 192.1.1.102 | 543 | 192.124.10.1 | 80 | 10.10.10.1 |
| 192.1.1.102 | 544 | 192.124.10.1 | 80 | 10.10.10.3 |

## **Load distribution algorithms**

* **Round robin**. Round-robin algorithms pair an incoming request to a specific machine by cycling through a list of servers capable of handling the request. It can not ensure even distribution of load, because each connection may stay open for a different time. It consumes little resources and can be used where load distribution algorithm consumes a lot of processing time and each request is roughly equivalent.
* **Least connections**. Load balancer sends a request to the server that has least connections. Load balancer needs to track number of active connections on each server. It is very simple and very popular.
* **Weighted distribution**. Since all servers in a farm can have different CPUs and memory, weighted distribution algorithm allows to assign a weight to each server depending on its performance. This method can be combined with another methods, like round robin or least connections.
* **Response time**. Load balancer measures response time of a request and provides the server with the best response time. Load balancer can create a http request for the web server to measure performance or TCP SYN/ACK time for other requests. It is pretty complex method and it’s not usually used alone, but it can be used with other methods.
* **Server probes**. Each server runs a server-side agent that sends detailed information about the server to load balancer including CPU load, memory usage, etc.

## **Load balancing layer 4/7**

Most Load balancers can work at two levels:

* **Level 4**. A load balancer has visibility on network information such as application ports and protocol (TCP/UDP). The load balancer delivers traffic by combining this limited network information with a load balancing algorithm such as round-robin and by calculating the best destination server based on least connections or server response times.
* **Level 7**. A load balancer has application awareness and can use this additional application information to make more complex and informed load balancing decisions. With a protocol such as HTTP, a load balancer can uniquely identify client sessions based on cookies and use this information to deliver all a clients requests to the same server. This server persistence using cookies can be based on the server’s cookie or by active cookie injection where a load balancer cookie is inserted into the connection.

## **Types of load balancer**

There are 3 types of load balancers:

1. Hardware load balancers
2. Cloud load balancers
3. Software load balancers

## **Hardware load balancers**

Some of the popular LB hardware vendors are:

* [F5](https://f5.com/)
* [TP-Link](https://www.tp-link.com/ae/products/biz-list-4910.html)
* [Barracuda](https://www.barracuda.com/products/loadbalancer/models)

They are usually very expensive but are the most flexible.

## **Cloud load balancing**

Cloud load balancers raise popularity these days. You pay for what you use. They provide monitoring, auditing and can provide global load balancing features (load balancing between data centers).

Some popular vendors are:

* [Oracle load balancing](https://cloud.oracle.com/load-balancing)
* [AWS elastic load balancing](https://aws.amazon.com/documentation/elastic-load-balancing/)
* [Google cloud load balancing](https://cloud.google.com/load-balancing/)
* [Azure load balancer](https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-overview)

Its advantages over traditional load balancing of on‑premises resources are the (usually) lower cost, ease of managing and installing and the ease of scaling the application up or down to match demand. The ease and speed of scaling in the cloud means that companies can handle traffic spikes (like those on Cyber Monday) without degraded performance by placing a cloud load balancer in front of a group of application instances, which can quickly autoscale in reaction to the level of demand.

## **Software load balancers**

* [HAProxy](https://www.haproxy.org/) – free open source TCP/HTTP load balancer. It offers good performance, monitoring/statistics,  health-check, HTTP/2 support. It has commercial version – [enterprise edition](https://www.haproxy.com/products/haproxy-enterprise-edition/). This load balancer is used by GitHub, Airbnb and Reddit.
* [Ngnix](https://www.nginx.com/) – free open source load balancer. It is very fast. Doesn’t have monitoring/statistics in the free version but you can buy Ngnix Plus that has more features and monitoring/statistics.
* [Seesaw](https://opensource.googleblog.com/2016/01/seesaw-scalable-and-robust-load.html) – Used by Google, a reliable Linux-based virtual load balancer server to provide necessary load distribution in the same network.