Introduction

The great explosion of Internet and the rapid advancement of the related technologies, such as the World Wide Web (WWW) and Java, have created an interest of researchers in the development of Internet based system management and application services. Today, web services are widely involved in all aspects of daily life as millions of users use Internet. These users appeal for high scalability, availability, and reliability of hosts (nodes) to provide rapid response and high throughput for their applications running at any time. In this scenario load balancing is a vital concept. Basically it is the distribution of the total computational load across available hosts in the network. These hosts may have centralized control (Cluster and Grid) or may be without centralized control (Ad Hoc networks). The distributed hosts can be organized in different scopes. They can be integrated into a cluster of hosts linked via local area network to act as a powerful server or can be deployed at different sites over wide area network to form a part of the Grid. Also the hosts may be without any centralized control viz., Ad hoc networks. The hosts forming the network may be heterogeneous in terms of environment processing power, storage capacity or some other metrics. Thus, load balancing is essential for getting the quick response time for an assigned task on heterogeneous networks.

Load balancing is an important technique for improving distributed system performance by considering the group of hosts in the system to share
their workloads. It results in a better utilization of host resources, a high system throughput and quick response time of user requests. In load sharing, the incoming client requests are evenly distributed among the participating hosts. The load on an overloaded host is transferred to an under loaded host to enhance the system throughput. Effective load balancing among the group of hosts in a distributed system relies on accurate knowledge of the state of the individual host. This knowledge is used to judiciously assign incoming computational tasks to appropriate hosts, by using specific load distribution policies [1], so that the requests can be processed quickly. The purpose is to avoid performance degradation caused by load imbalance. This is achieved by allowing the hosts to cooperatively monitor the global system load and distribute / re-distribute the load according to the load sharing algorithms which can be static or dynamic and can have either centralized or distributed control.

In static load sharing algorithms, decisions about the allocation of requests / tasks are made before execution. This is not practical for some systems and applications environments. Another limitation of this approach is that it does not take the fluctuation of the system load into account.

In dynamic load sharing algorithms, the system attempts to find the best host for an incoming computational request/task at the runtime. Decisions are made using the information on the current system states. However, this approach requires gathering and maintaining system state information because in order to achieve load sharing, a snapshot of the global load information in a specific time frame must be acquired.

In centralized approach, the global state information is collected or estimated at a single host which makes request (task) distribution decisions based on the collected information. This approach may impose fewer overheads for maintaining state information, but has lower reliability. Failure
of the central server will make load sharing inoperable. Furthermore, a centralized approach is not appropriate for large scale systems distributed over a wide area, since it requires all other hosts to first contact the central server. This incurs a long delay.

In distributed approach, each host in a group communicates with the others and makes its own decisions for sending the requests to other hosts in the group, or for obtaining the requests from them.

1.1 Motivation

For generalizing a system, portability is a big issue. Existing load balancing solutions are based on the Message Passing Interface (MPI) communication model, using various communication services such as unicast message sending and receiving, multicast protocols, and remote procedure calls. So, there is a large possibility of inaccuracy in decisions due to the use of outdated information. This is because message passing based schemes need to wait for load information sent by other hosts. As a result, response with accurate global load information cannot be guaranteed. Thus, it is difficult to accommodate dynamic changes in the systems that are adaptive and scalable.

Some of the achievements related to load balancing are as follows: Static and dynamic load balancing in parallel and distributed computing have been presented by researchers [4-12]. Homogeneous and heterogeneous load balancing schemes are also been proposed [13-15]. A variety of load balancing schemes having centralized and distributed control are presented on cluster and grid recently [16-30].

Besides this there are still several key challenges- high throughput, quick response time, load distribution, selection of appropriate load balancing policy, fault tolerance, resource discovery & resource utilization, design of routing schemes for load balancing, service discovery, security and intrusion detection.
In order to overcome the above challenges, researchers are using Mobile Agent Technology (MAT) for load balancing. Mobile Agent (MA) is a software program that can migrate on its own or under host control from one host to another in a heterogeneous network and continue execution without human interruption. In other words, the program running at a host is able to suspend its execution at an arbitrary point, transfer itself to another host, or request the host to transfer it to its next destination and resume execution from the point of suspension, is called Mobile Agent. After being dispatched, a MA becomes independent of the launching host and can operate asynchronously and autonomously [3]. MAs can be created and dispatched by individual hosts in the group for their specific purposes. They can also be owned by a system and shared among other hosts. It has been found that MAs are especially suitable for structuring and coordinating wide area networks and distributed services that require intensive remote real time interactions. MAs follow the predefined itinerary consisting of $n$ hosts, $Host_0, Host_1, ..., Host_n$ and a parent host, Home.

![Figure 1.1: MAs movement and execution across different hosts](image)

Home launches the MA to $Host_0$ reliably, so we consider that the MA starts its execution on $Host_0$ and visits from host to host. After performing
the final task on $Host_n$, it returns the result to its parent host. These MAs execute load balancing algorithms to balance the load on the network.

1.2 Work Carried Out

In this research, we have considered the above challenges and developed a multiagent system for load balancing across the different types of networks. An extensive simulation is done using Platform for Mobile Agent Distribution and Execution (PMADE [2]) along with other tools depending upon the type of networks. For experimenting the developed system two types of networks are chosen: one which has centralized control (Cluster and Grid), second which is free from centralized control (P2P and Ad hoc networks). A set of MAs are implemented in the system for balancing the load. Each MA executes a specific type of algorithm for load balancing on a specific type of network.

A brief outline of the presented work is as follows:

- We have considered Input/Output (I/O) data intensive applications for load balancing on cluster and designed a load balancing algorithm for such applications. (Chapter 4, Section 4.2.2, page 46)

- There are applications running in parallel mode on grid and demanding resources that are on distributed servers forming the grid. For such applications, a multi level distributed tree algorithm is developed for the grid (Chapter 4, Section 4.3, page 53)

- A load balancing algorithm for P2P networks is also designed (Chapter 5, Section 5.3, page 70)

- Two algorithms are implemented for the ad hoc networks- one is for route discovery (Chapter 6, Section 6.2.2, page 81) and load balancing using MAs and the other is for service discovery (Chapter 7, Section 7.2, page 94)
• MAs based checkpointing algorithm is implemented in the system for providing the fault tolerance (Chapter 8, Section 8.4, page 110). For optimizing the resource utilization on the grid, system uses Genetic Algorithm (GA) (Chapter 8, Section 8.7, page 118)

• An intrusion detection algorithm is also integrated in the system for the ad hoc networks (Chapter 9, Section 9.5.2, page 141)

All the algorithms are implemented and tested over the developed SFLBMS and also a comparison is made with some existing systems.

1.3 Organization of Thesis

This thesis is divided into eleven chapters. Chapter 1 presents motivation, problems considered and finally organization of thesis. Chapter 2 contains literature review and overview of various mobile agent systems (MASs).

In Chapter 3, we have presented Secure and Fault Tolerant Load Balancing Multiagent System (SFLBMS) Architecture. This chapter is initiated with design issues for SFLBMS followed by various components of the architecture (Policy Manager, Resource Manager, Mobile Agents, and layers for communication).

In Chapter 4, we have presented the load balancing on Cluster and Grid. Two algorithms –one is MA Based Load Balancing Algorithm (MALB) for Cluster (Section 4.2.2, page 42) and the other is Multi Level distributed tree algorithm (MLDTA) for grid are presented(Section 4.3, page 53).

In Chapter 5, we have presented load balancing on P2P networks. A Load Balancing Algorithm for P2P (LBAP) systems is presented (Section 5.3, page 70).

Chapter 6 gives Load balancing and Routing using MAs in Ad Hoc networks. An algorithm named Routing and Load balancing (RLBMA) is also presented. The performance of RLBMA is compared with other existing routing protocols (Section 6.2.2, page 81).
Chapter 7 explores an MA Based Service Discovery Protocol (MASD) for discovering services in Ad Hoc networks. The performance of MASD is compared with respect to adaptive, push and pull methods used by some other researchers (Section 7.2, page 94).

In Chapter 8, Fault tolerance using optimal resource utilization on grid is presented. The algorithm for fault tolerance is named as MA based Checkpointing algorithm (MACP) (Section 8.4, page 110). For optimal resource utilization, GA concepts are used (Section 8.7, page 118).

In Chapter 9, security on Ad Hoc networks using two key metrics called trust and confidence is presented. These metrics form the basis for trust establishment in Ad Hoc networks. Also an algorithm for intrusion detection is explored (Section 9.5.2, page 141).

In Chapter 10, Secure and fault tolerant data dissemination is implemented over open network (Section 10.1, page 166).

Finally Chapter 11 concludes the work.

In the next chapter, we will deal with literature review and some existing MASs.