Dynamic Multilateral Peering: An Efficient Mechanism for Controlled Resource Sharing

by

Kumaran Subramoniam

A dissertation submitted in partial satisfaction of the
requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

Faculty of Graduate Studies
University of Manitoba

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

Dynamic Multilateral Peering: An Efficient Mechanism for Controlled Resource Sharing

BY

Kumaran Subramoniam

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

Of

MASTER OF SCIENCE

Abstract

Peering systems are designed for efficient and fair load distribution among different autonomous resource sharing entities. Resource sharing entities allow computing resources such as disk space, memory space, network bandwidth, CPU cycles and specialized processing power to offer service to other entities on the network or to use the services offered by other entities. A working agreement is established between the resource sharing entities, and these working agreements that are established "offline" dictate how the peering systems operate, while sharing resources among different entities. In this thesis, we have designed a new peering system called, *Dynamic Multilateral Peering System*, in which the working agreement between resource sharing entities changes dynamically with change in the load on the sharing entities.

The peering system designed in this thesis is compared with a peering system called *Static Peering System*. Unlike dynamic peering system, static peering system has a fixed working agreement between different sharing entities; therefore, variation in load on resource sharing entities will not affect the sharing agreement. This problem of overlooking the load on sharing entities is addressed in dynamic peering system, where the peering policies among the sharing entities changes with the load. Pricing scheme is used in dynamic peering system to evaluate the value of a resource on an entity, and the value of the resource will change with the load. In dynamic peering system, working agreement between entities is determined by the value of the resource. We show that by changing the working agreement with change in the load increases the efficiency of the peering system. In this thesis, we examine the dynamic peering system with the following questions in mind: (a) Will dynamic peering system give better load distribution than static peering system? (b) What are the limitations of dynamic peering system? (c) What are

the different conditions in which a dynamic peering system outperforms a static peering system and vise versa? Load balancing is used as a baseline mechanism to compare the efficiency of different peering systems.

Acknowledgements

There are many people who helped me in pursuing my master's. First of all, I would like to thank my advisor Dr. Maheswaran, who gave an insight of my thesis work. I would like to thank my advisor, for guiding me with great patience, immense amount of support, and without my advisor's support this thesis would have gone nowhere.

I would like to thank *The University of Manitoba* and *Department of Electrical and Computer Engineering* for giving me the opportunity to pursue my master's degree. I would like to thank my committee members Prof. Rasit Eskicioglu and Prof. Ekram Hossain for spending their valuable time in evaluating my thesis. I would like to thank TR*Labs*, for providing me financial support and resources. I am very much grateful to TR*Labs* for providing me a great research environment.

I would like to thank my family and friends for providing me a great support. I would like to thank Rajesh for giving me valuable feedback on my thesis, and providing me great support throughout my studies. I would like to thank Maniy for constantly helping me in my research as well as other parts of my master's. I would like to thanks my brother Amudhan for constantly pushing me to learn new things.

Contents

List of Figures

Chapter 1

Introduction

Peering concept is successfully implemented and used in the Internet by different network service providers (NSPs). As an NSP by itself has limited capacity and reach, it is beneficial and necessary for it to establish working agreements with other NSPs, to extend its capacity and reach. These working agreements that are established "offline" dictate how the NSPs operate, while carrying each other's traffic. They outline the exchange of bandwidth capacities among the domains. To gain a better understanding of the peering process, let us examine the two motivations for peering: to extend reach and to extend capacity. The reach extension in the Internet, makes peering mandatory. Without working peering relationships, an NSP won't be able to route traffic to portions of the Internet. This can be crucial for smaller NSPs with correspondingly smaller spans. The capacity extension makes peering an optional requirement. Nevertheless, the additional capacities provided by the peering arrangements can provide alternative paths for a domain to route its traffic toward the destinations. This flexibility can be useful, when the domain is subjected to load spikes.

One of the main technology that has helped the world to shrink is Internet. As more and more people have started to use the Internet as a source for accessing and trading

information, the resource for satisfying these demands could not keep up. One possible solution for the increasing demand is to increase the supply, for which all the resources has it own threshold. This effectively means that there should be some way for an overloaded system to access the resource of an underloaded system. The above discussion leaves us with different systems peering with each other and these systems that peer with each other are called *Peering Systems.*

Hence, the questions that arise in our mind is: What means do the different systems use to peer with each other? One way of sharing resources among peers is through a network. The network can be a wireless network or a wired network, but ultimately, all the peers have to be connected through some network. This thesis considers that all peers in a peering system are connected through a network, and since the peering system is connected in a network, it is called as *Network Peering System*(NPS).

The situation in network peering systems is different than that in the Internet. In network peering systems, peering can be hardly considered mandatory. There are two contradicting observations. A domain should have sufficient resources to enter into a peering agreement with another domain. On the other hand, a domain with sufficient resources does not have to enter into a peering agreement. This implies the peering agreements should be optional. In [ZhK01], a new peering scheme was suggested, where each agreement had mandatory and optional peering levels that are independently set. To examine the drawbacks of this scheme, we need to consider the core requirements of peering in network peering systems.

Network peering systems are made of computing resources that are willing to co-operate in addressing the requirements of their clients. In non-trivially large network computing systems such as Grid computing systems, computing utilities and Peering overlays, the cooperation is among domains that are clusters of resources with the same ownership or administrative policies. Peering arrangements are necessary to coordinate

the inter-operation among the different domains such that a domain has control over what is being shared with other domains. The overall objective of peering is to maximize some global performance measure, while minimizing the impact on the local activities. More specifically, a domain wants to share its resources, when it is under-loaded and the resources are not critically required for local consumption; and don't want to share anything if it is over-loaded and the resources are critically required for local consumption. The peering policies define the parameters that can be used to determine, whether a domain is under-loaded or over-loaded. When a domain is under-loaded, we need to determine how much of its resources will it lend to other domains.

What is the gain for a domain to join a peering system? Why should a domain with sufficient resources should join the peering system? In the case of a overloaded system, it can use the resource of an underloaded peer in the peering system, and with the underloaded system it can get some revenue out of the unused resource. In some cases there are systems that need resources for preserving some backup information. In the case of systems that need to preserve information, the main advantage is to increase the redundancy of some important data. Peering system will be very useful in increasing the redundancy of data in a data archiving system. Making several copies of data is very important, but another important issue is to distribute the data in various autonomous domains. If one domain fails, then the data can be retrieved from another autonomous domain. In data preservation, reliability is a very important metrics, and reliability is measured for both local and global peering domains. There are domains, which have peering policy with one domain and share resources of some other domain.

Network peering systems are made of computing resources that are willing to cooperate in addressing the requirements of their clients. Peering arrangements are necessary to coordinate the inter-operation among different domains such that a domain has control over what is being shared with other domains. The overall objective of peering is to

maximize some global performance measure, while minimizing the impact on the local activities. More specifically, a domain wants to share its resources, when it is under-loaded and the resources are not critically required for local consumption; and don't want to share if it is over-loaded and the resources are critically required for local consumption. The peering policies define the parameters that can be used to determine, whether a domain is under-loaded or over-loaded. When a domain is under-loaded, we need to determine how much resources it will lend to other domains.

In this thesis, we propose a new peering system, which has peering policies that changes with load on the system. As the peering policy changes with the load in the peering domains, the peering system proposed in this thesis is named as *Dynamic Peering System.* Dynamic peering system uses economic model to determine the peering policy. The peering system proposed in this thesis is compared with the peering system proposed in [ZhK01], and load balancing is used as a baseline to compare the peering systems. More details on different peering systems are explained in chapter 3. The peering systems are simulated, and the results are compared to show the advantages and disadvantages of the peering systems.

Chapter 2

Literature Review

In this chapter, we examine projects related to peering systems and pricing algorithm. Section 2.1 deals with projects related in calculating the price of the resource. Section 2.2 talks about projects related to different peering systems.

2.1 Pricing Algorithm

2.1.1 Price Calculation Using Tâtonnement Process

In tâtonnement process, excess demand is used to determine the equilibrium price, where it is considered that the equilibrium price is reached when the demand is equal to the supply. The tâtonnement process is an iterative process and the process will not end until the equilibrium price is achieved. Price Tâtonnement process is an iterative process where the price changes in the same direction as excess demand. In [ChW98], pricing of the resources is done using the price-tâtonnement process. In [ChW98], WALRAS algorithm is used for the price-tâtonnement process. In WALRAS algorithm, interdependence of the resources are not considered. The main argument in the WALRAS algorithm is that, the price of a particular resource is dependent only on its own demand and the influence

of the price of the other resources might increase or decrease, and eventually all of the changes cancel each other.

In peering systems, there are situations where the demand is less than the supply. In this case, the process waits until a demand is found for the existing supply and if there is no demand for the existing supply then the process will run into an infinite loop. In this pricing scheme, one of the major drawbacks is the processing time required to calculate the value of a resource. The greater the time it takes to achieve the equilibrium, the higher is the processing time.

In [FeN96], pricing of the resource is done using the price-tâtonnement process. The drawback is, there is no control of price when there is a spike in the demand. The demand is calculated by adding the demand proposed by the client and the supply is calculated by polling information from the resource producer. [FeN96] uses tâtonnement process and the price of the resource is changed when there is an imbalance in the supply or the demand. The tâtonnement process is an iterative process and the price of the resource is calculated until the excess demand is equal to zero. If the excess demand is greater than zero, the price is increased and if the excess demand is less than zero then the price is decreased. The value of the new price depends on the value of the old price, which means that there should be an initial reference price, which will be used to calculate the new price. The economic model developed in [FeN96] is used to control the resource between different autonomous computer systems. [FeN96] and [ChW98] give an insight of the tâtonnement process that can be used to calculate the value of a resource. The economic model used in my thesis does not use tâtonnement process, but some ideas of calculating the demand of the resource is borrowed from [FeN96]. The main reason for not using tâtonnement process for calculating the value of the resource is, when there are lots of spikes in the demand, it is difficult to achieve an equilibrium price.

2.1.2 Commodity Market

Wolski et al in [WoP00], has done a comparison in efficiency of calculating a price for the resource using commodity market and auctioning. Both auction and commodity market strategies are compared in terms of price stability, market equilibrium, consumer efficiency, and producer efficiency. [WoP00] uses two methods for calculating the price of the resource; one method deals with the polling of information from resource producers and consumers, where, it is assumed that the information provided by both resource producers and consumers are reliable. Another method of calculating a price for the resource is by using an approximation excess demand function of Smale's method [WoP00]. In Smale's approximation, the excess demand is calculated using a large degree polynomial. Statistical data is used to calculate the approximate Smale's excess demand function. Ideas mentioned in [WoP00], gave some insight for calculating a value for the resource. In our research, we have calculated the price of the resource by polling the resource consumers and producers.

In [SaK98], a vertical differentiated pricing strategy is used, where vertical differentiation is near-universal agreement among consumers of what constitutes higher or lower quality. Five different pricing strategies for the sellers are evaluated. The strategies are game-theory, my-optimal, computationally-limited my-optimal, trial-and-error and derivative-following. The buyers, in [SaK98], make decision for buying the resource by using the utility function. In our project, we are using the utility function as the demand of the client. The utility function describes the demand proposed by the client and it is similar to the polling technique used in [WoP00].

2.2 Peering Systems

2.2.1 Enforced Resource Sharing For Web Clusters

In [ZhK01], two levels of peering are introduced. The mandatory peering level is active at all times irrespective of the state of the lending domain (i.e., whether the domain is overloaded or not, it should honor the peering requirement and give away the specified amount of resources). When all domains are overloaded, the mandatory peering policy will cause unnecessary resource flows among the domains that would not improve the overall performance. [ZhK01] uses a static peering model is to share resource among different peers. [ZhK01] uses bilateral peering policy, which has a peering arrangement with only two peering domains. [ZhK01] also uses a static peering policy, which creates a lot of management overhead, but in our project we are using dynamic peering policy which will change the peering policies according to the demand of the resource in a domain. More details on different peering systems and peering policies are discussed in Chapter 3.

In our peering model, we use a Central Information Board(CIB) to enable a multilateral peering system, which decreases the redundancy of the peering system. The redundancy in a multilateral peering system is less because, if the CIB fails then the entire peering system will fail. Since [ZhK01] uses bilateral peering policy, the overhead created due to having peering policies with individual peer is more when compared to multilateral peering system. If there are hundreds of peers in a bilateral peering system and if all the peers needs to peer with each other, then number of policies created is very high when compared to multilateral peering. More detail description on the advantages and disadvantages of multilateral and bilateral peering policies are given in Chapter 3. In [ZhK01], optional tickets can be used by a peering system when the lending peering

system is under loaded. By using the optional tickets, the load distribution efficiency is increased. But in case when both the peering domains are overloaded, both domains need to allow the foreign peering domains to use the mandatory resource that it has promised.

For example, let us consider that there are two peering domains PD1 and PD2 that are sharing 100 CPU cycles and 200 CPU cycles, respectively, with each other. When two peering domains are equally overloaded, it will be better for both peering domains to avoid peering; but as the peering system has mandatory peering policy, the peers need to share the mandatory peering resource. In this case PS2 will loose 100 CPU cycles to peering system PS1 even if the load in both peering system is the same, and this is an unbalanced load distribution. To overcome this issue in [ZhK01], we have introduced a peering model in which the peering policy changes with the local load on the domain.

2.2.2 Data Archiving Using Peering System

In [CoM02], work is done to find a highly reliable data trading mechanism among different autonomous systems. Two trading algorithms are described collection trading and deed trading, as a mechanism to facilitate trading blocks of space. [CoM02] concentrates more on improving the redundancy of data storage in different autonomous systems. Bilateral peering policy is used to store multiple copies of data at different sites through a network by making replicas of the data.

As mentioned in [CoM02], collection trading can be better explained with an example, and the example that I have used to explain collection trading is archival site that store data. Audio data of one giga bytes is stored in archiving site A. Another archiving site B approaches site A for archiving some of site B's data. If site A has enough space in its system then it will accept site B's proposal. Site B archives two giga bytes of video data

in site A, but site A has only one giga bytes of audio data to be archived. Now, there are two copies of audio and video data and the redundancy of both audio and video data has increased. The process continues, when more sites contact each other for archiving data and the system grows with increase in number of sites.

Deed trading used in [CoM02] is similar to bilateral static peering done in [ZhK01]. In deed trading, deeds similar to property deeds are traded between different sites. The deeds can be used as a whole or as part by the sites which holds the deeds. In collection trading example, site A will loose one giga bytes of storage space because it had only one giga byte of audio data to be archived in site B. In the case of site B, it gains one giga byte of data storage because site A archived only one giga byte of information in site B. If we consider the same example that is used in collection trading for deed trading, site A will give site B two giga bytes of storage space and site B gives two giga bytes of storage space to site A. Site B uses its deeds immediately and site A will use half of its deed immediately and keep the remaining half deeds for later use.

In [CoM02], simulation is done for testing the global and local reliability, and scalability of the trading mechanism. The simulation results show that the best reliability is achieved with few sites. Simulation results also show that the reliability is better when the number of sites is less than ten. In [CoM02], static peering is used and the sites need to contact another site for archiving information, this way of polling every known site for archiving formation introduces a lot of overhead. We have used the multilateral peering system in our work to overcome the overhead created due to polling different sites.

2.2.3 Lottery Scheduling

[WaW94] presents a novel randomized resource allocation mechanism that provides responsive control over the relative execution rates of computations. The resource allo-

cation mechanism used is called *Lottery Scheduling.* Lottery scheduling also provides support for modular resource management. Lottery scheduling is done using tickets, where lottery tickets encapsulate the rights for the resource. Scheduling by lottery ticket is probabilistic and the expected allocation of resource for a client is proportional to the number of tickets a client holds. Lottery scheduling algorithm is a randomized algorithm, therefore, the actual allocated proportions are not guaranteed to be exactly the same as the expected proportion. But [WaW94] claims that the disparity between actual and expected proportions decreases as the number of allocation increases.

Lottery scheduling has a modular resource management algorithm and the modularity is brought into the algorithm using tickets. The tickets can be explicitly transfered from one client to another. A unique currency is used to denominate a ticket and the ticket is valid only within a trusted boundary. Lottery scheduling has been implemented to quantify its ability to flexibly, responsively and effectively control the relative execution rates of computation and the fairness of the scheduler. The lottery scheduler was implemented in a multi-threaded client server application for searching text and competing MPEG video viewers.

[WaW94] gave an insight of using tickets for resource allocation. The resource allocation scheme in lottery scheduling uses randomized method to choose the winner for the resource. The clients, which have more tickets have higher possibility of winning the resource. In this thesis, we find a value for the resource and the clients get the resource according to the amount of dollars he owns. By having a price for the resource, the possibility of starving the low demand clients will be relatively less when compared to lottery scheduling. Unlike lottery scheduling, the expected and actual resource allocation for a low demand client is the same even if the demand for the resource is less.

2.2.4 Effectiveness Of Request Redirection On CDN Robustness

[LiV02] talks about an effective request redirection on content delivery networks (CDN). Some of the important issues addressed are the response time and system throughput. Response time is defined as a cumulative distribution of latencies and system throughput is defined as the average number of satisfied requests for each second. System throughput represents the overall robustness of a system, when there is a flash crowd or a distributed denial of service.

[LiV02] designs a request distribution mechanism that is both responsive across a wide range of loads and distributed denial of services attacks. [LiV02] has used different strategies to implement a good request distribution mechanism. [LiV02] gave me an insight of the different performance metrics that can be used to evaluate a system. The concept of finding throughput, completion time, and system capacity is borrowed from [LiV02]. We have also used the concept of calculating the threshold of a system from [LiV02], where threshold of a system or system failure occurs when the queue length of the system exceed five times the parallelism parameter. The number five for determining the system failure is randomly chosen.

Chapter 3

Peering Systems

Peering can be defined as a relationship between two or more domains in which the domains create a direct link between each other and agree to forward each other's resources directly across the link. Peering can be categorized by the policy used by the peers to share their resources. The different peering policy that are discussed in this work are load balancing peering system (LBPS), static peering system(SPS) and dynamic peering system(DPS). The domains in the peering system can either have bi-lateral or multi-lateral peering policy. A more detailed description about each system and their peering policy is discussed in the following sections.

3.1 Bilateral Peering System

Bilateral peering system (BPS) can be defined as a peering system which has peering policy with individual peering domains. Load balancing, static, and dynamic peering can be implemented by using bilateral peering policy. Bilateral peering system will have more control of its resource, because it knows the peer that it is peering its resource. As each peering domain has individual peering relation with another peering domain, the

redundancy of the bilateral peering system is high. This is because, failure of one peering domain will not cause the entire peering system to fail.

A bilateral peering system with sixteen domains is shown in Figure 3.1. Figure 3.1 shows that each peering domain has a peering policy with another peering domain and there are some peering domains that have peering policy with more than one peering domain. In Figure 3.1, we can see that PD10 has peering policy with both PD7 and PD4.

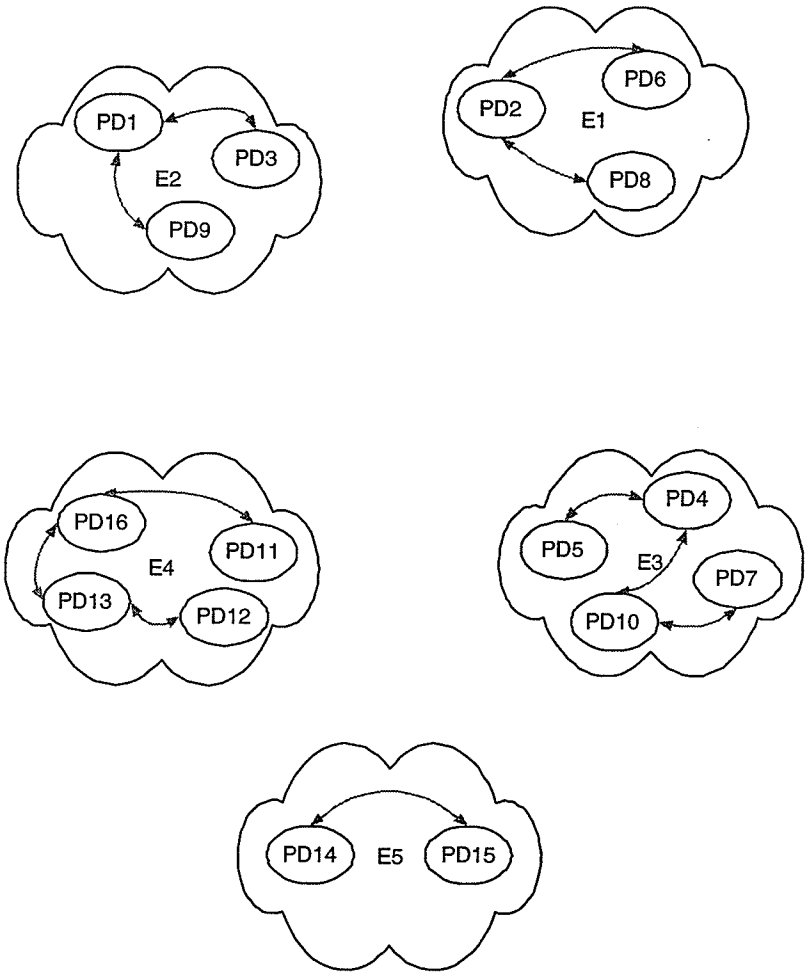

Figure 3.1: Bilateral Peering System

From Figure 3.1, we see that the sharing is restricted within a small entity. Each

individual entities in the peering system is shown as E1, E2, E3, E4, and E5. Each entity have a different number of peering domains. Peering domains that are either directly or indirectly connected to each other form an entity. In figure 3.1, entity E2 has peering domains PD1, PD3, PD9, and only PD1 is connected directly to both PD3 and PD9. PD3 and PD9 has an indirect connection between each other, and the indirect connection is established through PD1. In entity E2, PD1 can use the local resource for local jobs, and use the peered resource from PD3 and PD9 for peering. For example, let us consider that PD1 and PD3 shares 100 CPU slots with each other, and PD1 and PD9 shares 100 CPU slots. In this example, PD1 can use all the local resource for local jobs, use PD3's resource for peering with PD9, and use PD9's resource for peering with PD3.

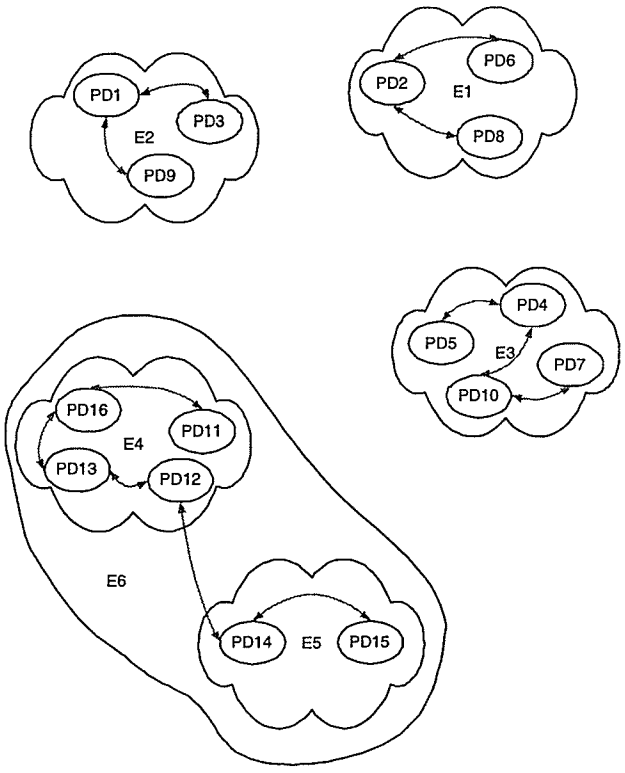

Figure 3.2: Amalgamation In Bilateral Peering System

Scalability of a bilateral peering system increases when more peering domains make

direct peering relation with another peering domain. When peering domain from one entity establishes peering relation with a peering domain in another entity, entity amalgamation occurs. Figure 3.2 shows that entities E4 and E5 is amalgamated into a single entity E6, and this amalgamation of entities occur when peering domain PD12 and PD14 established a peering relation with each other. When more and more entities amalgamate, the scalability of the peering system increases.

One of the main disadvantage of bilateral peering system is the restriction of its resources to a certain entity. If we consider a case where entity E1 is overloaded and entity E5 is underloaded, all the idle resource in entity E5 will go as a waste, and the jobs in entity E1 will be queued because of the overload. As each peering domains has to establish individual peering relationships with another peering domain, there are more overheads in distributing the peering policies.

3.2 Multilateral Peering System

Multilateral peering system (MPS) is a system, which has peering policy with all domains in the peering system. MPS has a centralized information board(CIB) which will have the peering information of all the peering domains, therefore, when compared to bilateral peering system, the overheads due to sharing peering policies are less in MPS. As the multilateral peering system requires a CIB to store the peering policy information of different peering domains, the redundancy is less but the resource utilization is more compared to bilateral peering system.

In Figure 3.1, which represents a bilateral peering system, all the peering domains can have a CIB and the resource distribution becomes more load distributed. The problem of wasting the idle resources of an entity, as explained in section 3.1 will be solved if a multilateral peering system is used. But, the important issue with multilateral peering

system is, as the the peering policy becomes more centralized the redundancy of the peering system decreases. If the CIB fails, then the entire peering system will fail. Therefore, there should be some trade-off between redundancy and resource utilization.

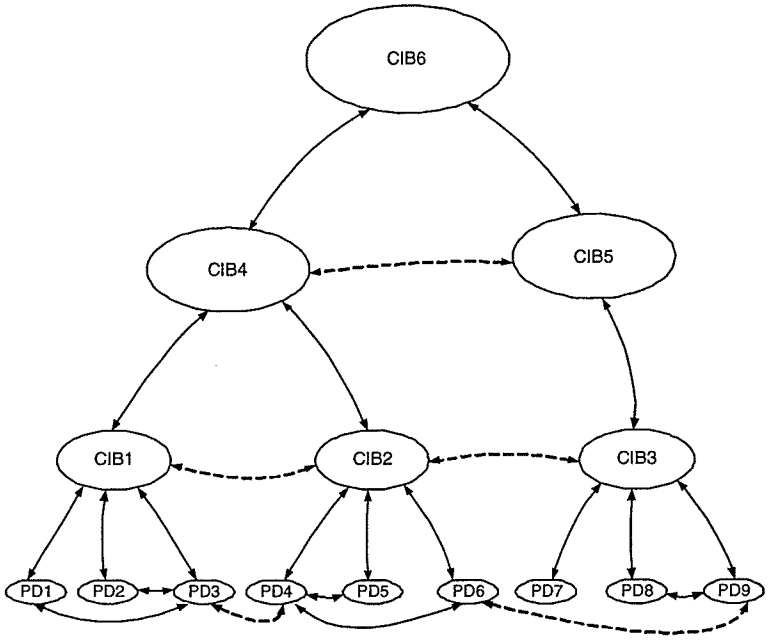

Figure 3.3: Hierarchical Peering Systems

The redundancy of the multilateral peering system can be improved by having a hierarchical structure for the peering system. The hierarchical multilateral peering policy structure is shown in figure 3.3. In figure 3.3, the lowest level of the hierarchy has the peering domains, and the rest of the levels in the hierarchy has the CIB. In figure 3.3, peering domains PD1, PD2 and PD3 form a peering system by submitting the peering information to CIB1. Similarly two different CIBs, CIB2 and CIB3 are formed by PD4, PD5, PD6, PD7, PD8 and PD9. CIB1, CIB2 and CIB3 are in the same level of the hierarchical structure. CIB1 and CIB2 are leaf nodes of the same branch, but CIB3 is a leaf node of a different branch. In figure 3.3, we can see that each CIB forms a separate entity, if there are no higher levels in the hierarchy, then the resource distribution will

be restricted. But, as there are higher levels in the hierarchy, the peering systems in the lower level of the hierarchy can amalgamate into a single peering system.

In Figure 3.3, we can see that CIB1 and CIB2 can be amalgamated into a single central information board CIB4. It is not mandatory for all the peering systems in the lower level to join the peering system in the higher level of the hierarchy, i.e., in Figure 3.3, it is not mandatory for CIB1 or CIB2 to have any peering policy with CIB4. If neither CIB1 or CIB2 has peering policy with CIB4, then the branch which has CIB4 will not exist. The dotted lines between peering domains in Figure 3.3, shows that peering is done between peering domains between different peering systems. The dotted lines that connect the CIBs, show some of the possible amalgamations of different peering systems. The main advantage in hierarchical multilateral peering system is, even if one CIB fails, the CIB in a different branch can still function.

With the hierarchical structure, the multilateral peering system is expected to be more redundant. Therefore with the improved redundancy, it is expected that a multilateral peering system would perform better than bilateral peering system. Due to the advantages of a multilateral peering system over a bilateral peering system, we have used multilateral peering system for the peering model used in this thesis.

3.3 Load Balancing System

Load balancing peering system can be defined as a peering system, which has no peering policy between peers and the resources in each peer are used according to the load in the entire peering system. In a load balancing peering system, none of the peers have control of its own resource and once a peer joins the peering system it will loose its autonomy. Depending upon the load on entire system, the load will be distributed among different peers so that the overall load in the entire peering system will be equally distributed to

all the peers. One of the major disadvantages is that the load balancing peer looses it autonomy.

As autonomy becomes a major concern for load balancing, the bilateral peering policy might restrict the usage of the resource to just one peer. In bilateral peering system peering, policy is restricted to only two domains and, therefore, this gives more control for the peers on its resource. But, as we saw in Section 3.1, as the bilateral peering system grows, the control of the resource for a peering domain decreases. In Figure 3.1, we can see that even though PD3 does not have direct peering relation with PD9, PD3 can have an indirect peering relation with PD9 through PD1. Therefore, the resources in an single entity of the bilateral peering system will be common to all the clients in the entity, i.e., all the resources in a single entity will be a common resource for the entire peering system.

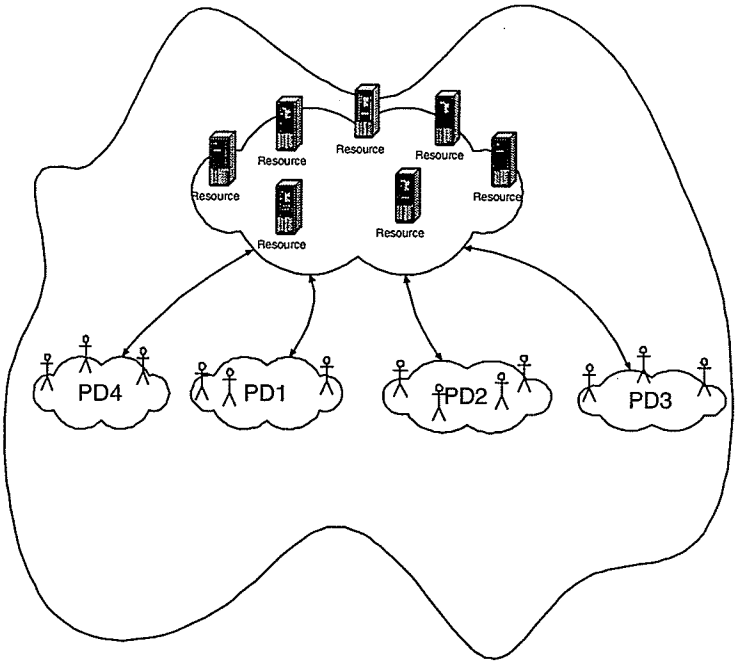

Figure 3.4: Load Balancing Peering System

Figure 3.4 shows a multilateral load balancing peering system, where PD1, PD2, PD3

and PD4 are the four peering domains in the peering system. The cloud in the center with resources, holds the virtual resources of all the peering domains. The resources will physically be in its own peering domain, but as soon as the peering domain joins the peering system the resources of the peers will virtually join the pool of resources of the entire peering system. The clients from different peers use the resources in the central pool, as a local resource without any restriction on the usage of the resource.

The load in all the domains in the peering system will be equally distributed. The big cloud in Figure 3.4, shows that PD1, PD2, PD3 and PD4 are the only peering domains that can access the resource from the common pool of resources. If some peering domain wants to join the peering system, it will put all its resources into the central pool of resources and will become a part of the peering system.

By looking at the load balancing peering system, it seems like peers with less resources and high demand can join the peering system and utilize more resources than they have contributed. The situation of over utilization of a resource by a peer brings the question of how to control the peering domain which shares low resource and demands more resource. One of the possible solution is to have a policy, where each peer can join the peering system only if the peer can contribute a certain amount of resource. If a policy is introduced to a load balancing peering system, it will become like a static peering system. But, if no policy is introduced for the peers in the peering system, then the possibility of reducing the efficiency of the peering system is more.

If we consider a situation, where only highly loaded peers join the peering system, then all the domains will be racing for the resource and the purpose of peering will be unsolved. For example, let us consider in Figure 3.4 that PS1, PS2, PS3, and PS4 are sharing 1000 mega bytes of hard disk space respectively. If PS1 has a demand of 1500 mega bytes, PS2 of 2000 mega bytes, PS3 of 1500 mega bytes and PS4 of 2000 mega bytes, and the jobs in each peer in the peering system has different start time. The jobs,

which starts first will try to grab all the resources and the jobs that appears in the end will have no resource. In this example, let us consider that the jobs in PS1 starts at 1:00 am and ends at 2:00 am, jobs in PS2 starts at 1:30 am and ends at 2:30 am, and similarly each peer has a difference, in the start time of the jobs, of one hour. In this case jobs from PS1 will use extra 500 mega bytes from other peer's resource and similarly PS2 will complete all its jobs by using the resource from other peers. But, peer PS3 will be able to use only remaining 500 mega bytes from the resource pool and PS4 will be left with no resource for its jobs.

This problem of unfair resource distribution can be solved by having some sharing policies between peering domains. The following sections discuss two peering systems, which distributes the resource with an enforced peering policy.

3.4 Static Peering

Static peering policy can be defined as the policy that does not change for a time unit regardless of the load in the domain. Static peering is the most simple peering technique, as the peering policy is changed only at every time unit or probably after few time units. As the policy for peering does not change very often, the overhead created due to the management of the peering policy is greatly reduced. Even though the overhead has been greatly reduced the efficiency of the peering is greatly reduced, because of the static peering policy, which does not change with changing load. Static peering systems can be defined as a system, which has fixed policy for sharing resources among different peers. The changes in the peering policy of a static peering system has to be done explicitly, and this leaves the peering system to overlook the future demand for a particular time interval and decide on the peering policy for a particular time interval.

In Figure 3.5, let us consider that PD1 shares 1000 units of resource, PD2 shares

1500 units of resource, and PD3 shares 1000 units of resource. If we consider a condition, where the overall local demand in PD1 is very less, then the only way in sharing the utilized resource in PS1 is by changing the peering system policy. But, in general, most of the peering system policy will not be changed very often. Going back to the example, let us consider that the demand in PD2 and PD3 is very high. Both, PD2 and PD3, can use the resource of PD1 only to the amount limited by the peering policy of PD1, therefore the unused resources in PD1 will go as a waste.

In the case of dynamic peering, as will be explained in Section 3.6, the value of the resource is being decided by the demand proposed by the clients in the local system. So, if there is high demand and if the supply for the resource is constant then the price of the resource has to increase, which actually changes the peering policy implicitly.

Like load balancing and dynamic peering, static peering can also have either bilateral or multilateral peering policy. In load balancing, once a peer joins the peering system, it will loose its autonomy and becomes one of the resource in the peering system. In the case of static peering, the peers does not loose its autonomy, because the peering policy restricts the amount of resource it is willing to share.

3.5 Dynamic Peering System

Dynamic peering policy can be defined as the policy that changes with the change in the load in the domain. Dynamic peering will have more management overhead than the static peering because of the changing policy with the varying demand. The increase in overhead is compensated with the increase in the efficiency of peering. This is so because the peering policy changes with the change in load in the domain, there by increasing the efficiency. The change in the load is used as a feedback to change the policy and the policy is changed implicitly by changing the value of the resource. Dynamic peering

policy can be incorporated in both bilateral and multilateral peering system.

Peering system is a system in which different systems agree to share resources among their peers. Dynamic multilateral peering system is a system which has sharing policy such that, as the load in an individual peering domain changes, the sharing policy also changes. The peering policy is controlled by the values of the sharing resource, and the value of the resource varies with the demand proposed by the clients in the peering domain. The value of the resource is calculated between a fixed time interval and changes in the value of the resource will be reflected on the peering policy, only of the resource in that particular time interval. The calculation of the value of the resource in a peering system is explained in detail in Section 3.6. The multilateral peering policy can be defined as the policy, which will allow all the peers in a peering system to share resource with all other peers, and the peers in the multilateral peering system need not have any knowledge about other peers. In the case of multilateral peering system, there should be some centralized board that will have the information on sharing policies of each domain. Figure 3.5 shows three peering domains PD1, PD2, and PD3 peering with each other and there is a Centralized Information Board (CIB), which has the information on the peering policies of all the different peering domains. If the peering domain PD1 needs to use the resource of either PD2 or PD3, it needs to know the peering policy of PD2 or PD3 and this information will be available in CIB. In general, the CIB holds the information of the peering domains in the format given in table 3.1.

Table 3.1 shows a sample of the peering information with the system ID in the first column, peering policy in the second column and the value of the resource in third column. Peering policy is the number of dollars worth of resource a peering system is willing to share. This concept of using dollars as the peering policy brings in the dynamics to the policy of the peering system. With the peering policies specified in table 3.1, PD1 will share 500 units of resource, PD2 will share 700 units of resource and PD3 will share 1000

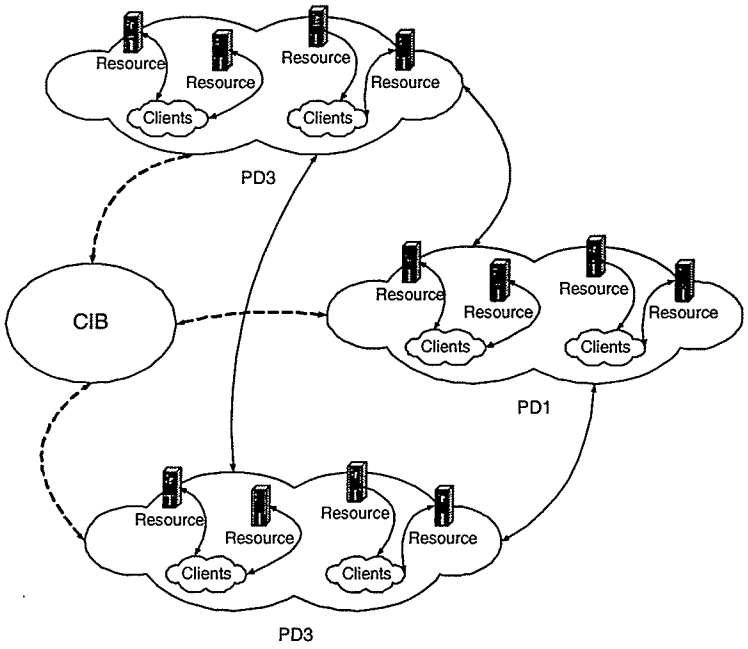

Figure 3.5: Multilateral Peering System

System ID	Peering dollars	Resource value in dollars
PD1	1000	2
PD2	2100	3
PD3	1000	1

Table 3.1: Example Dynamic Peering Sharing Information

units of resource. In this example peering domain PD3 is considered to have relatively low demand and, therefore, it has a very low value for its resource. The value of peering domain PD2 is the highest and therefore it shares relatively less resource. In the case of PD1, even though the value of the resource is less when compared to PD2, the amount of dollars worth of resource PD1 shares is less than PD2. Therefore, PD1 shares less resource when compared to PS2.

The value of the resource in a peering system is valid only for a specified time interval.

If the demand for the resource increases and if the supply for the resource remains constant, then the value of the resource will be increased. For example, if we consider that the value of the resource in PD1 increases from $2 to $4, then the number of resource shared by PD1 will decrease to 250 units, which brings the dynamics in the peering policy. Dynamic peering policy of a peering system need not be changed explicitly between each time interval, but as the load in the peering system varies, the value of the resource also varies and the peering policy of the peering system changes implicitly.

3.6 Pricing Mechanism

The value of the resource can be calculated by different pricing mechanisms. Some of the most common methods for calculating the value of the resource are auctioning [San02] and commodity market [WoP00]. Auctioning is a very common value determination mechanism, where clients bid for the resource and the client quoting the highest price will win the auction. There are different types of auctioning mechanisms that can be used in determining the value of the resource. A commodity market is relatively simple for the clients, but it introduces lots of complexity in the market end. In a commodity market, the value of the price is fixed for a certain time interval after which the value of the resource changes according to the demand of the client.

Auctioning is a very old mechanism which is used to determine the value of a resource. Some of the most common auctioning methods are closed bid auction and open bid auction. Lots of research has been conducted to find the best auctioning system that finds the correct value of the resource. Other than calculating the correct value of the resource, the most important criteria that should be considered, is the time taken for calculating the value of the resource.

Auctioning mechanisms can be categorized into parallel auctions and series auctions

[San02]. By the name it can be easily understood that in the case of a parallel auction, different resources are auctioned in parallel and in the case of a series auction one resource is auctioned after another. In network peering systems (NPS), resources can vary from CPU cycles, memory space, disk space, etc. Most of the resources are interdependent on each other and if one of the resource is not available then buying another resource might not do any good.

For example, let us consider that a job requires 1000 CPU cycles, 512 Kilo bytes of memory, and 100 Mega bytes of disk space. In the case of series auction, where each resource is auctioned in a serial manner, let us consider that CPU cycles are auctioned and the clients send bids for the CPU cycles, and win the auction. Now the situation is that the client has bought CPU cycles and he has to do the bidding for memory and disk space. If the client could not get one of the resource, then the resource that client has already bought will be wasted. This problem in buying one resource and losing the dependent resource can be avoided by using parallel auctioning. In the case of parallel auctioning all the resources will be auctioned in parallel and the chances of winning the auction for all the wanted resources is high, but still both series and parallel auction does not promise the resources needed by the client.

As mentioned before, one of the most important criteria in calculating a value for the resource is the time taken for finding the value of the resource. In the case of open bid auctioning, it is difficult to have a fixed time to calculate the value of the resource. The clients can keep on increasing the bid and at some particular time the auction might end with a highest bidder. But, the time taken to find the correct value of the price can even be infinite. To speedup the calculation of the value for the resource, auctioning can be done for a particular time interval. One of the very good example is e-bay which is one of the world's most popular on-line auction system. But once again the value of the resource might not be correct because of lack of network or lack of time. In some cases,

the possibility that the clients, who have high demand, might not be able to bid for the resource at the same speed as the client with low demand and loose track of bidding. Therefore having a time restriction in auction might result in having an undesirable price for the resource.

To solve the problem in open bid auctioning, there is another auctioning method know as closed bid auction. In the case of closed bid auction, the client can send only one bid and that will decide the demand and the price the client is willing to pay. Closed bid auction reduces a lot of overhead that will appear in a open bid auctioning system, because one client can only send one bid. It is assumed that there will be some security service that controls the clients from sending multiple closed bids. Closed bid auction can be conducted for a particular time interval and since a client can send only one bid, network delay will not play an important role. But, still we have a problem of not getting the dependent resource, because the bids sent by each client is closed, and the probability of not getting the dependent resource is more in the case of a closed bid auction than open bid auction.

There is another way of determining the value of the resource which is called as commodity market [WoP00]. In commodity market, the value of the price is fixed for a particular time interval and the client who have the money can buy the resource. It seems to be very simple, but the question is how is the price fixed? One way of finding the value of the resource is by finding the demand and the supply of the resource. It is easy to calculate the supply as the market knows the amount of resource available in the inventory. The demand can be calculated either by some statistical data or by somehow determining the current demand. Commodity market has fewer overhead, when compared to auctioning, and since the price is fixed for a time interval, the time for calculating the value of the resource is also fewer.

The concept in peering system is to share resources among different entities. Network

peering system peers resources that can be accessed through a network. The resource are mostly CPU cycles, memory, network resource, etc. Using auctioning mechanism to calculate the share for an individual client is almost impossible. So, it is always better to fix a price of the resource for a particular time interval and allow the clients to buy the resource depending on their demand.

In this thesis, we have used the concept of closed bid and commodity market to determine the value of the resource. The clients send a client demand vector to the system domain manager, and the client demand vector has the price the client is willing to pay and the demand for the specified price. The client vector $cv(a, b)$ is represented as P_i, Q_i, Where $i = 1$ to N, where a is the client ID and b is the peering system ID. N is the number of varying demands proposed by the clients and the value of N is restricted to some fixed integer, so that client can propose only a fixed number of demands. P_i is the price of the i^{th} demand proposed by the client and Q_i is the number of quantities that the client is willing to buy for the price P_i. In the case of the example discussed in Section 3.5 the system ID is PS1, PS2 and PS3. For example, a client vector may look like $(2, 1000; 3, 6888; 1, 7877)$. In this example, the client proposes three demand parameters and each demand parameter is separated by a semicolon.

The demand proposed by the client is assumed to be following the economics model, where as the number of quantity requested increases, the price decreases. The demand curve is assumed to be as shown in figure 3.6. The demand curve in Figure 3.6 shows that demand elasticity is negative, which means that as the number of quantity requested by the client increases the price of the resource will decrease. In the case of a peering systems, the supply can vary with the amount of resource contributed by the supplier, and the demand proposed by the client can also vary. For example, if the supply increases, the price proposed by the client will decrease, and as the supply decreases the price proposed by the client will increase.

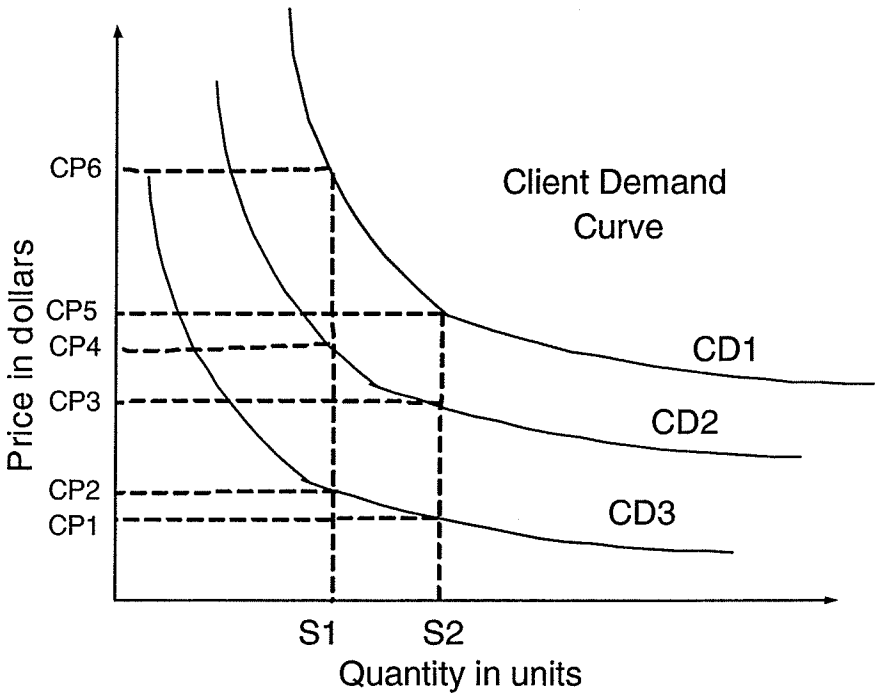

Figure 3.6: Client Demand Curve

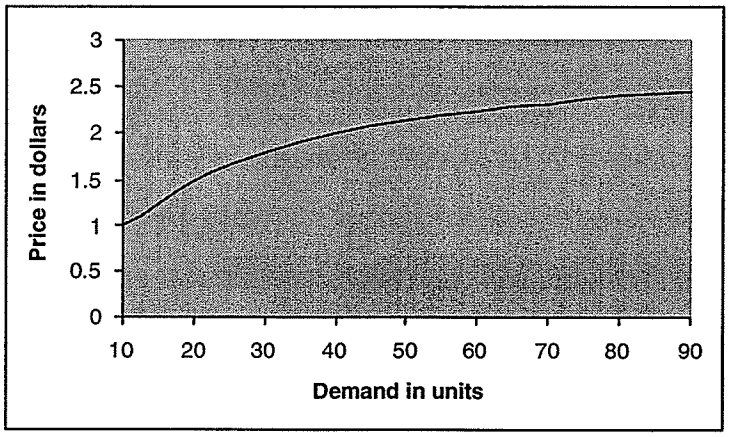

Figure 3.7: Market Demand Curve

In dynamic peering model, the higher the price the lower is the amount of resource shared by the peering system. In this pricing model, it is assumed that as the load increases, the price of the resource will increase. If the price is not increased with the increase in load, then the peering policy will not change and the possibility of loosing more resources to a foreign job is more. The main objective in dynamic peering is that, as the load increases, the sharing resource should decrease, and this is possible only if the price of the resource is increased. Keeping in mind the dynamic peering objectives, the market demand curve is expected to be as shown in Figure 3.7. Price elasticity in Figure 3.7 is positive and the supply is fixed, therefore as the demand increases, the price also increases. Using the demand curve proposed by the client, a value for the resource is calculated and the value remains constant for a certain time interval. In dynamic peering, the client vector is sent only by the host clients, which means that the value of the resource is determined only by the local clients.

With the client vector we can draw a demand curve and we can find the Price Elasticity of Demand (PED). PED can be defined as the ratio of the percentage change in demand to the percentage change in the price, the formula for PED is shown in equation 3.1 [Me76]. With the knowledge of PED, demand, and supply; we can calculate the price using the basic PED formula from equation 3.1

$$PED = \frac{\%\ Change\ \ in\ demand}{\%\ Change\ in\ Price} \tag{3.1}$$

In Figure 3.6, let us consider that the initial demand sent by the client represents the demand curve CD3, and the initial supply for the peering system is S2.With the supply S2 and demand curve CD3, the price of the resource will be CP1. Let us consider that the supply for the resource in the next time interval decreases from S2 to S1. If the local clients did not change the proposed demand, then the new price will be CP2. But, there is always a possibility that as the supply decreases the demand of the resource might

increase, and the demand curve might move away from the axis. In general economics ??, as the demand curve moves away from the axis, it is expected that the supply for the resource will also increase. But, if the demand curve is not shifting with the decrease in supply, then the resource might get obsolete. Figure 3.6 shows three different client curves, and the client demand curve CD1 is the demand curve with the highest demand and the client demand curve CD3 is the demand curve with the least demand.

In dynamic peering, the resources in a peering domain is shared with other peering domain, and as the price of the resource decreases more resource is peered with foreign clients. This situation makes the client to propose a client vector with higher price, because if the client proposes low price then the possibility of loosing the local resource to foreign clients is more. In dynamic peering, the price of the resource is more used to control the access of the local resource by the foreign clients. The commodity market used in this thesis assumes that the client vectors sent by the client represents the real demand, and no measure is made to restrict the clients from sending false demand vectors.

In this thesis, I have done some simulation to show the operation of dynamic, static and load balancing peering systems. In the simulation for dynamic peering system, clients have to send some demand vector to calculate the value of the price. It is difficult to simulate the client with varying behavior, so I have normalized the price of the resource with a equilibrium price, which is equal to one dollar. Equilibrium price can be defined as the price when the excess demand is equal to zero. By using equation 3.2, we can derive an equation for calculating the price for a given demand and supply.

$$\epsilon = \frac{\frac{D_n - D_c}{D_c}}{\frac{P_n - P_c}{P_c}} \tag{3.2}$$

$$P_c = \frac{P_n \times D_c \times \epsilon}{D_c(\epsilon - 1) + D_n} \tag{3.3}$$

Equation 3.3, gives the derived price equation. where ϵ is the elasticity, D_n is the normalized demand, P_n is the normalized price, D_c is the current demand, and P_c is the price that needs to be calculated. The normalized demand D_n is equal to the supply of the resource and the normalized price P_n is equal to one. Current demand D_c is calculated by adding all the processing time requested by the client. Assuming a value for elasticity and applying all known values, we can calculate a new price for the resource.

Calculation of a price can be better explained with an example. For calculating the value of the resource with fixed supply, the normalized demand will remain constant and with varying supply the normalized demand also varies with the varying supply. In this example, the supply is fixed to 10 processing units and the demand increases each time unit. Table 3.2, shows the value for the price for a given demand. The elasticity in this example is assumed to be 1.5. We can see that as the demand increases, the price of the resource also increases.

Demand in processing units	Price in dollars
10	1.0
20	1.5
30	1.8
40	2.0
50	2.14
60	2.25
70	2.33
80	2.4
90	2.45

Table 3.2: Example For Price Calculation With Varying Demand

3.7 Spectrum Of Cooperation

In this section, we will talk about the spectrum of cooperation between peering domains in different systems. In Figure 3.8, the left corner of the spectrum bar indicates that the peering domains in the system is fully cooperative, and the right end of the spectrum bar indicates that the peering domains in the system is fully non-cooperative. We can see from Figure 3.8 that the LBSs are fully cooperative, and the DPSs can have a wide range of cooperation. In the case of SPSs, as the peering policy is fixed, the cooperation between peering domains will remain in a fixed position in the spectrum. In Figure 3.8, the static peering system shows that the peering policy is less than 50% of the total resource owned by the peering domains, and therefore, the figure shows that SPS is closer to the right end of the spectrum.

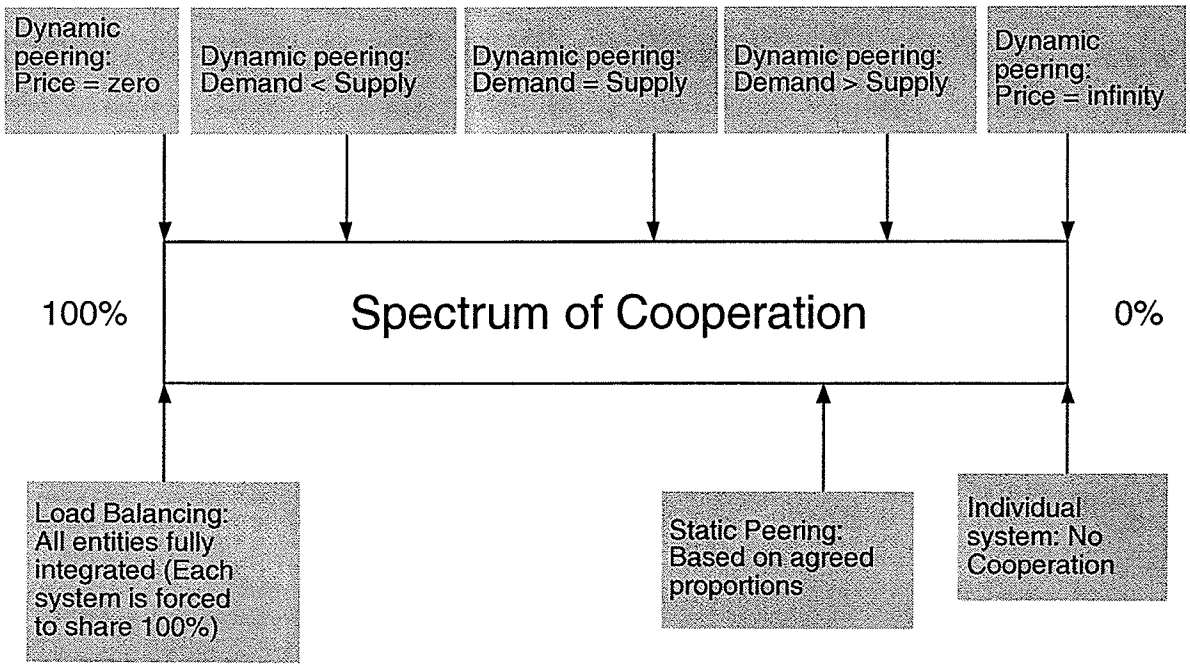

Figure 3.8: Spectrum Of Cooperation

In the case of DPSs, the cooperation between peering domains varies according to

the demand of the resource in each peering domain. If the value of the resource in a peering domain is zero, then the peering domain will have full cooperation. In DPSs, if the demand of the resource in a peering domain is less than the supply, then the cooperation will be closer to the left end of the spectrum bar. In DPSs if the demand and the supply in a peering domain is equal, then the cooperation will be in the center of the spectrum bar. In Figure 3.8, we can see that if the value of the resource in a peering domain is infinity, then cooperation of the peering domain will be zero. From the spectrum of cooperation figure, we can see that the DPSs will have a wide range of cooperation according to the load in the peering domain, and other systems will have a fixed cooperation irrespective of the load.

Chapter 4

Simulation Setup And Results

In this chapter, performance is evaluated for different NPSs, and in each simulation, peering domains are represented by a multi-processor parallel system. Simulation is done for two peering systems; dynamic peering system and static peering system, and simulation is also done for load-balancing, which is used as a baseline mechanism for evaluating the peering systems. The performance metrics of different peering systems are measured and a comparison is done to show the merits and demerits of different peering systems. The simulation setup for all the peering system is done using multilateral peering policy. Simulation setup for load balancing is explained in subsection 4.4.2, static peering is explained in subsection 4.4.3, and dynamic peering is explained in subsection 4.4.4. The input for the simulation is obtained from a workload generator program from [Cw01]. Section 4.1 talks about the different metrics that we have used to measure the performance of peering systems.

4.1 Performance Metrics

4.1.1 Throughput

Throughput is the number of jobs completed in a unit of time (e.g., number of jobs completed in a second). To measure this performance metric, the system is subjected to a stream of jobs that are injected at some known rate. The throughput measures the serving capacity of the domain. In this thesis, throughput is used to measure the performance at the domain level as well as at the global level.

The domain level throughput measurement is done for each individual peers in the peering system. When a peer is overloaded its is expected that the domain level throughput measurement will give a good insight of the load distribution of overloaded domains job to the underloaded domains. The global level throughput gives a insight of the performance of the entire peering system for different peering policy.

4.1.2 Turn Around Time

This is the difference between the finishing time and the arrival time of a job. The average turn around time measures how effectively the jobs were completed. In the best case, the turn around time of a job will be equal to the service time. The best case happens when the job waits zero time to receive the service.

4.1.3 Utilization

This measures the percentage of time a resource is busy. Over a domain, we measure the average utilization of the resources in that domain. We can compute two types of utilizations here. Native utilization is the percentage of time a resource is kept busy by executing local jobs (i.e., jobs that belong to the local domain). Foreign utilization is

the percentage of time a resource is kept busy by executing foreign jobs (i.e., jobs that belong to other domains). Total resource utilization of a domain is the sum of native and foreign utilizations. This metrics gives very good idea of the distribution of the load among the peering domains. When the load in a peering domain is high, then the foreign jobs will avoid using the resource from the overloaded domain. Therefore, it is expected that the foreign utilization is inversely proportional to the load on the peering domain, and native utilization is directly proportional to the load.

4.1.4 System Capacity

System capacity is the aggregate processing time of jobs in the queue and jobs running in the system when the system fails. The system is considered to have failed when the number of jobs in the queue exceeds three times the number of parallelism nodes in a system. A parallel system with n processors, have a maximum queue length as three times n. The system is considered to have failed even if one of the peering domain in the peering system has failed, therefore this metrics will give a very good overview of the load distribution in a peering system.

To evaluate this metrics the job injection rate must be very high. When the job injection rate increases, the load applied to the system also increases, and therefore the system fails as the queue length in the system exceeds the threshold.

4.2 Assumptions And Limitations

The assumptions and limitations of the dynamic peering system are as follows:

- *Misuse*: Dynamic peering system determines the value of the resource by polling the clients for demand a vector. The demand vector sent by the client is assumed

to be correct, and I have not proposed any method from stopping the clients from sending wrong demand vectors.

- *Network delay*: When different domains (NSPs) are trying to access a resource of a remote domain, it is assumed that all the domains have the same priority to access the foreign domain and the network delay is constant for all the domains. In other words, the network delay and access priority is assumed to have no effect in our model.

- *Unique Resource*: The peering model used in this thesis assumes that only one resource is peered with different domains.

4.3 Workload Generator

The peering system described in this thesis has been simulated in a parallel system. The input for the peering system simulation is obtained from a workload generator from [Cw01]. The workload generator was built on the sole purpose for testing the performance of the supercomputer schedulers. Testing the scheduling using realistic job is very important to determine the performance of the scheduler. The workload generator program is used just because it can generate input for different loads, number of jobs, and number of nodes for a parallel system.

The workload generator takes few input parameters to generate a log file which has the workload for a particular parallel system. The input parameters for the workload generator are; number of nodes in the parallel system, number of jobs to be created, type of parallel system for which the workload is generated, and load multiplier. The first parameter, number of nodes is basically the moldability of the jobs, where moldable jobs are jobs that can run on multiple processors in a parallel system. The third parameter

is the parallel system type, which defines the parallel system for which the workload is created. The workload generator can generate workload logs for four different parallel system. The fourth parameter, load multiplier is a integer factor that varies the load on the jobs of a parallel system. As the load multiplier increases the job arrival rate will also increase, which is basically the load on the parallel system.

The jobs created by the workload generator is characterized by the arrival time, requested time, execution time, and partition size. The workload generator program is created using the log files from four different supercomputers and all four supercomputers are IBM SP2s with different number of processors. The workload generator, generates a log file with the above specified input parameters, and the output log file created by the workload generator has information on start time, processing time, completion time, moldability, job status, job name, finish time, job owner, and tag for the job. For peering system simulation, we use only the start time and processing time of the job, all the other information in the log file is not important for the simulation done in this thesis. Other information available in the log files gives performance information specific to a parallel system, but in this simulation we need only the input information to the parallel system.

4.4 Simulator

This section explains the parallel system and peering system simulator. The simulator is written in C language and it runs on Linux operating system. The input to the simulator is the log file generated by the workload generator. The peering system simulator have different functions for simulating a static peering system, a dynamic peering system and a load balancing system.

4.4.1 Parallel System Simulator

The parallel system simulator is programmed using linear linked list and each node in the linked list refers to a processor in a parallel system. Each node in the linked list has information about a single processor. The head node has the information about the entire parallel system.

Each processor has the following local information:

- *Available time*: This variable gives information on the available time of the processor.

- *Status*: The status variable shows the availability of the processor at a specific time. If the processor is not available at the specified time then the status flag is set, which means that the processor is busy.

- *Processor ID*: This variable is an integer which holds the ID of the processor in the parallel system.

The parallel system also has some global variables which are defined in the head of the linked list; the global information are listed below.

- *Queue length*: This is an integer variable and has the count of the number of jobs queued in the parallel system.

- *Start time of job in the queue*: This variable is an array and it has information of the start time of the queued jobs. The values in the array are used by the parallel system to process the queued job in a FIFO system.

- *System failure*: This is an integer variable and is used as a flag for indicating the system failure. The parallel system is considered to have failed if the queue length

exceeds a certain threshold. In this simulation the threshold is equal to three time the number of processors in the parallel system.

The above mentioned local information from the processor and the global information of a parallel system is used to simulate the parallel system. The parallel system simulator uses different functions to do the simulation and the functions are listed below.

- *Initialization*: This function takes two integers as the input parameter, the first integer will represent the number of processors available in a parallel system. The second input to the function specifies the threshold of the parallel system and the threshold is used to determine the failure of the parallel system. This function returns the structure of the processor, and the only meaningful value in the processor structure will be the processor ID, other values in the structure will be initialized to zero. The values that are initialized to zero holds the peering information.

- *Find available time*: This function is used to determine the available time of a specified processor in the parallel system.

- *Remove job*: This function is used to remove an assigned job from a processor in the parallel system.

- *Schedule job*: This function is used to schedule a job in a processor in the parallel system. The job is assigned to the processor which has the least available time.

- Add Queue: This function is used to add a job into the queue. The job is added into the queue only when the start time of the job is greater than the available time of all the processors in the parallel system.

- Remove job from queue: This function is used to remove jobs from the queue. This function looks at the start time of the job in the queue and assigns the job to a processor and removes it from the queue.

4.4.2 Load Balancing Simulation Setup

Load balancing system is a system which has no peering policies between domains. The simulation is done for parallel systems with different number of processors, and the input to the simulation is a log file and details about the parallel systems and the log file is mentioned in section 4.5. As mentioned in section 4.3, the simulation uses only the start time and requested processing time from the log file. In load a balancing system, the resources of all parallel system will join the common pool and the jobs from any parallel system can access the resource from the common pool without any restriction. The simulator looks at the start time of the jobs coming from different parallel system, and the job with the least start time is scheduled in one of the processor from the common pool. The job is assigned to the parallel system which has the least completion time. The flow chart in figure 4.1, shows the process of accessing the information from the log file. The data blocks with labels P0, P1, P2, P3, and P4 are the blocks which has the log files. Each parallel system has its own workload log file, therefore, in this simulation five log files are used. In the start of the simulation, a job from each log file is passed to the common data pool block. The common data pool block has information of the start time and processing time of local jobs of each parallel system, and the job with the least start time is scheduled first and we can see from the flow chart that the first decision block checks if the start time of job from P0 has the least start time. If the job from P0 has the least start time, then the P0 job becomes the current job and it is scheduled; next job from the P0 log file is moved to the common data pool. The flow chart shows five decision block which makes decision on jobs from five different parallel system.

If a parallel system has jobs starting earlier than any other parallel system's job, then jobs from the parallel system with the earliest start time will have share of the resource from the common pool. The jobs from the decision block is moved to the schedule job

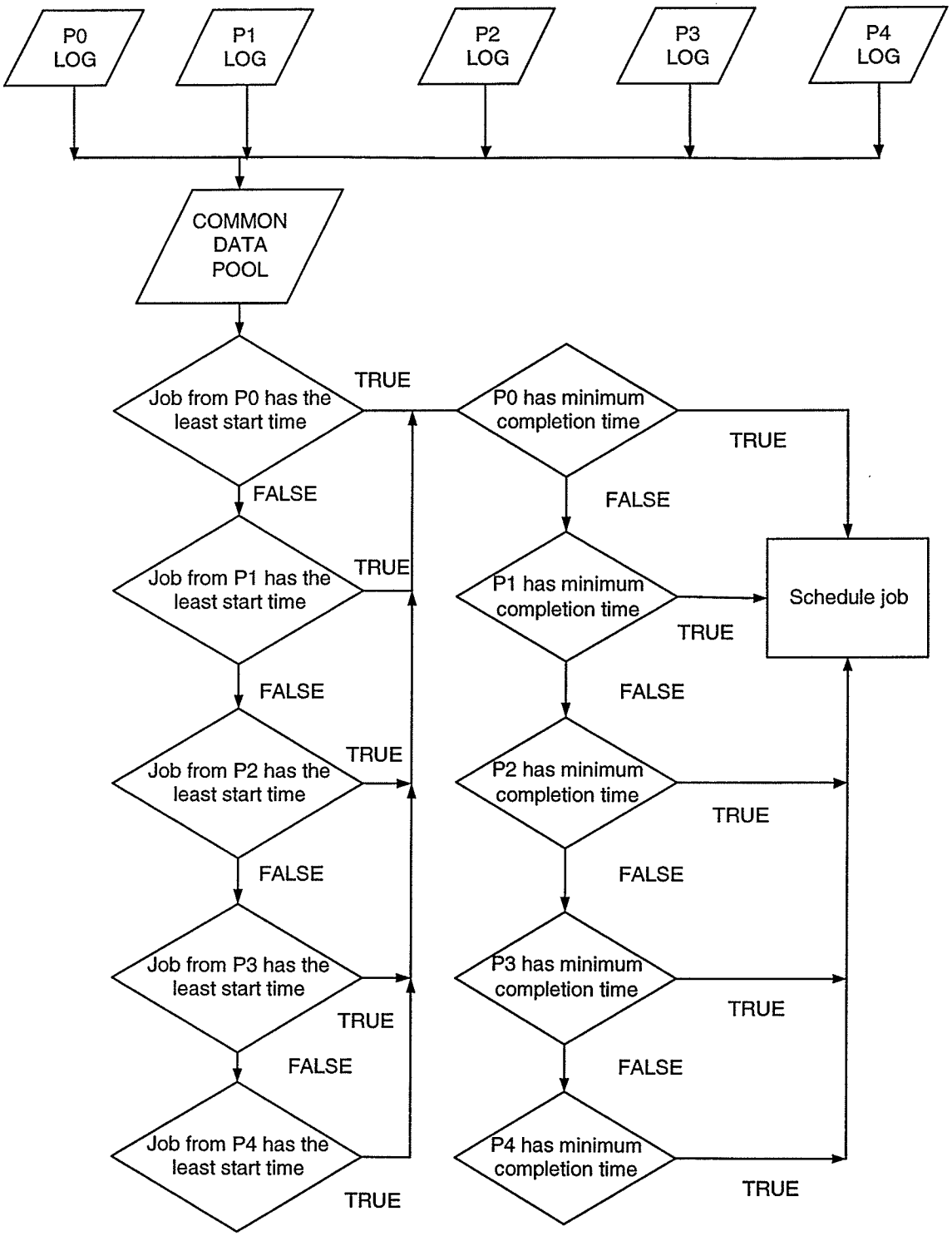

Figure 4.1: Flow Chart For Load Balancing

block. The schedule block is a process block, and jobs entering this block is scheduled in one of the parallel system with the best completion time.

4.4.3 Static Peering Simulation Setup

Static peering system is a system in which domains has fixed peering policy between different peering domains. The simulation for the peering system is done using multilateral peering policy. Therefore a central information board is used to control the sharing between domains. Each parallel system has different number of processors and therefore the supply of processing time in each parallel system will also be different. In this simulation setup, the sharing policy is set such that each parallel system will share 40% of its resource. The sharing policy of each parallel system is fixed and it does not change with the load on the parallel system.

The flow chart in figure 4.2 shows the process of accessing information from the log file for each parallel system. The procedure of accessing information from the log file is the same as load balancing; but when the job is scheduled the policy of the parallel system will determine the scheduling of the job. The peering policy of each parallel system is shown in table 4.1.

Parallel system Name	Total supply	Sharing policy	No. of processors
ANL	36	14.4	120
CTC	129	51.6	430
KTH	30	12	100
SDSC	38.4	15.36	128
CTC	129	51.6	430

Table 4.1: Input For Parallel System

Table 4.1 has information for each parallel system for a time unit of 0.3×10^6 seconds. The time unit 0.3×10^6 seconds is a fixed time interval for which the peering policy holds good, and the value of the time unit is fixed throughout the simulation. The first column shows the name of the parallel system and the second column shows the total supply of the processing resource in 10^6 seconds. The third column shows the resource shared by each parallel system, which is 40% of the total supply. Static peering uses completion time and sharing policy of the parallel system to schedule the job in a peering system. The parallel system with the least completion time will be the best candidate to process the job, but, if the parallel system with the least completion time is not a local parallel system, then the requested job should have a processing time less than or equal to the shared resource. If the requested processing time of the foreign job is more than the peering resource of the parallel system that has the least completion time, then the job has to assigned to the parallel system with the next least completion time. The shared resource decreases as more and more foreign jobs are assigned to the parallel system.

The static peering process is shown in figure 4.3. In flow chart shown in figure 4.3, the processing block BCT1 is the parallel system with the best completion time, BCT2 is the parallel system with the next best completion time, etc. If the parallel system with the best completion time is a local parallel system then the job is assigned to the parallel system without verifying the peering policy. If the parallel system with best completion time is not a local parallel system, then the peering policy check is done. If the peering policy is not satisfied, then the job has to look for a parallel system with the next best completion time. In the flow chart, if requested processing time (RPT) is less than peering policy (PP), then the job has to look for another parallel system. This process continues until the job finds a parallel system for executing the job, and if the job could not find a parallel system, then, the entire system is considered to have failed.

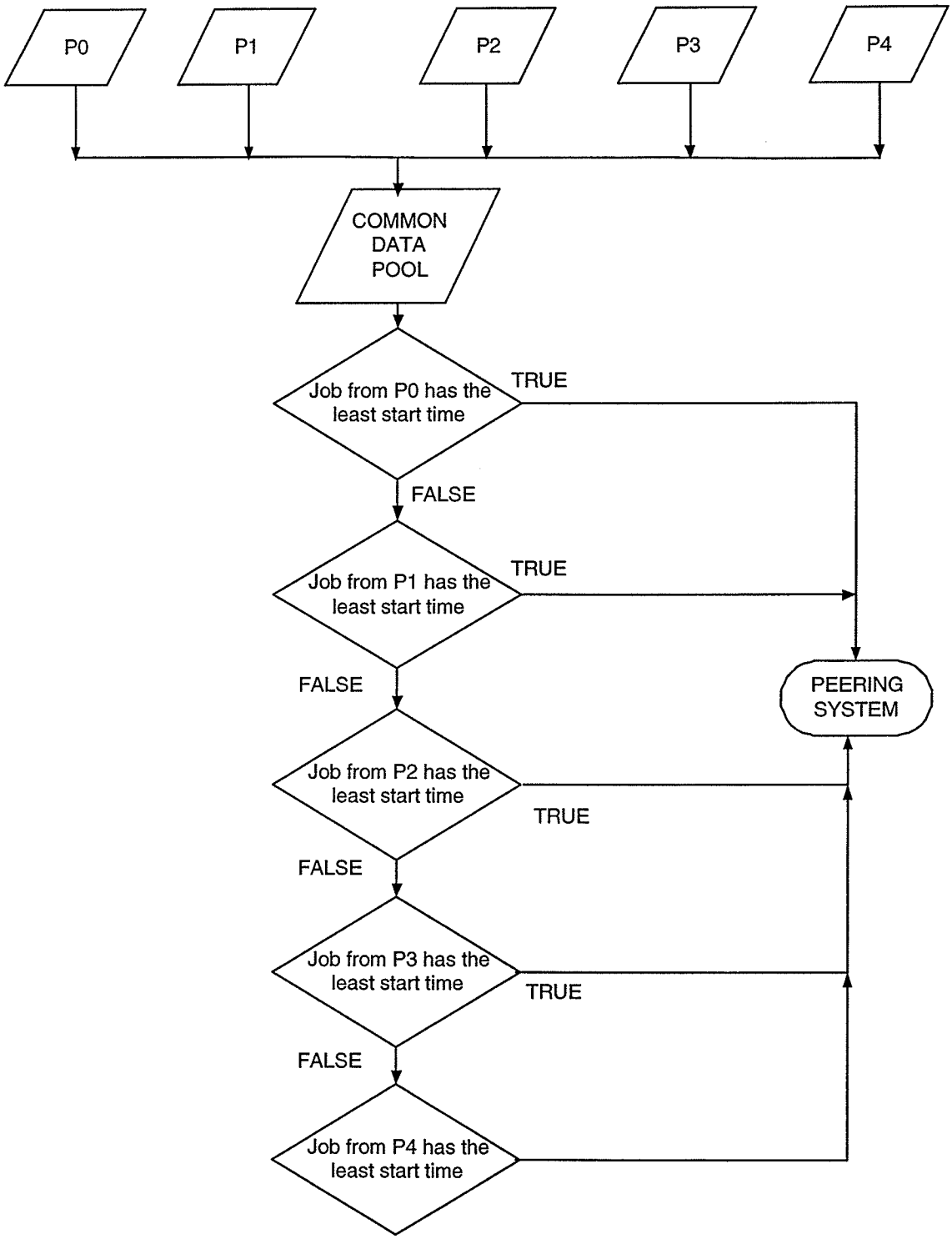

Figure 4.2: Flow Chart For Accessing Log Info In Peering Systems

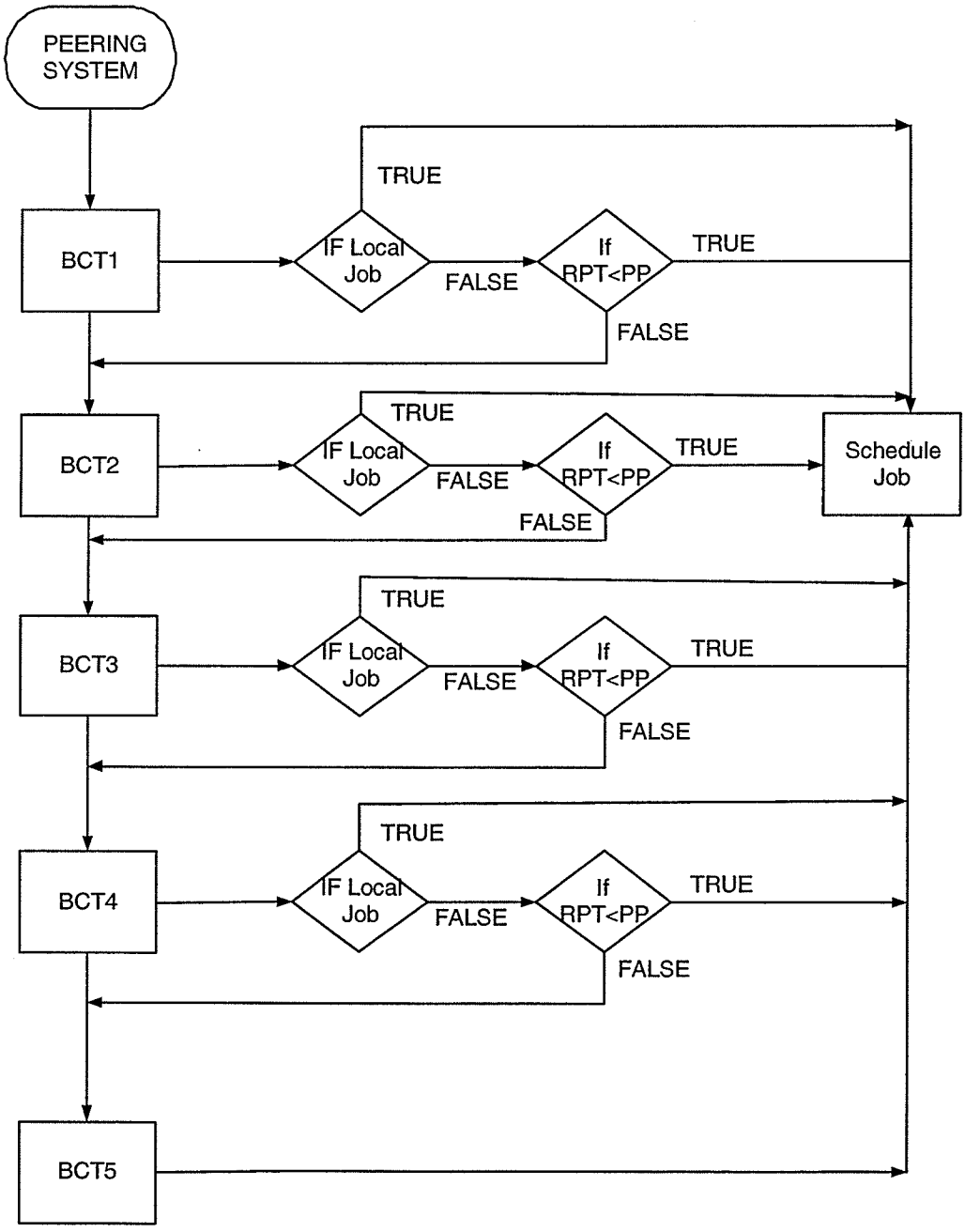

Figure 4.3: Flow chart For Scheduling Jobs In Peering Systems

4.4.4 Dynamic Peering Simulation Setup

Dynamic peering system is a system in which the peering policy changes with the change in the load of the domain. The simulation is similar to the static peering system, except the peering policy changes with the load. The peering policy in the dynamic peering changes with change in the demand and in the supply of the resource. In this simulation the peering policy of the parallel system changes after every 0.3×10^6 seconds. Table 4.2, shows the peering information provided by each parallel system.

Parallel system Name	Sharing policy	No. of processors
ANL	14.4	120
CTC	51.6	430
KTH	12	100
SDSC	15.36	128
CTC	51.6	430

Table 4.2: Input For Parallel System

In table 4.2, first column gives the name of the parallel system. The second column is the amount of dollars worth of resources the dynamic peering is willing to share. The third column shows the total number of processors in the parallel system. The simulation procedure for dynamic peering is exactly the same as the static peering, except the price of the resource will determine the amount of resource the parallel system is sharing for the current time unit. The price of the resource will depend upon the demand of the local jobs and the supply of the resource.

When the demand and the supply are equal, then the price of the resource will be equal to one. With this normalization, the dynamic peering will behave like a static peering. The peering policy of the dynamic peering is set for a normal load, where the

demand and the supply are equal. Therefore at normal load like the static peering, the dynamic peering will be sharing forty percent of its resource. When the load in the parallel system increases the price of the resource will increase so that the number of resource shared will decrease.

4.5 Simulation Setup

Simulation is done for two different peering systems and a load balancing system with varying load for each peering domains. Each peering domain in the simulation is represented by a parallel system. The simulation uses the log file created by the workload generator [Cw01] as the input. This section has three subsections, subsection 4.5.1 deals with input to parallel systems, with all the parallel systems overloaded. Subsection 4.5.2 talks about input to parallel systems, with one parallel system overloaded, three parallel system underloaded and one parallel system normally loaded. Subsection 4.5.3, deals with input to parallel systems, with one parallel system overloaded and all other parallel systems underloaded.

4.5.1 All Peering Domains Overloaded

In this simulation setup all the peering domains are overload so that the system fails after executing a few thousand jobs. As mentioned before, each domain represents a parallel system and each parallel system is differentiated with the number of processors in the system. System capacity can be calculated in this simulation, because all peering systems and load balancing system fail after execution a few thousand jobs. Fairness of resource allocation is an important metric that needs to be measured in all peering systems. This simulation is useful in measuring the fairness of resource allocation for the least overloaded parallel system.

Information of the simulation setup is shown in table 4.3. The first column shows the parallel system name, where ANL is Argonne National Laboratory SP2, CTC is Cornell Theory Center SP2, KTH is Swedish Royal Institute of Technology SP2 and SDSC is San Diego Super computer Center SP2. The workload generator mentioned in Section 4.3 takes the parallel system name as an input to generate the workload log file. The second column gives the aggregate requested processing time of all the jobs submitted to each parallel system. The third column gives the available processing time in each parallel system and the fourth column gives the total number of processor available in each parallel system. All the data in table 4.3, is provided for 286419 seconds, after which dynamic peering system fails. In all simulations, calculation for all the systems is done until one of the system fails, and in this simulation, dynamic peering system fails after 286419 seconds.

The supply for each parallel system is calculated by the formula given in equation 4.1. Where S_i is the supply for the i^{th} parallel system, N_i is the number of processors for the i^{th} parallel system and T is the duration for which the supply is calculated. The demand is calculated by adding the requested processing time.

$$S_i = N_i \times T \tag{4.1}$$

Figure 4.4 shows the requested processing time for different parallel system. From the figure 4.4 we can see that load in parallel system 1 and parallel system 4 has relatively higher load, and the load on parallel system 2 is relatively closer to the supply.

4.5.2 Peering Domains With Differential Load

In this simulation setup, the load in one parallel system is high compared to all other parallel systems. In this simulation the fairness of resource allocation in the normally

Domain Name	Demand (sec.)	Supply (sec.)	No. of processors
ANL	91249530	34357440	120
CTC	228187400	123109860	430
KTH	40608110	28636800	100
SDSC	112704400	36637056	128
CTC	240835100	123114160	430

Table 4.3: Input For Parallel Systems With Parallel Systems Overloaded

loaded parallel system is examined, where in a normally loaded parallel system, the supply and demand are equal. In this simulation all metrics except system capacity can be calculated. For calculating the system capacity the peering system must fail, and with the load provided in this simulation all peering systems will not fail. A system is considered to have failed, if the number of jobs in the system queue exceeds the threshold. The threshold for system failure is equal to three times the number of processors in the parallel system, therefore the failure threshold for each parallel system is different. All metrics calculation is done for the time until the first peering system fails. For example, if static peering fails while executing a job at t1 seconds, then the metrics calculation for dynamic peering system, and load balancing is done until t1 seconds. System capacity is the only metrics that is calculated until the individual peering system fails.

Information on the simulation is shown in table 4.4. From table 4.4 we can see that the demand is the only parameter that have changed. For calculating the throughput, turn around time, and completion time, the requested processing time is reduced so that system does not fail after executing few jobs. As mentioned in Section 4.5.1, supply is calculated until the duration for which the jobs are injected to the parallel systems. In this simulation the supply is calculated for 912003 seconds. In all simulations, calculation

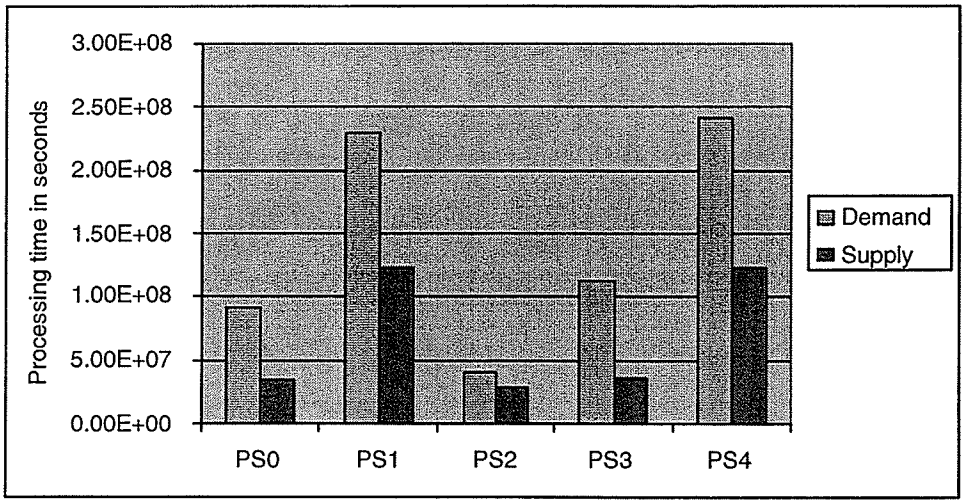

Figure 4.4: Supply And Demand For Overloaded Parallel Systems

for all the systems is done until one of the system fails, and in this simulation, static peering system fails after 912003 seconds.

Domain Name	Demand (sec.)	Supply (sec.)	No. of processors
ANL	415274100	109440360	120
CTC	235518200	392161290	430
KTH	39270390	91200300	100
SDSC	116666400	116736384	128
CTC	241928300	392161290	430

Table 4.4: Input For Parallel System

Figure 4.5 shows that the requested and available processing time of different parallel system. From figure 4.5, we can see that the demand on PS0 is higher than all the other parallel systems. PS3 has load equal to the supply, therefore this simulation will be used to measure fairness in resource allocation for parallel system 3. In this simulation, it is also necessary to see load distribution of parallel system 0 with all other parallel system.

Information on the simulation results are provided in following sections.

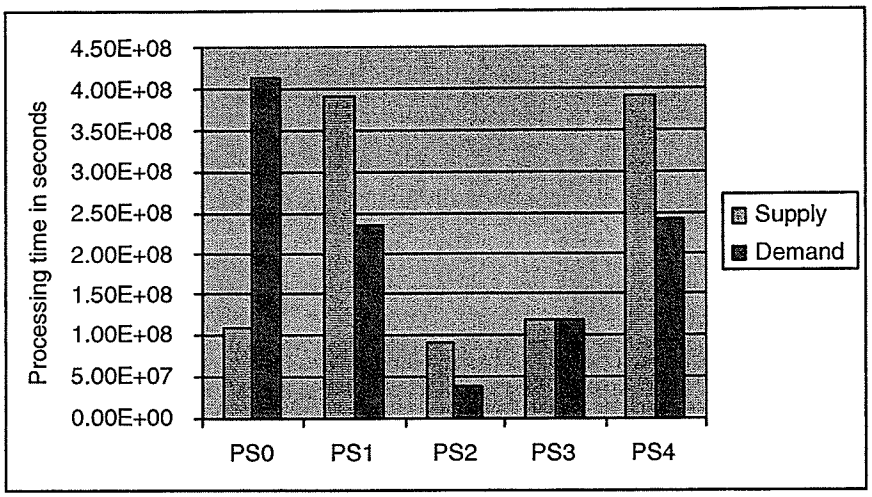

Figure 4.5: Supply and Demand Graph For Differential Load

This simulation setup is good for calculating the throughput, completion time, and other metrics. In this simulation setup parallel system 0 is heavily loaded, and except parallel system 3 all other parallel systems are underloaded loaded. The peering policy for dynamic peering will change as the load in the domain changes, but if the domain is normally loaded then the demand and the supply will be the same and the price for the resource will be the normalized price. The normalized price in dynamic peering is equal to one and if the price is one, then the sharing policy of both static and dynamic peering in the normally loaded domain will be the same. If the sharing policy in static and dynamic peering are the same then both the system will behave the same.

4.5.3 Peering Domains With Extreme Load

In this simulation setup parallel system PS0 is heavily loaded and all other parallel systems are underloaded. The main aim in doing this simulation is to see if there is a difference in the peering policy of dynamic and static peering. Except PS0, all the other

parallel systems are underloaded and when compared with static peering, the underloaded domains in dynamic peering will share more resource. The simulation is done for 575774 seconds, therefore the supply for the simulation is calculated for 575774 seconds.In all simulations, calculation for all the systems is done until one of the system fails, and in this simulation, static peering system fails after 575774 seconds. Figure 4.6, shows that load in parallel system 0 is great when compared to all other parallel system.

Domain Name	Demand (sec.)	Supply in (sec.)	No. of processors
ANL	405064400	69092880	120
CTC	30787660	24758280	430
KTH	43840455	57577400	100
SDSC	14449600	73690720	128
CTC	30576090	24582820	430

Table 4.5: Input For Parallel System

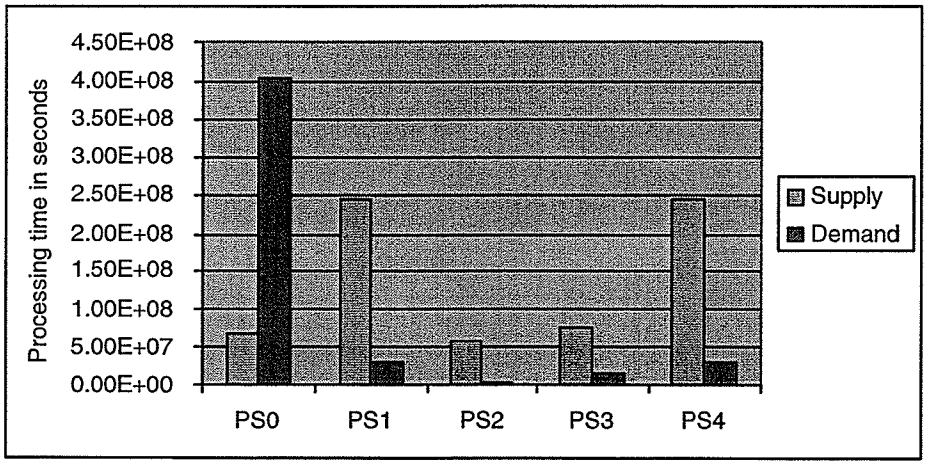

Figure 4.6: Supply And Demand For Extreme Load

4.6 Results

This section discuss the results of the simulation done for dynamic and static peering system and load balancing system. This section is divided into subsections, and each subsection talks about the results obtained from different simulation setup. In the results, ANL parallel system is referred as PS0, SDSC as PS3, and KTH as PS2. The simulation has used two CTC machines, the CTC in the second row of table 4.3 is refereed to PS1, and CTC in the fifth row of table 4.3 is referred as PS4.

4.6.1 All Peering Domains Overloaded

This subsection, discusses the results obtained for simulation setup explained in section 4.5, subsection 4.5.1, where all peering domains are overloaded. As shown in figure 4.4, the demand for processing time in all the parallel systems are greater than the supply. The demand for processing time of parallel system 2 is not as high as compared with other parallel systems, and the supply of processing time for parallel system 2 is relatively less than all other parallel systems. In this simulation, we can measure the system capacity and all other metrics mentioned in section 4.1. Table 4.6 shows the metrics for load balancing, table 4.7 shows the metrics for static peering and table 4.8 shows the metrics obtained from dynamic peering. In all the tables, ACT is the average completion time, TAT is the turn around time, NRU is native resource utilization, FRU is foreign resource utilization and SC is system capacity.

First, let us look at the throughput for the simulation. Figure 4.7 shows the domain level throughput for the different systems. The domain level throughput of load balancing and static peering for PS1, PS3 and PS4 are higher than dynamic peering. But in the case of PS2, which is the least loaded domain, dynamic peering has higher throughput. This result shows that in dynamic peering, the least overloaded parallel

Parallel system	ACT (sec.)	TAT (sec.)	NRU	FRU	Throughput	Queue	SC (sec.)
PS0	334714	161816	84.243	121.405	481	360	236632300
PS1	338774	168184	106.881	101.227	1042	1290	845988500
PS2	345191	175948	62.297	129.03	143	300	201724700
PS3	350212	175973	105.32	106.841	480	384	259109300
PS4	344598	174003	115.82	89.183	1026	1290	871659400

Table 4.6: Load Balancing Metrics

Parallel system	ACT (sec.)	TAT (sec.)	NRU	FRU	Throughput	Queue	SC (sec.)
PS0	340997	168099	181.641	41.911	468	360	212093800
PS1	334443	163853	151.598	41.913	1044	792	473961100
PS2	316344	144548	124.729	41.9	198	70	57991980
PS3	364371	190131	203.162	41.922	474	376	242248100
PS4	353237	182641	170.217	41.911	996	867	548777200

Table 4.7: Static Peering Metrics

system PS2 is not starved by other highly loaded parallel systems. Load balancing has the smallest throughput for PS2, which means that in load balancing the highly loaded parallel systems are grabbing the resources from the least loaded parallel systems. In peering systems we expect that load in the peering domains should be distributed, but it should not starve the domain with minimum load.

Figure 4.8 shows the average price of the resource when all the peering domains are overloaded. An average price is shown, because in dynamic peering system the value of

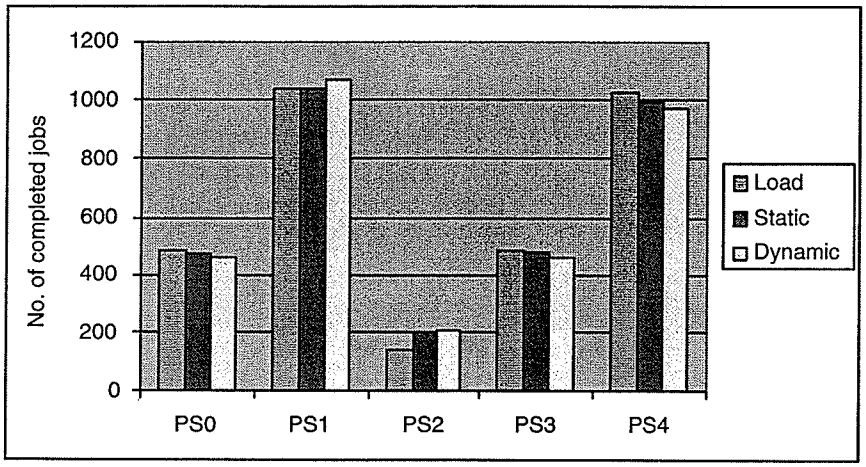

Figure 4.7: Domain Level Throughput For Overloaded Parallel Systems

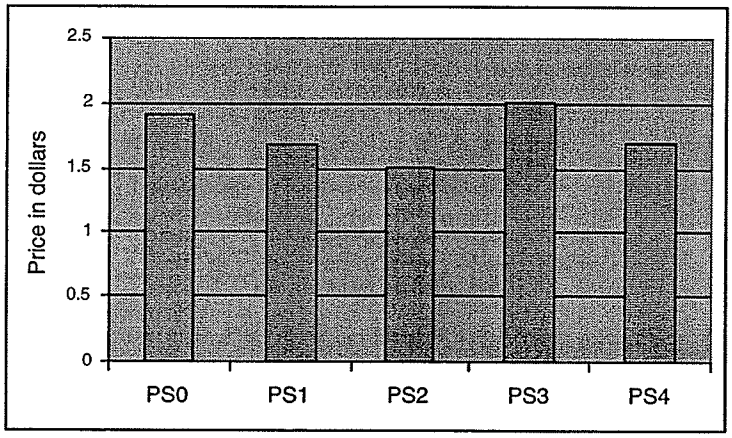

Figure 4.8: Price Of Resource For Overloaded Parallel Systems

Parallel system	ACT (sec.)	TAT (sec.)	NRU	FRU	Throughput	Queue	SC (sec.)
PS0	347977	175079	210.45	21.975	454	351	207830000
PS1	330016	159426	168.219	24.945	1068	723	43141070
PS2	306949	135153	130.187	27.854	204	47	44452380
PS3	374108	199976	231.196	20.879	456	384	224188400
PS4	355230	184634	184.991	24.823	974	862	529357500

Table 4.8: Dynamic Peering Metrics

the resource is calculated between a fixed time interval, and in a simulation there are possibility of having different prices in different time intervals. As mentioned in chapter 3, when the price of the resource in a dynamic peering system is equal to the nominal price, then dynamic peering system will behave like static peering system. But in this simulation setup, the price of the resource is greater than the nominal price, therefore, when compared to static peering dynamic peering will share less resource.

The behavior that we see in the throughput of parallel system 2 can be better explained with the foreign resource utilization graph in figure 4.9. We can see that the resource utilization of the foreign jobs is less in the case of dynamic peering. In dynamic peering, as the load in a peering domain is increased the resource shared with other peering domains will decrease. In this simulation, the load in all the peering domains are high, therefore, the foreign utilization of all the peering domains are less in dynamic peering. Jobs in the queue of all the parallel systems are also taken into consideration while calculating the resource utilization, and this is the reason for some of the resource utilization being greater than 100%.

In this simulation, as all the domains are overloaded, the native resource utilization

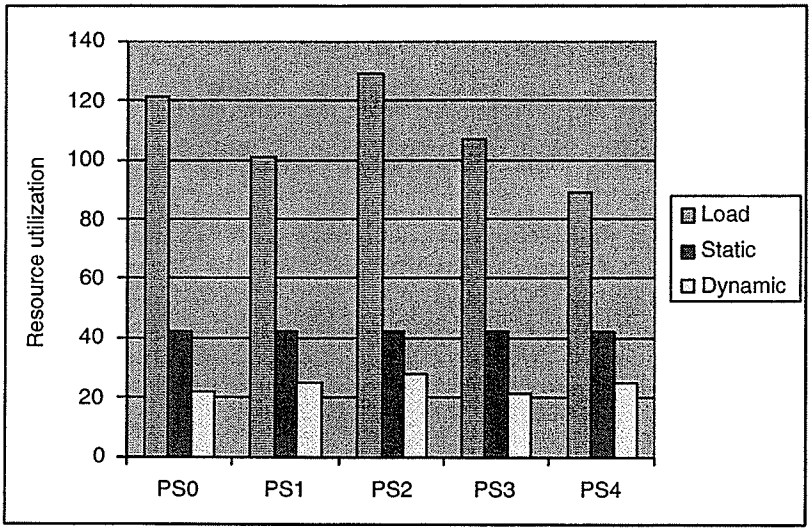

Figure 4.9: Foreign Resource Utilization For Overloaded Parallel Systems

is expected to be high. In the case of dynamic and static peering system as there is restriction in the foreign resource utilization, the native resource utilization for both dynamic and static peering is high. Figure 4.10 shows the native utilization for different parallel systems. Figure 4.10 also shows that the native resource utilization for dynamic peering is the highest. This is because in dynamic peering system as the load increases, the price of the resource increases and therefore sharing with foreign domain decreases.

In the case of average completion time, Figure 4.11 shows that the average completion time for parallel system 2 is the lowest in dynamic peering. This can be explained by the resource utilization of the native and foreign jobs of the different parallel systems. The foreign utilization of parallel system 2 is the least for dynamic peering, and therefore most of parallel system 2's resource is used by the native jobs, and therefore, the average completion time of PS2 is less in the case of dynamic peering. The average completion time of PS2 in static peering is also less when compared to load balancing and static peering. As parallel system 1 is overloaded, it is expected that in load balancing, parallel system 1 will get a greater share of the global resource.

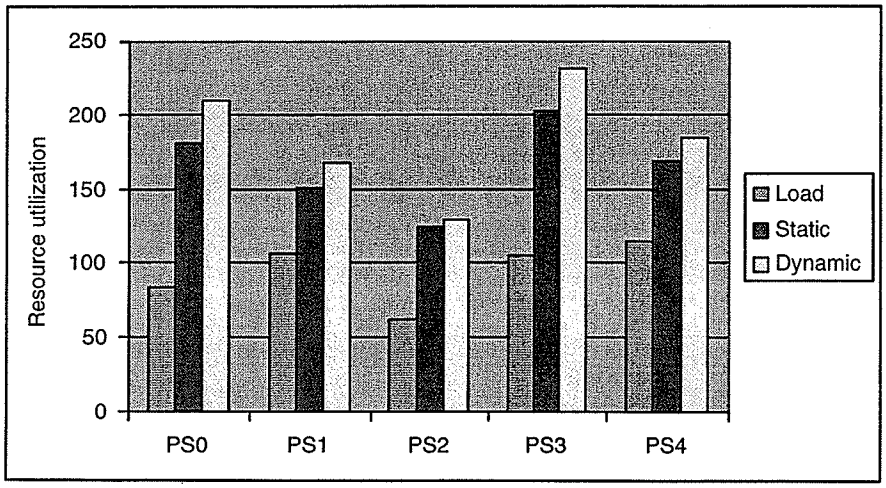

Figure 4.10: Native Resource Utilization For Overloaded Parallel Systems

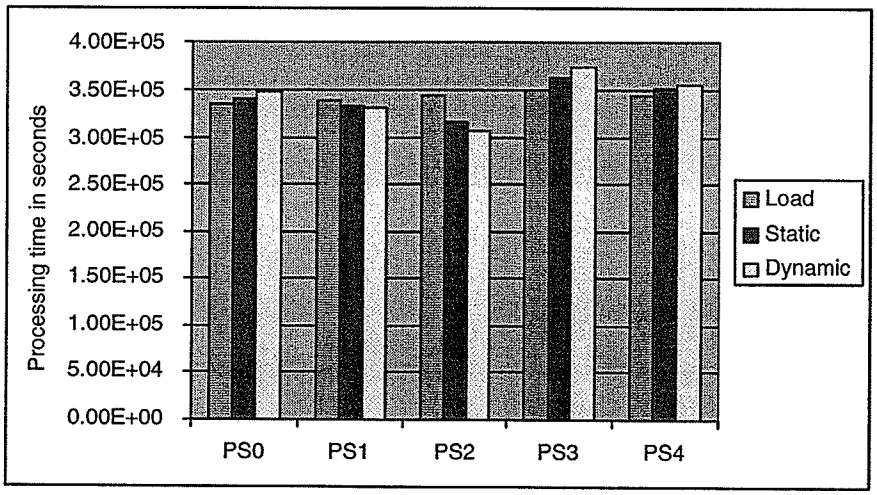

Figure 4.11: Average Completion Time For Overloaded Parallel Systems

Turn around time gives almost the same information as average completion time. Figure 4.12, shows the turn around time of different parallel systems under different peering policies. The behavior in turn around time has the same explanation as average completion time.

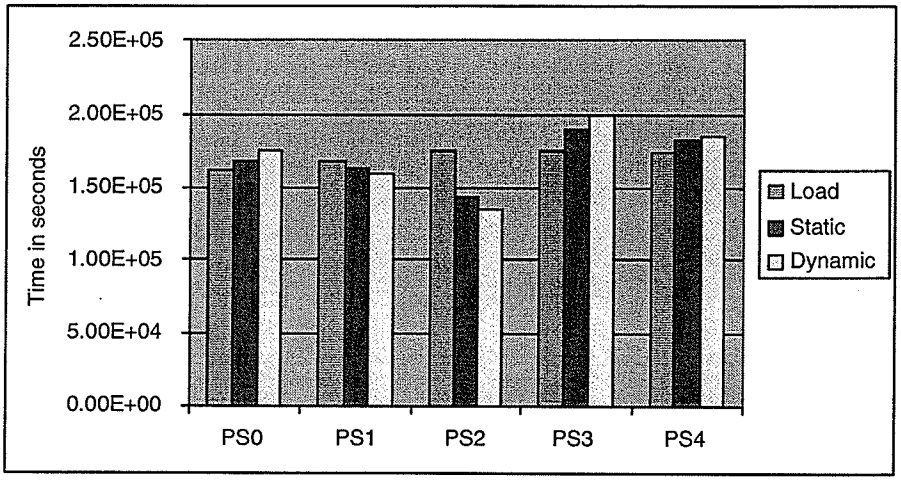

Figure 4.12: Turn Around Time For Overloaded Parallel Systems

When the system fails, the addition of processing time of jobs in the queue, and unfinished jobs, will give the system capacity. Figure 4.13 shows the system capacity, and it is clear that the system capacity of load balancing is the highest. This is because, load balancing fails only when the queue in all the parallel systems reaches its threshold. In load balancing, when one parallel system's queue reaches its threshold, it will move its job to another parallel system, and this happens until all the parallel systems reaches its threshold.

In simulation setup explained in subsection 4.5.1, all the domains are overloaded and therefore the dynamic peering will be sharing the least amount of resource. Therefore, when one parallel system reaches its queue threshold, it has very few peering resources to execute its jobs. This is evident from figure 4.14, which shows the queue length on

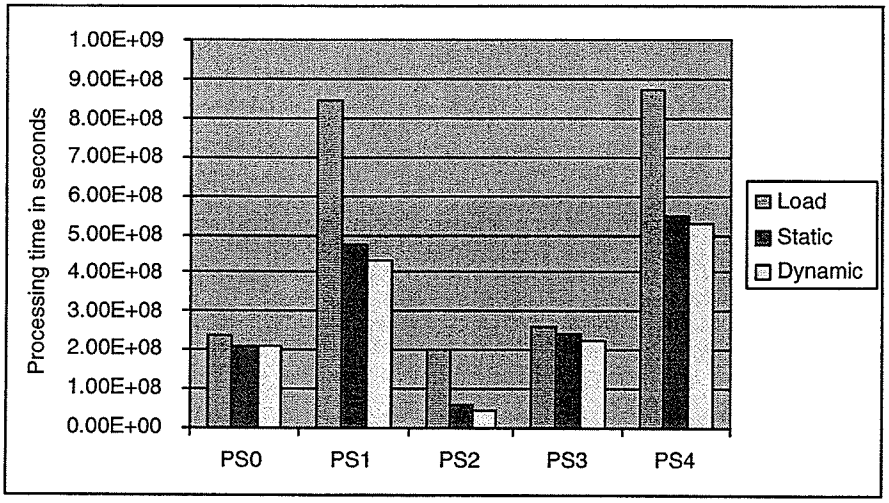

Figure 4.13: System Capacity For Overloaded Parallel Systems

each parallel system. In figure 4.14, we can see that the overall jobs in the parallel system queue is less in dynamic peering. In static peering, as the sharing policy is fixed irrespective of the load on the system, each parallel system shares more resource than in dynamic peering. This simulation setup show that, as all the parallel systems are overloaded, dynamic peering will have a poor system capacity. But, as we can see from other metrics, dynamic peering has more fair resource allocation.

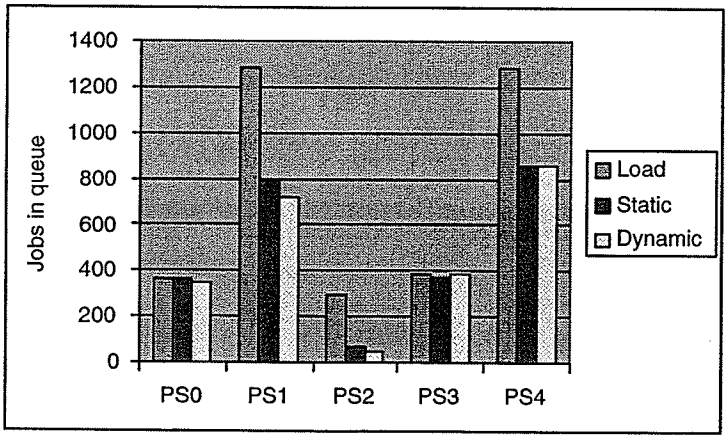

Figure 4.14: Queue For Overloaded Parallel Systems

In conclusion, when all the system are overloaded, dynamic peering system will not starve the least loaded domain. This is evident from the resource allocation for PS2 parallel system. The global system performance of load balancing is better than any other peering system, but, the resource allocation becomes unfair when least loaded domains are starved.

4.6.2 Peering Domains With Differential Load

This subsection talks about the results obtained for simulation setup explained in section 4.5, subsection 4.5.2. As shown in figure 4.5, the demand for processing time in parallel system PS0 is greater than the supply. The demand in parallel systems PS1, PS2, and PS4 are less than the supply, and the demand in PS3 is almost same as the supply. This simulation is done to see the load distribution from the overloaded domain to the underloaded domains. We also have one parallel system PS3 which is normally loaded, for which we can measure the fairness of resource allocation. Table 4.9 shows the metrics obtained from simulating a load balancing system, table 4.10 shows the metrics for static peering system, and table 4.11 shows the metrics obtained from simulating dynamic peering system.

As mentioned in subsection 4.5.2, PS0 is the only parallel system which is overloaded. Therefore, it will be useful to see how the load in PS0 is distributed to other parallel systems. Figure 4.15 shows that the throughput for parallel system PS0 is the least in the case of static peering. Load balancing has the highest throughput for PS0, this is because both dynamic and static peering have restriction in the resource usage for foreign jobs. As PS1, PS2 and PS4 are underloaded, load balancing was able to use many of resources from these underloaded parallel systems. As static peering has rigid peering policy, it was not able to use many of resource from the underloaded parallel systems.

Parallel system	ACT (sec.)	TAT (sec.)	NRU	FRU	Throughput
PS0	594773	110275	78.64	29.02	3337
PS1	540116	55369	36.16	62.96	4164
PS2	543475	55394	21.12	70.64	689
PS3	545520	58703	52.73	47.33	1862
PS4	541512	56780	45.46	45.12	4170

Table 4.9: Load Balancing Metrics

Parallel system	ACT (sec.)	TAT (sec.)	NRU	FRU	Throughput
PS0	632682	148184	145.78	8.99	3121
PS1	537898	53151	50.86	39.99	4202
PS2	541315	53235	37.73	39.99	693
PS3	548548	61731	65.12	36.85	1863
PS4	541506	56774	55.8	32.66	4170

Table 4.10: Static Peering Metrics

In dynamic peering the sharing policy changes with varying load, therefore the resource shared by underloaded domain are more than static peering. But, when compared to load balancing, dynamic peering has less throughput because of the restriction in the resource usage of the underloaded parallel systems.

Figure 4.16 shows the average price of different peering domains. From figure 4.16, we can see that the average price of resource in PS0 is the highest and the average price of resource in PS3 is close to the nominal price, which is one dollar. The average prices

Parallel system	ACT (sec.)	TAT (sec.)	NRU	FRU	Throughput
PS0	623755	139304	144.74	6.607	3182
PS1	537806	53059	48.29	44.4	4203
PS2	551501	55387	33.16	52.69	702
PS3	544880	58063	72.98	24.78	1902
PS4	541456	56724	56.9	31.4	4121

Table 4.11: Dynamic Peering Metrics

of resource in PS1, PS2 and PS4 are less than the nominal price, and this indicates that the demand in PS1, PS2 and PS3 are less than the supply.

Foreign resource utilization will also give a good idea on the load distribution in the parallel systems. In this metrics, we need to look at parallel systems which are overloaded and underloaded. In overloaded parallel systems, it is better if the foreign resource utilization is less, and in underloaded parallel systems the foreign utilization is expected to be high. Figure show that foreign resource utilization in parallel system PS0 is the least in the case of dynamic peering. This shows that as the load increases the dynamic peering, enforces a poor resource sharing, so more resource can be used by the local jobs. Load balancing has the highest foreign resource utilization for parallel system PS0, which shows that, without any control of resource sharing, even domains with high load is overloaded with foreign jobs. The reason for high foreign resource utilization in load balancing is, PS0 uses other parallel systems resource and overloads the underloaded parallel system, and jobs from underloaded domains use PS0 resource.

The foreign resource utilization of the underloaded domains are the highest in load balancing. This is because, unlike dynamic peering system, load balancing does not have

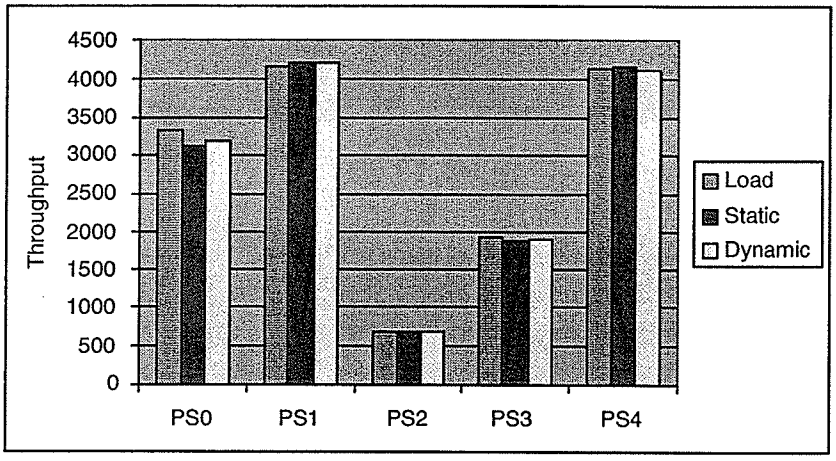

Figure 4.15: Throughput For Differential Load

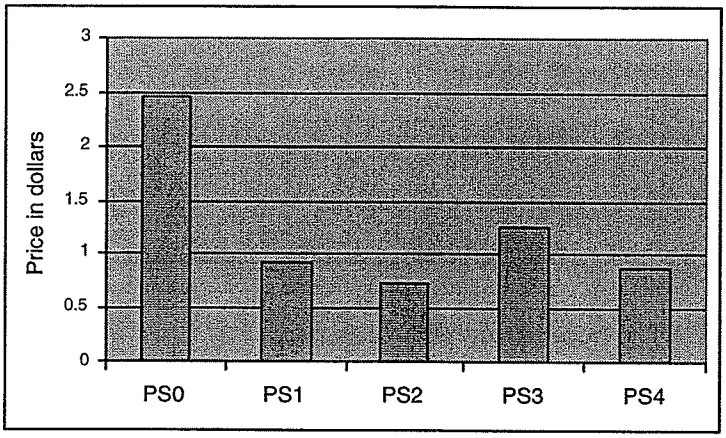

Figure 4.16: Average Price For Differential Load Condition

any restriction in using the resource of the underloaded peering domains. In the case of static peering system, as the peering policy is fixed, the overloaded domains uses the least resource from the underloaded domains.

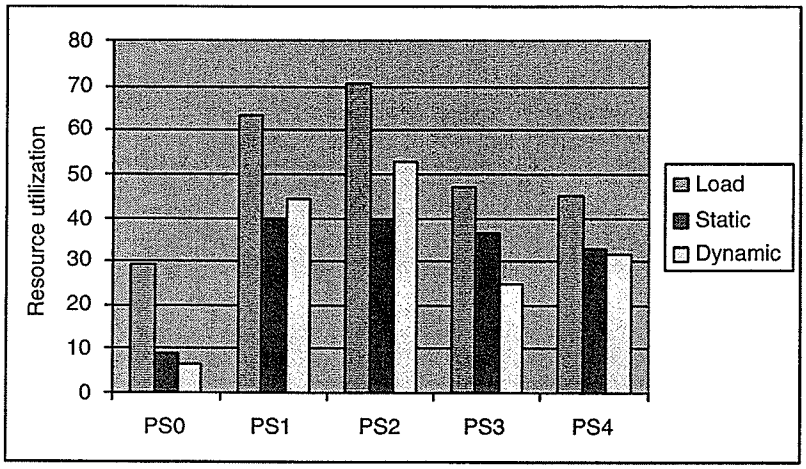

Figure 4.17: Foreign Resource Utilization For Differential Load

When compared to static and dynamic peering, load balancing has high foreign resource utilization. Figure 4.18, shows that native utilization for load balancing in all the parallel systems are less. There are two reasons for this condition, one is because of the over usage of local resource of parallel systems by foreign jobs. The second reason is, in dynamic and static peering, there is restriction in local resource usage by foreign jobs, which will constraint lots of local jobs to be executed in the host parallel system.

In the case of load balancing, since there is no restriction in local resource usage, the jobs are executed in the parallel system which gives the best completion time. Therefore, it is expected that load balancing has the best completion time and turn around time. Figure 4.19, shows that average completion time for load balancing is better in all parallel systems except PS3. PS3 is the parallel with normal load, and we see that dynamic peering has better average completion time and turn around time. This shows that, dynamic peering has better fair allocation of resource.

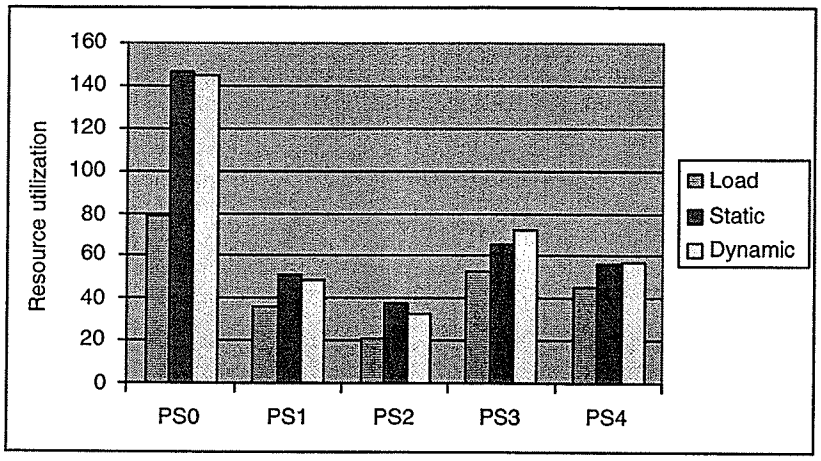

Figure 4.18: Native Resource Utilization For Differential Load

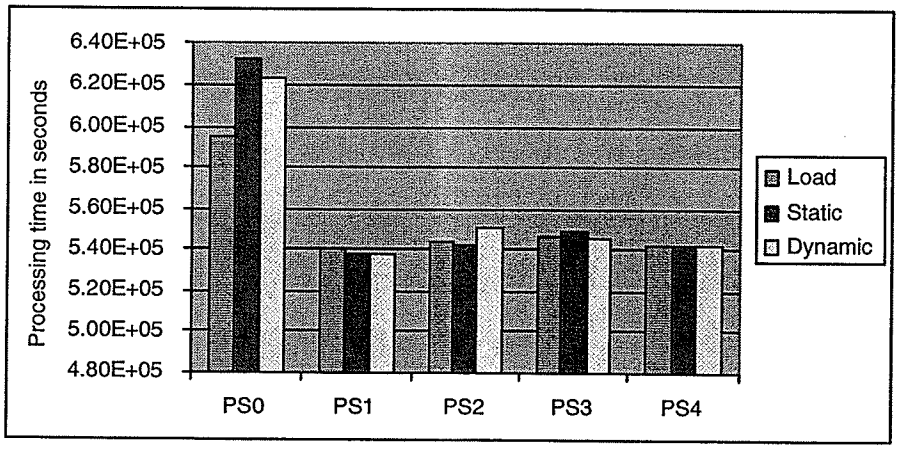

Figure 4.19: Average Completion Time For Differential Load

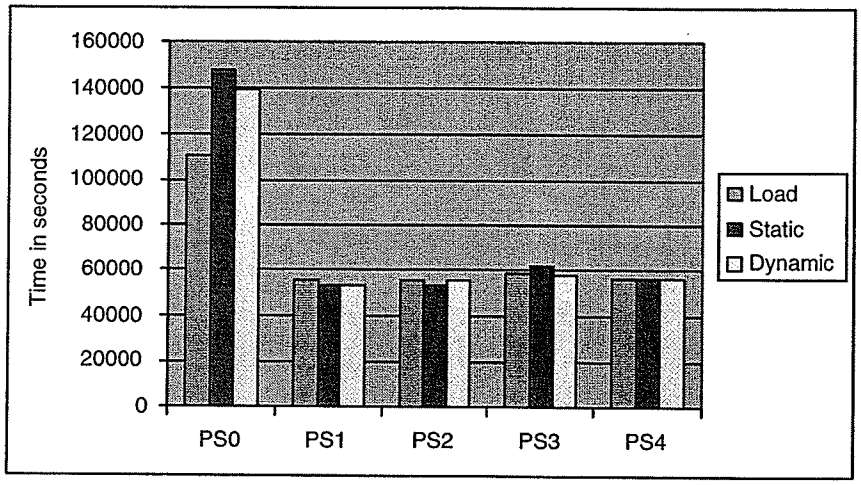

Figure 4.20: Turn Around Time For Differential Load

4.6.3 One Domain Overloaded And All Other Domains Underloaded

In this simulation, except PS0 all other parallel systems are under-loaded. The main aim in doing this simulation is to see how bad the static peering performance. As static peering has rigid peering policy, the under-loaded resource are not properly used. The input for this simulation is shown in table 4.5. Table 4.12 shows the metrics for load balancing, table 4.13 shows the metrics for static peering and table 4.14 shows the metrics for dynamic peering.

In this simulation, load balancing and dynamic peering has the same results. This is because only parallel system PS0 is overloaded and all other parallel systems are underloaded. In the case of dynamic peering, the underloaded parallel systems will share almost the entire resource and the overloaded resource will have very few foreign jobs, this is because all the foreign parallel systems are underloaded, and ultimately dynamic peering will be the same as load balancing.

The throughput for PS0 is higher in the case of load balancing and dynamic peering.

Parallel system	ACT (sec.)	TAT (sec.)	NRU	FRU	Throughput
PS0	434370	130666	134.36	0.85	2405
PS1	325422	23342	10.42	100.07	1252
PS2	324743	19732	6.73	69.53	210
PS3	329471	24043	19.2	33.73	576
PS4	325272	23181	12.34	1.29	1249

Table 4.12: Load Balancing Metrics

Parallel system	ACT (sec.)	TAT (sec.)	NRU	FRU	Throughput
PS0	478708	175003	208.77	0	2220
PS1	325422	23342	12.43	41.67	1252
PS2	324743	19732	7.6	41.67	210
PS3	329471	24043	19.2	41.67	576
PS4	325272	23181	12.34	41.67	1249

Table 4.13: Static Peering Metrics

This shows that in static peering, because of the rigid sharing policy, the overloaded parallel system could not use the resource from under-loaded parallel systems. Figure 4.21, shows the result for the throughput for different parallel systems. The throughput for all the underloaded parallel systems are the same, but for the overloaded parallel system PS0, dynamic peering and load balancing has better performance.

Figure 4.22, shows the average price of the resource in each parallel system. It is clear from figure 4.22, that the overloaded parallel system has price higher than the nominal

Parallel system	ACT (sec.)	TAT (sec.)	NRU	FRU	Throughput
PS0	434370	130666	134.36	0.85	2405
PS1	325422	23342	10.42	100.07	1252
PS2	324743	19732	6.73	69.53	210
PS3	329471	24043	19.2	33.73	576
PS4	325272	23181	12.34	1.29	1249

Table 4.14: Dynamic Peering Metrics

price and therefore, the resource sharing in overloaded domain is less in dynamic peering. But the price of the resource in underloaded domains are less than the nominal price, which allows a greater sharing of the underloaded resource.

In static peering, foreign resource utilization in all the underloaded parallel systems are restricted to 40% resource usage, which is the sharing policy. In the case of dynamic peering systems, the foreign resource utilization varies according to the load, and gives a better performance. Figure 4.23, shows that the foreign resource utilization in dynamic peering and load balancing is more in the underloaded parallel system PS1. Foreign utilization of PS3 and PS4 is more in static peering, this is because in dynamic and load balancing, PS1 and PS2 where able to satisfy the demand for PS0 jobs, and the necessity for using PS3 and PS4 resource was very less.

Native utilization in static peering is more than dynamic peering and load balancing. This is because, in static peering there is a rigid peering policy, which will share a fixed resource irrespective of the load, and ultimately even if a parallel system is highly loaded, it will be able to use only a fixed resource from underloaded domains. Figure 4.26, shows the native resource utilization of different peering systems.

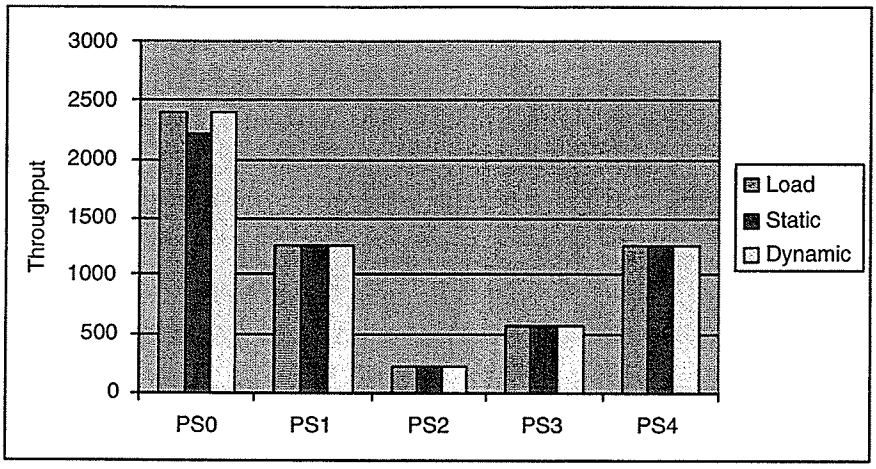

Figure 4.21: Throughput For Extreme Loads

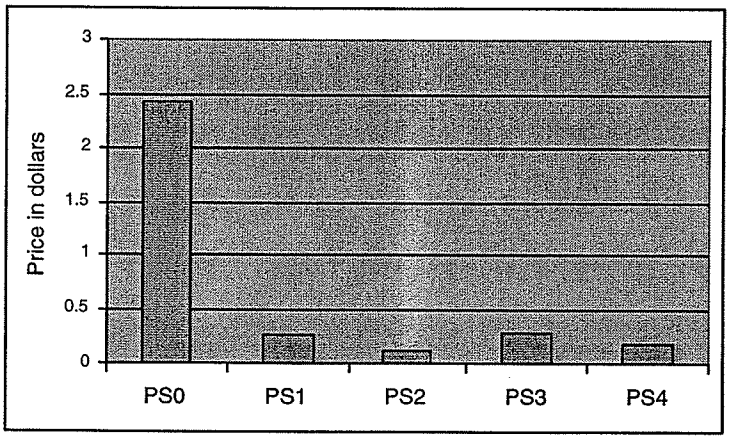

Figure 4.22: Price For Extreme Loads

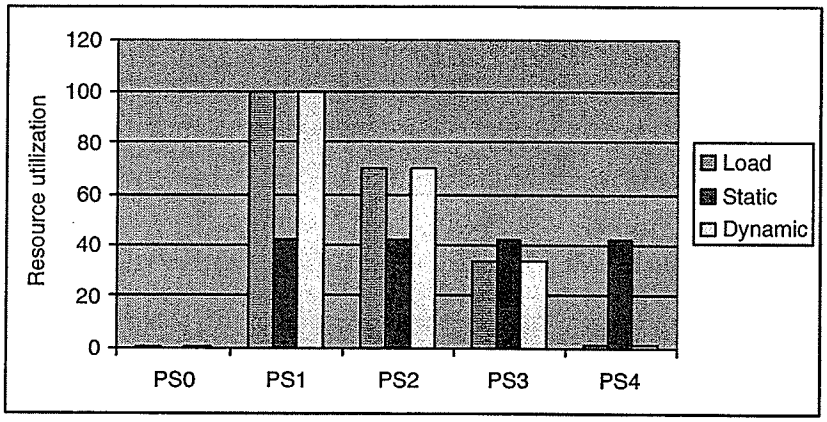

Figure 4.23: Foreign Resource Utilization For Differential Load

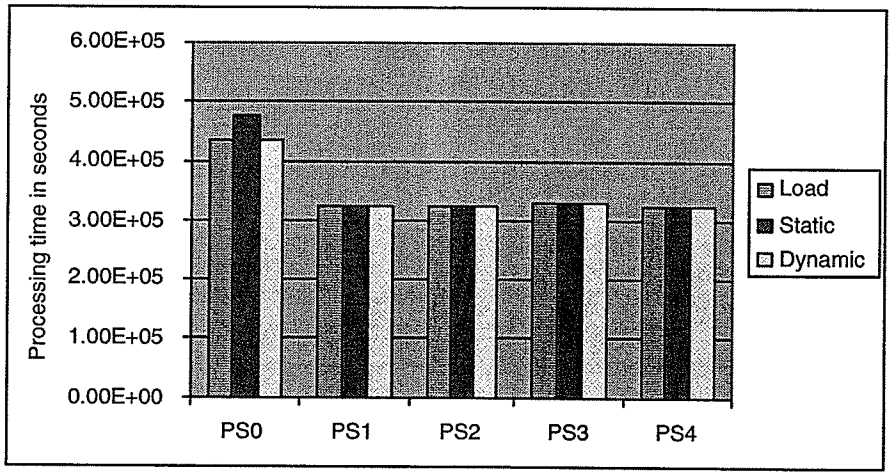

Figure 4.24: Average Completion Time For Extreme Load

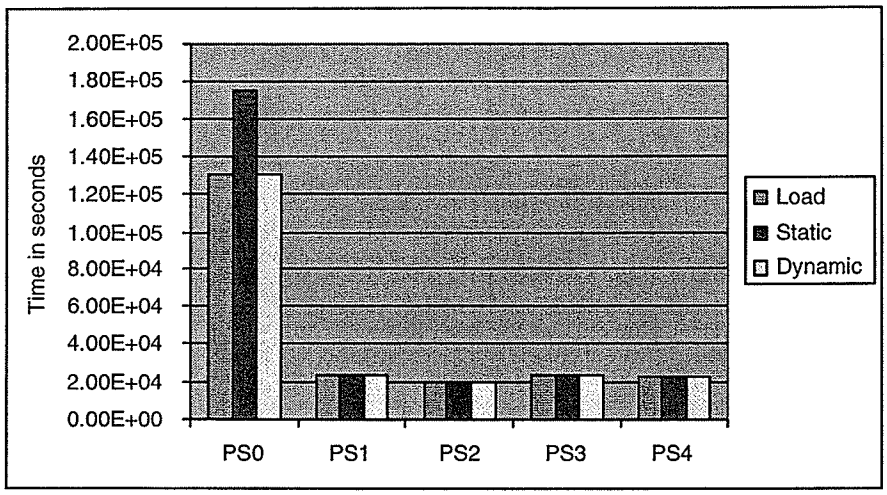

Figure 4.25: Turn Around Time For Extreme Load

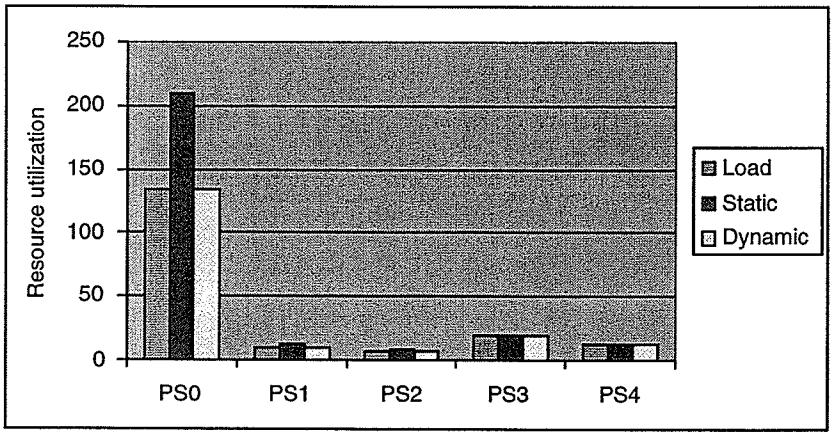

Figure 4.26: Native Resource Utilization For Extreme Load

The average completion time and turn around time for PS0 is better for load balancing and dynamic peering. Figure 4.24, shows the average completion time, and from the graph it is clear that the average completion time for PS0 is better in dynamic peering and load balancing. This shows that in dynamic peering, PS0 uses lots of resource from underloaded domains and ultimately decreasing its average completion time.

Chapter 5

Future Work And Conclusion

This chapter gives a summary on the overall work done in this thesis. This chapter is divided into three section; section 5.1 deals with the contributions made in this thesis, section 4.2 deals with the limitations and assumptions made in this thesis, and section 5.2 deals with the future work that needs to be done.

5.1 Contributions

This thesis make the following contributions:

- *Peering Model:* Dynamic peering system is a new peering system proposed in this thesis. In dynamic peering system, the peering policy of the peering domains change with the changes in load. Peering policy of peering domains in a dynamic peering system is determined using micro-economics concepts. The pricing mechanism used in this thesis combines concepts from auctioning and commodity market to give an efficient and valid pricing scheme.

- *Fair resource allocation:* As the peering policy of peering domains in dynamic peering system is decided by the load on the peering domain, unfair resource allocation

has been avoided, and this is evident from the simulation results. In static peering system, peering policy is fixed and therefore unfair resource allocation is more common. A baseline mechanism is used to compare both static and dynamic peering system.

In dynamic peering system, load on individual peering domain will decide the peering policy of that peering domain, therefore, when all the peering domains in a peering system are overloaded, global system performance in dynamic peering system might be less than static peering. This is because, even if all the peering domains are overloaded, the load in each peering domain will be different, and in static peering, as the peering policy is fixed, the highly overloaded peering domain can use more resource from less overloaded peering domains. But once again, the fairness in the resource distribution will be better in dynamic peering.

- *Resource Utilization:* As the peering policy in static peering system is fixed, the unused resources from the under-loaded peering domains are not properly utilized. But, in the case of a dynamic peering system, as the peering policy changes with the load in the peering domain, the unused resources in the under-loaded peering domains are more efficiently used.

- *Multilateral Peering Model*: In this thesis, we have proposed a new peering model which is called as multilateral peering model. This model is more centralized when compared to the the existing bilateral peering model. We have also proposed a hierarchical model of the multilateral peering system to increase the redundancy of the peering system.

5.2 Future Work

In this thesis, simulations are done to show the behavior of all the features explained in this thesis. I hope that someone will use the ideas from this thesis and do more research to develop the proposed model. I have listed the future work that can be done to improve and show some benefits of the proposed model.

- Simulations are done only for multilateral peering system, therefore as a future work more simulations can be done to show the advantages and disadvantages of bilateral and multilateral peering systems.
- In this thesis, simulations are done only for a multilateral peering system with just one node. Therefore, the redundancy of the multilateral peering system cannot be analyzed. If simulation is done with more nodes in the multilateral peering system, then the redundancy of the peering system can be better analyzed.
- More simulations can be done to show the scalability of the peering system. Simulations with more nodes in the peering system can better explain the scalability of the multilateral peering model.
- The demand vector sent by the clients are assumed to be correct. We have not proposed any security model to prevent incorrect information send by the client.

Appendix A

Abbreviations

ACT	Actual Completion Time
BPS	Bilateral Peering System
CIB	Central Information Board
DPS	Dynamic Peering System
FRU	Foreign Resource Utilization
LBS	Load Balancing System
MPS	Multilateral Peering System
NPS	Network Peering System
NRU	Natural Resource Utilization
NSP	Network Service Provider
SC	System Capacity
SPS	Static Peering System
TAT	Turn Around Time

Bibliography

[ChW98] Q. Cheng and P. Wellman, "The WALRAS algorithm: A convergent distributed implementation of general equilibrium outcomes," *Computational Economics*, Vol. 12, 1998, pp. 1–24.

[CoM02] B. Cooper and H. Garcia-Molina, "Peer-to-peer data trading to preserve information," *Information Systems*, Vol. 20, No. 2, 2002, pp. 133–170.

[Cw01] W. Cirne, "A comprehensive model of the supercomputer workload," *In Proceedings of 4th Ann. Workshop on Workload Characterization*, Dec. 2001, pp. 140–148.

[FeN96] D. F. Ferguson, C. Nikolaou, J. Sairamesh, and Y. Yemini, "Economic models for allocating resources in computer systems," in *Market-Based Control: A Paradigm for Distributed Resource Allocation*, S. Clearwater, ed., World Scientific, Hong Kong, 1996.

[LiV02] L. Wang, V. Pai, and L. Peterson, "The Effectiveness of Request Redirection on CDN Robustness," Technical Report, Princeton University, June 2002.

[Me76] Robert A. Meyer, *Microeconomics Decisions*, Houghton Mifflin Company, Houghton Mifflin Company, Boston, 1976.

[SaK98] J. Sairamesh and J. Kephart, "Price dynamics of vertically differentiated information markets," *Proceedings of First International Conference on Information and Computation Economies*, Oct. 1998.

[San02] T. Sandholm, "Algorithm for optimal winner determination in combinatorial auctions," *Artificial Intelligence*, Vol. 135, No. 1-2, 2002., pp. 1–54.

[WaW94] C. A. Waldspurger and W. E. Weihl, "Lottery scheduling: flexible proportional-share resource management," *In Proceedings of the First Symposium on Operating System Design and Implementation*, Nov. 1994, pp. 1–11.

[WoP00] R. Wolski, J. S. Plank, J. Brevik, and T. Bryan, "G-commerce: Market formulations controlling resource allocation on the computational Grid," *International Parallel and Distributed Processing Symposium (IPDPS)*, Apr. 2001.

[ZhK01] T. Zhao and V. Karamcheti, "Enforcing resource sharing agreements among distributed server clusters," Technical Report, New York University, Oct. 2001.