Load Balancing in P2P Networks

This chapter deals with P2P load balancing problems. A P2P network is that which provides a distributed directory (DD) ([76-79]) for information sharing among peer hosts. In such networks, a unique identifier is associated with each data item and host in the network. The identifier space is partitioned among the hosts to form the P2P network such that each host is responsible for storing all the data items that are mapped to an identifier in its portion of space. This chapter is organized as follows:

- Section 5.1 deals with the load balancing issues for P2P networks
- Section 5.2 explores the designed algorithm including problem formulation and assumptions
- Section 5.3 gives the implementation
- Section 5.4 deals with result and discussion
- Section 5.5 summarizes this chapter

5.1 Issues

P2P networks are based on hashing functions having an uneven load distribution problem. The objective of P2P load balancing is to balance the workload of hosts in proportion to their capacity to eliminate traffic bottleneck. The workload of a host is measured in terms of metrics like file size and traffic volume. Load balancing in P2P networks remains challenging because of their unique features of dynamism, in which hosts join and leave the network with passage of time. Also P2P networks are often highly heterogeneous. This requires a load balancing solution not only to distribute
the application load (e.g. file size, access volume), but also the load balancing overhead among the hosts in proportion to their capacities. Following key issues are identified for load balancing in P2P networks. A consistent hashing results in an uneven load distribution. Consistent hashing produces a bound of $O(\log n)$ imbalance of file keys between hosts, where $n$ is network size. In addition, factors like non-uniform file size, time varying file popularity, and host heterogeneity make the load balance problem even more difficult in practice. It is challenging because of the dynamic nature of P2P networks and time-varying load characteristics.

5.2 Problem Formulation and Assumptions

DD among peer hosts forms a Distributed hash table (DHT) network. It is a content addressable network that maps files to each network node based on a consistent hashing function. DD is constructed using Virtual Cluster (VC). Each VC has Virtual Server Host (VSH), together with a group of regular hosts, and VSH acts as a server to the others. In general, VSHs are hosts with high capacity and fast connections while regular hosts are hosts with low capacity and slower connections. VSHs in DHTs provide fast routing between the hosts. Each VSH operates as a server to its associated regular nodes.

VC arranges physically close hosts in one group by clustering. They take static gateways (or routers) as VSH. Network tools for finding gateway, such as trace route, are too heavy-weight and intrusive for use in large scale applications, because it generates excessive load on the network

Key is stored in a host whose Id is the closest to the key. A regular host is assigned to a VSH whose Id is closest to the host’s Id, i.e., regular hosts are connected to their physically closest VSHs since host Id represents host physical location. As a result, the physically close hosts will be in the same VC or nearby clusters. The consistent hashing for key assignment protocol
requires relatively little re-association of regular hosts to dynamically
designated VSHs, as nodes join and leave the system.

We use a nearest-neighbor VC technique to build each VSH routing table
in the VSH’s DD, i.e., it selects the routing table entries pointing to
physically nearest among all the hosts with IDs in the desired portion of Id
space. As a result, hosts in one VC are physically close to each other.

We do not assume a particular resource, but we assume that there is only
one bottleneck of resources in the system to be balanced. Each agent also has
a movement cost. Each agent that enters the system has an associated load in
terms of the number of bits required to store the agent, the amount of
processor time needed to serve the agent. We assume this cost is the same
regardless of which two hosts are involved in load transfer.

5.3 Algorithm
We have developed a Load Balancing Algorithm for P2P networks (LBAP)
{Figure 5.1}. The goal of LBAP is that every host would have the same
utilization. It is of primary concern that no host’s load is above the host’s
capacity. Also moving a large amount of load uses bandwidth and may be
infeasible if a host’s load changes quickly with respect to the time needed to
move it.

In LBAP load information is stored at the respective hosts in form of DD.
This DD is maintained by DA at each host. Also this directory is being
refreshed by LCA after a finite interval of time. Each directory has an ID
known to all other hosts in the network. Load is transferred from one host to
another subject to the following conditions-

- Load on the source host is above the predefined threshold
  value..........................(5.1)
- The destination host does not become
  overloaded...........................(5.2)
We have used the concept of VSHs for load balancing. A VSH looks like a single peer. The key advantage of splitting load into virtual servers is that we can move a virtual server from any host to any other host in the system. This

![Figure 5.1: Pseudo Code for LBAP](image)

operation looks like a leave followed by a join. A VSH transfer is initiated, if
one of the hosts is heavily loaded and the other is lightly loaded. This policy is easy to implement in a distributed manner. Each lightly loaded host can periodically pick a random task and then perform a lookup operation to find the host that is suitable for that task. If that host is a heavily loaded host, then a transfer may take place between the two hosts.

To execute the designed algorithm, following agents and policies are chosen- LCA, DA, one-to-one, one-to-many, many-to-many.

The algorithm consists of following two steps- Step1 explores hosts joining and leaving the network and construction of DD, Step 2 describes the load balancing.

5.4 Implementation

MA based load balancing scheme is executed on network consisting of 100 hosts over Fast Ethernet (100Mbps) and employing a decentralized JuxtaNet [164, 165] which is an open and decentralized P2P network model. The JuxtaNet is significant in the sense that it is an open, general-purpose P2P network model. JXTA’s JuxtaNet is abstracted into multiple layers namely core, service and application with the intention that multiple services will be built on the core. In fact, there is no constraint against the simultaneous existence of the JuxtaNet multiple services or applications designed for the similar purpose. As an example, just as a PC's operating system can simultaneously support multiple word processors, the JuxtaNet can simultaneously support multiple file sharing systems. It enables users to access each others resources in a decentralized manner. Category 5 (twisted pair) copper wire runs among the PCs and an Ethernet hub. Following metrics are chosen to present the developed scheme:

- Load movement factor- It is defined as the total movement cost incurred due to load balancing divided by the total cost of moving all agents in the system once
• Node utilization- It is defined as the node load by its capacity
• System utilization- It is defined as the overall system load by its capacity

5.5 Result and Discussion
We have studied the impact of designed algorithm on the metrics chosen as follows:

Impact of load movement on host utilization
Figures 5.2 and 5.3 present that even for high system utilization of 92%, the LBAP is able to keep more than 94% of the hosts underloaded while incurring a load movement factor of less than 0.08.

Figure 5.2: Impact of Load Movement Factor on host Utilization with varying System Utilization

Also as the number of directories in the system has only a small effect on chosen metrics for a particular load movement factor, i.e., default choice of
16 directories produces host utilization less than 3% than in the fully centralized case of single directory.

**Figure 5.3:** Impact of Load Movement Factor on host Utilization with Varying Number of Directories

**Impact of Number of Virtual Servers**

Figures 5.4 and 5.5 present host utilization and load movement factor as a function of system utilization for various numbers of virtual servers per host. The LBAP achieves a good load balance in terms of system utilization. In particular, the host utilization increases roughly linearly with system utilization. Second, while an increased number of virtual servers help to balance the load at fairly high system utilizations; its beneficial effect is most pronounced on load movement factor.
Figure 5.4: Host Utilization VS. System Utilization

Figure 5.5: Load Movement Factor VS. System Utilization
Impact of Host arrival and Departure

We considered the impact of the host arrival and departure rate on the chosen metrics. The arrival rate is modeled as Poisson distribution, and the lifetime of a host is chosen as Exponential distribution. We varied inter-arrival time from 10 and 50 seconds. To analyze the overhead of LBAP, we have studied the load moved by it as a fraction of the load moved by the underlying DHT due to host arrival and departure. Figure 5.6 presents load movement factor as a function of system utilization. The main point to take away is that the load moved by LBAP is considerably smaller than the load moved by the underlying DHT especially for small system utilizations. More precisely, with the default value of 12 VSHs, LBAP never moves more than 65% of the load that is moved by the underlying DHT. Figure 5.7 presents that by increasing the number of VSHs, the fraction of load moved by LBAP significantly reduces.

Figure 5.6: System Utilization VS. Load Movement Factor
**Figure 5.7:** Impact of Virtual Servers on Load Movement Factor

### 5.6 Summary

In this chapter, we have presented LABP for load balancing on P2P networks using MAs. This algorithm is applied to optimize the use of available resources. The LBAP handles heterogeneity in form of varying agent load and host capacity. Moreover, it handles dynamism in the form of continuous insertion and deletion of hosts. Simulation results show that LBAP is effective in achieving load balancing for system utilizations as high as 95% while transferring only about 5% of the load that arrives in the system. Moreover, it performs more effectively than fully centralized load balancer algorithm. In addition, we have found that heterogeneity of the system improves scalability by reducing the necessary number of VSHs as compared to a system in which all hosts have the same capacity. In the next chapter, we will focus on routing and load balancing in Ad Hoc networks.