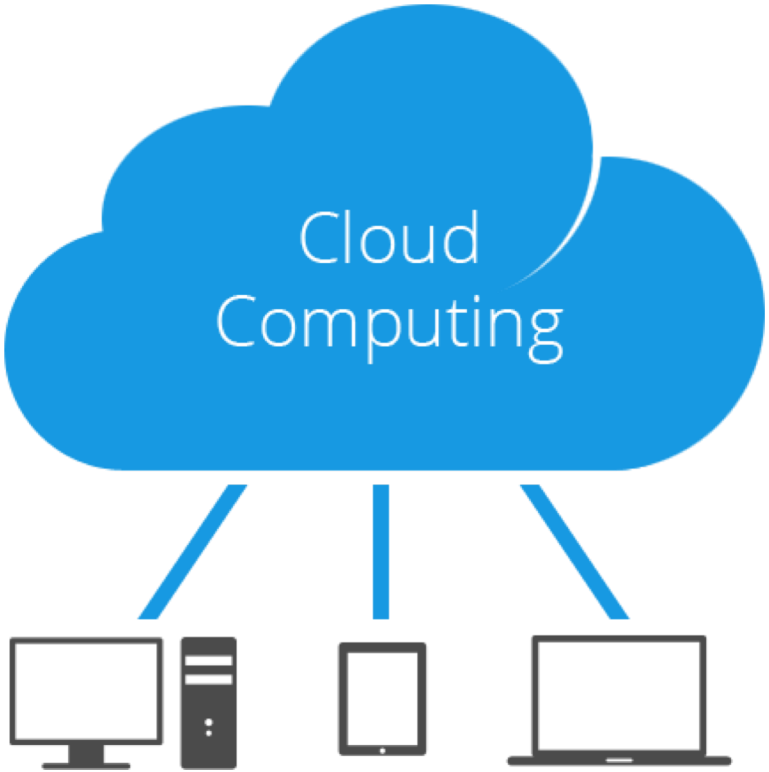
Load Balancing in Cloud Computing

Nicholas M. Plucker
2019 CSCI Senior Seminar
Division of Science and Mathematics
University of Minnesota Morris
Morris, Minnesota USA

October 24, 2019

The Big Picture

- 3.6 billions users estimated to access a cloud service in 2018 [1]
- Public cloud revenue estimated to reach $278.3 billion in 2021, compared to $145.3 billion in 2017 [2]
- Demand is increasing everyday
- Load balancing improves the likelihood that cloud services run efficiently and uninterrupted



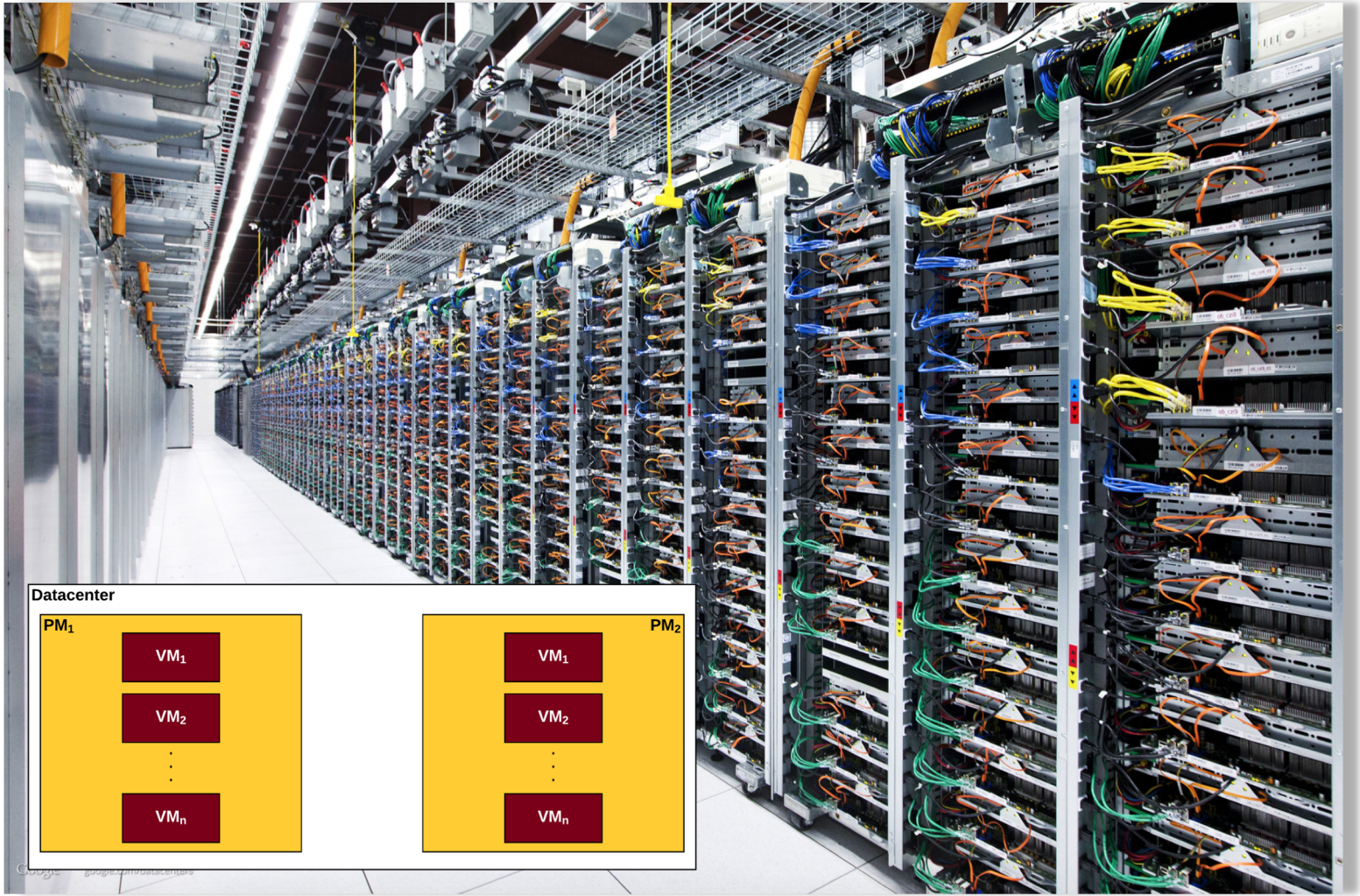
Outline

1. Background
2. Length Based Weighted Round Robin Algorithm
3. Honey Bee Behavior Inspired Load Balancing
4. Conclusions

Outline

1. **Background**
   a. **What is Cloud Computing?**
   b. **What is Load Balancing?**
   c. **Round Robin Algorithm**
2. Length Based Weighted Round Robin Algorithm
3. Honey Bee Behavior Inspired Load Balancing
4. Conclusions

What is Cloud Computing?


Datacenter
PM1
VM1
VM2
VMn
PM2
VM1
VM2
VMn

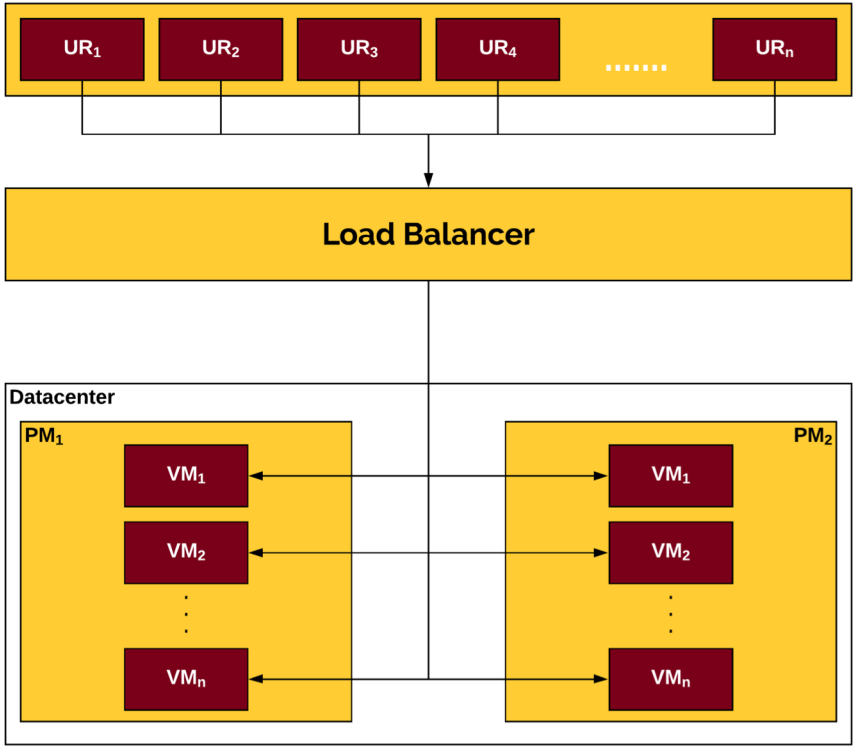
What is Cloud Computing?

- Uses computing resources to provide a wide range of services over the internet
- Content is located in a data center that is connected to the internet
- Content is delivered to the user through a client, such as a web browser

What is Load Balancing?

- Process of distributing traffic to multiple servers to improve reliability and performance
- Increased reliability is achieved by routing incoming traffic away from inactive servers
- Improved performance is achieved by balancing incoming traffic across multiple healthy servers so no one server is congested with traffic

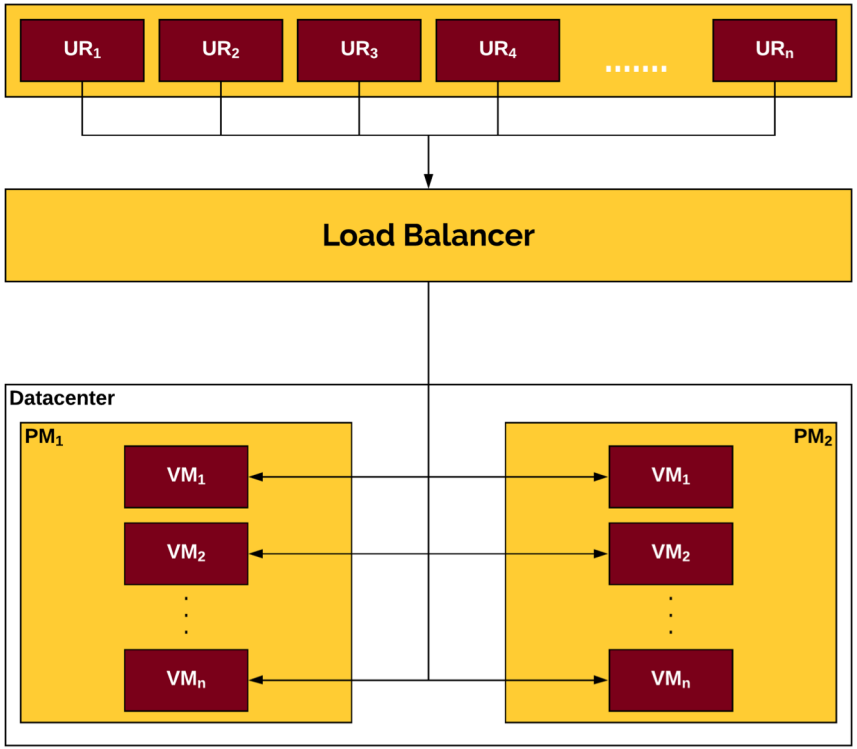
Load Balancing System Architecture



UR = User Request
PM = Physical Machine
VM = Virtual Machine

Load balancing model based on [3]

Load Balancing System Architecture

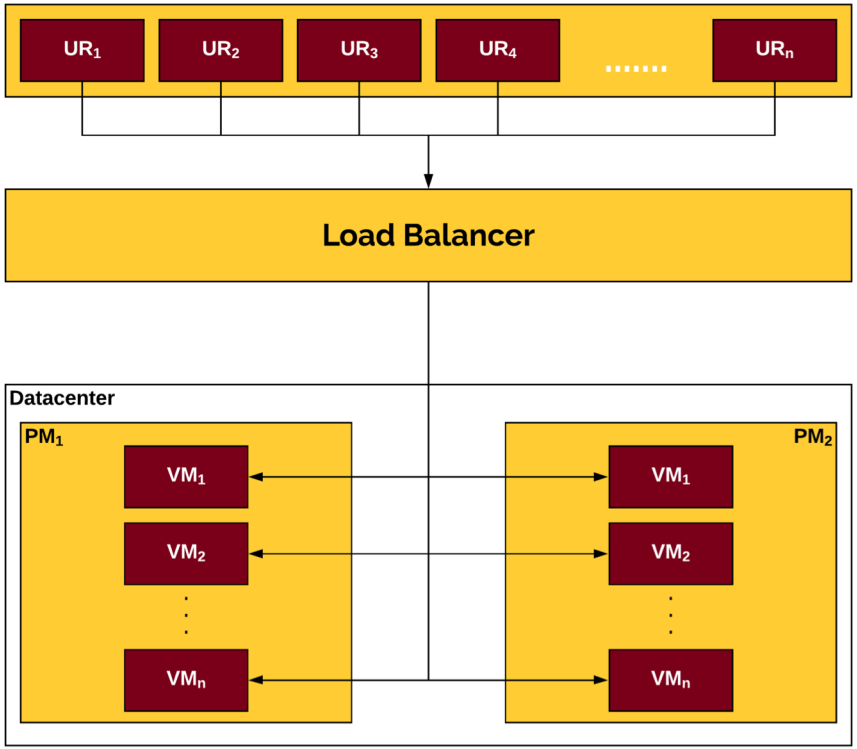


Users make requests to cloud service

UR = User Request
PM = Physical Machine
VM = Virtual Machine

Load balancing model based on [3]

Load Balancing System Architecture



Users make requests to cloud service

Allocates requests to Virtual Machines (VM) for processing

UR = User Request
PM = Physical Machine
VM = Virtual Machine

Load balancing model based on [3]

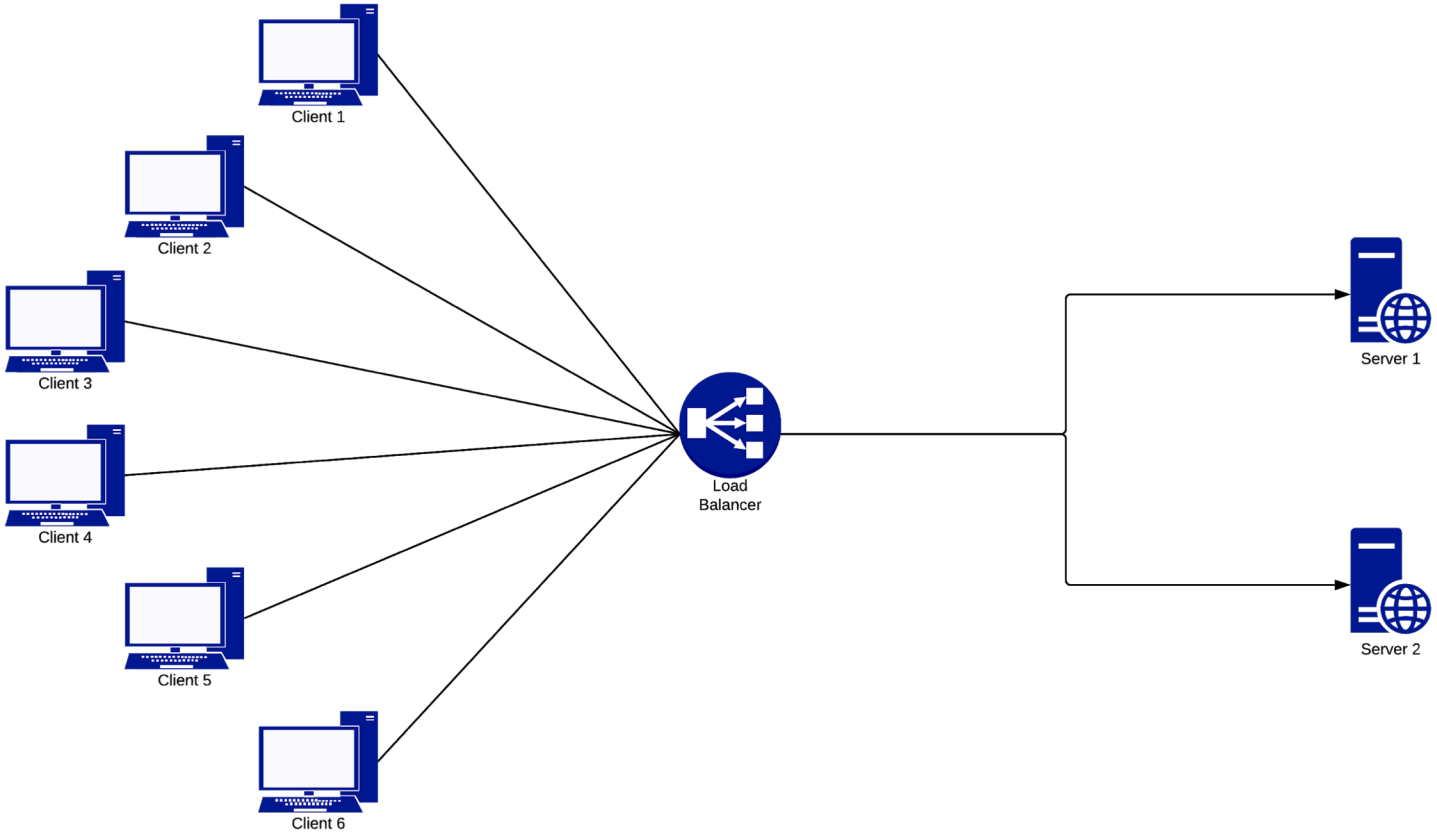
How does Load Balancing work?

- Two Types
  - Static Load Balancing
    - Information about the system state is gathered **before** run-time
    - May lead to uneven distribution of resources [3]
  - Dynamic Load Balancing
    - Information about the system state is updated **during** run-time
    - Allows transferring of tasks from overloaded to underloaded machine [3]
- Preemptive vs non-preemptive
  - Preemptive: tasks are executed for a given amount of time
  - Non-preemptive: tasks are executed until they finish

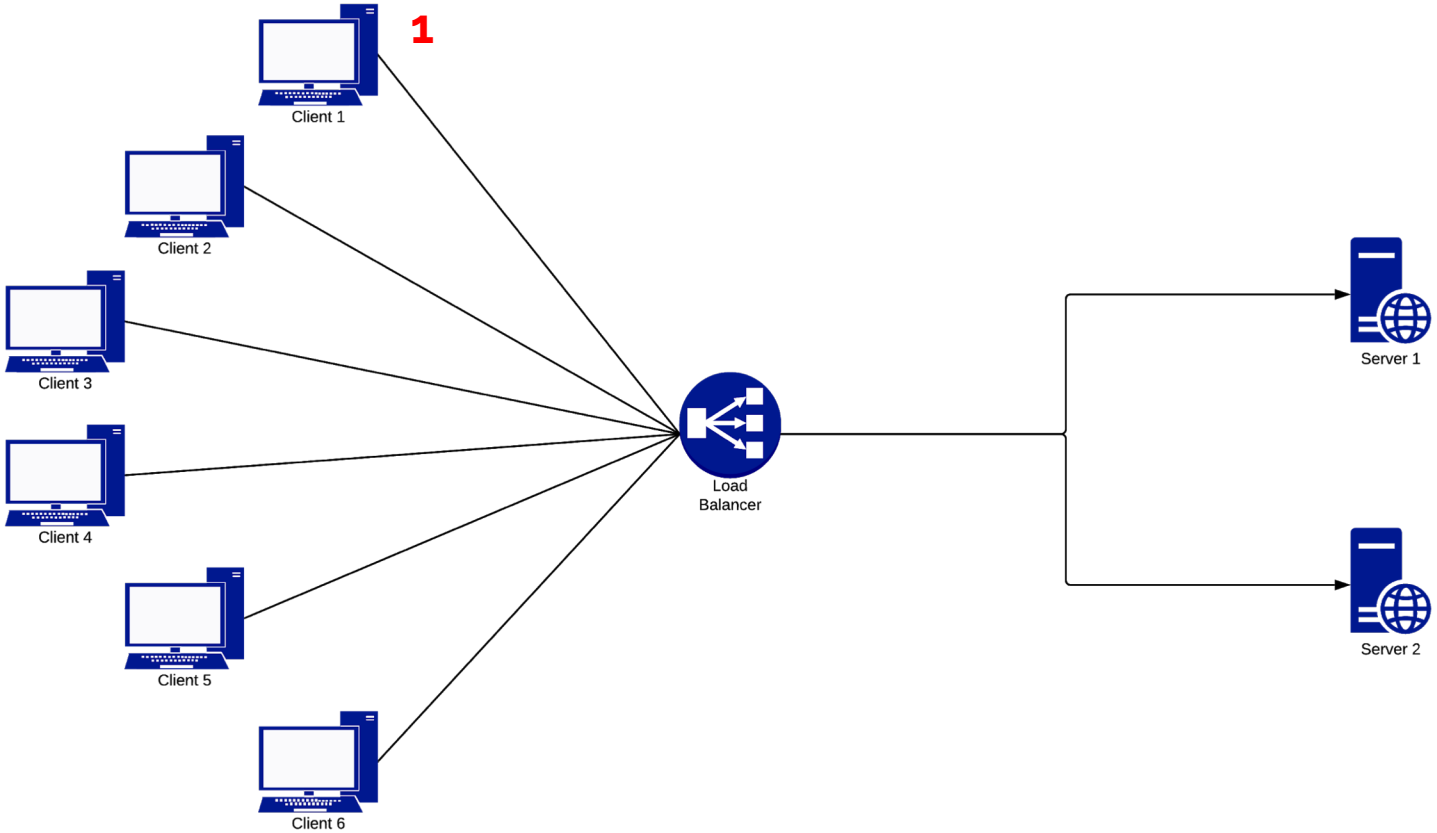
Round Robin Algorithm

- Most widespread static load balancing algorithm [3]
- Tasks are assigned to VM's in a cyclical manner
- Suitable for homogeneous environments

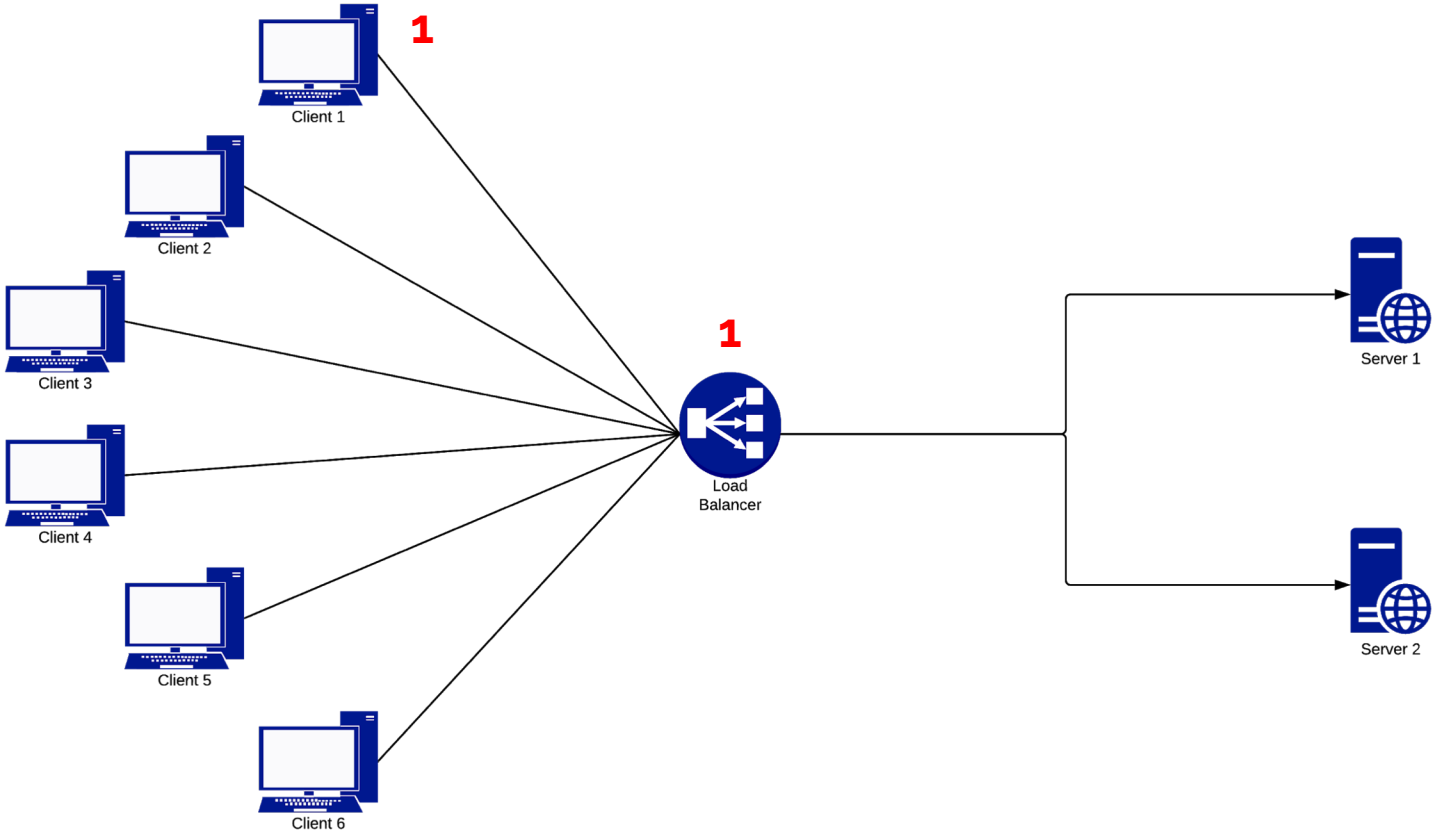
Round Robin Algorithm



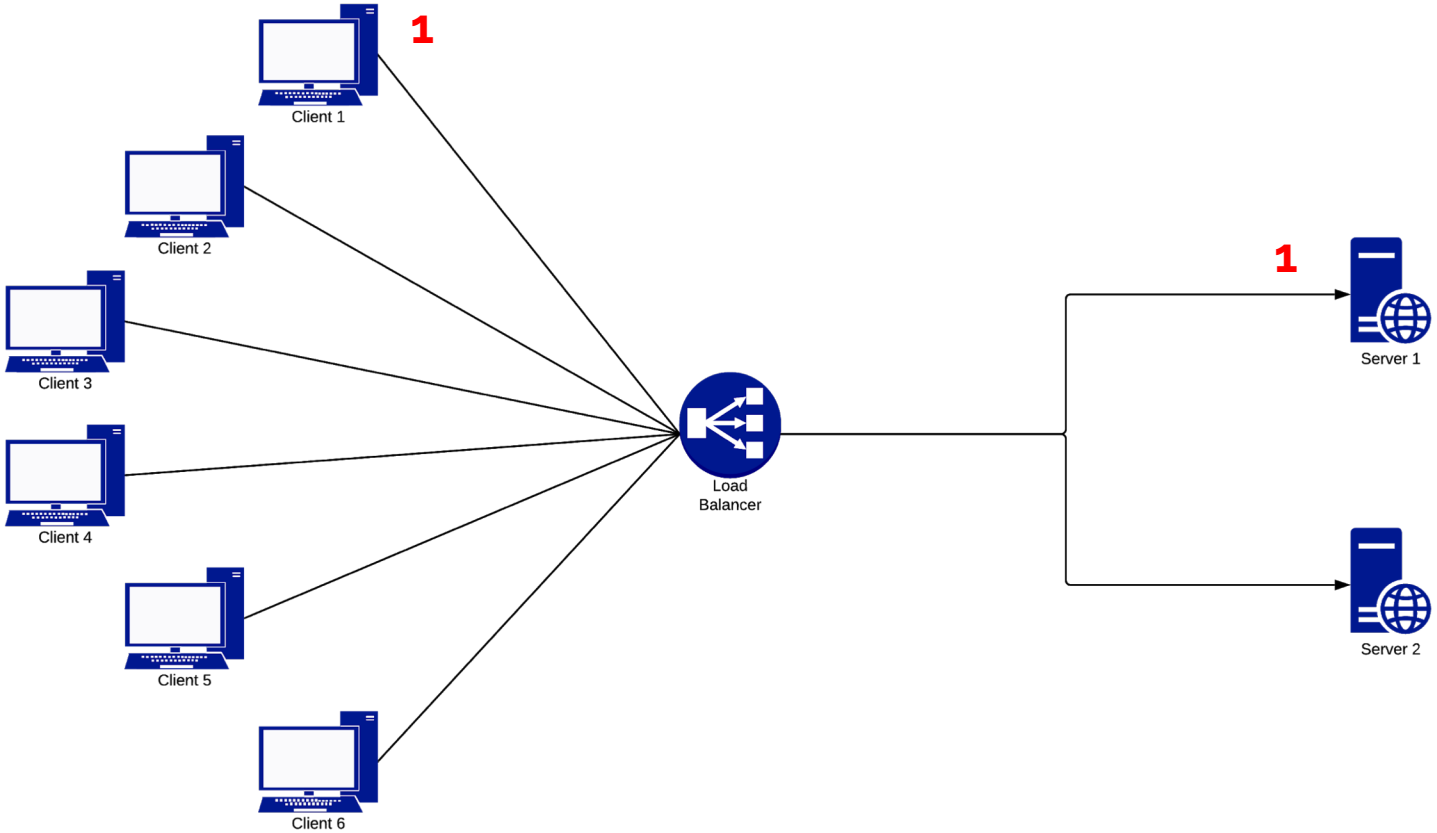
Round Robin Algorithm



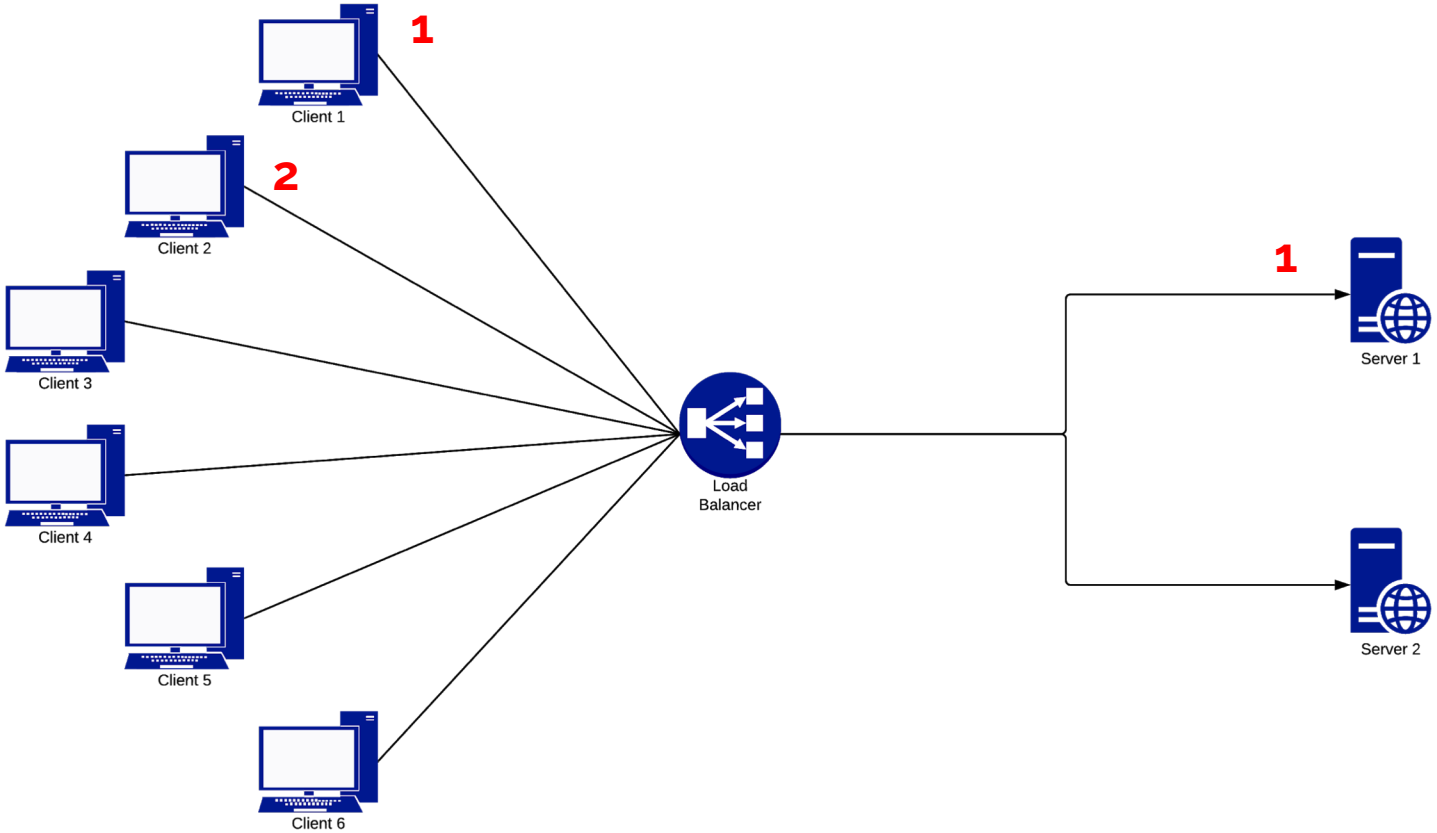
Round Robin Algorithm



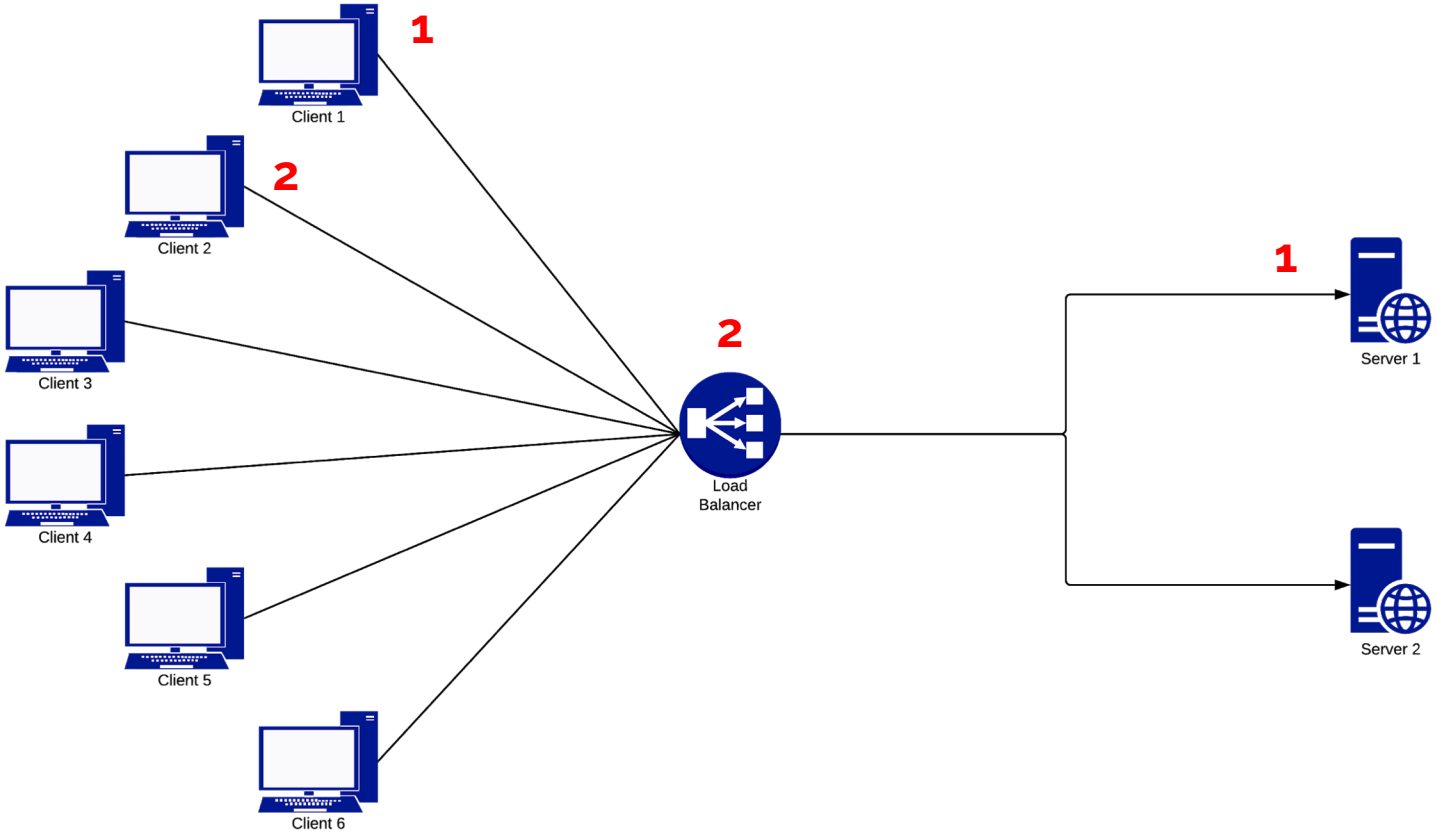
Round Robin Algorithm



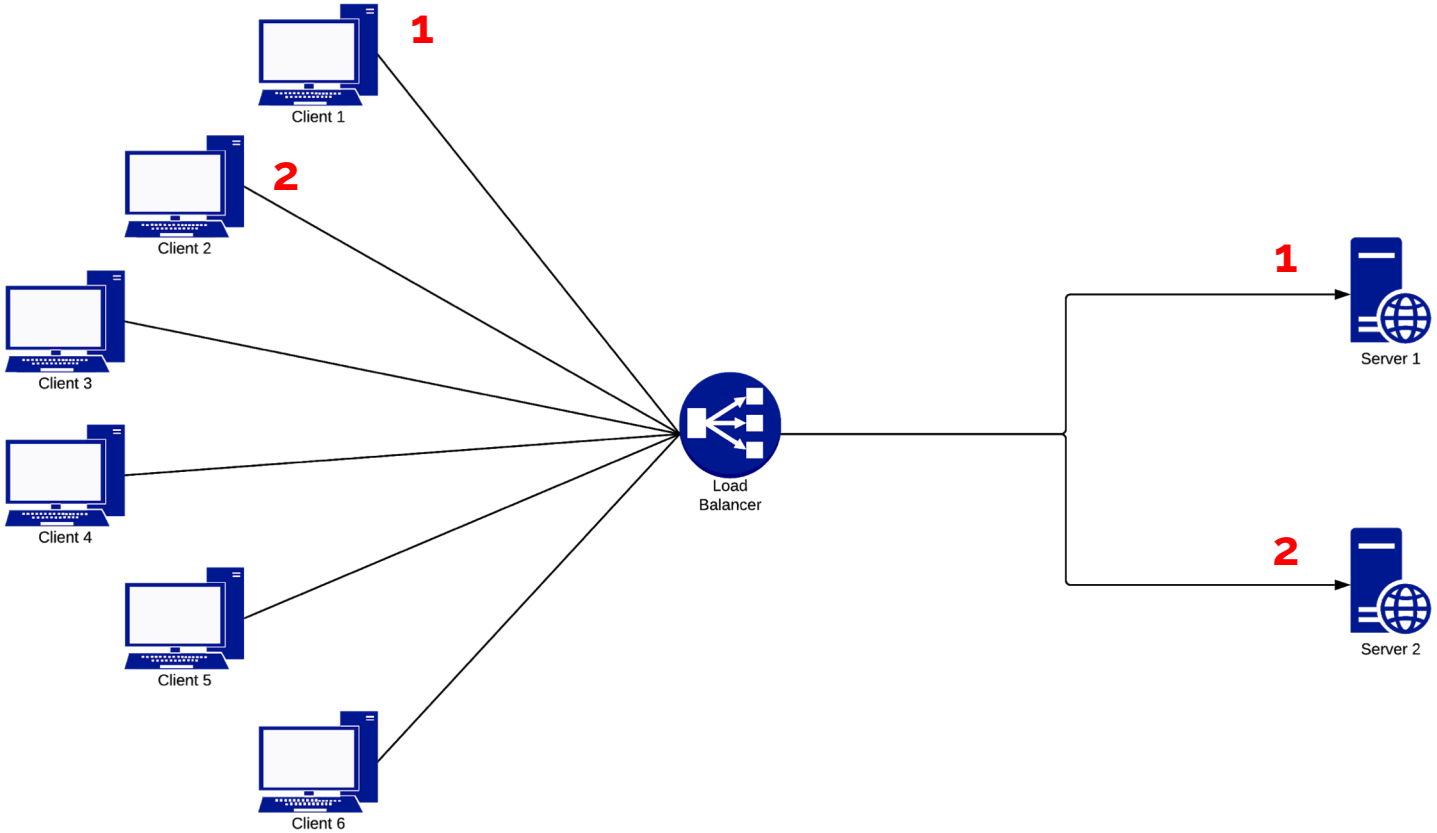
Round Robin Algorithm



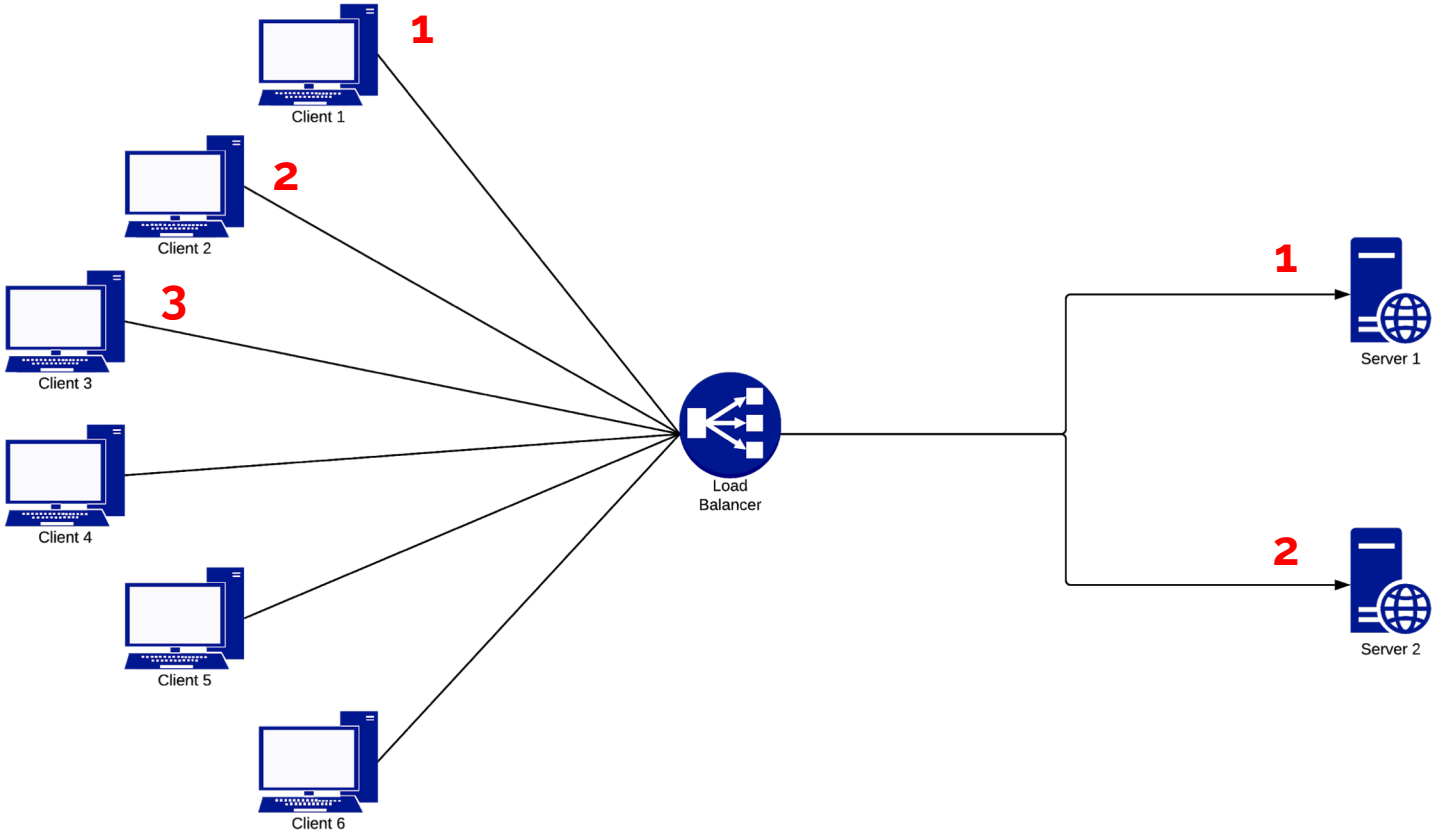
Round Robin Algorithm



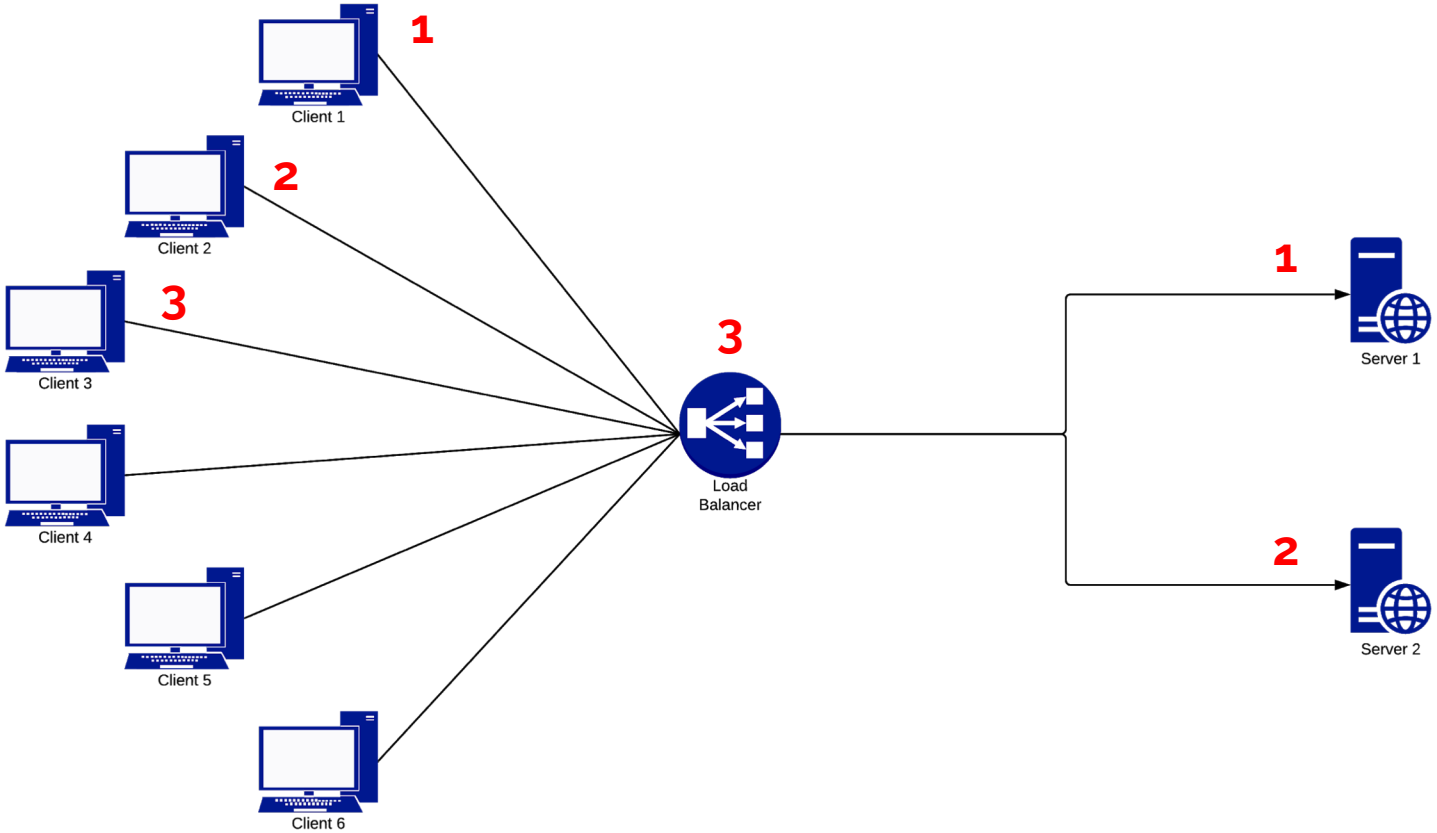
Round Robin Algorithm



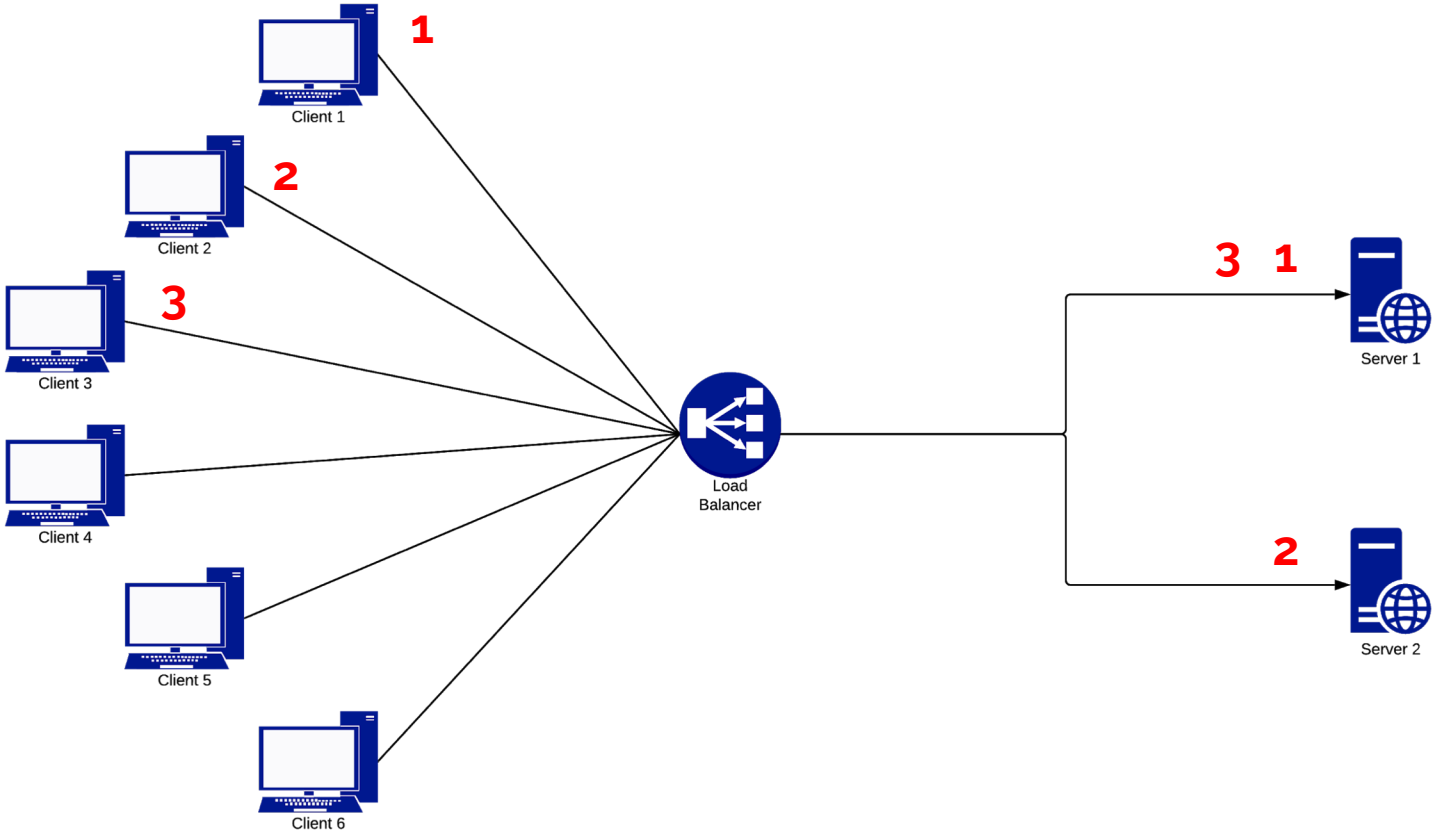
Round Robin Algorithm



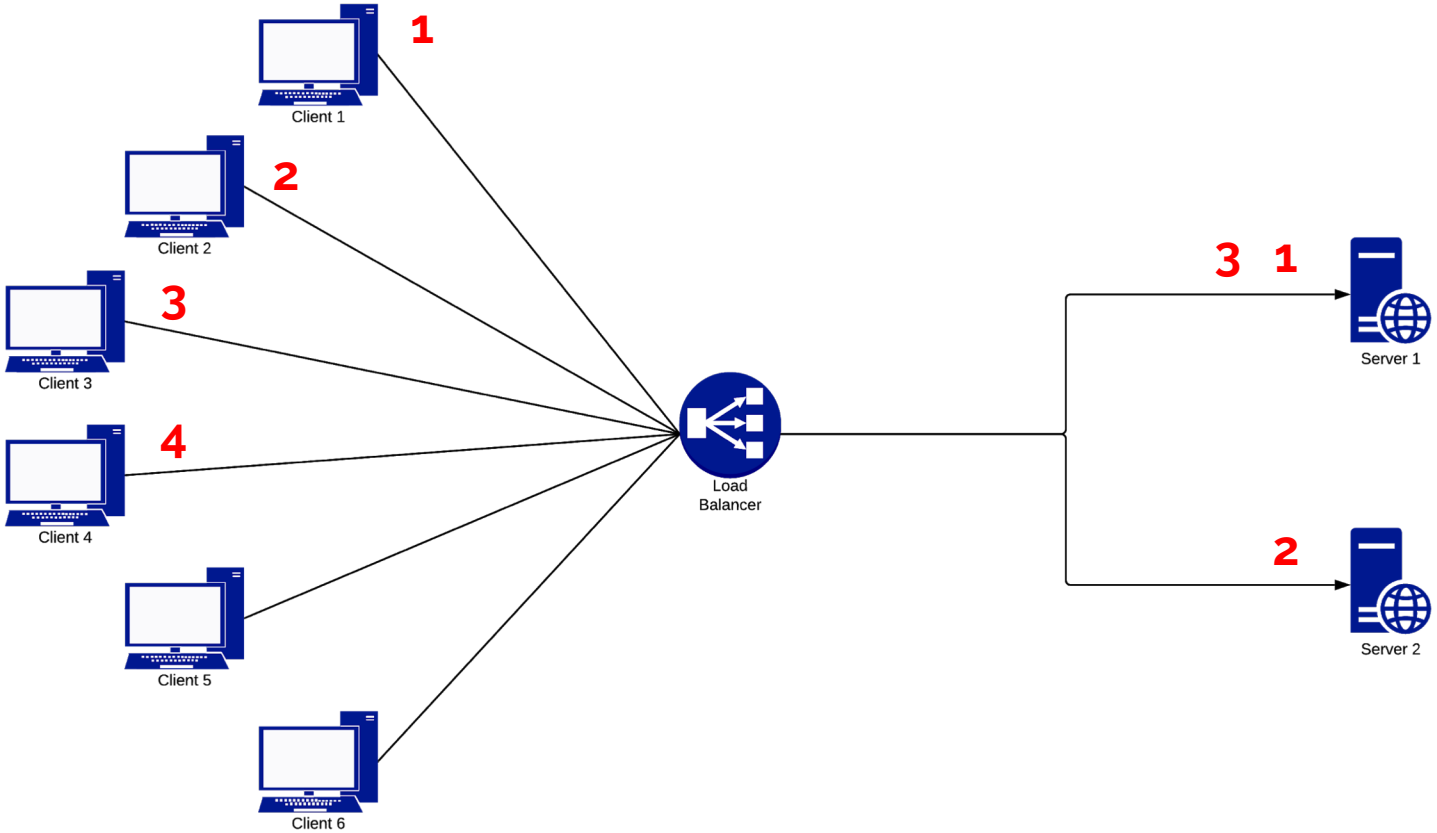
Round Robin Algorithm



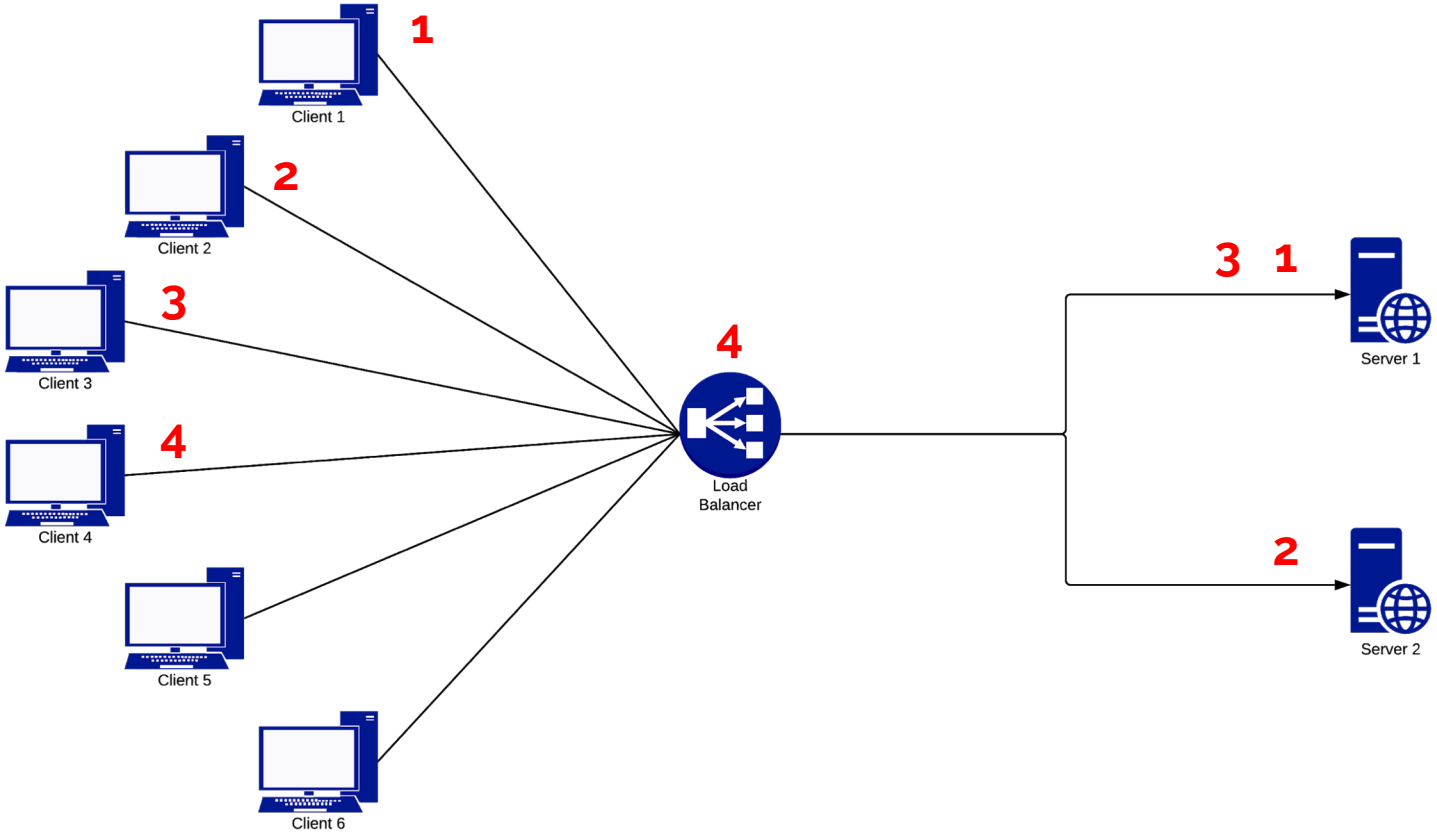
Round Robin Algorithm



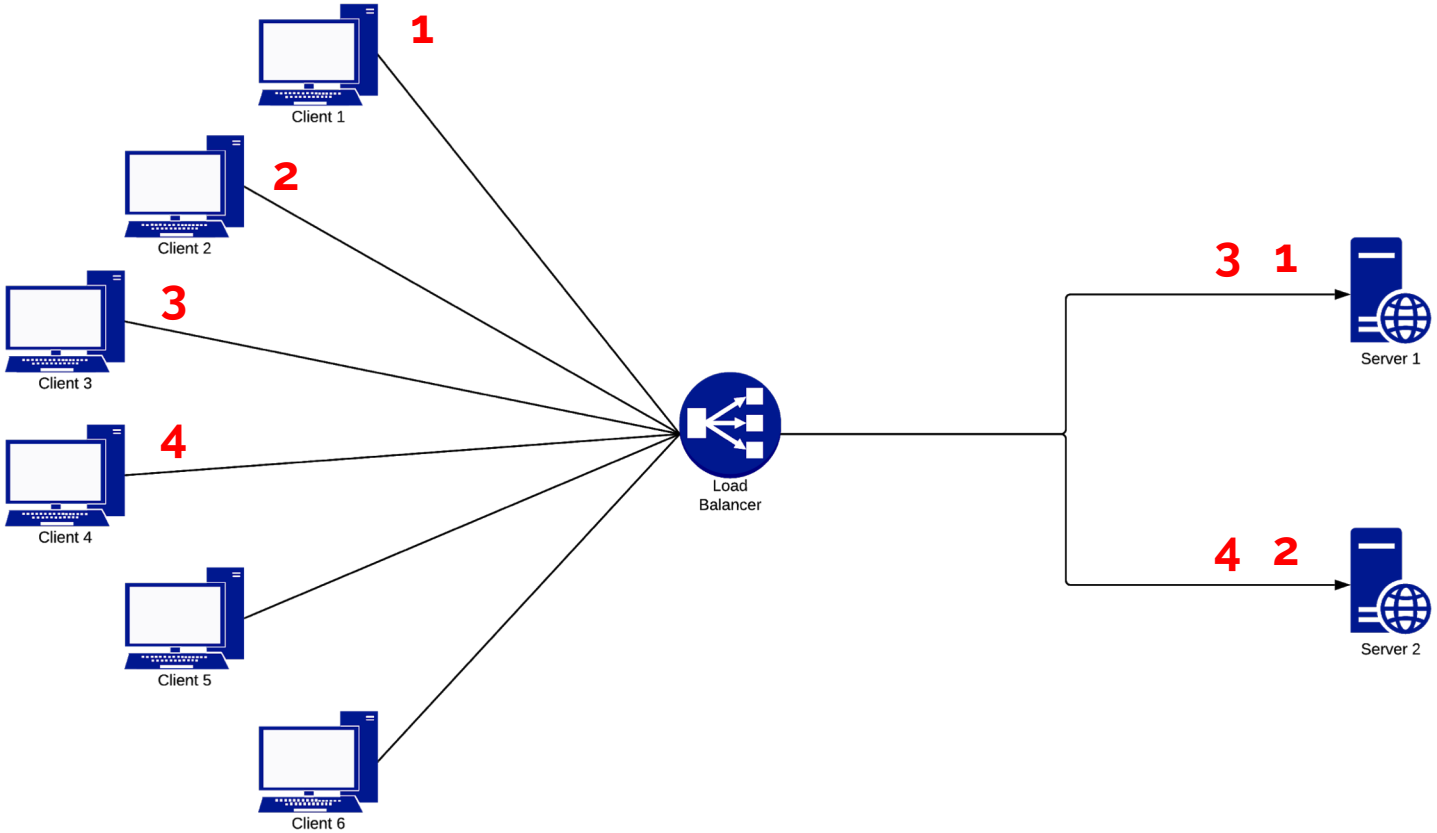
Round Robin Algorithm



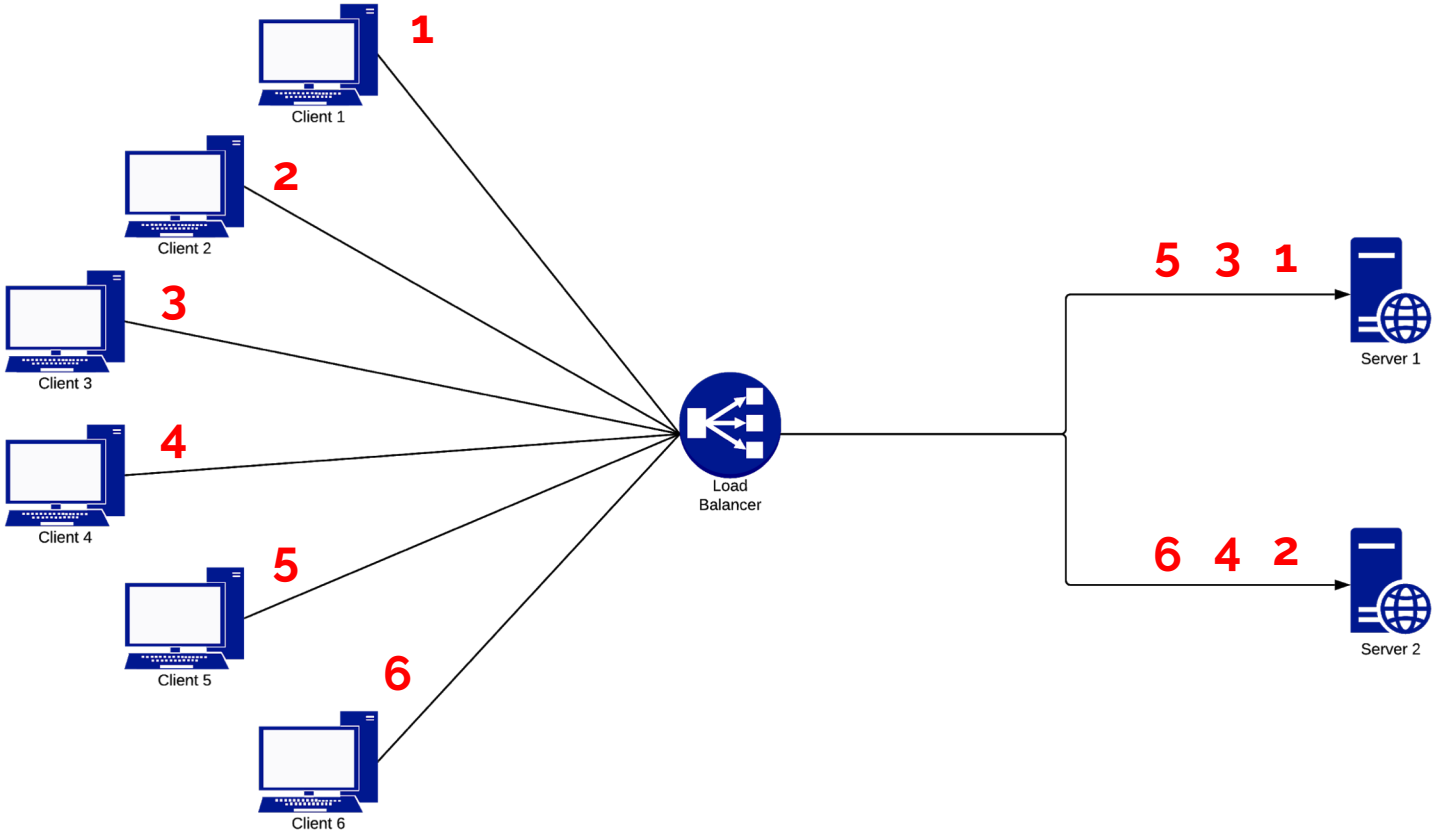
Round Robin Algorithm



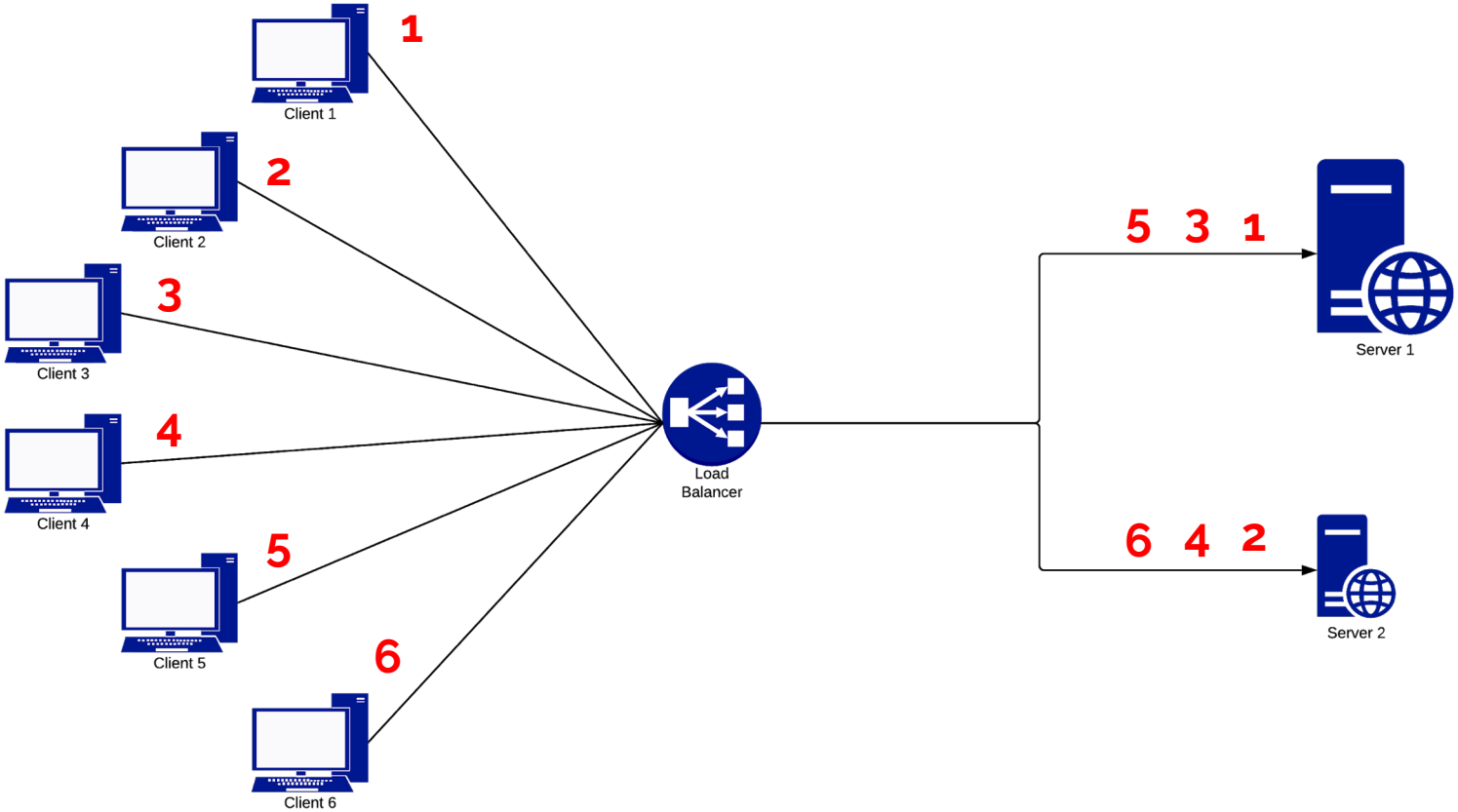
Round Robin Algorithm



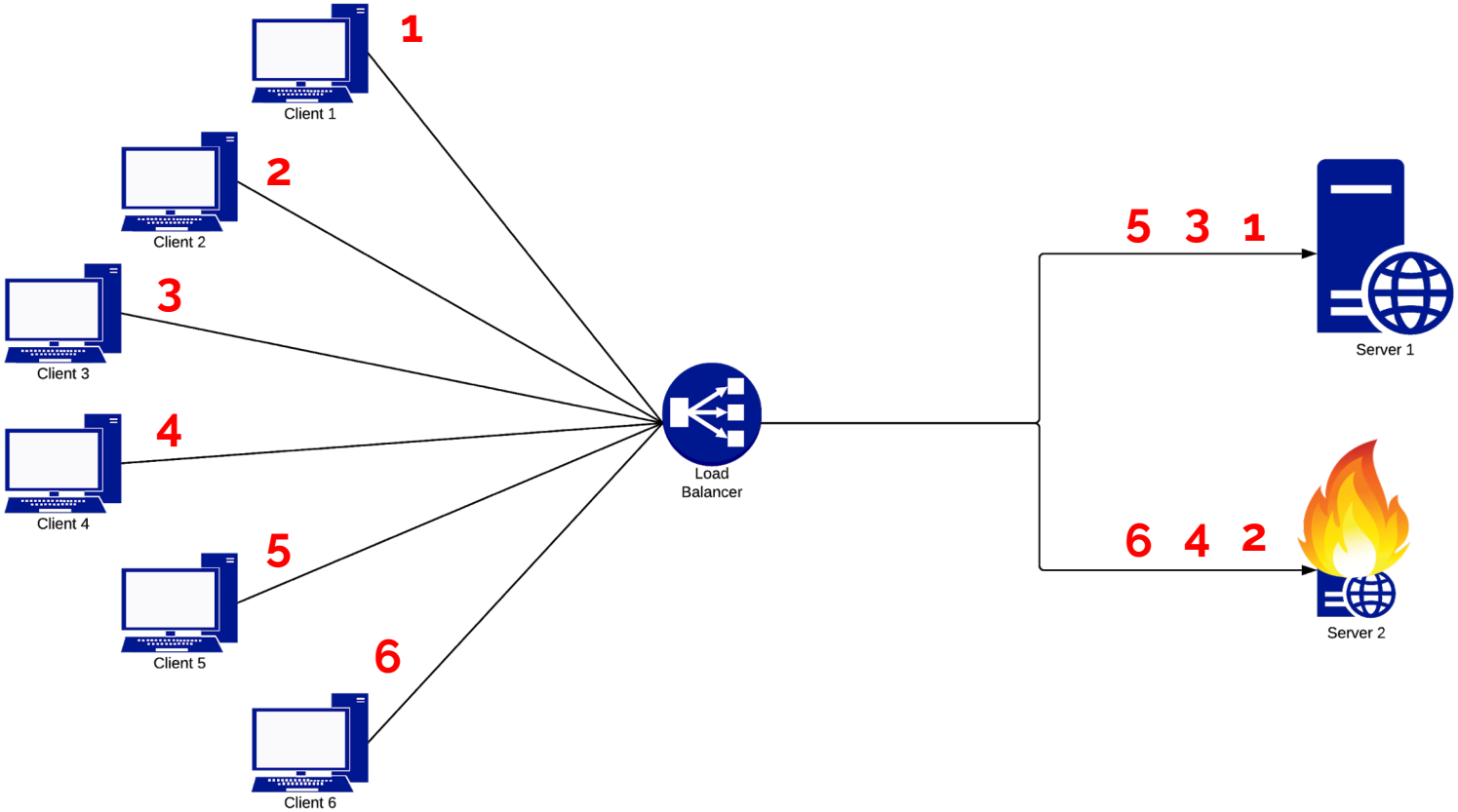
Round Robin Algorithm



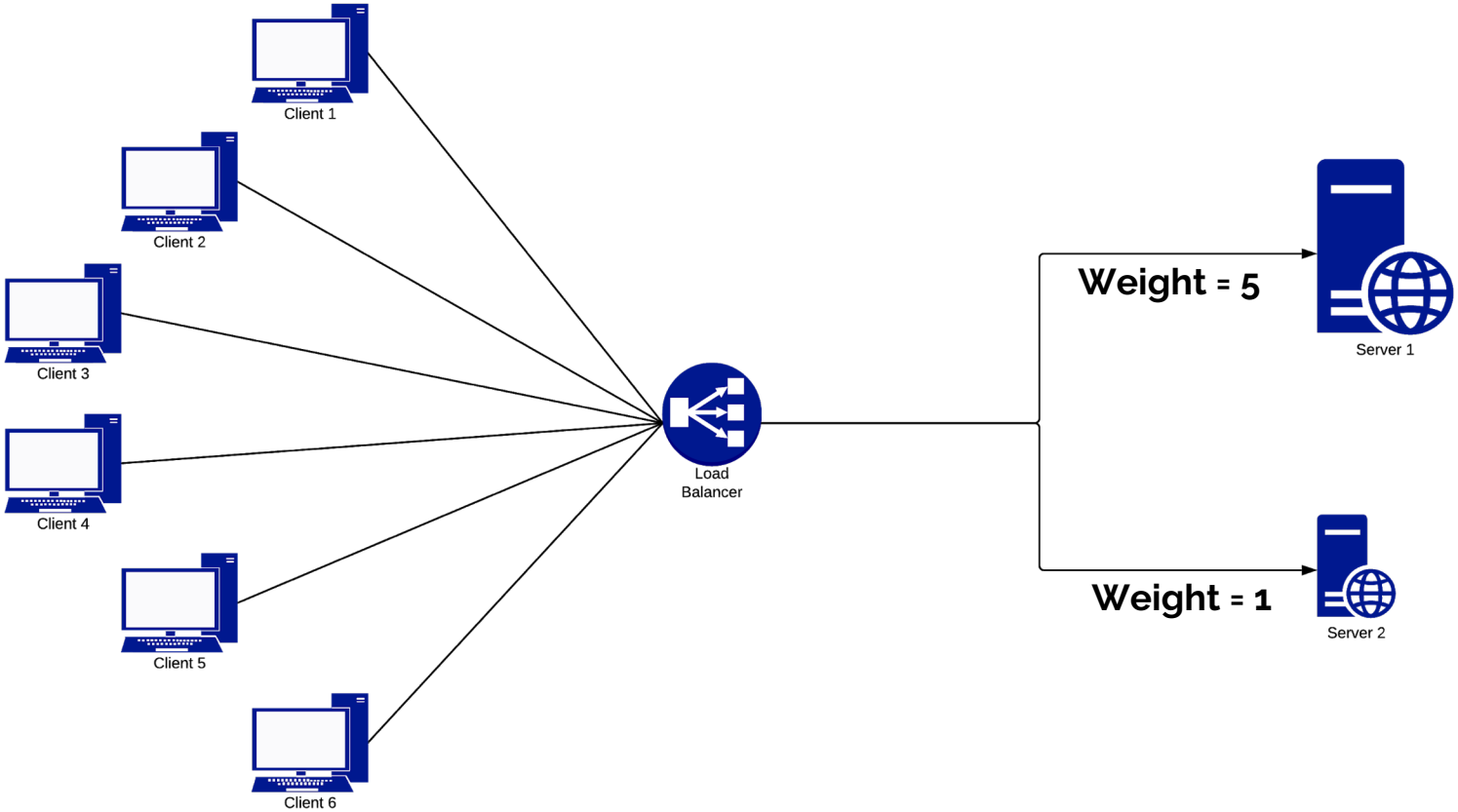
Round Robin Algorithm



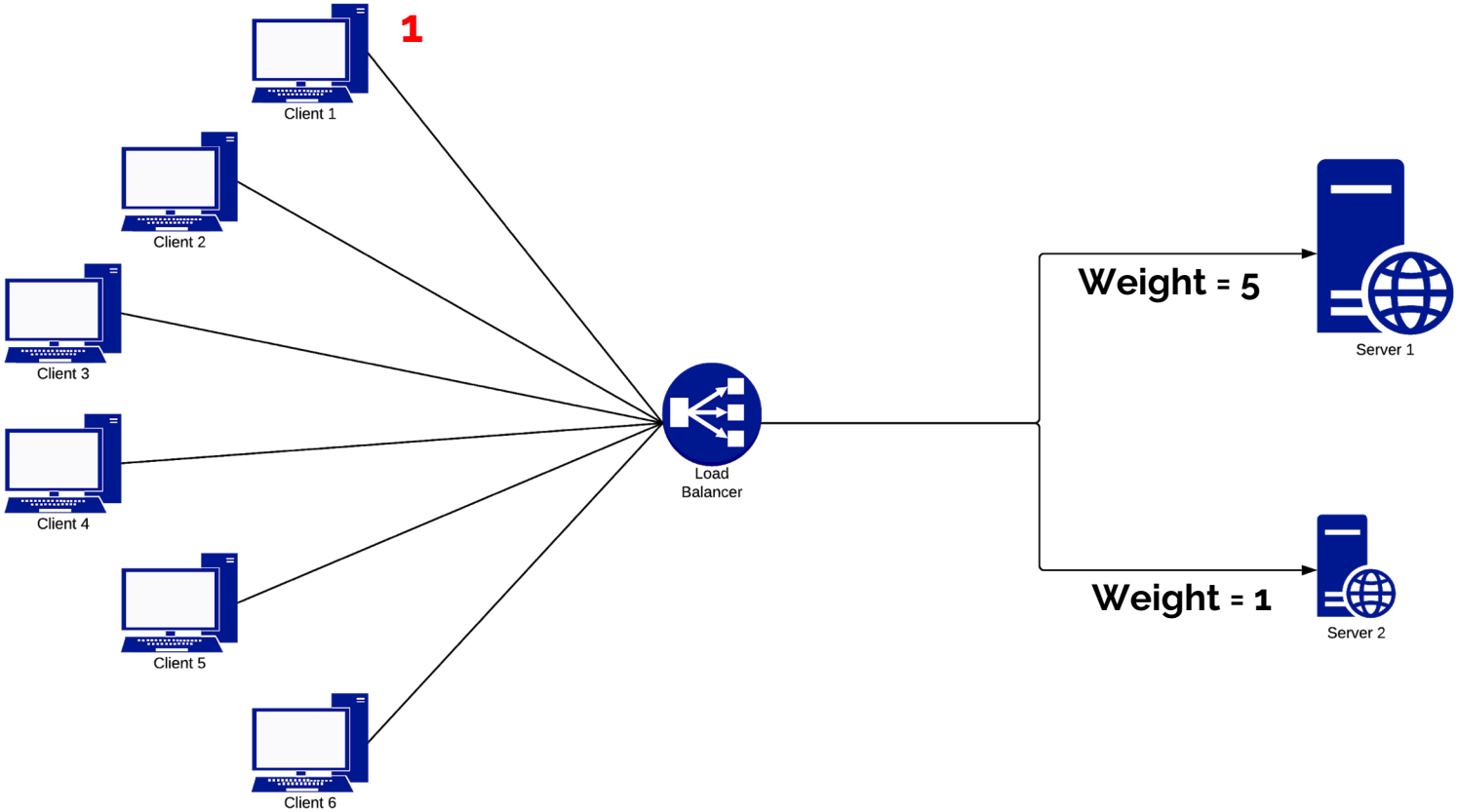
Round Robin Algorithm



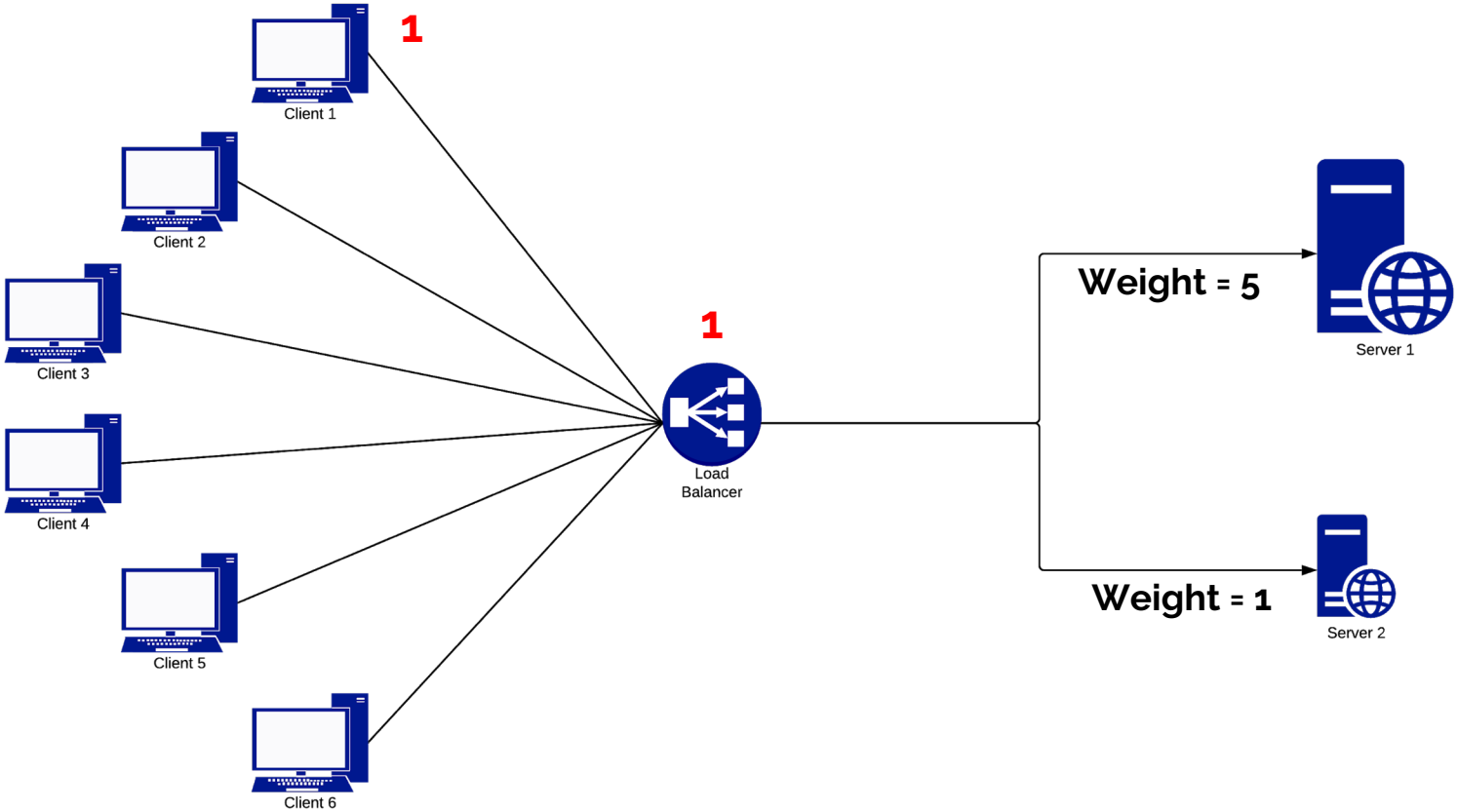
Weighted Round Robin Algorithm



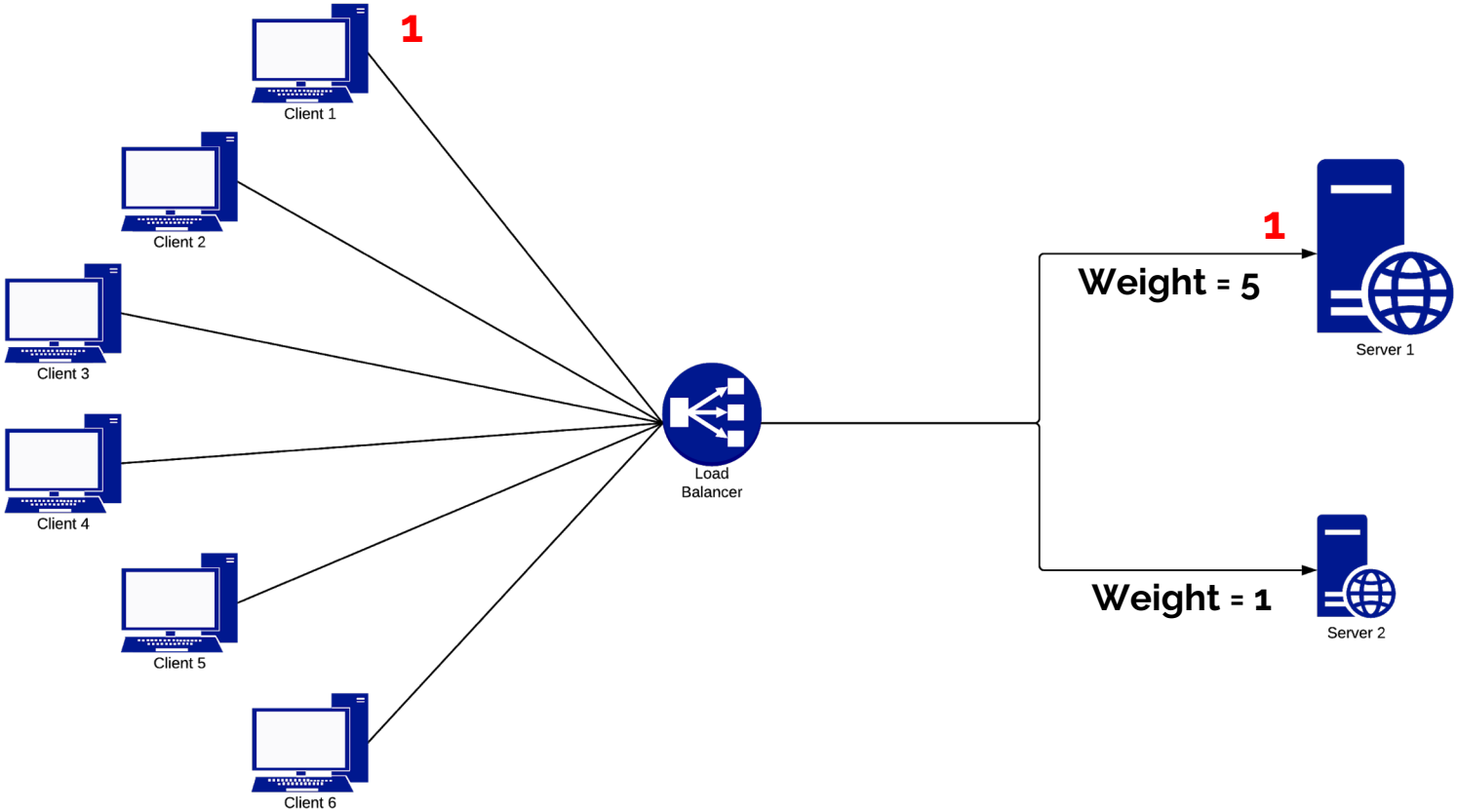
Weighted Round Robin Algorithm



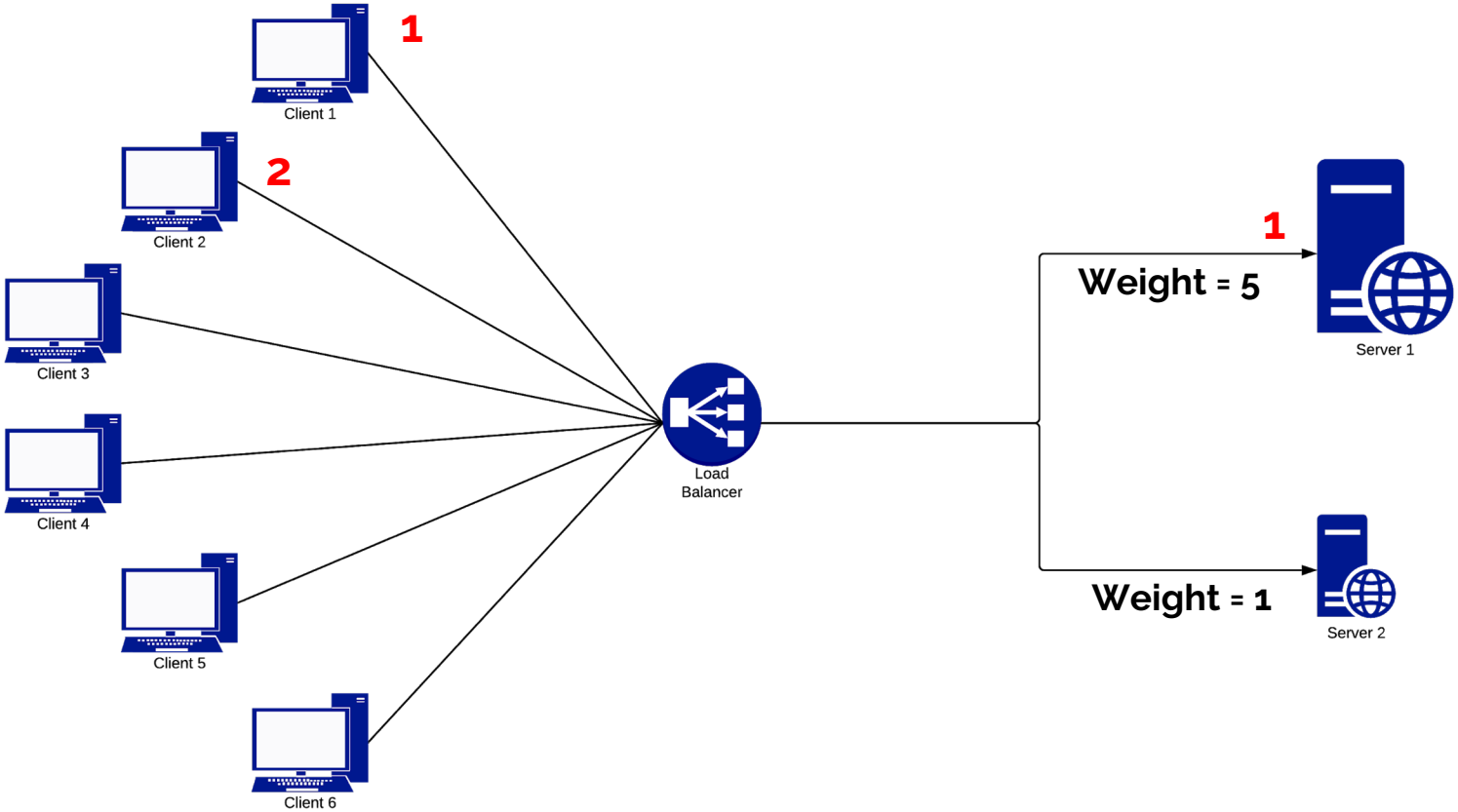
Weighted Round Robin Algorithm



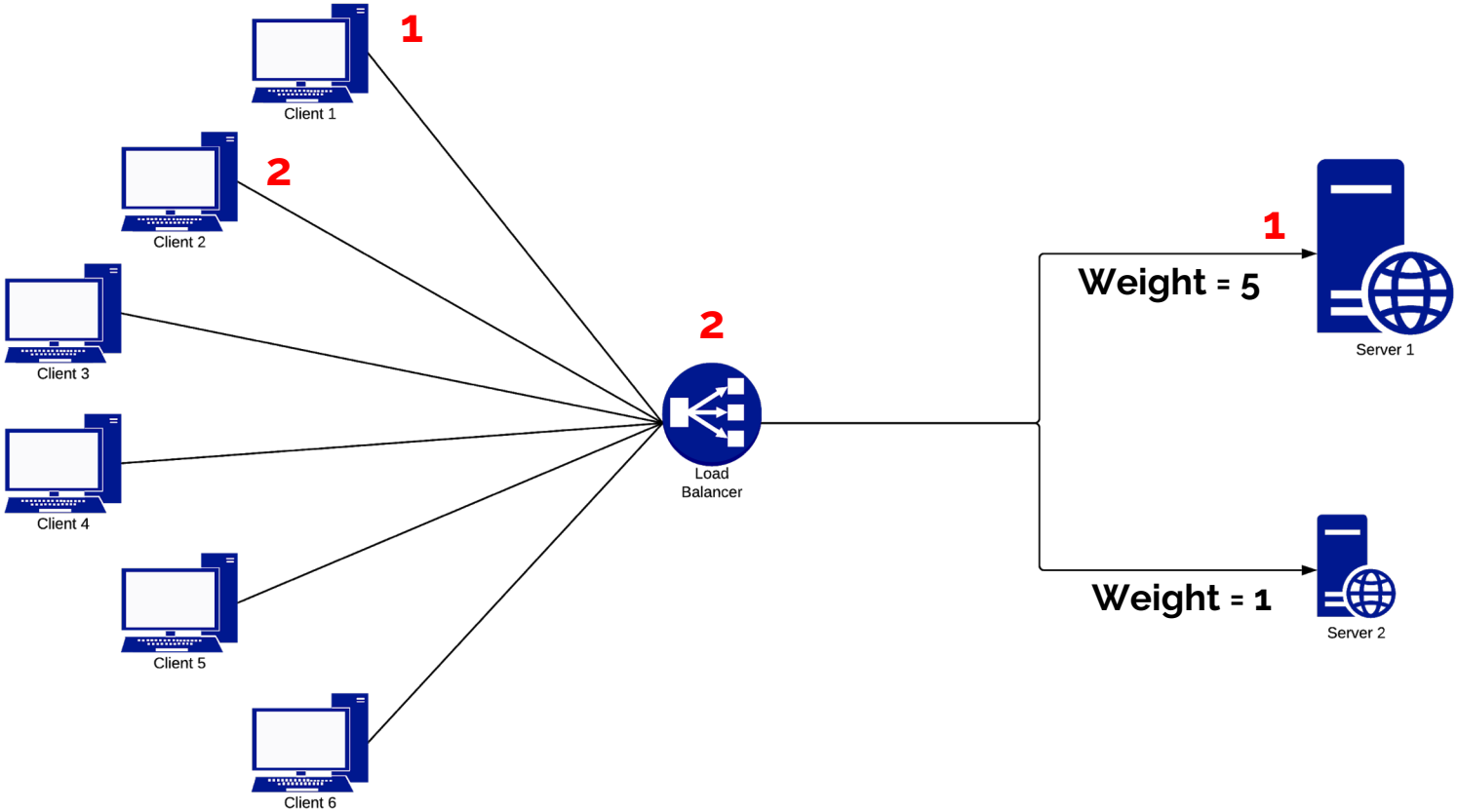
Weighted Round Robin Algorithm



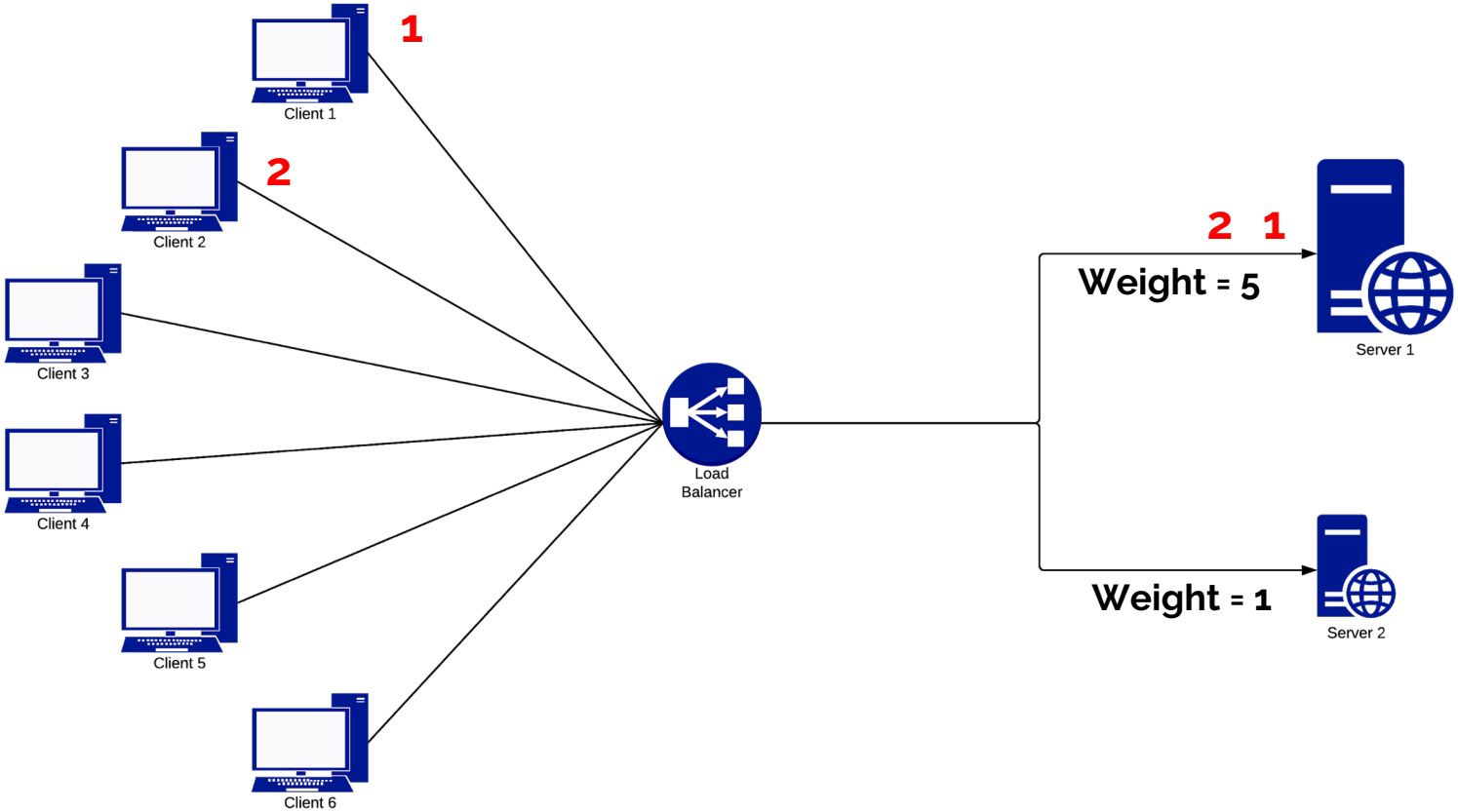
Weighted Round Robin Algorithm



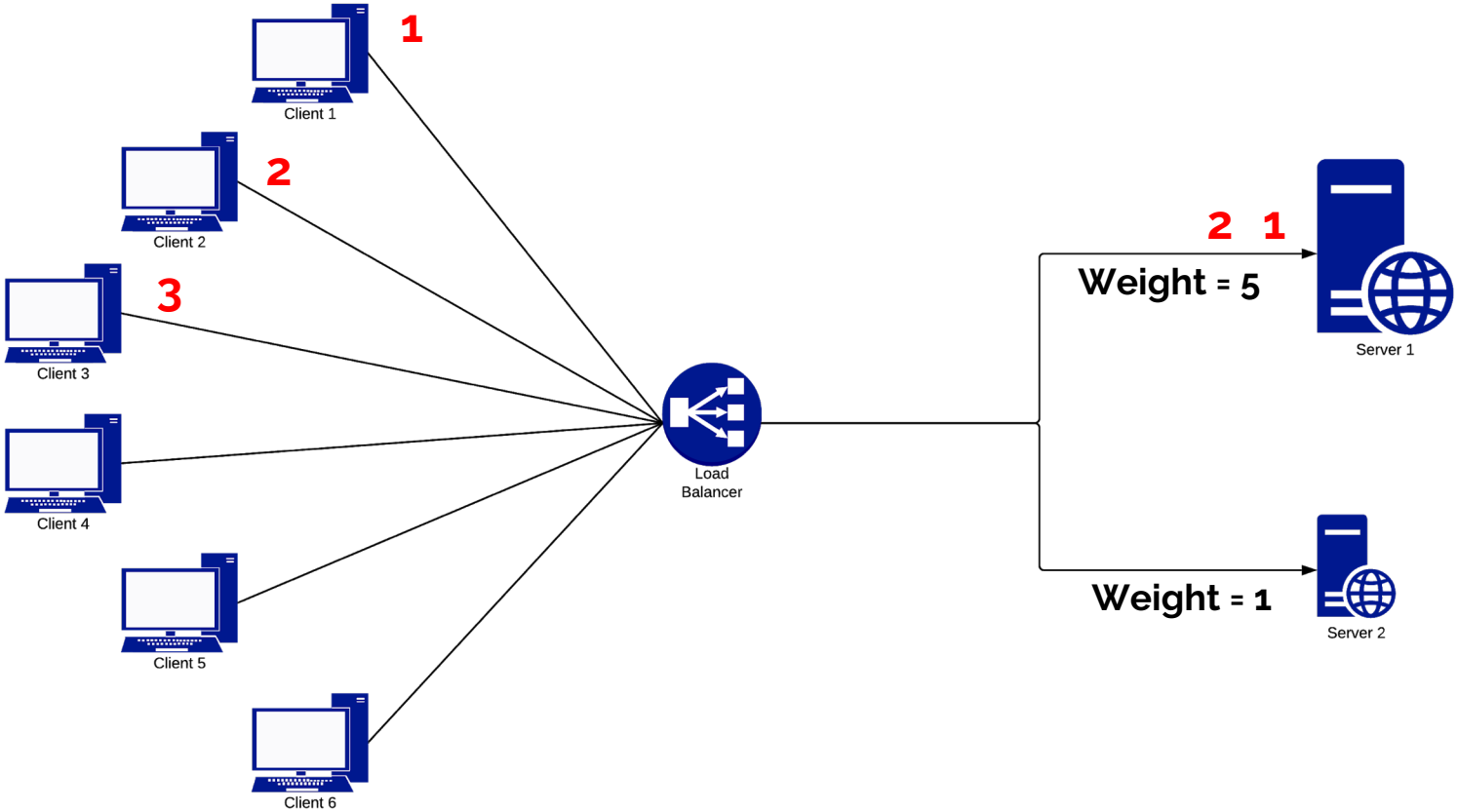
Weighted Round Robin Algorithm



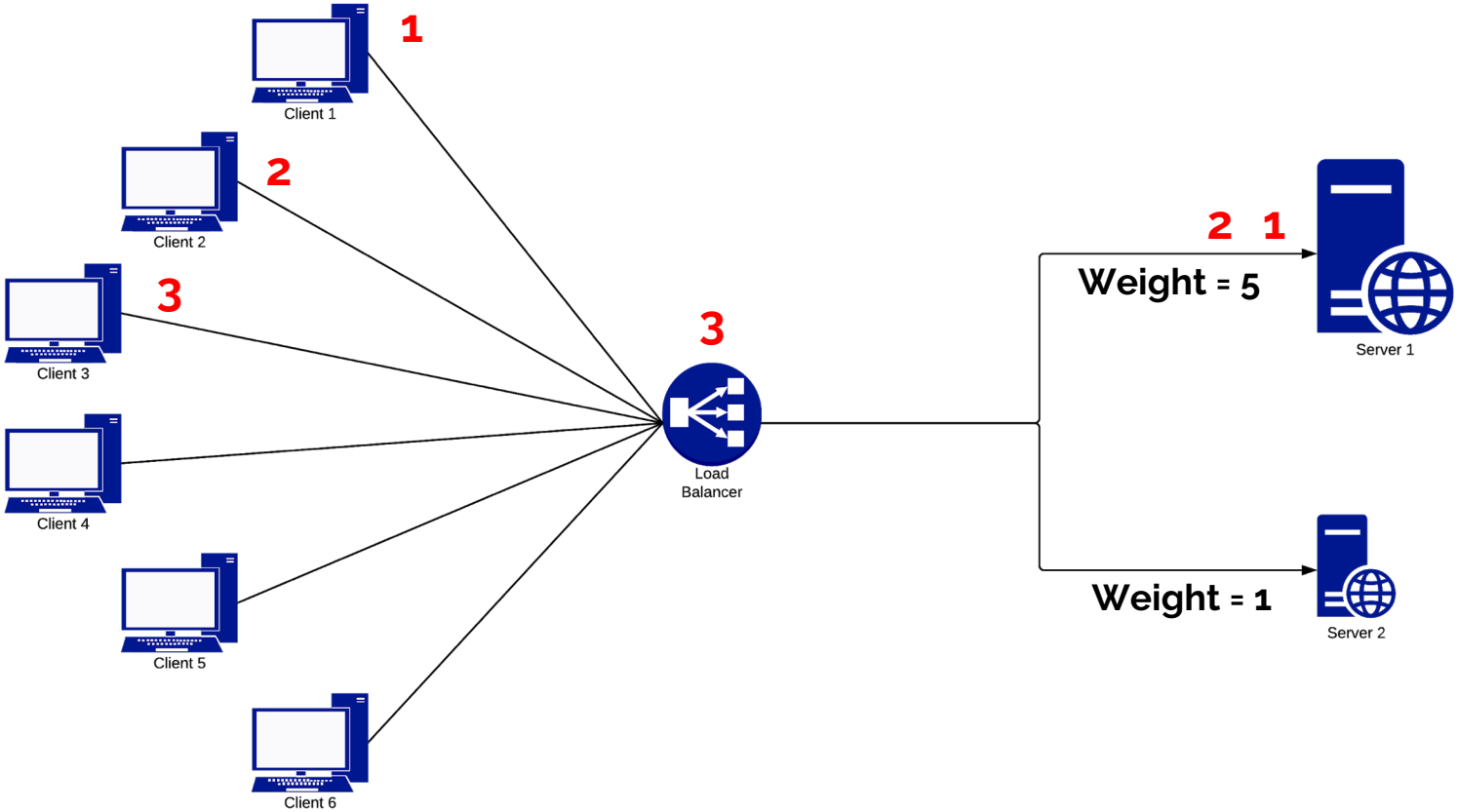
Weighted Round Robin Algorithm



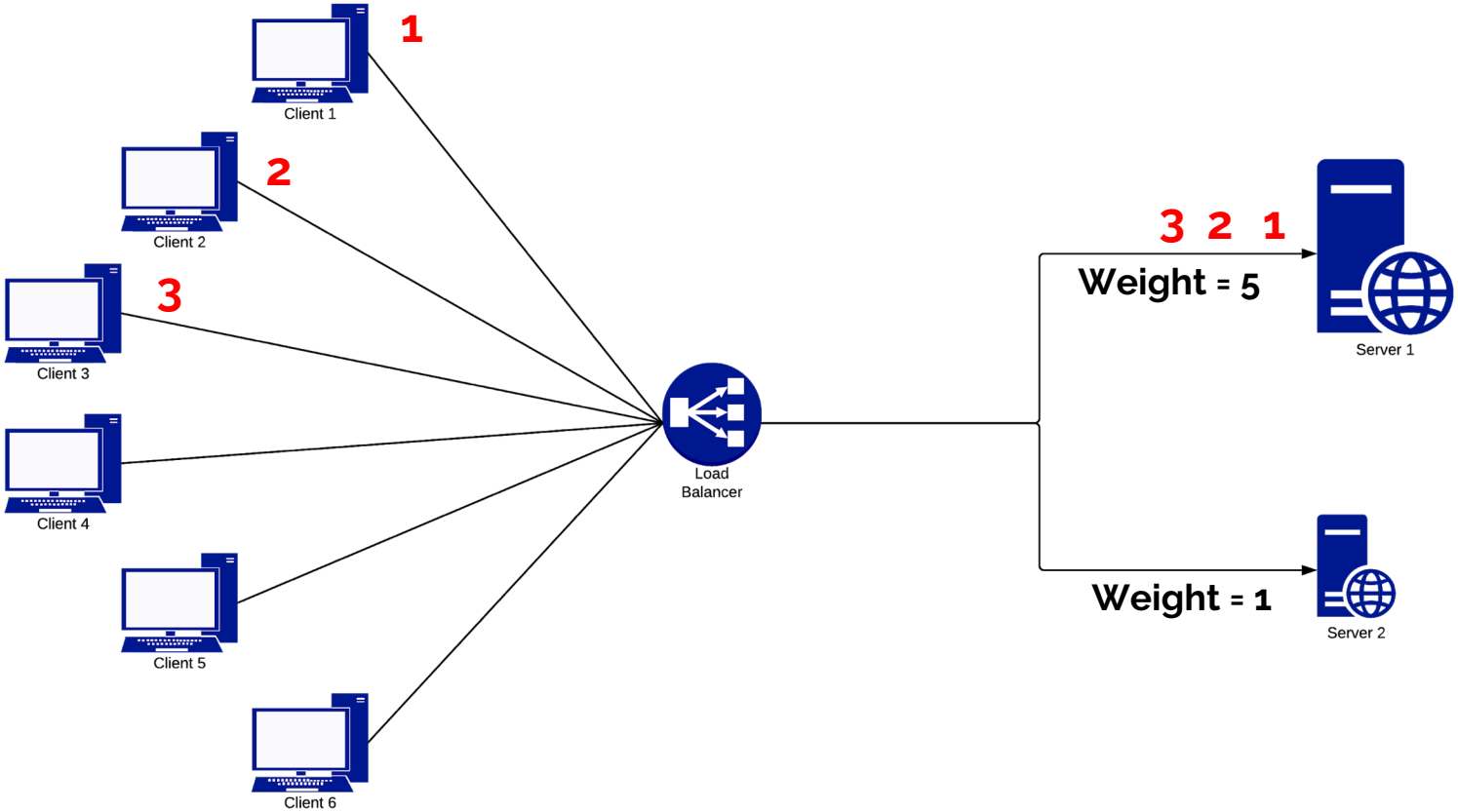
Weighted Round Robin Algorithm



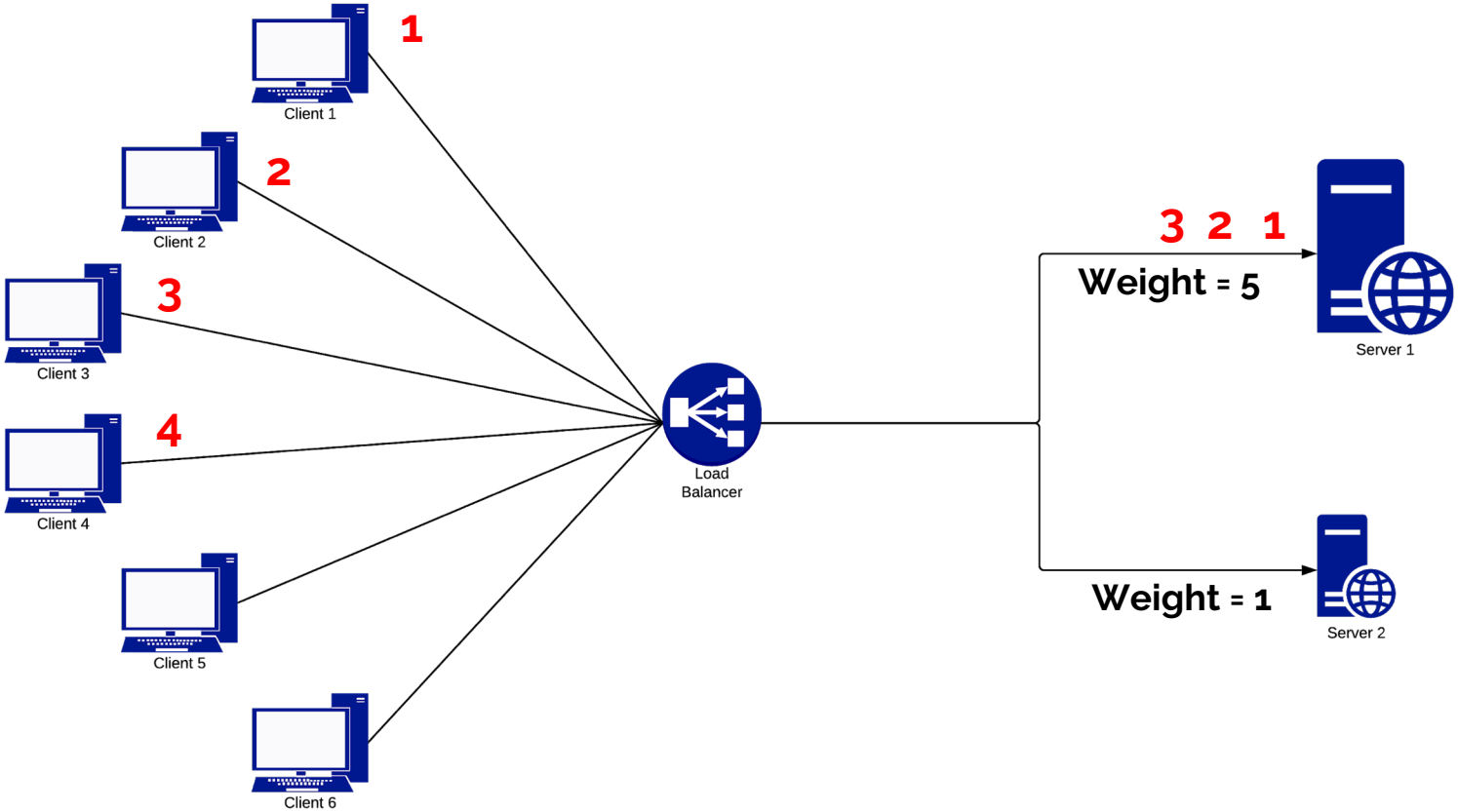
Weighted Round Robin Algorithm



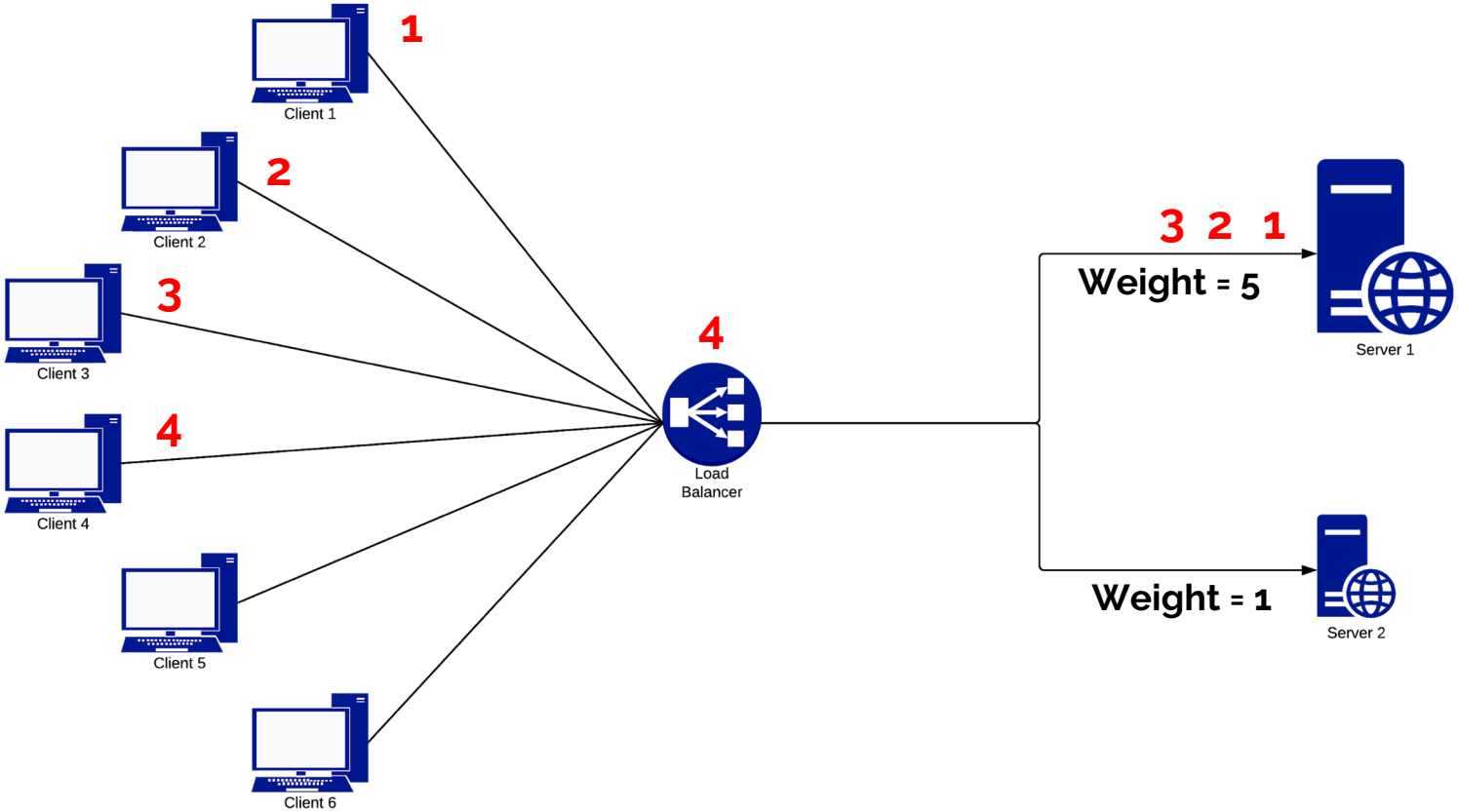
Weighted Round Robin Algorithm



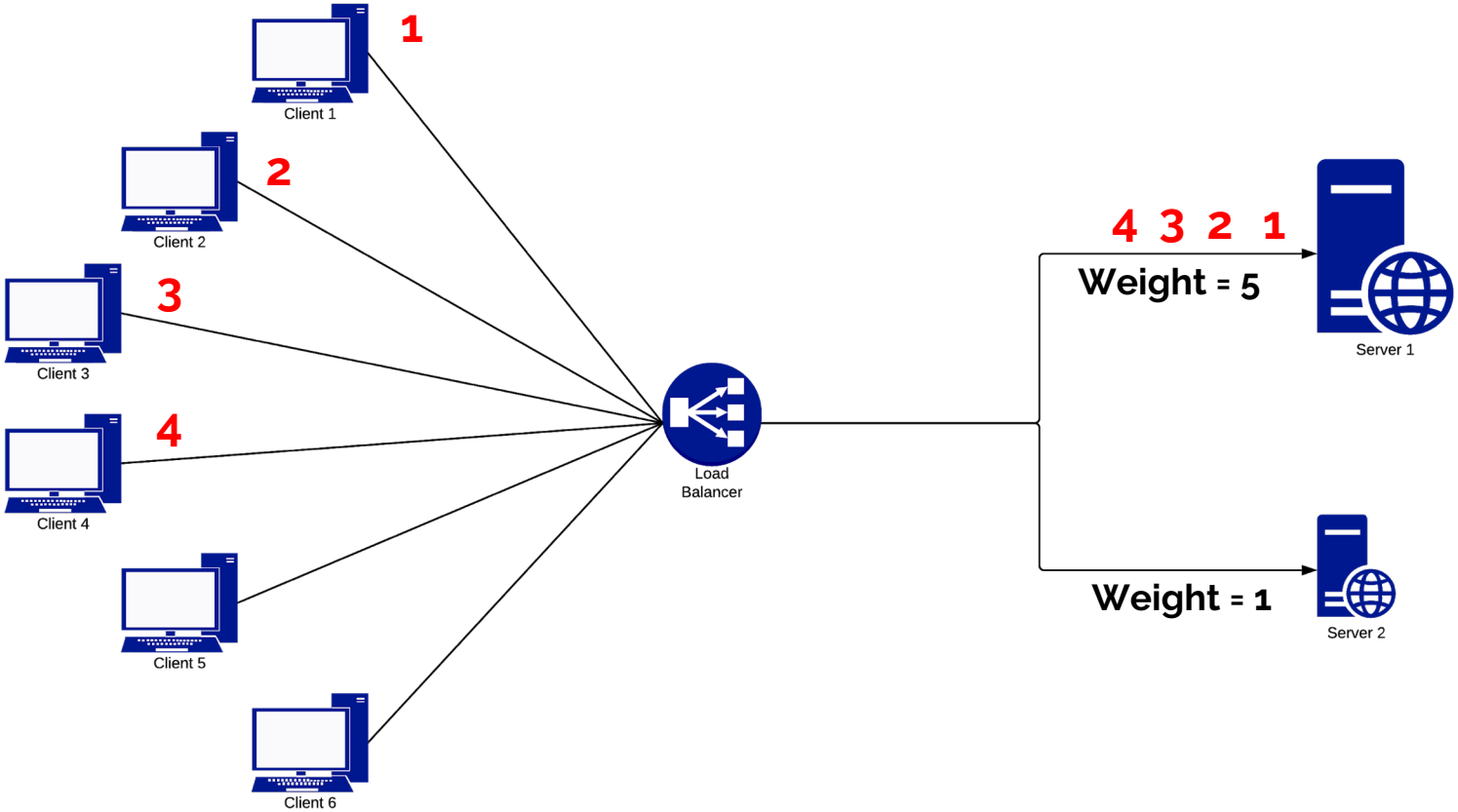
Weighted Round Robin Algorithm



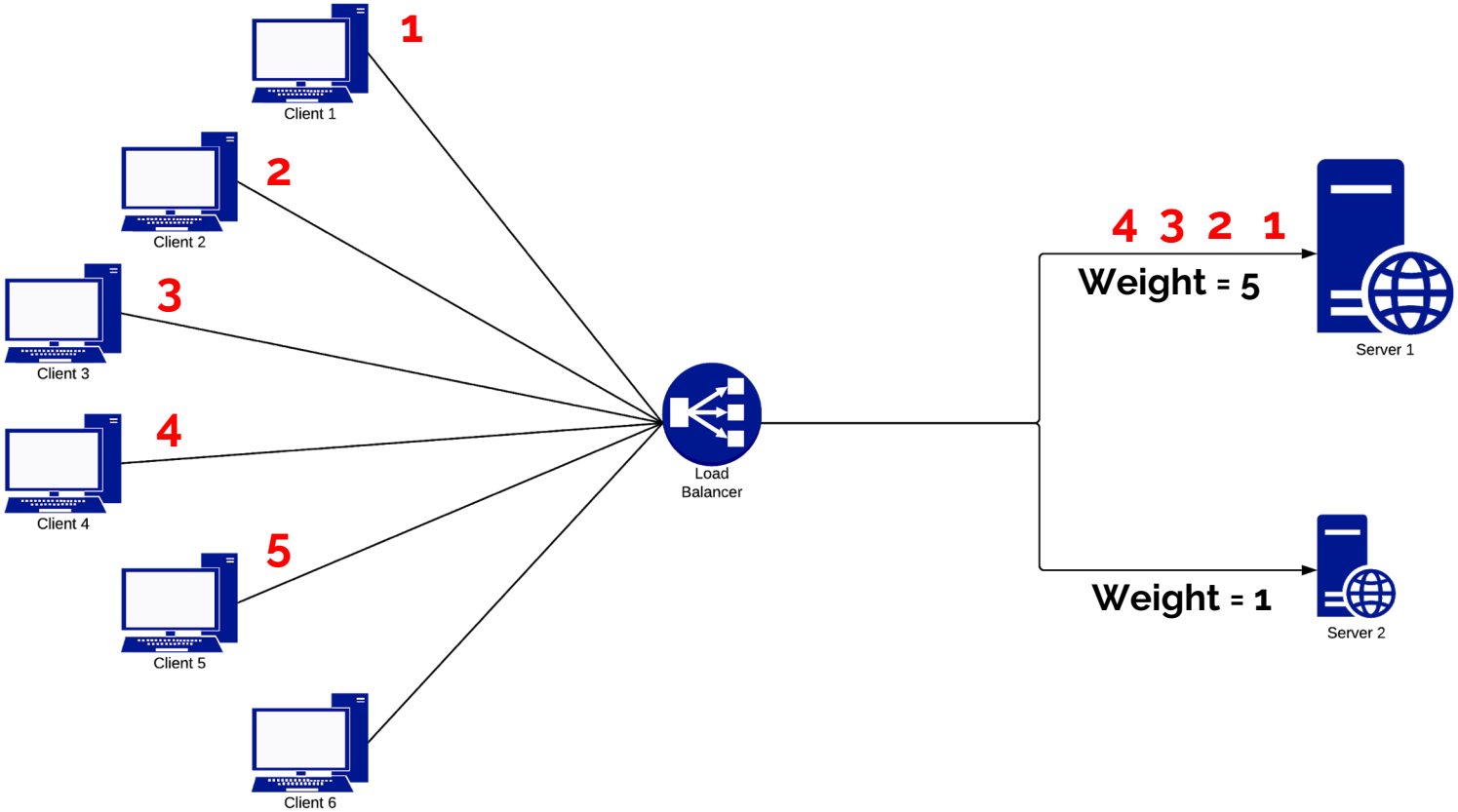
Weighted Round Robin Algorithm



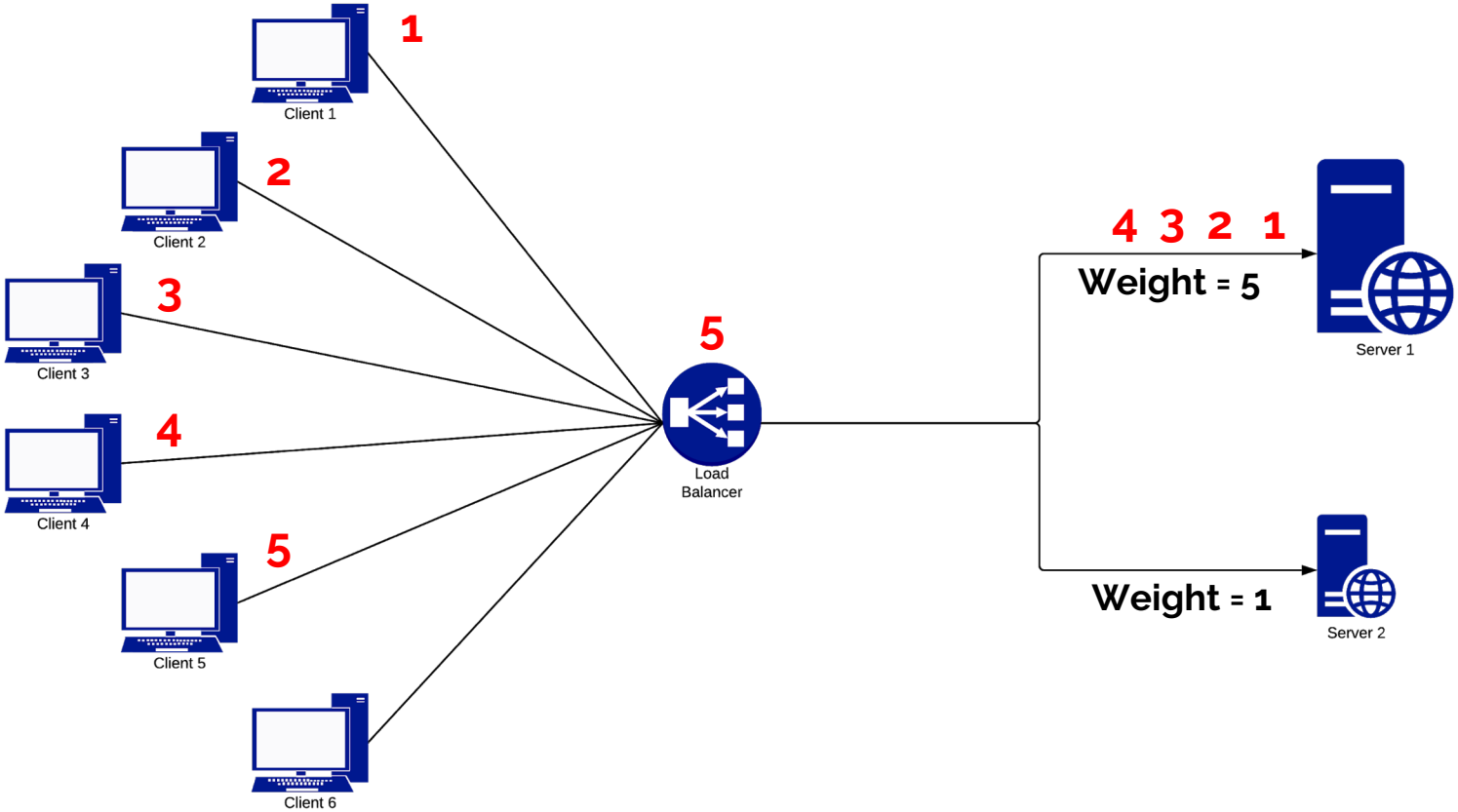
Weighted Round Robin Algorithm



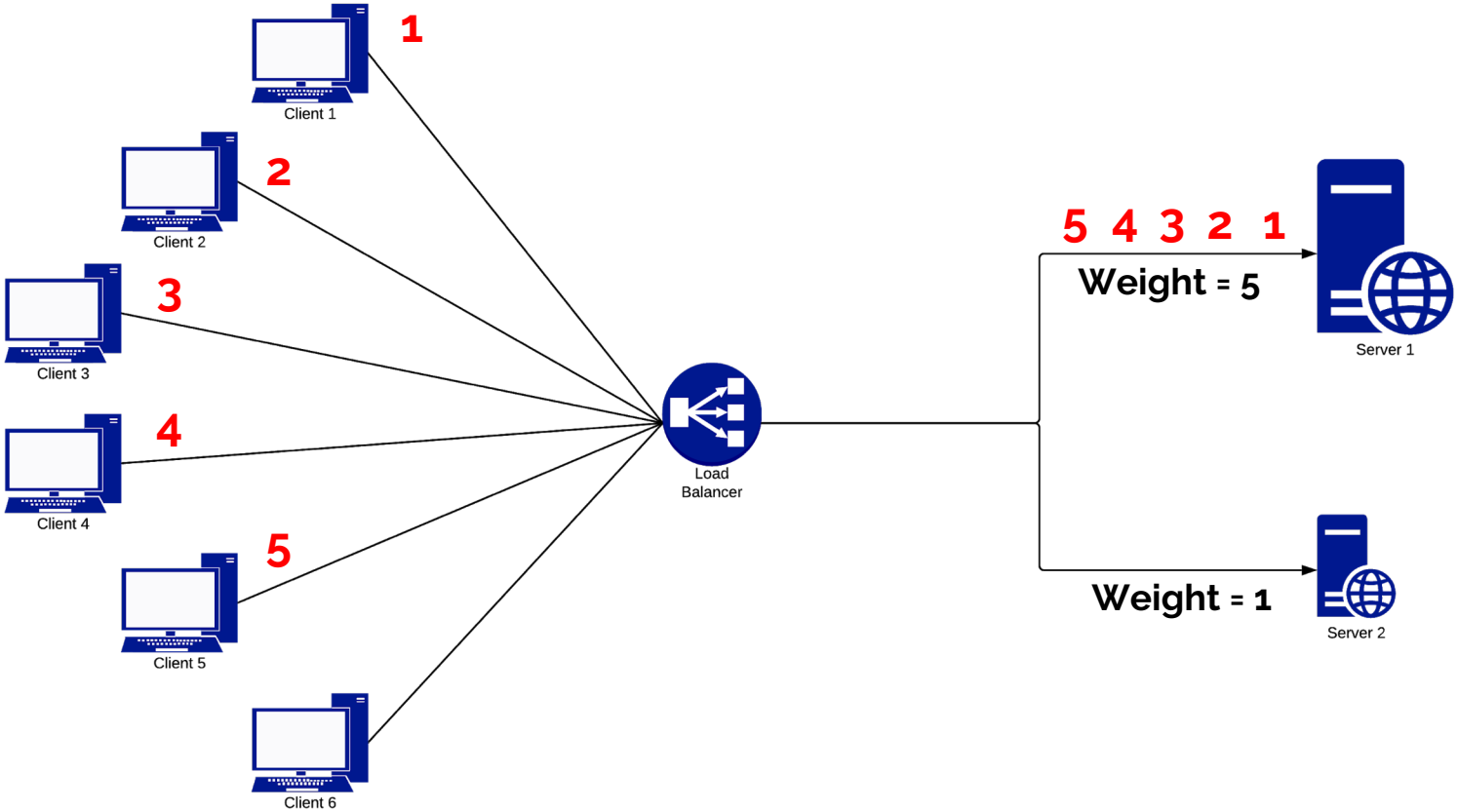
Weighted Round Robin Algorithm



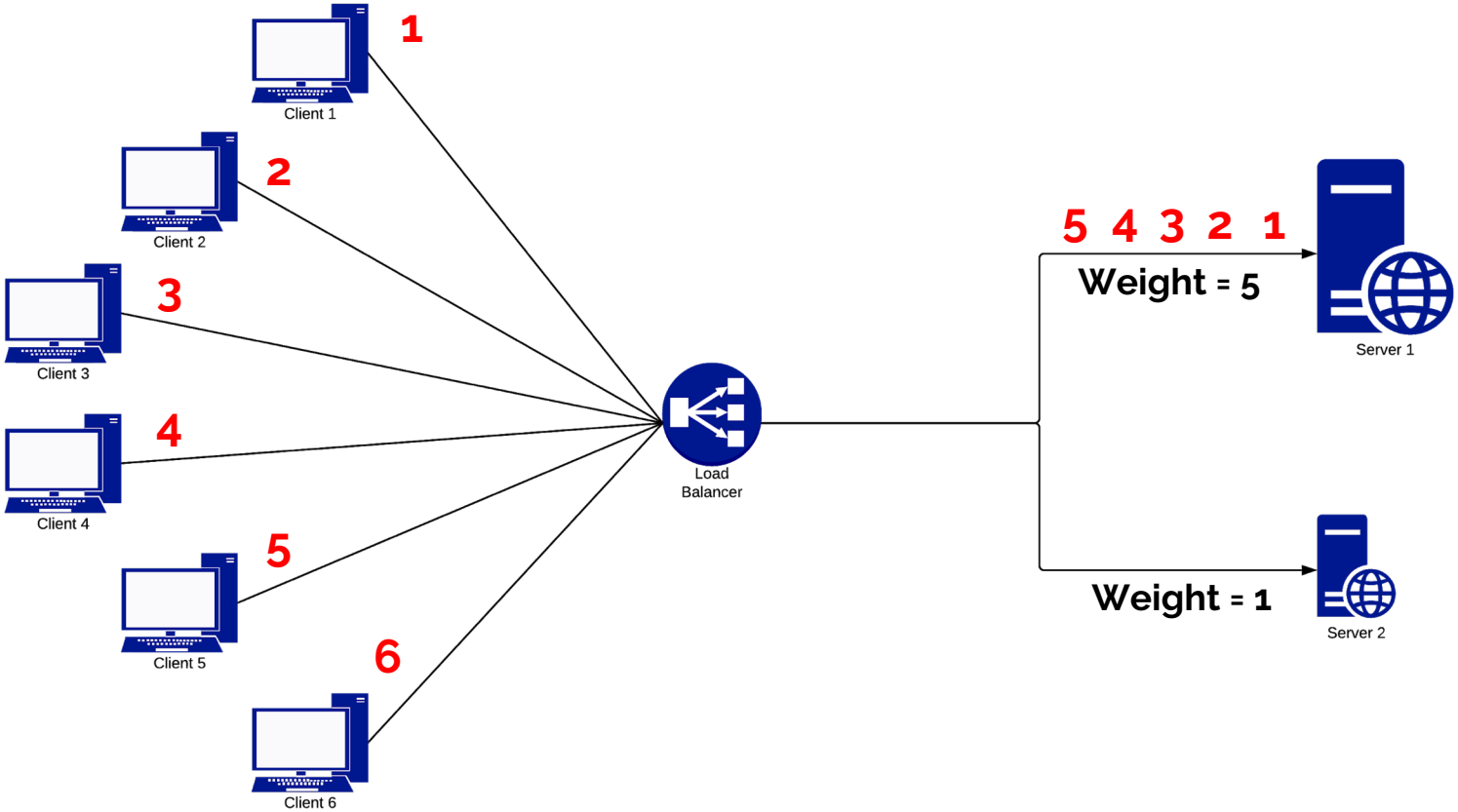
Weighted Round Robin Algorithm



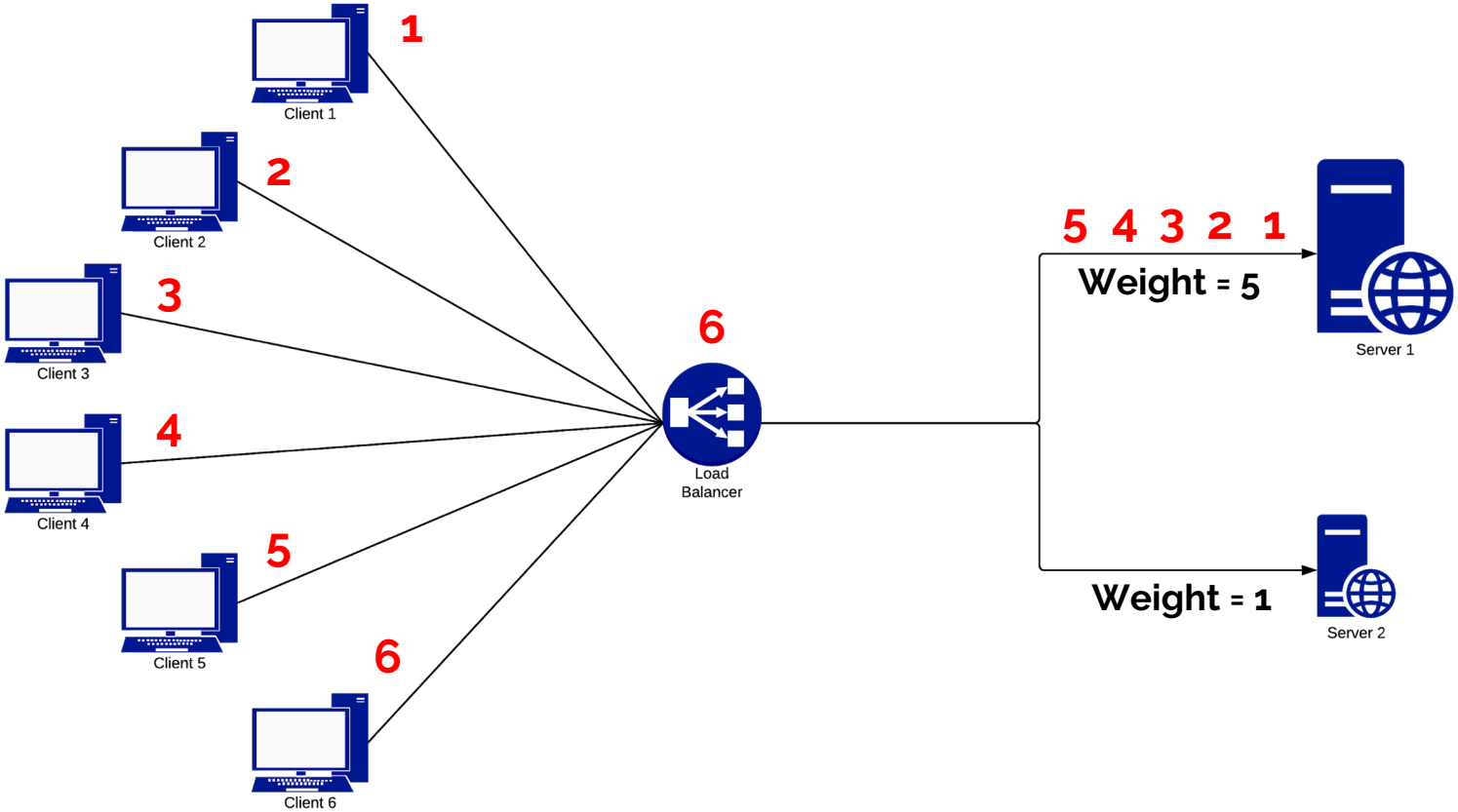
Weighted Round Robin Algorithm



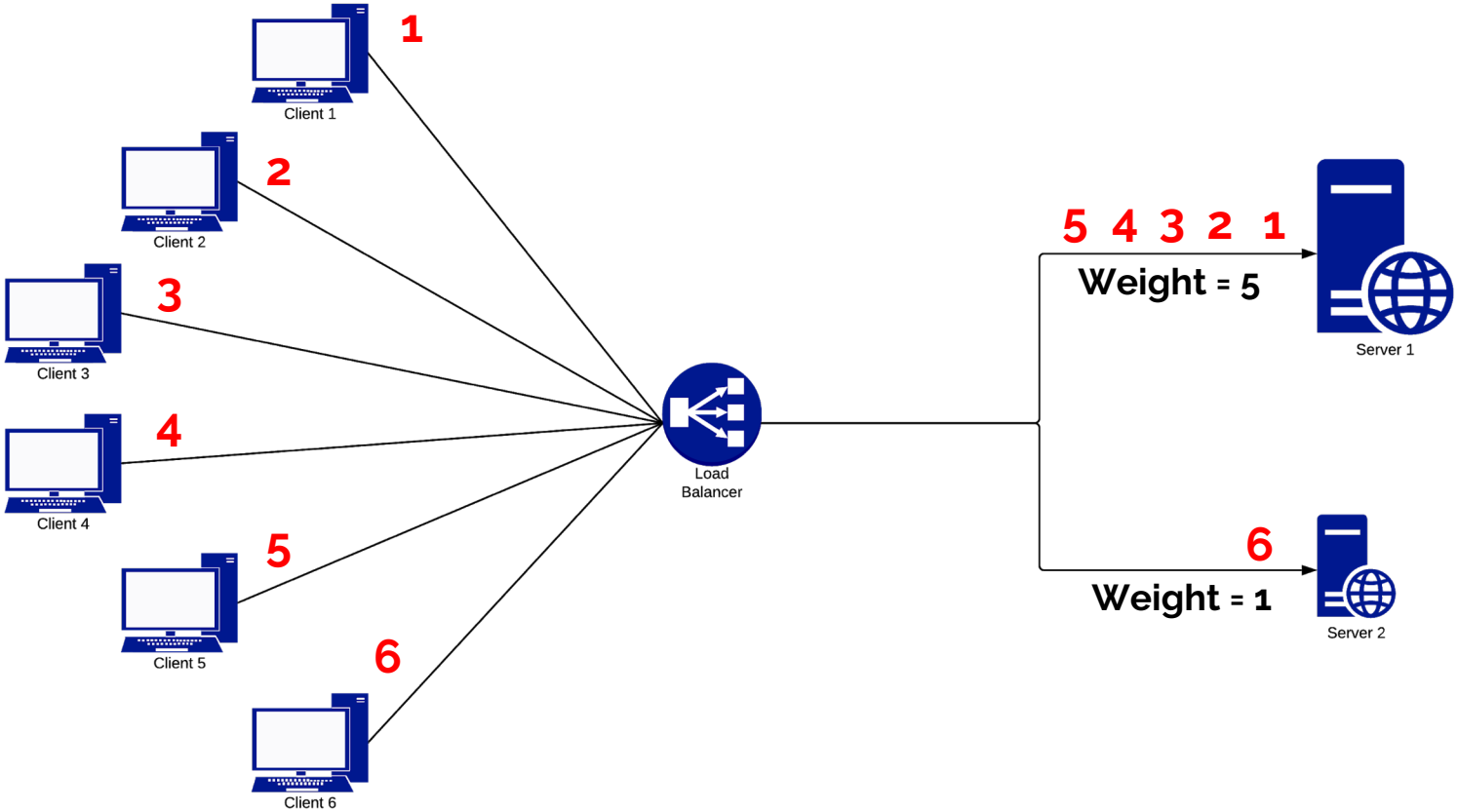
Weighted Round Robin Algorithm



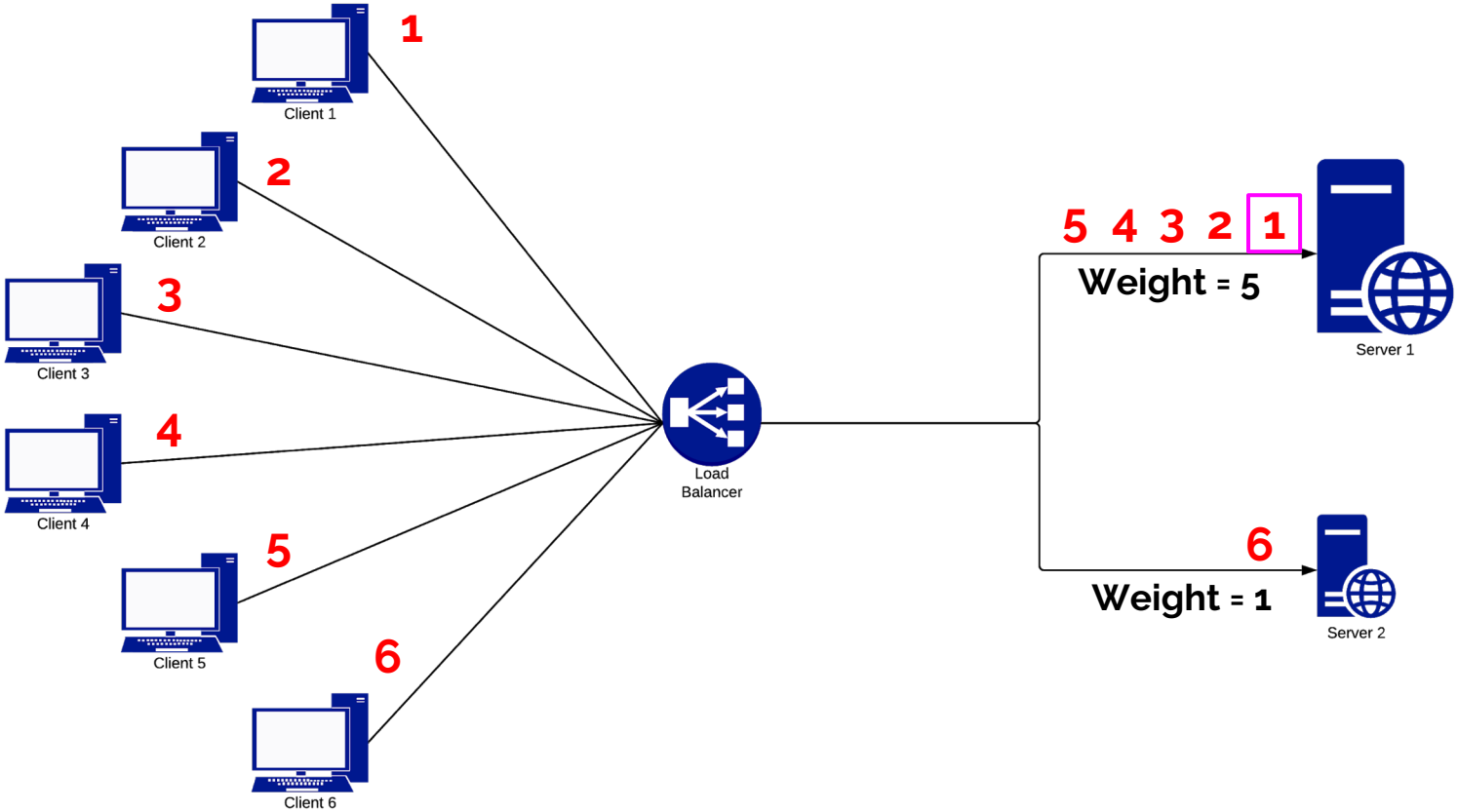
Weighted Round Robin Algorithm



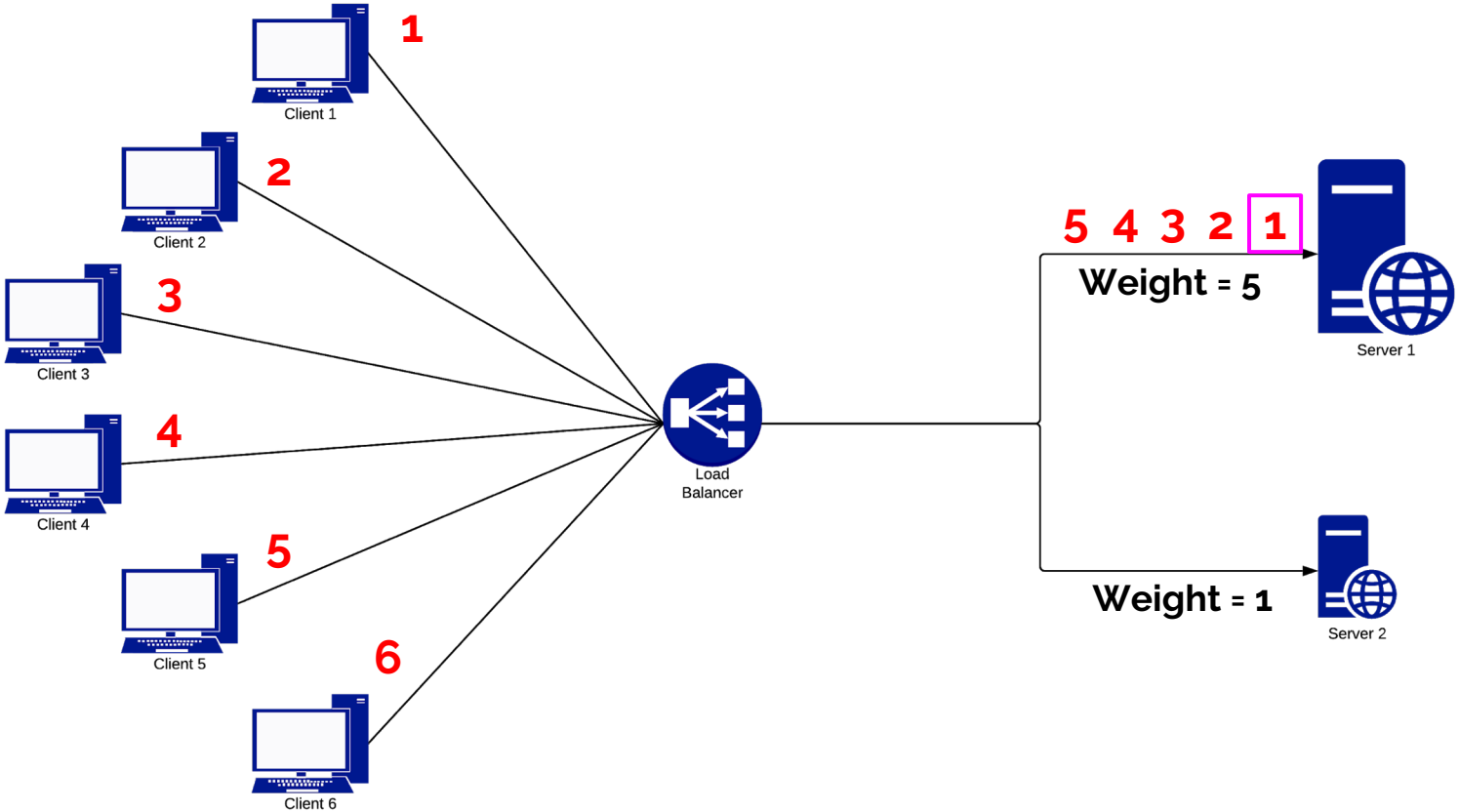
Weighted Round Robin Algorithm



Weighted Round Robin Algorithm



Weighted Round Robin Algorithm



Outline

1. Background
2. **Length Based Weighted Round Robin Algorithm**
   a. **Overview**
   b. **Architecture**
   c. **Threshold Value**
   d. **Simulation Results**
3. Honey Bee Behavior Inspired Load Balancing
4. Conclusion

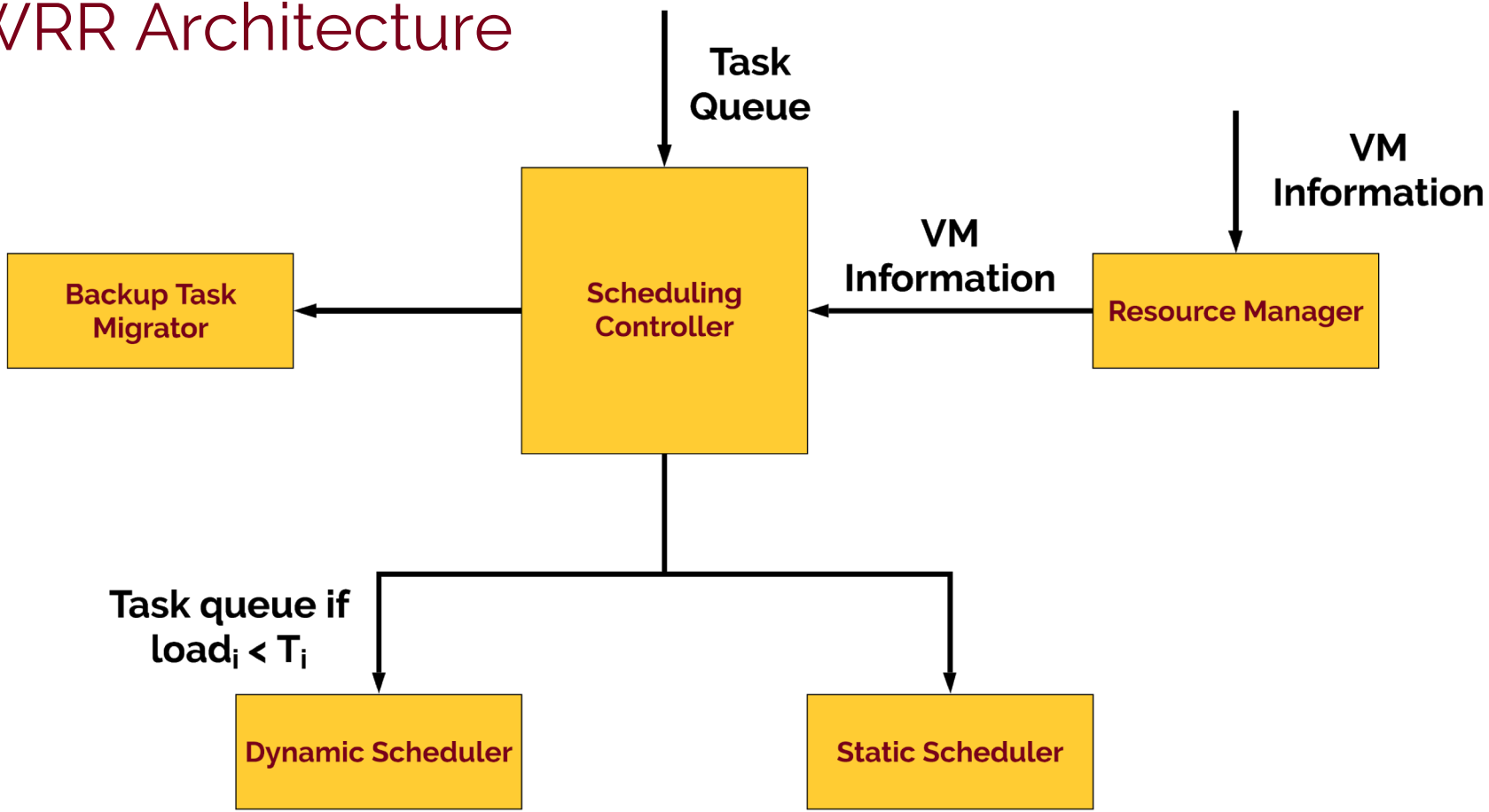
Length Based Weighted Round Robin Algorithm (LWRR)

- Developed by Devi and Uthariaraj [4]
- Non-preemptive algorithm
  - Tasks are executed without interruption
- Goal is to combine static and dynamic algorithms to reduce number of task migrations

Task Migration

- Process of moving a task from an overloaded to underloaded VM
- Expensive
- Less is better
- Idea is to effectively load balance **new** tasks to minimize task migrations at runtime

LWRR Architecture



LWRR Architecture based on [4]

LWRR Threshold Value

K = 4	VM_1	VM_2	VM_3	VM_4
c_i	20	25	25	30
l_i	16	22	12	24

LWRR Threshold Value

$$C = \sum_{i=1}^{k} c_i$$

K = 4	VM_1	VM_2	VM_3	VM_4
c_i	20	25	25	30
l_i	16	22	12	24

LWRR Threshold Value

$$C = \sum_{i=1}^{k} c_i$$

C = 100

K = 4	VM_1	VM_2	VM_3	VM_4
c_i	20	25	25	30
l_i	16	22	12	24

LWRR Threshold Value

$$L = \sum_{i=1}^{k} l_i$$

C = 100

K = 4	VM_1	VM_2	VM_3	VM_4
c_i	20	25	25	30
l_i	16	22	12	24

LWRR Threshold Value

$$L = \sum_{i=1}^{k} l_i$$

C = 100

L = 74

K = 4	VM_1	VM_2	VM_3	VM_4
c_i	20	25	25	30
l_i	16	22	12	24

LWRR Threshold Value

$$LPC = \frac{L}{C}$$

C = 100

L = 74

K = 4	VM_1	VM_2	VM_3	VM_4
c_i	20	25	25	30
l_i	16	22	12	24

LWRR Threshold Value

$$LPC = \frac{L}{C}$$

C = 100

L = 74

LPC = 0.74

K = 4	VM_1	VM_2	VM_3	VM_4
c_i	20	25	25	30
l_i	16	22	12	24

LWRR Threshold Value

$$T_i = LPC \cdot c_i$$

C = 100

L = 74

LPC = 0.74

K = 4	VM_1	VM_2	VM_3	VM_4
c_i	20	25	25	30
l_i	16	22	12	24

LWRR Threshold Value

$$T_i = LPC \cdot c_i$$

C = 100

L = 74

LPC = 0.74

K = 4	VM_1	VM_2	VM_3	VM_4
c_i	20	25	25	30
l_i	16	22	12	24
T_i	14.8	18.5	18.5	22.2

LWRR Threshold Value

$$T_i = LPC \cdot c_i$$

C = 100

L = 74

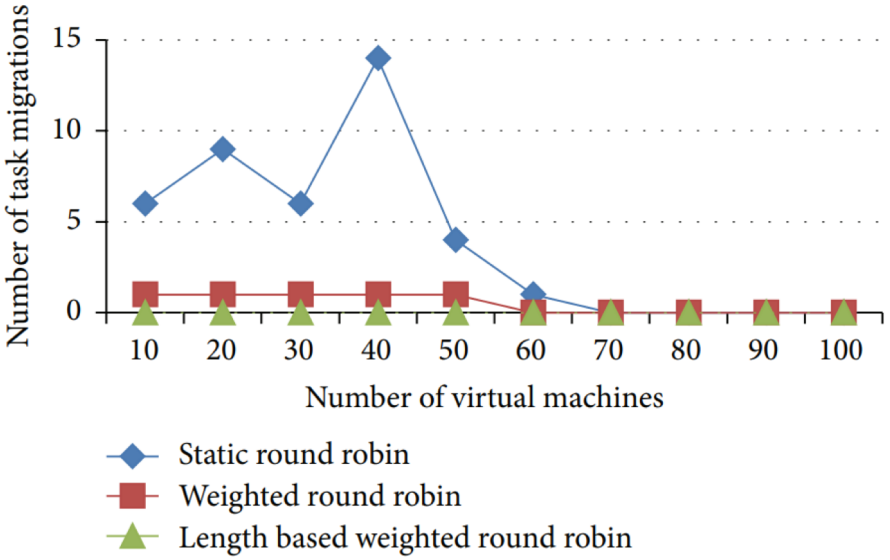
LPC = 0.74

K = 4	VM_1	VM_2	VM_3	VM_4
c_i	20	25	25	30
l_i	16	22	12	24
T_i	14.8	18.5	18.5	22.2

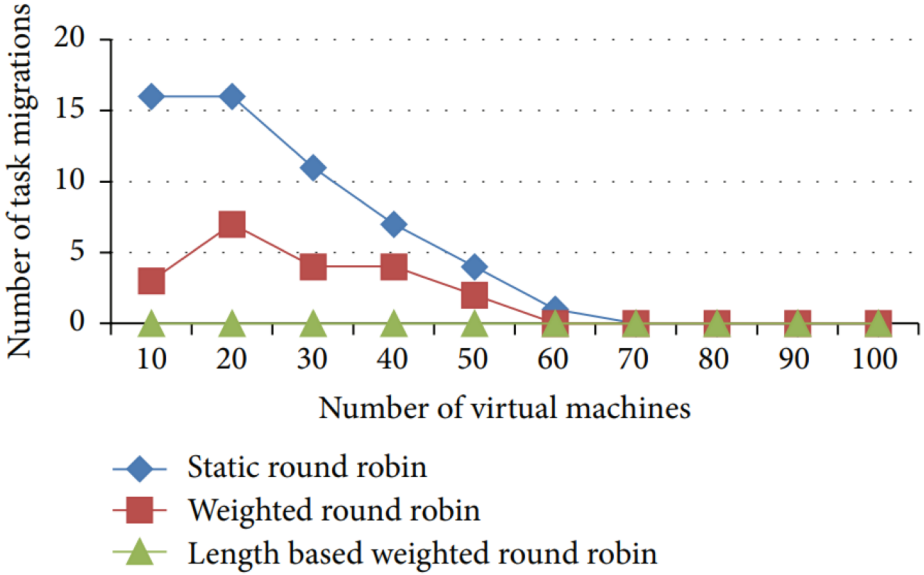
LWRR Simulation Results

- CloudSim was used
  - Provides environment to implement load-balancing and scheduling algorithms
  - Most commonly used cloud simulator
  - Written in Java
- Why use simulations?
  - Implementation can be expensive
- Simulated with homogeneous and heterogeneous tasks in heterogeneous environment

LWRR Simulation Results - Task Migrations

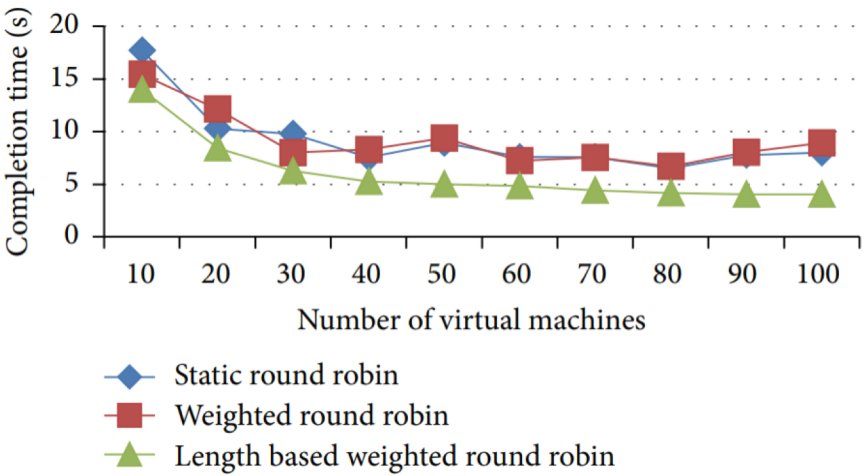


Number of homogenous task migrations [4]

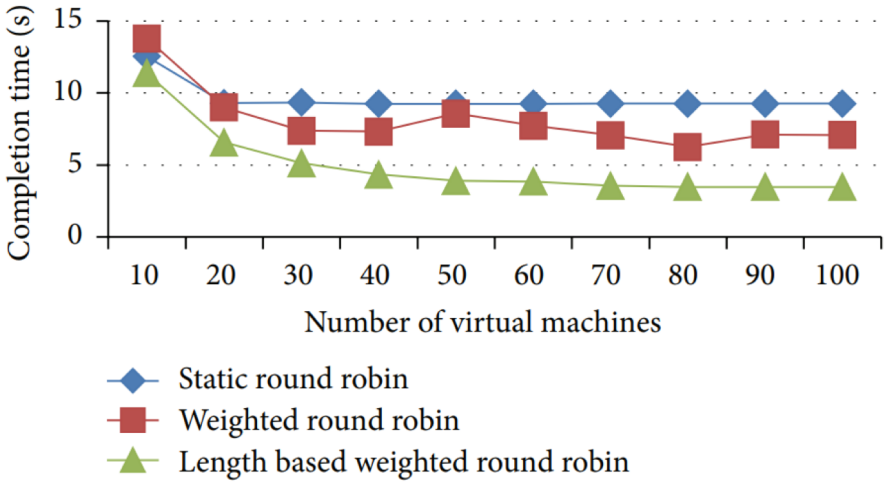


Number of heterogeneous task migrations [4]

LWRR Simulation Results - Completion Time



Homogenous task completion times [4]



Heterogeneous task completion times [4]

Outline

1. Background

2. Length Based Weighted Round Robin Algorithm

3. **Honey Bee Behavior Inspired Load Balancing**
   a. **Overview**
   b. **Honey Bee Behavior vs Cloud Environment**
   c. **Algorithm**
   d. **Simulation Results**

4. Conclusion

Honey Bee Behavior Inspired Load Balancing (HBB-LB)

- Developed by Babu and Krishna [5]
- Non-preemptive algorithm
  - Tasks are executed without interruption
- Utilizes honey bee foraging behavior based on Johnson and Nieh [6]
- Combines existing dynamic load balancing techniques with honey bee foraging behavior
- Goal is to reduce task migrations and completion time

Honey Bee Behavior vs Cloud Environment

Honey Bee Behavior	Cloud Environment
Honey Bee	Task
Food Source	VM
Honey bee foraging a food source	Task assigned to a VM
Honey bee running out of food at food source	VM in an overloaded condition
Foraging bee finding a new food source	Task will be moved from overloaded to underloaded VM

HBB-LB Algorithm

1. Find capacity and loads of all VMs

HBB-LB Algorithm

1. Find capacity and loads of all VMs
2. Make load balancing decision

HBB-LB Algorithm

1. Find capacity and loads of all VMs
2. Make load balancing decision
    a. If Load > Maximum Capacity, exit

HBB-LB Algorithm

1. Find capacity and loads of all VMs
2. Make load balancing decision
    a. If Load > Maximum Capacity, exit
    b. If $\sigma \leq$ Threshold Condition Set, exit

HBB-LB Algorithm

1. Find capacity and loads of all VMs
2. Make load balancing decision
    a. If Load > Maximum Capacity, exit
    b. If $\sigma \leq$ Threshold Condition Set, exit
3. Execute load balancing until system is balanced

HBB-LB Algorithm - Load Balancer

1. Group VMs based on load

HBB-LB Algorithm - Load Balancer

1. Group VMs based on load
    a. Underloaded VM (UVM)

HBB-LB Algorithm - Load Balancer

1. Group VMs based on load
    a. Underloaded VM (UVM)
    b. Balanced VM (BVM)

HBB-LB Algorithm - Load Balancer

1. Group VMs based on load
    a. Underloaded VM (UVM)
    b. Balanced VM (BVM)
    c. Overloaded VM (OVM)

HBB-LB Algorithm - Load Balancer

1. Group VMs based on load
    a. Underloaded VM (UVM)
    b. Balanced VM (BVM)
    c. Overloaded VM (OVM)
2. Sort UVM and OVM sets

HBB-LB Algorithm - Load Balancer

1. Group VMs based on load
    a. Underloaded VM (UVM)
    b. Balanced VM (BVM)
    c. Overloaded VM (OVM)
2. Sort UVM and OVM sets
3. Sort tasks in OVM based on priority

HBB-LB Algorithm - Load Balancer

1. Group VMs based on load
    a. Underloaded VM (UVM)
    b. Balanced VM (BVM)
    c. Overloaded VM (OVM)
2. Sort UVM and OVM sets
3. Sort tasks in OVM based on priority
4. For some tasks in each OVM, find suitable UVM

HBB-LB Algorithm - Load Balancer

1. Group VMs based on load
    a. Underloaded VM (UVM)
    b. Balanced VM (BVM)
    c. Overloaded VM (OVM)
2. Sort UVM and OVM sets
3. Sort tasks in OVM based on priority
4. For some tasks in each OVM, find suitable UVM
5. Update UVM and OVM sets

HBB-LB Algorithm - Load Balancer

1. Group VMs based on load
    a. Underloaded VM (UVM)
    b. Balanced VM (BVM)
    c. Overloaded VM (OVM)
2. Sort UVM and OVM sets
3. Sort tasks in OVM based on priority
4. For some tasks in each OVM, find suitable UVM
5. Update UVM and OVM sets
6. Repeat steps 1-5 until system is balanced

HBB-LB Algorithm - Load Balancer

1. Group VMs based on load
    a. Underloaded VM (UVM)
    b. Balanced VM (BVM)
    c. Overloaded VM (OVM)
2. Sort UVM and OVM sets
3. Sort tasks in OVM based on priority
4. **For some tasks in each OVM, find suitable UVM**
5. Update UVM and OVM sets
6. Repeat steps 1-5 until system is balanced

Finding a suitable VM

$$VM_d \in UVM$$

Finding a suitable VM

$$VM_d \in UVM$$

$$T_h \rightarrow VM_d | min(\sum T_h) \in VM_d$$

Finding a suitable VM

$$VM_d \in UVM$$

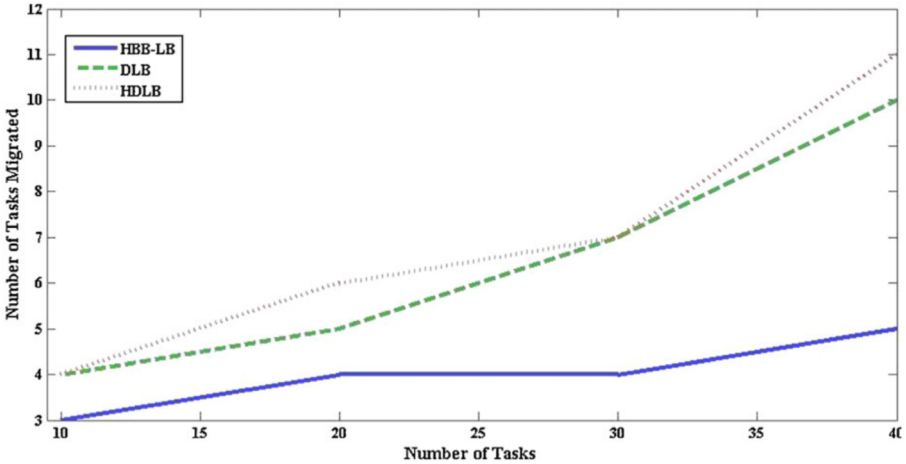
$$T_m \rightarrow VM_d | min(\sum T_m + \sum T_h) \in VM_d$$

Finding a suitable VM

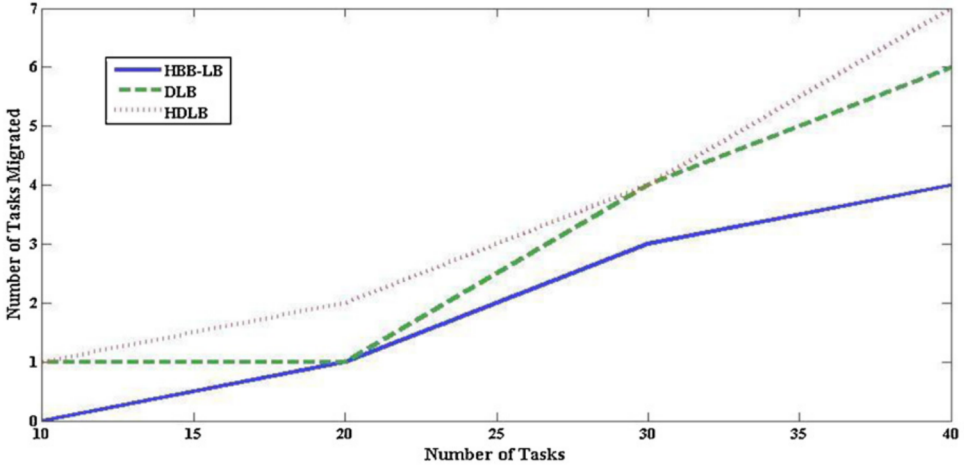
$$VM_d \in UVM$$

$$T_l \rightarrow VM_d | min(\sum T) \in VM_d$$

HBB-LB Simulation Results - Task Migrations

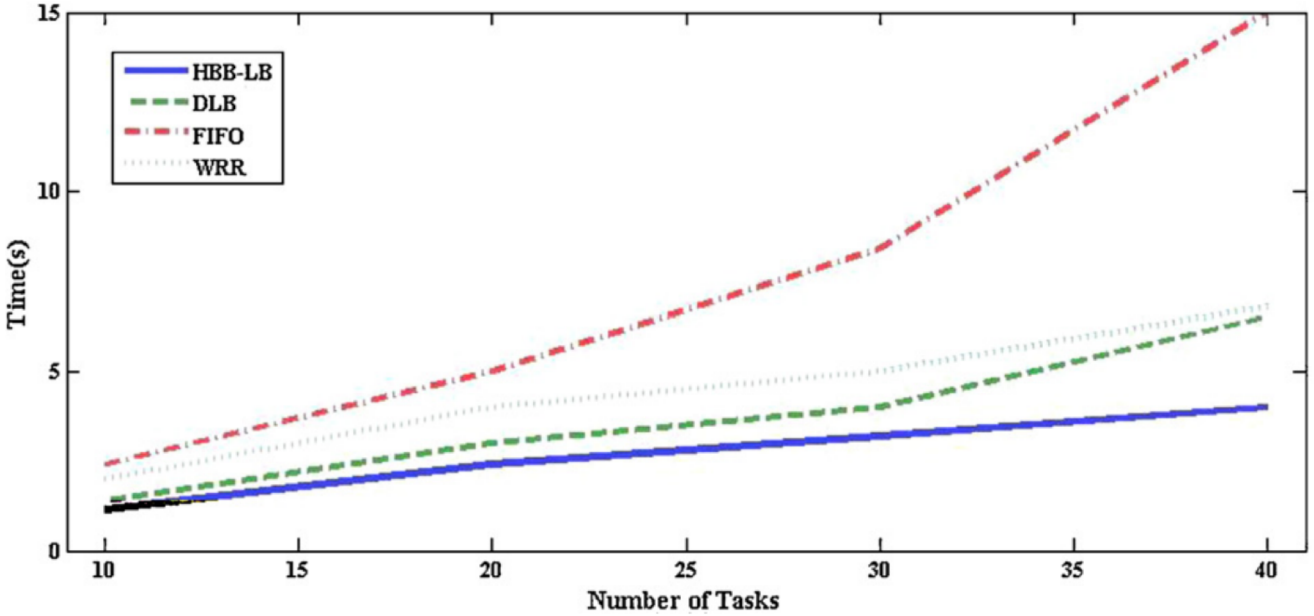


Task Migrations vs Number of Tasks for 4 VMs [4]



Task Migrations vs Number of Tasks for 7 VMs [4]

HBB-LB Simulation Results - Completion Time



Completion Time vs Number of Tasks [4]

Outline

1. Background
2. Length Based Weighted Round Robin Algorithm
3. Honey Bee Behavior Inspired Load Balancing
4. **Conclusions**

Conclusions

- Load balancing increases reliability and performance of cloud services
- LWRR improves load balancing by using a combination of static and dynamic techniques
- HBB-LB improves load balancing by utilizing honey bee foraging behavior
- LWRR vs HBB-LB

Thank you to Nic McPhee and Elena Machkasova!

Questions?

Work Cited

[1] https://www.statista.com/statistics/321215/global-consumer-cloud-computing-users/

[2]https://www.gartner.com/en/newsroom/press-releases/2018-09-12-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-17-percent-in-2019

[3] Pawan Kumar and Rakesh Kumar. 2019. Issues and Challenges of Load Balancing Techniques in Cloud Computing: A Survey. ACM Comput. Surv. 51, 6, Article 120 (February 2019), 35 pages. DOI: https://doi.org/10.1145/3281010

[4] D. Chitra Devi and V. Rhymend Uthariaraj, "Load Balancing in Cloud Computing Environment Using Improved Weighted Round Robin Algorithm for Nonpreemptive Dependent Tasks," The Scientific World Journal, vol. 2016, Article ID 3896065, 14 pages, 2016. https://doi.org/10.1155/2016/3896065.

[5] Dhinesh Babu L.D. and P. Venkata Krishna. 2013. Honey bee behavior inspired load balancing of tasks in cloud computing environments. Appl. Soft Comput. 13, 5 (May 2013), 2292-2303. DOI=http://dx.doi.org/10.1016/j.asoc.2013.01.025

[6] Johnson, Brian & Nieh, James. (2010). Modeling the Adaptive Role of Negative Signaling in Honey Bee Intraspecific Competition. Journal of insect behavior. 23. 459-471. 10.1007/s10905-010-9229-5.