### **Introduction to Load Balancers**

**Purpose of Load Balancers:**

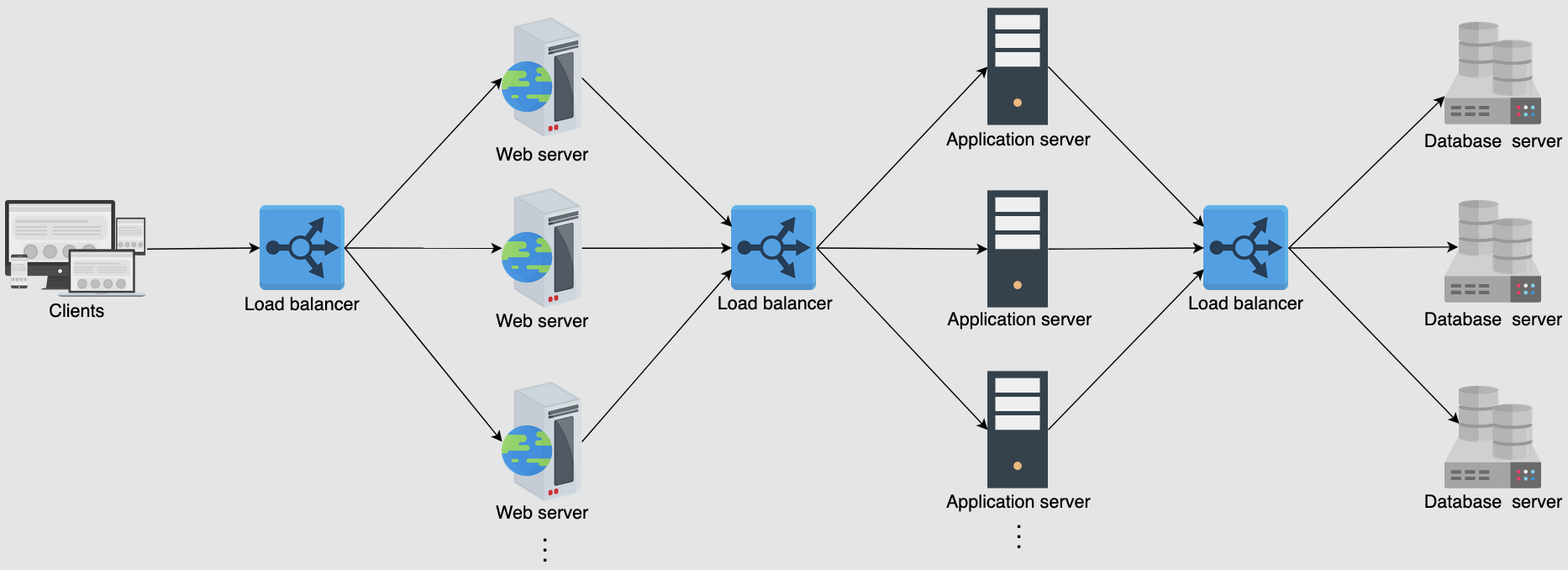
* Load balancers distribute incoming network traffic across multiple servers to ensure no single server bears too much load. This helps increase reliability and availability of applications by rerouting live traffic from failed servers to healthy ones and by balancing the load efficiently.

**Key Functions:**

* **Scalability**: Facilitates easy addition or removal of servers without downtime.
* **Availability**: Maintains application availability through automatic failure handling.
* **Performance**: Enhances user experience by reducing server load, which decreases response time.
* **Security**: Protects against direct attacks on the backend server and enhances overall security posture.

**Placement in System Architecture:**

* Placed between client devices and the servers hosting applications, or between different layers of the application stack (e.g., between application and database servers).



LBs not only enable services to be scalable, available, and highly performant, they offer some key services like the following:

* **Health checking**: LBs use the heartbeat protocol to monitor the health and, therefore, reliability of end-servers. Another advantage of health checking is the improved user experience.
* **TLS termination**: LBs reduce the burden on end-servers by handling TLS termination with the client.
* **Predictive analytics**: LBs can predict traffic patterns through analytics performed over traffic passing through them or using statistics of traffic obtained over time.
* **Reduced human intervention**: Because of LB automation, reduced system administration efforts are required in handling failures.
* **Service discovery**: An advantage of LBs is that the clients’ requests are forwarded to appropriate hosting servers by inquiring about the service registry.
* **Security**: LBs may also improve security by mitigating attacks like denial-of-service (DoS) at different layers of the OSI model (layers 3, 4, and 7).

### **2. Global and Local Load Balancing**

**Global Server Load Balancing (GSLB):**

* Distributes user requests across multiple data centers based on factors like geography, server health, and server capacity. GSLB improves application reliability and user experience by directing traffic to the nearest or most optimal location.

**Local Load Balancing:**

* Operates within a data center to manage traffic across servers. Ensures efficient use of server resources and improves overall system performance by distributing incoming requests based on current load and server health.

Round-robin in DNS forwards clients to data centers in a strict circular order. However, round-robin has the following limitations:

* Different ISPs have a different number of users. An ISP serving many customers will provide the same cached IP to its clients, resulting in uneven load distribution on end-servers.
* Because the round-robin load-balancing algorithm doesn’t consider any end-server crashes, it keeps on distributing the IP address of the crashed servers until the TTL of the cached entries expires. Availability of the service, in that case, can take a hit due to DNS-level load balancing.

Despite its limitations, round-robin is still widely used by many DNS service providers. Furthermore, DNS uses short TTL for cached entries to do effective load balancing among different data centers.

If the session information isn’t kept at a lower layer (distributed cache or database), load balancers are used to keep the session information. Below, we describe two ways of handling session maintenance through LBs:

* Stateful
* Stateless

#### **Stateful load balancing**

#### As the name indicates, **stateful load balancing** involves maintaining a state of the sessions established between clients and hosting servers. The stateful LB incorporates state information in its algorithm to perform load balancing.

Essentially, the stateful LBs retain a data structure that maps incoming clients to hosting servers. Stateful LBs increase complexity and limit scalability because session information of all the clients is maintained across all the load balancers. That is, load balancers share their state information with each other to make forwarding decisions.

#### **Stateless load balancing**

**Stateless load balancing** maintains no state and is, therefore, faster and lightweight. Stateless LBs use consistent hashing to make forwarding decisions. However, if infrastructure changes (for example, a new application server joining), stateless LBs may not be as resilient as stateful LBs because consistent hashing alone isn’t enough to route a request to the correct application server. Therefore, a local state may still be required along with consistent hashing.

**DNS and Load Balancing:**

* DNS can be used for rudimentary load balancing by rotating the order of IP addresses in DNS responses. This method, known as round-robin DNS, distributes client requests across a number of servers to balance the load.

### **3. Advanced Details of Load Balancers**

**Load Balancing Algorithms:**

* **Round-robin scheduling**: In this algorithm, each request is forwarded to a server in the pool in a repeating sequential manner.
* **Weighted round-robin**: If some servers have a higher capability of serving clients’ requests, then it’s preferred to use a weighted round-robin algorithm. In a weighted round-robin algorithm, each node is assigned a weight. LBs forward clients’ requests according to the weight of the node. The higher the weight, the higher the number of assignments.
* **Least connections**: In certain cases, even if all the servers have the same capacity to serve clients, uneven load on certain servers is still a possibility. For example, some clients may have a request that requires longer to serve. Or some clients may have subsequent requests on the same connection. In that case, we can use algorithms like least connections where newer arriving requests are assigned to servers with fewer existing connections. LBs keep a state of the number and mapping of existing connections in such a scenario. We’ll discuss more about state maintenance later in the lesson.
* **Least response time**: In performance-sensitive services, algorithms such as least response time are required. This algorithm ensures that the server with the least response time is requested to serve the clients.
* **IP hash**: Some applications provide a different level of service to users based on their IP addresses. In that case, hashing the IP address is performed to assign users’ requests to servers.
* **URL hash**: It may be possible that some services within the application are provided by specific servers only. In that case, a client requesting service from a URL is assigned to a certain cluster or set of servers. The URL hashing algorithm is used in those scenarios.

Algorithms can be static or dynamic depending on the machine’s state. Let’s look at each of the categories individually:

* **Static algorithms** don’t consider the changing state of the servers. Therefore, task assignment is carried out based on existing knowledge about the server’s configuration. Naturally, these algorithms aren’t complex, and they get implemented in a single router or commodity machine where all the requests arrive.
* **Dynamic algorithms** are algorithms that consider the current or recent state of the servers. Dynamic algorithms maintain state by communicating with the server, which adds a communication overhead. State maintenance makes the design of the algorithm much more complicated. Dynamic algorithms require different load-balancing servers to communicate with each other to exchange information. Therefore, dynamic algorithms can be modular because no single entity will do the decision-making. Although this adds complexity to dynamic algorithms, it results in improved forwarding decisions. Finally, dynamic algorithms monitor the health of the servers and forward requests to active servers only.
* **Note:** In practice, dynamic algorithms provide far better results because they maintain a state of serving hosts and are, therefore, worth the effort and complexity.

### **Tier-0 and tier-1 LBs#**

If DNS can be considered as the tier-0 load balancer, equal cost multipath (ECMP) routers are the tier-1 LBs. From the name of ECMP, it’s evident that this layer divides incoming traffic on the basis of IP or some other algorithm like round-robin or weighted round-robin. Tier-1 LBs will balance the load across different paths to higher tiers of load balancers.

ECMP routers play a vital role in the horizontal scalability of the higher-tier LBs.

### **Tier-2 LBs#**

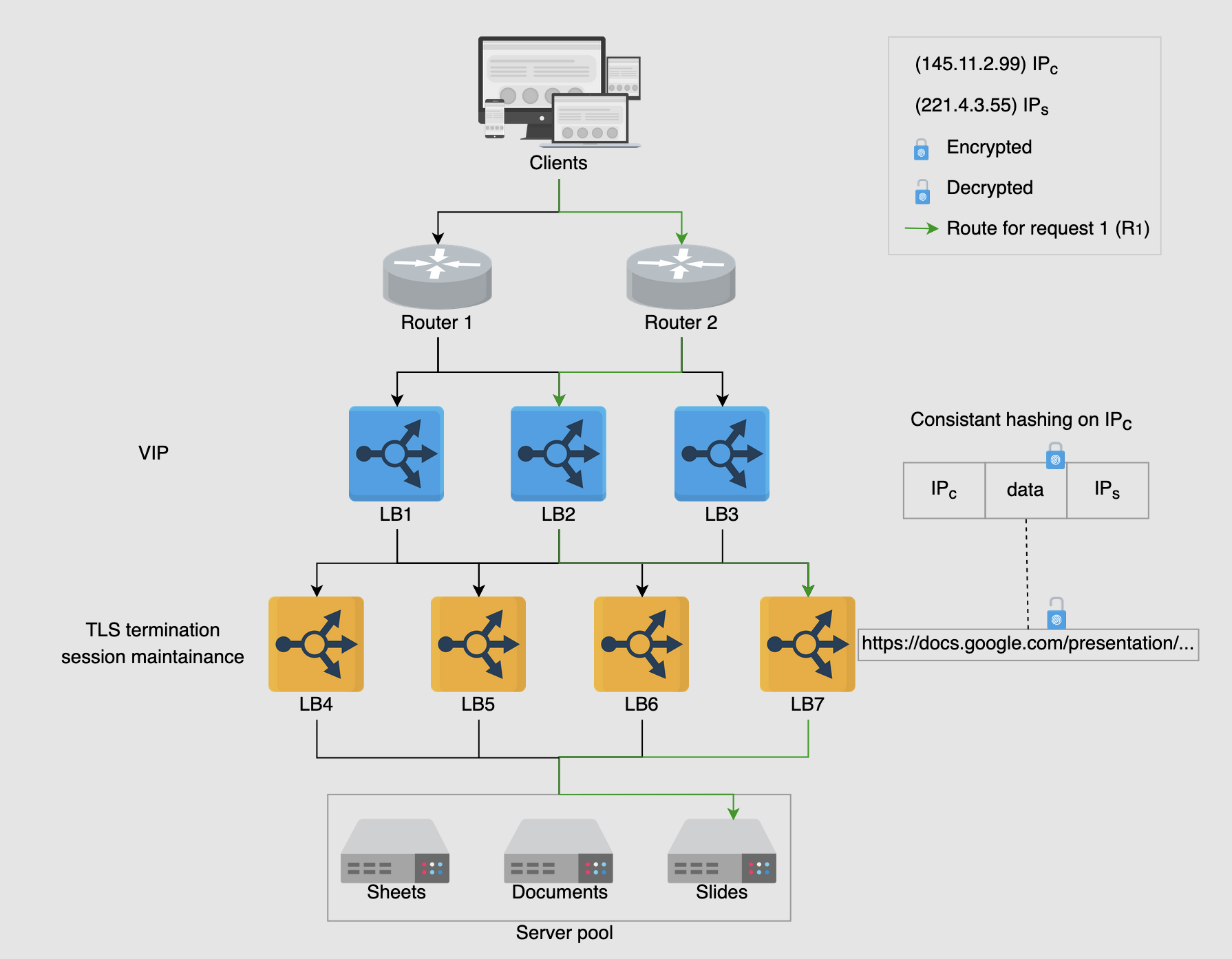
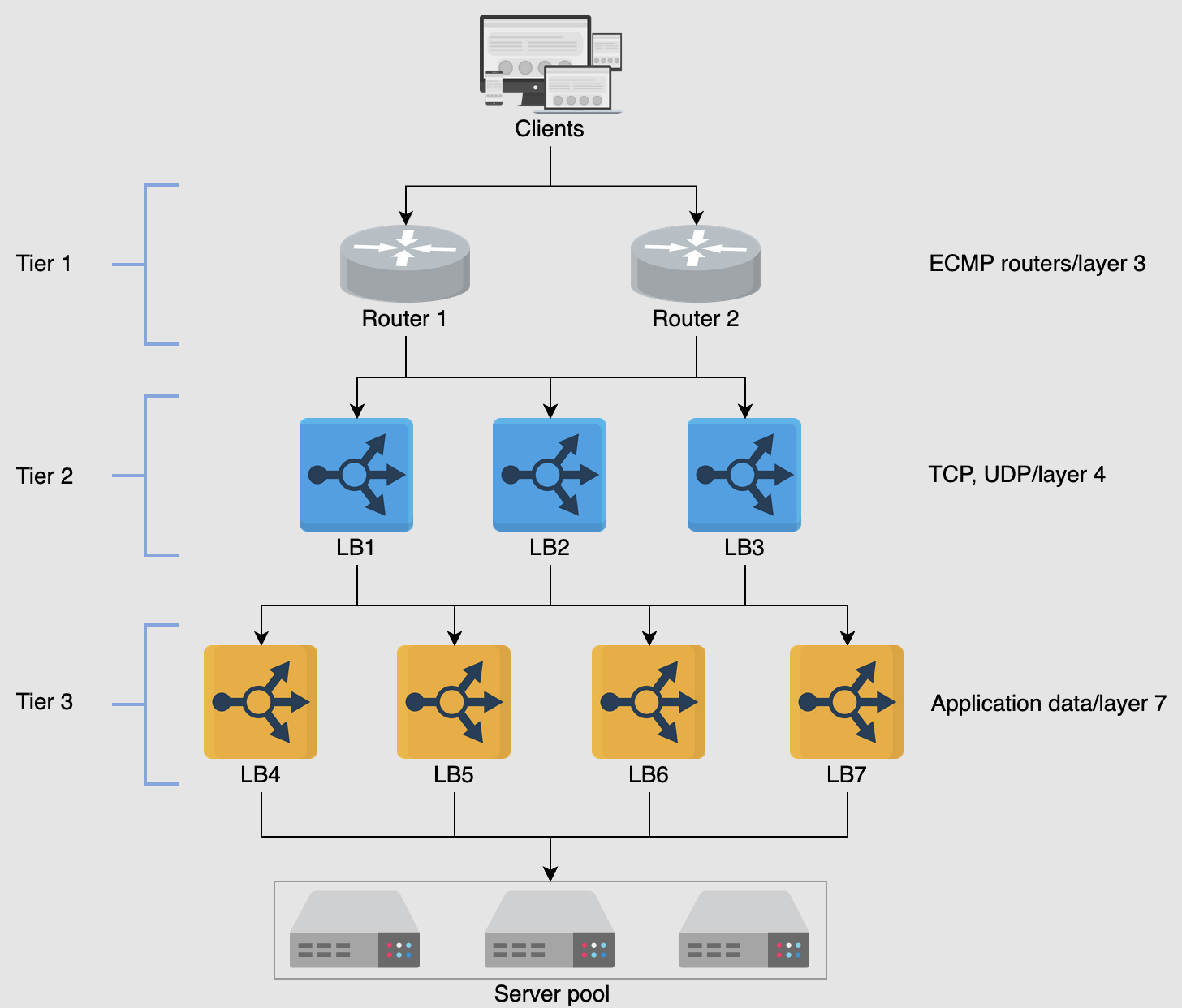
The second tier of LBs include layer 4 load balancers. Tier-2 LBs make sure that for any connection, all incoming packets are forwarded to the same tier-3 LBs. To achieve this goal, a technique like consistent hashing can be utilized. But in case of any changes to the infrastructure, consistent hashing may not be enough. Therefore, we have to maintain a local or global state as we’ll see in the coming sections of the lesson.

Tier-2 load balancers can be considered the glue between tier-1 and tier-3 LBs. Excluding tier-2 LBs could result in erroneous forwarding decisions in case of failures or dynamic scaling of LBs.

### **Tier-3 LBs#**

Layer 7 LBs provide services at tier 3. Since these LBs are in direct contact with the back-end servers, they perform health monitoring of servers at HTTP level. This tier enables scalability by evenly distributing requests among the set of healthy back-end servers and provides high availability by monitoring the health of servers directly. This tier also reduces the burden on end-servers by handling low-level details like TCP-congestion control protocols, the discovery of Path MTU (maximum transmission unit), the difference in application protocol between client and back-end servers, and so on. The idea is to leave the computation and data serving to the application servers and effectively utilize load balancing commodity machines for trivial tasks. In some cases, layer 7 LBs are at the same level as the service hosts.

To summarize, tier 1 balances the load among the load balancers themselves. Tier 2 enables a smooth transition from tier 1 to tier 3 in case of failures, whereas tier 3 does the actual load balancing between back-end servers. Each tier performs other tasks to reduce the burden on end-servers.



### Different kinds of load balancers can be implemented depending on the number of incoming requests, organization, and application-specific requirements:

### **Hardware load balancers#**

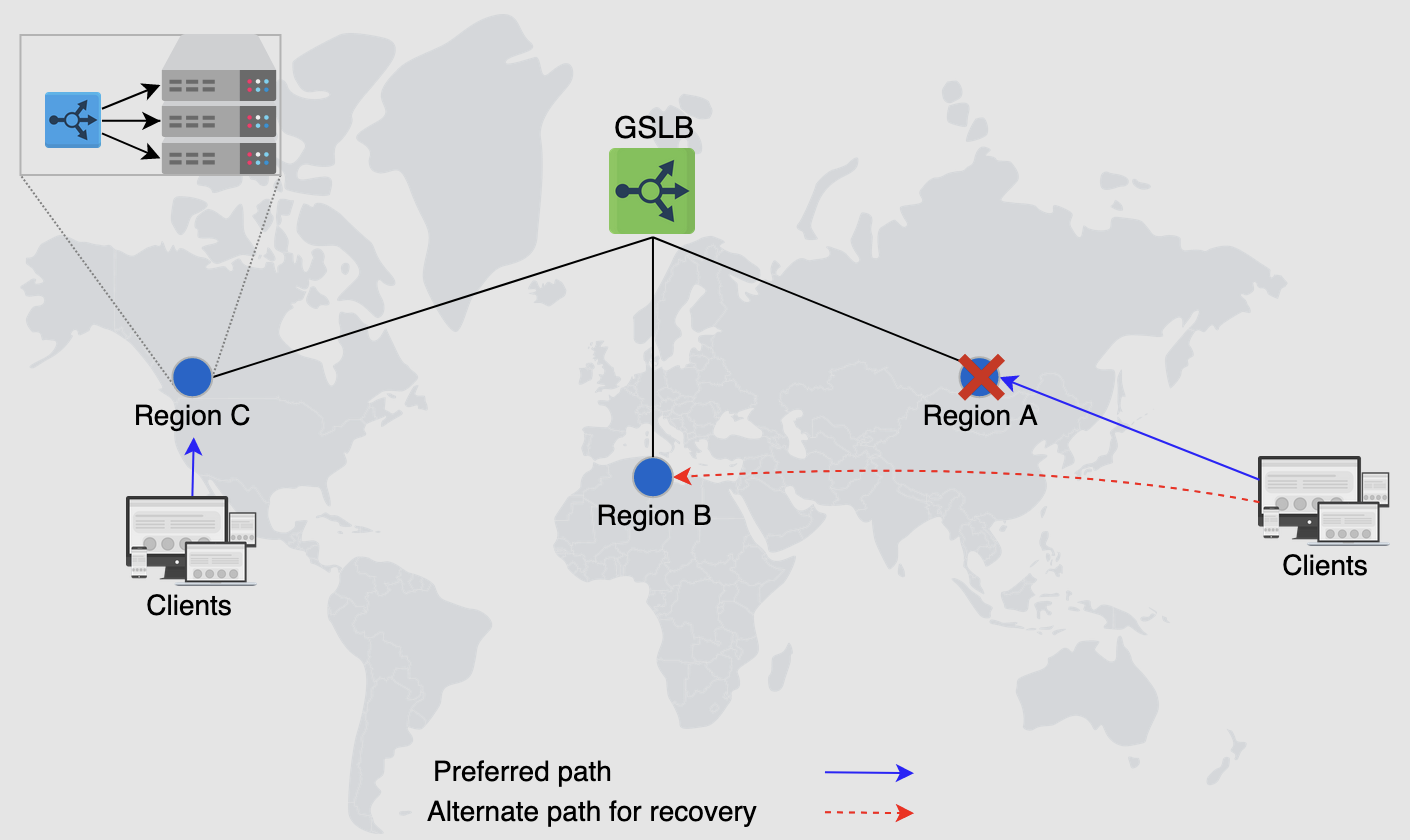
* Load balancers were introduced in the 1990s as hardware devices. Hardware load balancers work as stand-alone devices and are quite expensive. Nonetheless, they have their performance benefits and are able to handle a lot of concurrent users. Configuration of hardware-based solutions is problematic because it requires additional human resources. Therefore, they aren’t the go-to solutions even for large enterprises that can afford them. Availability can be an issue with hardware load balancers because additional hardware will be required to failover to in case of failures. Finally, hardware LBs can have higher maintenance/operational costs and compatibility issues making them less flexible. Not to mention that hardware LBs have vendor locks as well.

### **Software load balancers#**

* Software load balancers are becoming increasingly popular because of their flexibility, programmability, and cost-effectiveness. That’s all possible because they’re implemented on commodity hardware. Software LBs scale well as requirements grow. Availability won’t be an issue with software LBs because small additional costs are required to implement shadow load balancers on commodity hardware. Additionally, software LBs can provide predictive analysis that can help prepare for future traffic patterns.

### **Cloud load balancers**

* With the advent of the field of cloud computing, Load Balancers as a Service (LBaaS) has been introduced. This is where cloud owners provide load balancing services. Users pay according to their usage or the service-level agreement (SLA) with the cloud provider. Cloud-based LBs may not necessarily replace a local on-premise load balancing facility, but they can perform global traffic management between different zones. Primary advantages of such load balancers include ease of use, management, metered cost, flexibility in terms of usage, auditing, and monitoring services to improve business decisions. An example of how cloud-based LBs can provide GSLB is given below:



GSLB is obtained through LBaaS, and the regions contain data centers that are the property of application providers

**Note:** Another interesting implementation of load balancers comes in the form of **client-side load balancing**. Client-side load balancing is suited where there are numerous services, each with many instances (such as [load balancing in Twitter](https://www.educative.io/collection/page/10370001/4941429335392256/5379128533975040)). Our focus, however, is on traditional load balancers because most three-tier applications employ these in their design.