This chapter deals with the Secure and Fault Tolerant Load Balancing Multiagent System (SFLBMS) architecture and is organized as follows:

- Section 3.1 gives issues regarding SFLBMS
- Section 3.2 deals with SFLBMS architecture which includes agent manager, policy manager, resource manager, agent communication layers, and results with discussion
- Section 3.3 gives summary of the chapter

3.1 Issues
There are various issues regarding the development of a SFLBMS. Key issues are- load balancing across heterogeneous networks, fault tolerance, optimal resource utilization, route discovery and topology maintenance, service and resource discovery, security and intrusion detection in Ad Hoc networks.

3.2 System Architecture
Keeping in view of the above defined issues, we have designed and implemented PMADE based SFLBMS on the different types of networks {Figure 3.1}. The main components of the system are- Policy Manager, Resource Manager, Mobile Agents, Interface, and Agent-Agent communication layers. We have tested SFLBMS on heterogeneous networks (Cluster, Grid, Ad Hoc, and P2P) for the above defined issues. In addition of
balancing the load across heterogeneous networks for the above defined issues, the system also provides Fault Tolerance, Optimal Resource Utilization, Route and resource Discovery, Path Maintenance, service discovery, and Security.

![Figure 3.1: Secure and Fault Tolerant Load Balancing Multiagent System Architecture](image)

This system implements the agents for Load Balancing depending upon their roles. These agents execute the predefined policies for Load Balancing and
share the valuable information with each other through different layers using interface for the communication. The various components of the system are illustrated in the following sections:

3.2.1 Policy Manager

It selects one of the predefined policies for executing by MAs. The network selection plays a major role for categorizing and grouping the policies. The policies defined in the developed SFLBMS are as follows:

3.2.1.1 Initiation - This policy is further subcategorized into the following:

   Sender initiative (S-I)- The load balancing process can be started by underloaded or the overloaded host. If the overloaded host is responsible for finding other hosts to share its workload, the strategy is called sender initiative (S-I).

   Receiver initiative (R-I)- This policy is started by underloaded host to find a suitable receiver who can take its load. If a lightly loaded host is the initiator then it is called receiver initiative (R-I).

   Symmetrical Initiative (Sy-I)- This policy is a mixture of first two(S-I and R-I). It is a hybrid initiative called Symmetrical Initiative (Sy-I).

3.2.1.2 Virtual Server Selection- This policy is further subcategorized into the following:

   One-to-One- It is based on the one-to-one mechanism, where two hosts are picked at random. A virtual server transfer is initiated if one of the hosts is heavily and the other is lightly loaded.

   One-to-Many- Unlike the first scheme, this scheme allows a heavily loaded host to consider more than one lightly host at the time of transfer of load.

   Many-to-Many- This scheme is a logical extension of the first two schemes (one-to-one, one-to-many). While in the first scheme we match one heavily loaded host to a lightly loaded host and in the second scheme, we
match one heavily loaded host to many lightly loaded hosts, in this scheme we match many heavily loaded hosts to many lightly loaded hosts.

3.2.1.3 Resource Selection- There are various resources available in distributed systems. These resources are either available at local site or at global site. To achieve quick response time and high throughput, proper resource scheduling is always necessary. The resource selection is further subcategorized into the following:

- **Route discovery**- This policy is used to select the path from the available host disjoint paths.
- **Topology update**- Ad Hoc network is a dynamic network where hosts leave and join frequently. In these types of networks, topology changes quickly. Hence this policy is used to update the topology of the network
- **Cost computation**- There are various host disjoint paths available from source to destination. The best path from the available paths is that which is most reliable in terms of data transmission. This policy is used to compute the cost of the available path in terms of reliability.

3.2.1.4 Service Selection- This policy is further subcategorized into the following:

For the sake of simplicity, we have selected three different types of policies for service selection namely multicasting, topology update, and broadcasting. MASD uses multicasting to fulfill the growing demands of streaming multimedia, and video conferencing. By using the busy tone mechanism to increase multicast reliability, MASD uses a variable length control frame to stipulate an order for the receivers to respond, such that the problem of feedback collision can be solved. This mechanism is used to avoid delay in transmission of multimedia contents to the hosts. Topology changes occur frequently in MANET, so to face the challenges due to frequent topological changes each host sends a hello message to its neighbors.
3.2.1.5 Intrusion Detection- This policy is further subcategorized into the following:

Data Collection - The functionality of the data collection is to collect the security related data from various data sources and preprocess them for input to detection engines. There are many data collection modules in the Intrusion Detection Agent (IDA). Each module is responsible for collecting data from a particular data source. There are two different data sources: network packets and host audit trails.

Detection - Different detection techniques can be deployed in different detection engines in order to improve the detection performance. Misuse-based detection techniques operate based on the known attack scenarios and system vulnerabilities. In this work, the anomaly based detection is used for detection of intrusion.

Local Aggregation and Correlation Engine (LACE)- Because different detection techniques can be deployed in the IDA, it is necessary for the LACE to aggregate and correlate the different detection results before further transmission. The local IDAs for a mobile host are capable of operating in a standalone mode and detect attacks against the host. Since wireless ad hoc networks are constrained by bandwidth, energy consumption, and process capability, it is desirable to correlate the alert information on the local hosts first, before transmitting every alert across the network.

Global Aggregation and Correlation Engine (GACE) - The functionality of the GACE depends on the host type: if the host is a gateway host, its GACE utilizes the aggregation and correlation to combine the detection results from the IDAs of intrazone hosts in the same zone and neighboring gateway hosts. If the host is an intrazone host, the functionality
of the GACE is to distribute the outputs of its LACE to all the gateway hosts in the same zone.

**Intrusion Response**- The countermeasures taken by the intrusion response are different due to the different intrusions, network services, applications and confidence in the evidence. Possible countermeasures may include identifying the intruders, reinitiating the communication channels and excluding the compromised hosts from the networks.

### 3.2.2 Agent Manager

This manager selects the agents as per their requirement. These agents are not fixed but vary according to their role. Agents are divided into different groups and they are picked by agent manager depending upon their requirement in the specified network. Each agent has a role defined and executes the predefined policies. The generalized architecture of the developed SFLBMS consists of the following groups of agents which are used in heterogeneous networks:

#### 3.2.2.1 Load Balancing-

For balancing the load across different types of networks, we have developed a set of agents which are briefly introduced as under. Details about load balancing are available in Chapter 4.

**Load Index Agent (LIA)**- This agent calculates the load index (LI) of each resource (CPU, Memory, I/O) on a particular host. LI of CPU is the sum of remaining CPU lifetimes of the tasks running on a host. LI of memory is the sum of page fault processing time of tasks on a host. Similarly LI of I/O is the sum of I/O processing time of tasks on a host. The sum of load index of each resource is the total value function (VF) for a host and is used for load transfer by the respective agent.

**Load Transfer Agent (LTA)**- This agent is used for migration of task from heavily loaded host to lightly loaded one. It executes two predefined policies namely–local and global. The policy is chosen according to the
response time of task submitted for execution. If the response time of a task at the local site is less than the global site then local policy is executed otherwise global policy.

**Resource Management Agent (RMA)**- This agent is responsible for gathering information about each host’s resource requirement and passing this information to Resource Manager (RM), which makes its entry in resource database and allocates the appropriate resource to the requesting host.

**Routing Agent (RA)**- It is a stationary agent responsible for updating the routing table that resides at each host. RA carries a route vector table containing the communication cost from the assigned host to other hosts in the network. This table has a lifetime measured by the number of hops. It plays an important role in informing each host in the network about the addresses of other hosts and if failure of link for a particular host is detected, RA spreads route failure information over the network by flooding the updated table.

**Load Computation Agent (LCA)**- This MA is responsible for information gathering. It travels around the hosts and collects the load information, and propagates this information to other hosts.

**Directory Agent (DA)**- This agent is activated whenever an overloaded situation arises on a host. DA finds the suitable receiver partner for the overloaded host that launched it.

3.2.2.2 **Resource Discovery**- For route discovery and balancing the load across different types of networks, a set of agents are developed. Details about resource discovery are available in Chapter 6. A brief introduction about these agents is as follows:

**Route Discovery Agent (RDA)**- This agent keeps a record of host disjoint paths which are not deemed failed yet. As soon as the rating of a path falls
below a given threshold, the path is discarded from the Active Path Set (APS) (set of node disjoint paths) and accordingly a new path is added for future references. This agent executes route discovery policy.

**Topology updates Agent (TUA)**- This agent keeps record of one hop neighbors using the hello message technique. As in MANET, hosts leave and join the network in a random interval of time, so every host should have knowledge about its neighbors. This agent executes topology update policy.

**Cost computation Agent (CCA)**- This agent computes the cost of sending the messages to another host in the network. The cost is measured by the number of hops traveled by the message and bandwidth lost due to the presence of selfish host in between the paths. As the battery power of mobile host is a sacred resource in MANET, this agent plays an important role in cost computation in terms of bandwidth loss. This agent executes the cost computation policy.

**3.2.2.3 Service Selection**- For selection of service across different types of networks, we have developed a set of agents which are briefly introduced as under. Details about service selection are available in Chapter 7.

**Advertising Agent**- This agent actively broadcasts service descriptions already registered. The Policy Manager controls the rate of advertisements. Various policies are employed to adjust the rate of advertisement. For example, if the network is fairly static, then the advertisement rate can be slowed down. Also policy is event driven (Events represents the arrival or departure of hosts in network). Advertisements can also be assigned different priorities.

**Forwarding Agent**- This agent receives service advertisements and requests for service messages. Then it decides whether to drop or to propagate the advertisement based on the policy. To prevent broadcast storms, this agent uses multicast tree for selectively forwarding service
advertisements. For example, this uses to forward advertisements to more active or resource rich hosts in the network.

**Cache Agent**- This agent is responsible for handling remote advertisements, storing remote advertisements of services, handling requests to match services present in the cache. The Forwarding Agent, on receiving an advertisement might also decide to forward it to other hosts or broadcast the advertisement to all other hosts. Each advertisement contains a lifetime. When a new advertisement is received by a Cache Agent, the agent decides to either accept it or reject it. An advertisement is accepted only when there is sufficient space in the cache to hold this advertisement or when an old advertisement is removed from the cache based on the policy chosen.

**3.2.2.4 Fault Tolerance:** For fault tolerance across different types of networks, a set of agents are developed. Details about fault tolerance are available in Chapter 8. They are briefly introduced as under-

**Process monitor agent**- This agent monitors the state and starvation of a process in a task queue. It classifies a process state into a processing state, a stop state, a silent state, and an unknown state.

**Processor monitor agent**- This agent monitors the crash state of a processor (shutdown, power value) and the normal state of a processor. During the normal execution of a processor, this collects the used and the available CPU utilization.

**Network monitor agent**- This agent monitors communication bandwidth, communication latency time, network disconnection, and partition between its own host and connected hosts.

**Fault decision agent**- This agent decides the occurrence of a failure by analyzing state information of each resource and identifies a process failure, a processor failure or a network failure.
**Rescheduling agent**- This agent evaluates the performance benefits that can be obtained due to task migration and decides whether task migration occurs or not. This agent also decides a new resource allocation for tasks.

**State display agent**- This agent shows the state of each resource and the type of failures occurred. Also it decides whether task migration occurs or not. If this agent receives a rescheduling result for migration from the rescheduling agent, it requests to allocate new selected resources and restarts execution.

### 3.2.2.5 Security:
A set of agents are developed for secure data transmission across different types of networks. Details about security are available in Chapter 9. These agents are briefly introduced as under:

**Multi path Secure Routing Agent**- This agent executes the multipath policy which has multiple paths to combat the frequent topological change and link instability problem in MANET, since the use of multiple paths could diminish the effect of possible link failures.

**Secret Sharing Agent**- This agent executes secret sharing policy. In this policy, the secret message is divided into $N$ pieces such that in order to get message, the adversary must compromise at least $T$ shares. With fewer than $T$ shares, the enemy cannot learn anything about the message and has no better chance to recover the secret than an outsider who knows nothing about the message. This gives the desirable security properties.

**Share Allocation Agent**- This agent executes share allocation policy with the objective of maximizing the message security. It chooses a host disjoint path for secret allocation so that adversary can never get the message.
3.2.2.6 **Intrusion Detection:** For detection of intrusion across different types of networks, a set of agents are developed. A brief introduction is as under.

**Intrusion Detection Agent (IDA)-** This agent is the major component for intrusion detection. The internal model of this agent contains the following policies- the data collection, detection, LACE, GACE, and the intrusion response.

3.2.3 **Resource Manager (RM)**

The resources are managed by RM. Each task in execution has its own resource requirement, which is provided by RM. It keeps track of which resource is available at which host. Resources may be at local site or at global site. As soon as the demand of resources comes from the tasks in execution, these are provided by the RM. MAs execute predefined policies to provide the desired resources to the demanding task. In this process they also consume certain resource like memory, processor time, etc. So to keep track of the all these resources, RM is included in the architecture which fulfills the demand of resources as per the requirement.

3.2.4 **Communication Layers for Agents**

The architecture for load balancing consists of multiagent where each agent has a specific role to play and also has facility for agent interaction with each others through interface. This interface provides the facility for agent-agent communication. There are three layers for agent-agent communication. Resource and Data Management Layer- RM which is the key component of the system architecture operates on this layer. As discussed earlier it is responsible for managing the resources consumed by MAs when it executes on a particular site. On each layer the respective MA operates. Communication and Coordination Layers- These MAs communicate and coordinate using communication and coordination layers. The request an
agent receives from the communication layer is submitted to the coordination layer for further processing. Agents communicate by exchanging messages using mobile group approach through reliable communications channels [160].

3.2.5 Result and Discussion
The developed SFLBMS is capable of balancing the load across different types of networks (Cluster, Grid, P2P and Ad Hoc). This system uses set of MAs which are executing predefined policies for finding the load and resource requirement status at each host. It generates less message transfer complexity and overhead compared to other existing systems developed earlier [143-149]. Also it has a major impact on the efficiency of the networks with the following performance measurement metrics - response time, throughput, fault tolerance and network delay. This system is capable of achieving the dual goal of security and intrusion detection.

3.3 Summary
In this chapter, we have presented SFLBMS for balancing the load across different types of networks (Cluster, Grid, P2P, and Ad Hoc). Due to the support for mobility of agents in heterogeneous networks, it is able to locate the resource and service in Ad Hoc networks. Inter-agent communication is supported by the system to get the updated load information on each host. Hence the system is well suited to be used in heterogeneous environment not only for load balancing but for other issues like fault tolerance, path discovery, resource management, security and intrusion detection.

Next chapter deals with the load balancing in cluster and grid types of networks.