LOAD BALANCING USING THE CONSENSUS CONTROL ALGORITHM FOR A NETWORK SERVER SYSTEM

By

ALAN TAN CHIN LOON

A dissertation submitted for partial fulfilment of the requirement for the degree of Master of Science (Electronic Systems Design Engineering)

August 2016

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my research supervisor, *Dr. Muhammad Nasiruddin Mahyuddin,* who have provided me abundance of guidance and advices during my research. His invaluable knowledge sharing and continual encouragement have helped me very much in the accomplishment of this research dissertation.

I would like to sincerely thank my career manager, *Mr. Ch'ng Sheng Cheang*, for giving me this priceless opportunity to pursue my Master degree in part-time. His graceful leniency in allowing me to manage my own work schedule has enabled me to achieve balance between my Master degree studies and my work.

I would like to express my gratefulness to my family, my mother, *Ooi Lay Leng*, and my brother, *Benny Tan Chin Chiu*, for all the supports they have been constantly giving me. Their dearest loves motivated and inspired me in my pursuit of happiness.

Lastly, I would like to express my earnest appreciation to all my course-mates from *MSc. Electronic System Design Engineering (ESDE) Batch 2015/16*, we have been great comrades together in this battle for betterment. Furthermore, I would also like to thank helpful staffs from USAINS, *Ms. Ong Bee Leng* and *Ms. Mazlin* in particular, for providing all the necessary facilitation for the course to be smoothly conducted.

ABSTRAK

Pusat data internet merupakan contoh rangkaian edaran dengan beban-beban dinamik. Algoritma pengimbangan beban amat diperlukan dalam sebuah pusat data internet untuk tujuan pengurusan dan pengagihan beban kepada nodus-nodus dalam rangkaian, algoritma pengimbangan beban membawa pelbagai manfaat kepada pihak pemilik dan pihak pengguna Cloud termasuk ia memanjangkan jangka hayat komputer pelayan dan meningkatkan kualiti perkhidmatan Cloud. Satu algoritma konsensus purata menggunakan kawalan PID telah dikemukakan dalam kajian ini. Algoritma yang dikemukakan diujikan dengan rangkaian yang berbeza bilangan nodus dan keputusan dibandingkan dengan algoritma yang sedia ada. Justerunya, alat pengimbang beban yang direka juga diujikan dengan topografi graf yang berbeza-beza untuk kajian keupayaan penyesuaian diri alat pengimbang tersebut. Keputusan prestasi yang diperolehi lebih baik berbanding dengan model PI yang sedia ada.

ABSTRACT

An internet datacenter is an apt example of a distributed network with dynamic loads, and a load balancing algorithm is needed in a datacenter to manage and distribute the loading to all server nodes appropriately. The implementation of a load balancer brings benefits to both the datacenter owner and Cloud users as it prolongs the lifespan of servers and improve the quality of service. An average consensus algorithm using PID control is proposed for the implementation in a network of server nodes and the result is compared to an existing model. The proposed algorithm is simulated with different number of nodes and edges. Furthermore, the designed load balancer is subjected to different graph topologies to test its adaptability. The performance results are found to be better than the existing model using PI.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CPU	Central Processing Unit
Digraph	Directed Graph
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
I/O	Input and Output
IAE	Integral Absolute Error
IDC	Internet Datacenters, also as Internet Data Center
IT	Information Technology
PID	Proportional, Integral, Derivative

LIST OF SYMBOLS

v_i	Vertex of *i*
V	Array of vertices
e_i	Edge of *i*, for undirected graph
e_{ij}	Edge from v_i to v_j, for directed graph
E	Array of edges
q_i	Load of server *i*
$\dot{q}_\iota$	Derivative of the load of server *i*
L	Laplacian matrix
l_{ij}	Element *(i, j)* of the Laplacian matrix
A_d	Adjacency matrix
a_{ij}	Element *(i, j)* of the Adjacency matrix
Ξ	Incoming request rate
O	Completed request rate
t	Time
$U(t)$	Control Term or Control Effort
K_p	Proportional Gain Coefficient
K_i	Integral Gain Coefficient
K_d	Derivative Gain Coefficient

CHAPTER 1

INTRODUCTION

1.1 Introduction

The technology of cloud computing is expanding exponentially in recent years as there is an increasing number of consumers boarding the bandwagon. The popularity of Cloud computing taps on the convenience of internet bandwidth and virtual storage space available today. It enables consumer to have fast and connected data sharing capability without owning the cumbersome facilities.

The datacenters experience drastic changes in loading at different time. Without a good load balancing algorithm distributing the workload to servers appropriately, a datacenter could be overloaded or underutilized, which means wastage of energy and resources. Hence, a good load balancing algorithm is a critical feature in datacenter's management system, it is required to increase working efficiency of a network of servers.

There are many load balancing algorithms proposed by researchers and implemented, each with different capabilities and limitations. Every load balancing algorithms can be categorized into one of static, dynamic, or hybrid family. The application of load balancing algorithm also differs based on design, some of the algorithms work best for centralized network; while some are more suited for distributed network.

Furthermore, there are many considerations when designing a load balancing algorithm. An algorithm may focus on only one aspect or multiple aspects of the servers during implementation. Aspects such as network traffic, thermal condition, and CPU process time are commonly used as load balancing parameters, and the algorithm may makes adaptive changes according to changes in these aspects.

The modern internet datacenters can be modelled with the graph theory, a mathematical theorem that studies relationship between objects. An internet datacenter can be seen a graph, with servers being the vertices and network as the edges. The graph theory originated in the 18th century as the scholars were finding solutions to the Königsberg Bridge Problem. Since then, the graph theory has received extensive studies and found its wide application in fields of electrical, telecommunication, control, etc.

The graph theory provides a framework which could facilitate the research of load balancing algorithms. Various modern control theory, including load balancing algorithms, have been modelled using the graph theory. Among many load balancing theorems, the average consensus algorithm is the focus in this dissertation.

The average consensus algorithm is the set of rules specifying how individual nodes in a network can reach an agreement based on a quantified amount of information. The intention of average consensus algorithm is to obtain an eventual common decision by the group of nodes. The average consensus algorithm has been widely studied and applied in various technologies such as multi-vehicular cooperative control and network load balancing.

1.2 The Problem Statement

In an internet datacenter, a “workload” or “task” is defined as the incoming network traffic into the network and is waiting to be serviced by the servers’ processor. The workload could be a webpage HTTP request, a file transfer FTP request, or any computational request that need to be addressed and served.

Without a controller in place, the traffic comes into the network and would goes to servers unregulated, thus servers which are high in service chain would have to gobble up most of the workload. Conversely, servers which are placed low would have little workload or none. Hence some systems, being physical or virtual, could be overloaded while some others are idle most of the time.

The load balancing algorithm is designed to address the issues mentioned above. It works as the manager cum controller that regulates the network traffic with the purpose of distributing workload to all subordinate systems by an appropriate scheduling. The intention of the implementation is to ensure all server nodes in the network are having an optimal workload and throughput. The objectives are to increasing efficiency of the datacenter and eliminate wastage of power and resources.

In this research, several existing load balancing algorithm are to be studied and analyzed for their advantages, disadvantages, and special features. So that an improved algorithm could be proposed, simulated and tested for justification.

1.3 The Research Objectives

The objectives of this research are as follow:

a. To propose and simulate a consensus-control-based load balancing algorithm which could improve the efficiency of a network of server nodes in comparison to the protocol proposed by M. Talavera (2014).

b. To characterize the performance of the proposed load balancing algorithm according to different network server topology.

c. To demonstrate the robustness of the load balancing algorithm against bounded disturbance in the network.

1.4 The Research Scope

The scope of research briefly covers the following:

Field:	Control Theory
Main branch:	Load Balancing Algorithm for Network
Specific branch:	Dynamically Distributed Load Balancing Algorithm
Specialized branch:	Average Consensus Algorithm

As part of the research, relevant topics such as Graph Theories, the Diffusion Scheme for Static and Stochastic models, and the PID control theories will also be studied for in-depth comprehension.

1.5 The Dissertation Outline

Chapter 2 is the Literature Review of the topics discussed in this dissertation. This chapter starts with presenting the background of datacenters and importance of load balancing. Followed with the review of various literature, mainly on the topics of server network, basics of load balancing, existing load balancing algorithms, the fundamental graph theories, the graph Laplacians, the concept of Averaging Consensus, and the model of Average Consensus Algorithm for server environment.

Chapter 3 is the Methodology of this research. It covers topics such as the fundamental average consensus theorem and different graph topologies to be used for the simulations. The proposed algorithm will be introduced and explained. Furthermore, the methodology of the experiment will be presented, which includes the simulation design, simulation setting, and the simulation environment. In this research, the experiment will be conducted based on 4-nodes and 8-nodes digraphs.

Chapter 4 is the Results from this research. The experimental results generated from the simulation tool are presented in this chapter. Moreover, critical analyses are made based on the empirical findings from the simulation results. Some comparisons and discussions are also produced for the purpose of evaluation and justification.

Chapter 5 is the Conclusion of this research. A conclusion is drawn from the findings and results from previous chapters. Additionally, expectation of possible future works will also be included as an end note.

CHAPTER 2

LITERATURE REVIEWS

2.1 Introduction

Cloud computing is a recent trend in information technology that moves digital data from desktops and laptops to large scale internet datacenters with greater computing power, better efficiency, and virtualization capability (IEEE Network Magazine, 2011). The trend can be classified as a new paradigm for dynamic provisioning of computing service supported by state-of-the-art datacenters. Today, almost a quarter of American companies are already using cloud-based application for virtual computing and data storage, and more companies are planning to join the trend.

Essentially, cloud is an infrastructure which provides platform and software as services that are made available to consumers in a pay-as-you-go model (Anton Beloglazov *et al.*, 2012). Cloud offers significant benefit to IT companies by relieving them from the necessity of setting up basic physical hardware and software infrastructures. Furthermore, consumer store and access data in the internet without the knowledge of the physical location of the servers and storage devices. Thus, the namesake “Cloud” appears virtually to its users.

Cloud is a collection of parallel and distributed systems (Mayanka Katyal *et al.*, 2013). There are three types of cloud: The private cloud, public cloud, and hybrid cloud.

Private clouds are owned by enterprises and business units for their own use. In a private cloud, the data and resources are kept strictly within an internal network and typically protected by a firewall. The client organization manages the cloud instead of the service provider, hence they have more freedom to the customization, upgrades, and security setting of the cloud to suit their need.

Public cloud may be owned or hosted by a service provider and is made available to general public use, some examples of public cloud service providers are Google Cloud and Microsoft Azure. While public cloud offers higher level of efficiency in shared resources, it is more vulnerable in security terms (Mayanka Katyal *et al.*, 2013). The security and privacy of public cloud have been major concerns of Cloud service provider. Public cloud service provider typically provide reliable access control to the user so that user's data are not publicly accessible. (Shiny, 2013)

Hybrid cloud is a cloud environment which an organization owns, provides and manages a private cloud, while having some other provided externally from public cloud. It is a combination of both private and public cloud, and carries the characteristics of various cloud models (Shiny, 2013). The advantage of hybrid cloud is the ease of moving data around from internal or private interface to public interface.

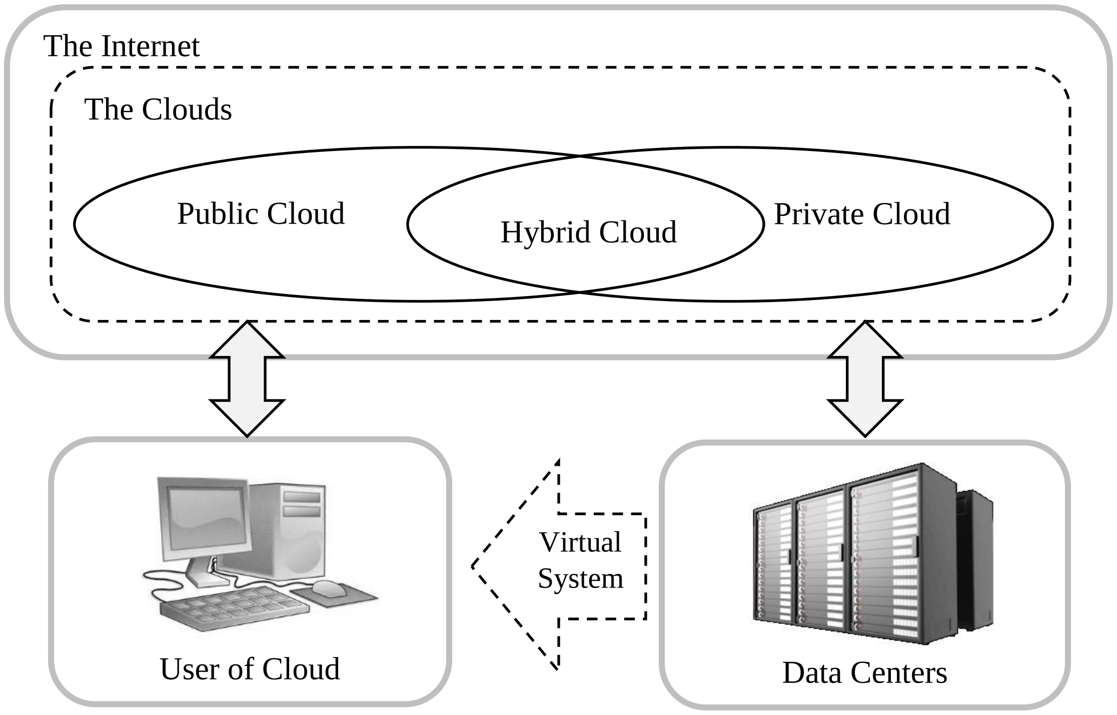

Figure 2.1: The Cloud as a collection of parallel and distributed systems.

The main component that forms the Cloud is the internet datacenter (IDC), it is metaphorically the backbones of cloud computing. Internet Datacenter (IDC) is a physical repository facility where a collection of servers are located, it is also known as Server Farm or Server Center.

In an IDC's server room, thousands of server racks line the sides of aisles, forming a city of thousands of server systems. In every rack, the individual servers are connected with network cables, which is further connected to larger network switches to form a huge network of systems. Each physical server is usually booted with multiple virtual Operating Systems to maximize the utilization of computing resource that single physical server provides.

2.2 The Distributed Network of Servers

Distributed network is a collection of self-governing computers that are connected via the same network. The network is facilitated with distribution middleware that coordinate resource sharing and traffic regulation, so that user of the network can perceive the entire network of computers as a single computing unit (Pathan A. *et al.*, 2011).

As the servers nodes in a distributed network attain certain degree of autonomy themselves as each system must have the capability to manage tasks with its own authority. There is no single processing element in a distributed system that is solely responsible for managing the entire load of an application. Instead the processing overhead is shared/distributed among all available server nodes. The prime benefits of distributed system are their combined elevated performance, accessibility, and flexibility (Shivaratri *et al.*, 1992).

The presence of a middleware, such as a load balancer, to coordinate and regulate workload among the server nodes in a distributed network signifies that the whole network has the ability of self-regulating/healing, fault tolerance, and an improved network reliability. If one of the server nodes is experiencing downtime, its workload can be quickly redirected to other available connected nodes without affecting the overall network integrity. Other than sharing of data and I/O devices, distributed systems also share the nodes' computational power and processing capability among others, which improves the performance of whole system. (Payal B. *et al.*, 2014)

Another benefit of distributed network is the scalability. (Payal B. *et al.*, 2014) Distributed network is able to expand to meet new needs. New servers can be added to the existing network and form new nodes without having to shut down the entire network. Similarly, a server can be removed when required. This scalability brings remarkable conveniences for existing IDCs to growth and cope with expanding demand, the convenience is especially visible when an IDC is moved fully or partially from one place to another, and when a defective server needs to be replaced without shutdown the whole rack of servers.

However, distributed network requires workload between interconnected servers be managed and regulated, and this requires a load balancing algorithm. Next section will elucidate the load balancing algorithm.

2.3 The Load Balancing Algorithms

The web servers in IDCs are known to have very drastic change in service request continually, resulting in an ungraceful profile for service load and long awaited queue of requests. IDC without a load balancer in place would see some systems constantly experiencing high load, while some other idle, this uneven distribution of workload lowers the overall efficiency of IDC, increased power consumption, and could bring bad reputation for the IDC owner of the slow service. It is crucial to have proper load balancing algorithm for IDCs to increase the service efficiency and to lessen the energy wastage.

The fundamental idea of load balancing in IDCs is to introduce elasticity elements into the server's service policies with the objective of maintaining a feasible computing capability with the least idle time and most utilization, thus enabling an efficient use of computing resources in IDCs. From an example provided by Xu Cheng *et al.*, the requests for a messenger software service spikes and fluctuates with respect to user login volume from time to time (2013), hence it is important for the software to have an elastic resource management to deal with the drastic changes demands.

The IDC's servers would have different loading if not regulated, the load refers to CPU load, memory used, delays, or network load waiting to be serviced. The unregulated traffic could cause some server to be heavily loaded while some too lightly. Hence, a load balancing algorithm is required to notice this unbalanced distribution of workload and provides a viable policy of traffic management so that one or more servers, network interfaces, hard drives or other computing resources have optimized utilization and job response time.

The load balancing is defined as a process of allocating the total work load to the individual nodes of the distributed systems to improve resource utilization and response time, also avoiding the condition in which some nodes are overloaded while others are under loaded or server failure. (Suriya Begum *et al.*, 2013) It is a new technique that facilitates network and resources distribution by providing a maximum throughput with minimum response time (Rajwinder K. *et al.*, 2014 and Chaudhari *et al.*, 2013).

The load balancer takes in requests from clients, usually other computers on the same network, and distribute the requests across multiple servers it manages based on the current status and loading of the servers. With the critical role of a load balancer, its algorithm has to be designed properly to avoid data lost (Shiny, 2013).

The main goals of load balancing algorithm listed by (A. Goscinski, 1991) are as follow:

a. To achieve greater overall improvement in system performance at a reasonable costs.
b. To treat all workload in a system equally regardless of their origin.
c. To have fault tolerance in the system
d. To have the ability to modify itself in accordance with any changes or expansion in the distribute system configuration. (This is applicable for Dynamic Load Balancing only)

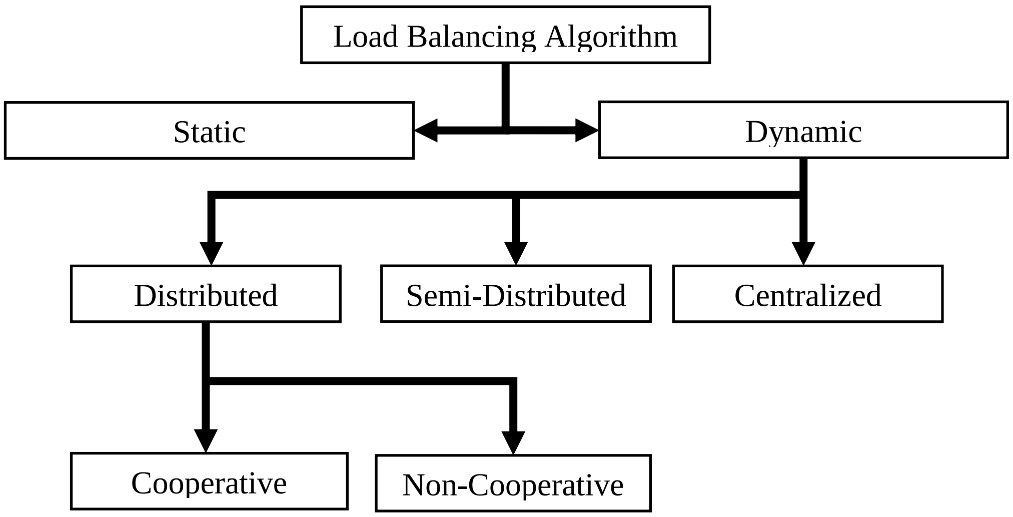

Figure 2.2: The categories and types of load balancing algorithms.

Load balancing algorithm can also be categorized according to the initiation types. The initiation types commonly seen are:

a. Sender Initiated – The load balancing algorithm is initiated by the sender node.

b. Receiver Initiated - The load balancing algorithm is initiated by the receiver node.

c. Symmetric – The combination of the both initiation types above.

The initiation types are the approaches to start a load balancing algorithm. The overloaded node initiates load balancing algorithm to distribute out its load is called Sender initiated. In Receiver initiated, an under-loaded or lightly-loaded initiates the load balancing algorithm to offer its willingness to accept new job.

Load balancing was identified as a crucial factor to allow Cloud computing to scale up to increasing demands (Martin Randles *et al.*, 2010). With the efficiency, practicality, and expandability of load balancing algorithm on an ecosystem of systems, the datacenters can be expanded and grow while maintaining its quality of service, cost, and security.

2.4 The Comparison of Static and Dynamic Load Balancing Algorithms

In general, all load balancing algorithms that operates based on system status are categorized into static methods and dynamic methods. Static methods make decision based on the static information of the systems. Conversely, dynamic method make decision based on both the current state and information of the systems.

The design of a static load balancing method is determined *a priori*, and the responding behavior is predestined (Ali M. Alakeel, 2010). Behavior of this approach is not altered with respect to the load transition or the current state of the network (Cybenko G., 1989). Static load balancing methods are non-preemptive, which means that once the load is allocated to a node it cannot be transferred to another node after execution commenced (Payal Beniwal *et al.*, 2014). Hence static load balance policy lacks the flexibility to respond differently with respect to changes in system loads. However, these methods requires less communication and computation, hence reduced the execution time and run-time overhead (Payal Beniwal *et al.*, 2014).

Conversely, dynamic load balancing method change its distribution policies with respect to different load profile and current state of the systems. The methods are adaptive, flexible, and require no prior knowledge of the incoming workload. Due to its nature, dynamic load balancing is more complex than static load balancing, it is also the more popular load balancing policy used contemporary.

Dynamic load balancing methods are applied in three forms: Centralized, Distributed, and Semi-distributed.

a. Centralized – A single node in the network acts as the "Central Node" and is responsible for all the load distribution to all other nodes.

b. Distributed – The responsibility of load balancing and job distribution is evenly distributed to all nodes in the network.

c. Semi-distributed – The network is segmented into clusters, where each cluster is centralized. The load balancing is achieved through the cooperation of central nodes of all connected clusters.

In dynamic load balancing algorithm for distributed network, the algorithm is further categorized under Cooperative and Non-cooperative forms.

a. Cooperative – Nodes work together to achieve a global objective. The nodes cooperate with each other to meet a common system performance goal, such as to improve system's overall response time.

b. Non-cooperative – Each node works independently toward a local goal. The node makes decision of job scheduling and migration based on local status and resources independently, such as to improve a node's response time.

There are three types of dynamic load balancing methods: Central Queue algorithm, Local Queue algorithm, and Least Connection algorithm.

2.5 The Mechanism of Dynamic Load Balancing

A dynamic load balancing algorithm makes load distribution decisions based on the current workload information at each node in the distributed system. Hence, a mechanism in the dynamic load balancing algorithm responsible for collecting and managing system information is required in the algorithm. This mechanism is referred as Information strategy.

The information strategy acts as the information center for the algorithm. It is responsible for providing location and transfer strategies at each node with the necessary information needed to make their load balancing decisions. There is a tradeoff between the amount of information provided and the frequency of the provision as the more amount of information the strategy provides, the more traffic it generates and thus the overhead is increased. (Ali M. Alakeel, 2010)

Moreover, a mechanism to make decision of which workload is eligible for redistribution is required for the algorithm. This mechanism which selects a workload for transfer from a local node to a remote node. Some research further separate this mechanism into two parts: Transfer strategy and Selection strategy.

The transfer strategy determines the condition under which a task should be transferred. It typically includes job migration and job rescheduling, where migration means suspending an executing job, transferring it to another processor and resumes the execution from halted point. The transfer strategy determines which node becomes eligible to act as a sender or a receiver.

The selection strategy is sometimes merged with transfer strategy as one mechanism. Discretely, selection strategy is the strategy which decides a node as a host, and selects a job to be transferred. This mechanism must be sophisticatedly designed as the important parameters of a job such as execution time, size, I/O, and memory requirements are only known at execution. A selection strategy considers the following factors (Sachin Kumar *et al.*, 2012) when selecting a job:

a. The overhead incurred by the transfer. The overhead should be minimal hence smaller tasks are preferred in selection.
b. The job should be long lived. Tasks with longer processing time or with higher priority are preferred as it is worthwhile for the overhead incurred per transfer.
c. The number of location-dependent system call made by the selected job. The number of location-dependent system call should be minimal as the job uses resources which are location-dependent.

There are many different implementations of transfer strategy. The simplest approach would be making decisions independently of the job's parameter except consideration of the job queue length in local node, a job is transferred only if the queue length exceeds a certain threshold. The main drawback is the inflexibility and the disregard of job sizes. Other approaches involves use of estimation and statistical prediction, such as using Fuzzy Logic, to approximate the job's parameters when making decisions. (S. Chowdhury, 1990 and A. Karimi *et al.*, 2009)

The last mechanism in a dynamic load balancing algorithm is the mechanism to determine which location or node for a workload to be transferred to. This mechanism selects a destination for the transferring workload is known as Location strategy. This mechanism is the executional stage of a load balancing algorithm, where it located a lightly-loaded node to share the load of a heavily-loaded node. The most common consideration in the decision is the current work load status at a node, which represented by CPU Queue Length (The total number of job waiting to be serviced by CPU and the one in servicing). Some approaches used in the location strategy are random location, probed location, and negotiated location (Ali M. Alakeel, 2010).

In random location strategy, the local or central node selects a remote node randomly and transfers the job there for execution. Upon receipt of the new job, the remote node executes the job if the job queue is below a predefined threshold. Or else, the new job is further transferred to another node. However, to avoid the situation where a job is constantly ping-pong around the nodes, a limitation on the number of transfer is usually imposed, the last node at the limitation is forced to take up the new job regardless of its job queue.

Probing location strategy identifies certain number of nodes and probe them for their status, one node is picked from the list for the transfer of new job. The criteria of selection include the job queue length, service time, and response time. This strategy is subcategorized into different sub-strategies: Threshold, Greedy, and Shortest. (D. L. Eager *et al.*, 1986)

In Probe-Threshold, a random remote node selected and is probe to see if the transferring of new job would cause it to go above its threshold. If not, the job is transfer to it; else, another node is identified for probing. The Probe-Greedy is a variation of threshold strategy, it uses a cyclic probing of nodes instead of random node. The Probe-Shortest strategy select a remote node randomly, and probes it to determine the job queue length currently at the remote node. A remote node with shortest job queue length is selected for the job transfer. (D. L. Eager *et al.*, 1986)

In negotiation location strategy, nodes negotiate with each other to select a suitable node for taking up new tasks for load balancing purpose. This strategy is commonly applied in distributed dynamic load balancing algorithm. Two subcategories are Bidding and Drafting. (J. A. Stankovic *et al.*, 1984 and L. Ni *et al.*, 1985) in bidding location strategy, an overloaded node broadcasts request message which contains its load information to all other nodes. Only lighter-loaded nodes respond to the request and answer it. Then the requesting node shall select the lightest-loaded node that answered and both initiate a negotiation on the transfer of job.

The drafting location strategy is the inverse of bidding strategy, where the lightly-loaded nodes which ready to take up more tasks broadcast their status across the network and initiate negotiation with other heavily-loaded nodes. Comparing the two negotiation strategies, drafting strategy outperforms the bidding strategy when given the same environment. The major problem with the bidding strategy is that a lightly-loaded node might be overloaded as a result of winning many bids in a short time. But this can be solved with imposing a limit on number of winning bids. (L. Ni *et al.*, 1985)

Thus, a dynamic load balancing algorithm is composed of three main components: the Information strategy, the Transfer strategy (includes Selection strategy), and the Location strategy. The relationship between the three components are depicted as followed:

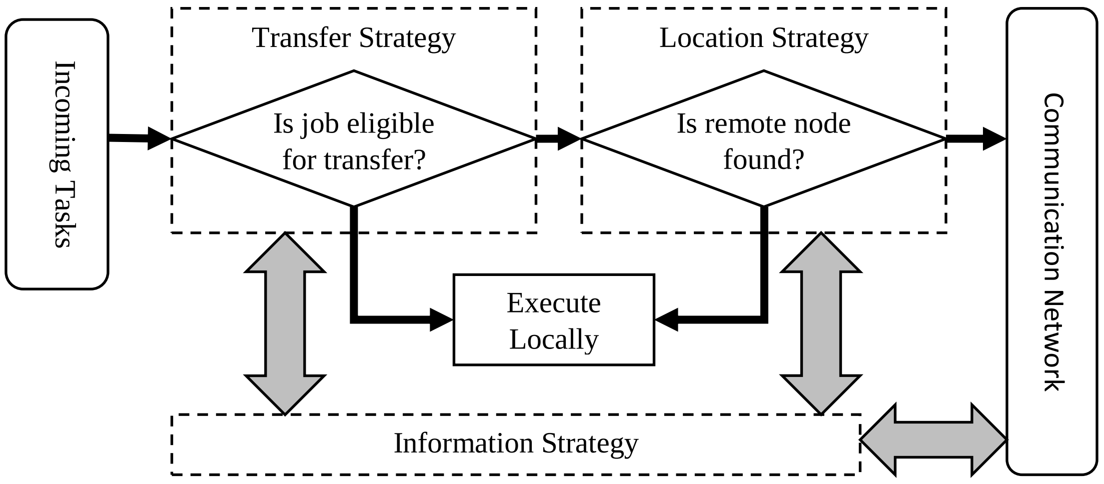

Figure 2.3: The relationship between transfer, location, and information strategies.

The incoming tasks are intercepted by the transfer strategy, which decides should the job be transferred for load balancing purposes. If the decision is made to transfer the job, the location strategy decides which remote node to get the job. The information strategy provides useful information to both transfer and location strategies to enable the decision making processes (Ali M. Alakeel, 2010).

The different categories of the dynamic load balancing algorithm are depicted in the figure below:

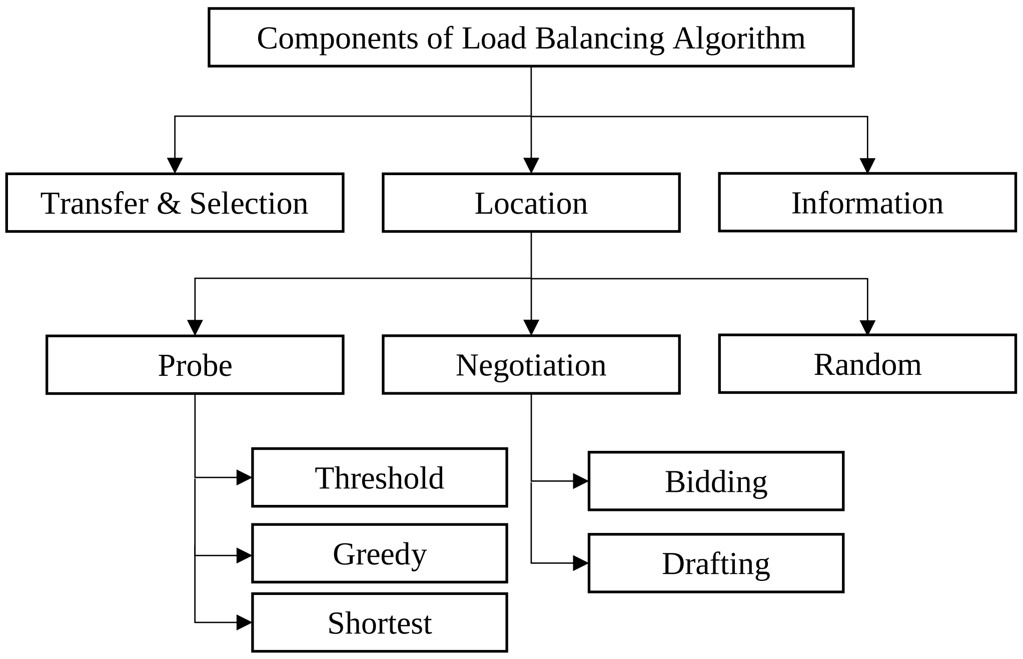

Figure 2.4: The mechanisms of dynamic load balancing algorithms.

2.6 The Assessment Metrics of a Load Balancing Algorithm

The performance of a dynamic load balancing algorithm is measured by several metrics (Sachin Kumar *et al.*, 2012), some of which are listed below:

a. Communication Overhead – The status information which every node has to be conveyed, including communication, resource utilization, and migration of job involved in the load balancing. It should be minimal for system to be efficient.

b. Load Balancing Time – The amount of time that elapses between the job arrival time and the time which the job is accepted by the remote node. It should be optimally short.

c. Scalability – The ability of the algorithm to perform load balancing with any finite number of nodes.

d. Fault Tolerance – The ability of the algorithm to perform uniform load balancing in spite of some failure in nodes and links.

e. Reliability – The ability of the algorithm to schedule tasks in predetermined time.

f. Stability – A characterization in terms of delays in the transfer of information between nodes, and the gains of the algorithm.

g. Throughput – The number of tasks or tasks which have been completed. It should be high for better performance.

2.7 The Existing Load Balancing Algorithms

There are many load balancing algorithms in existence, mainly grouped under static and dynamic according to the algorithm's nature. As examples, three types of static load balancing algorithms are given by Payal Beniwal *et al.* (2014) as followed:

a. Round Robin Algorithm

The round robin algorithm is performed in the namesake pattern of distributing load in a circular round robin fashion to a chain of nodes. The assignment of load and each node are not given any priority. Equal load is simply assigned from the first to the last node, then back to the first node for another cycle. This algorithm requires little communication, simple to implement, and node starvation free.

b. Central Manager Algorithm

The central manager algorithm has a central node acting as the manager for the network of nodes. It selects a managed node as the destination for assigning load. The load assignment can be based on different criteria such as load priority or node availability. Usually the node with the least workload will be selected for new load transfer. This algorithm requires constant updates between central manager and nodes for the best performance, hence inter-process communication level is high.

c. Randomized Algorithm

The randomized algorithm is static and the simplest in implementation. It selects node for load receiving randomly. As the information of nodes are not required in load assignment, the inter-process communication level is low, however, for the same reason the algorithm is best suited for a network of nodes with equal loads. The nodes are expected to have certain degree of variability in performance to avoid overloading.

All the static load balancing algorithms have limitations in terms of flexibility when dealing with different load levels and load assignment. Randomized algorithm, for example, lacks of the knowledge of the slave nodes' status, and could easily cause overloading for one node if that particular node is consecutive selected randomly. To give the flexibility to load balancing algorithms, a new family of algorithm is created –

Dynamic load balancing algorithms. Three of the existing dynamic load balancing algorithms compiled by Payal Beniwal *et al.* (2014) are listed below:

d. Central Queue Algorithm

In the central queue algorithm, the new load are queued in a cyclic First In First Out formation. New coming load is inserted to the last of queue. When an available node is requesting for load, the first load is removed from the queue and transferred to the node.

e. Local Queue Algorithm

The local queue algorithm is a process of load check-and-balance among the network of nodes. When a node has load number falls under a predefined threshold of minimum, it initiates the call of load transfer from other node. The remote nodes check their list of activities and pass some of their loads to the requesting node.

f. Least Connection Algorithm

In this algorithm, the load balancer tracks a record of the number of connection for each node. The node with the least number of connection is selected first for new load transfer.

There are many more existing load balancing algorithms which are not covered here. The algorithms have different nature and implementation for different requirements and environments.