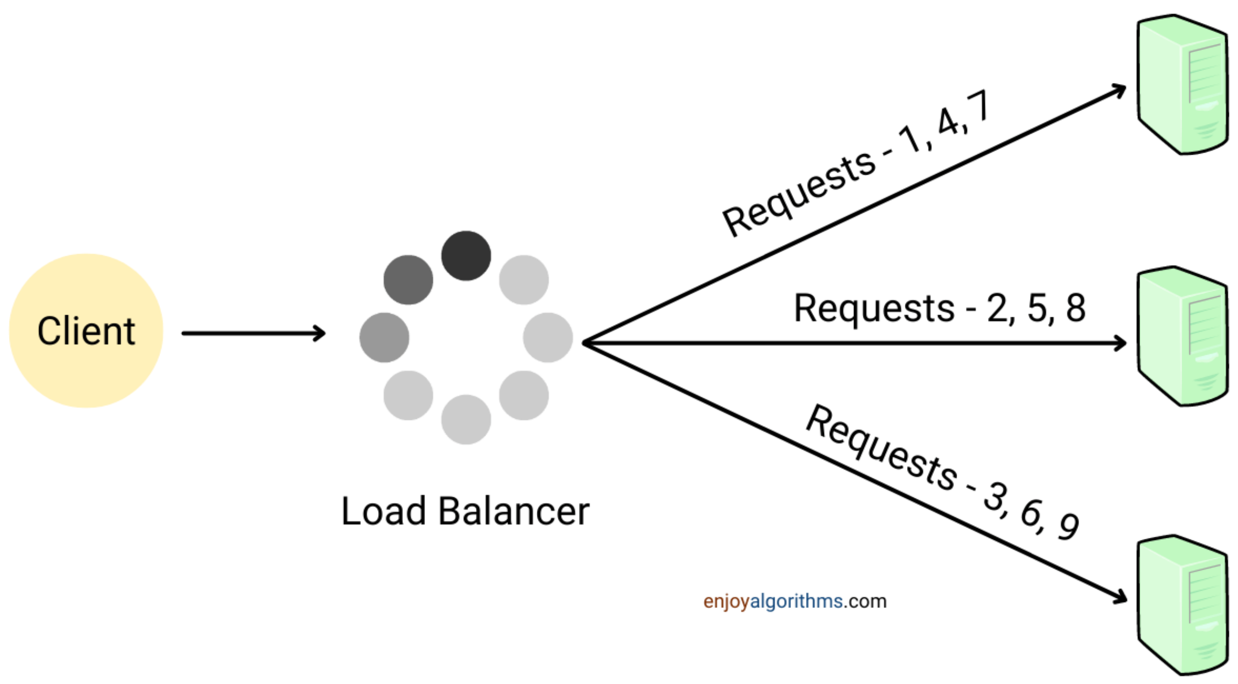
Different Types of Load Balancing Algorithms

Load balancers can distribute requests to specific servers based on various load balancing techniques that use different algorithms to select servers based on a particular configuration. Load balancing algorithms can be divided into two categories:

- **Static load balancing algorithms:** These algorithms work the same way regardless of the state of the backend server serving the requests. They are simpler and more efficient to implement, but they can lead to an uneven distribution of requests. Examples of static load balancing algorithms include **round robin**, **weighted round robin**, **source IP hash**, **URL hash**, **randomized algorithm,** etc.
- **Dynamic load balancing algorithms:** These algorithms take into account the state of the backend server and consider server load when distributing requests. They require communication between the load balancers and servers, so they are slightly more complex but can distribute requests efficiently. Examples of dynamic load balancing algorithms include the **least connection method**, **weighted least connections method**, **least response time method,** etc.

Round Robin Load Balancing Algorithm

Round robin is a simple load balancing algorithm that directs client requests to different servers based on a rotating list. The load balancer maintains a list of available servers and directs incoming requests in a round-robin fashion: the first request goes to the first server, the second request goes to the second server, and so on. When the load balancer reaches the end of the list, it goes back to the beginning and starts from the first server again.



Java example code of round robin scheduling

```
class Server {
  private final String address;
  public Server(String address) {
    this.address = address;
  }

  public String getAddress() {
    return address;
  }
}

class LoadBalancer {
  private static int currentServer = 0;
  private final List<Server> servers = new ArrayList<>();

  public void addServer(Server server) {
    servers.add(server);
  }

  public void removeServer(Server server) {
    servers.remove(server);
  }
  public Server nextServer() {
    Server server = servers.get(currentServer);
    currentServer = (currentServer + 1) % servers.size();
    return server;
  }
}

public class Main {
  public static void main(String[] args) {
    LoadBalancer loadBalancer = new LoadBalancer();
    loadBalancer.addServer(new Server("Server 1"));
    loadBalancer.addServer(new Server("Server 2"));
    loadBalancer.addServer(new Server("Server 3"));

    for (int i = 0; i < 11; i++) {
      Server server = loadBalancer.nextServer();
      System.out.println("Request " + (i + 1) + " directed to " + server.ge
    }
  }
}
```

Output

```
Request 1 directed to Server 1
Request 2 directed to Server 2
Request 3 directed to Server 3
Request 4 directed to Server 1
Request 5 directed to Server 2
Request 6 directed to Server 3
Request 7 directed to Server 1
Request 8 directed to Server 2
Request 9 directed to Server 3
Request 10 directed to Server 1
Request 11 directed to Server 2
```

Benefits and drawbacks of the round robin algorithm include:

- It is easy to implement.
- It evenly balances traffic between servers.
- It does not consider the server's load or specifications, so there is a risk that a server with low capacity will receive a large number of requests and become overloaded.
- It works best if every server in the load balancer list has roughly the same specification. Otherwise, a low processing server may have the same load as a high processing server.

Weighted Round Robin Balancing Algorithm

The weighted round-robin load-balancing algorithm is a more advanced version of the simple round-robin algorithm. It distributes incoming requests based on the weighted score of the servers. The weight can be an integer that reflects the server's processing power or specifications. This allows the algorithm to consider server specifications when distributing traffic.

Some benefits and drawbacks of the weighted round-robin algorithm include:

- It is slightly more complex than the simple round-robin algorithm, but it works well with servers with different specifications.
- It does not consider the current load of each server or the relative computation cost of each request.
- Based on the weighted score, some servers may receive a higher proportion of the overall request count.

Random Load Balancing Algorithm

- The randomized load-balancing algorithm randomly distributes requests to servers using a random number generator. When a load balancer receives a request, the randomized algorithm evenly distributes it to the servers. Like the round robin algorithm, this algorithm works well for a group of servers with similar configurations.

Java example code for random load balancing

```java
class Server {
    private String name;

    public Server(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}

class LoadBalancer {
    private List<Server> servers = new ArrayList<>();
    private Random random = new Random();

    public void addServer(Server server) {
        servers.add(server);
    }

    public Server getRandomServer() {
        int index = random.nextInt(servers.size());
        return servers.get(index);
    }
}

public class Main {
    public static void main(String[] args) {
        LoadBalancer loadBalancer = new LoadBalancer();
        loadBalancer.addServer(new Server("Server1"));
        loadBalancer.addServer(new Server("Server2"));
        loadBalancer.addServer(new Server("Server3"));

        for (int i = 0; i < 10; i++) {
            Server server = loadBalancer.getRandomServer();
            System.out.println("Request" + i + " assigned to " + server.get
        }
    }
}
```

Source IP Hash Load Balancing Algorithm

The source IP hash load-balancing algorithm selects a server based on a unique hash key. It generates the hash key by combining the source and destination IP addresses and allocates the request to a specific server. If the session is broken, the key can be regenerated and the client request can be directed back to the same server that was originally handling it. In other words, this algorithm is useful when a dropped connection needs to be returned to the same server.

URL Hash Load Balancing Algorithm

The URL hash load-balancing algorithm generates a hash value based on the URL present in client requests. The requests are forwarded to servers based on the hash value. The load balancer caches the hashed value of the URL, so subsequent requests that use the same URL result in a cache hit and are forwarded to the same server.

URL hash algorithm is useful when load-balanced servers serve mostly unique content per server. In other words, all requests related to one process (e.g., running code) will go to one server, and all requests related to another process (e.g., payments) will go to another server, and so on.

Least Connection Method

The least connection load-balancing algorithm considers the current load on a server and aims to improve performance. It does this by sending requests to the server with the least number of active connections.

Some benefits and drawbacks of the least connection algorithm include:

- The load balancer does additional calculations to determine the server with the least number of connections.
- It is useful when there are many persistent connections in the traffic

that are unevenly distributed between the servers. If the servers are busy with long computations, the connections between client and server stay alive for a long period of time.
- The additional calculations required to determine the server with the least number of connections may increase the load on the load balancer itself.

Weighted Least Connections Method

The weighted least connections load-balancing algorithm distributes load based on both the number of current and active connections to each server and the relative capacity of the server.

- It takes into account that some servers can handle more connections than others.
- Servers are rated based on their processing capabilities, allowing for a more fine-grained distribution of load.

Least Response Time Method

The least response time load-balancing algorithm is a more advanced version of the least connection method. It forwards requests to the server with the fewest active connections and the least average response time.

- It relies on the time taken by a server to respond to a health monitoring request, using the speed of the response as an indicator of the server's load.
- It also considers the number of active connections on each server.
- The backend server that responds the fastest receives the subsequent request.
- It may require additional calculations to determine the server with the least average response time.

Note: Our system might have multiple load balancers that use different server selection strategies. Enjoy learning, enjoy system design!