CHAPTER 2

LITERATURE SURVEY

Nowadays, Web service treats client requests evenly, while either being processed by servers or transmitted over a network according to the best-effort model (Yoon-Jung Rhee et al 2002). These requests are stored and forwarded without guarantee of reliable service. Since Web servers have limited resources, they are not performing well up to the satisfaction of the users in terms of the number of requests that are processed. Some applications treat the level of service requests differently using priority schemes. The IETF (Internet Engineering Task Force) community has proposed differentiated service as an efficient and scalable solution to provide better services for the next generation Internet (Nong Ye et al 2005). In reality, service differentiation is achieved, not without any overheads. Overheads should not exceed the average timeliness performance of the best effort model. Earlier research suggest that the network and end-system OS are the best places to provide differentiated services. However, there are some practical difficulties in deployment mechanisms.

Yoon-Jung Rhee et al (2002) proposed two different levels of Web services at the application level by setting different timeout values for TCP connections. Service differentiation is done after the requests are entered into the server log. Persistent and Non-Persistent connections are established for high and low level priority requests. The different holding times are allotted for each client request after user level differentiation. The idle user’s connections are preserved till the end of the holding time assigned. The levels
of priorities for each request are assigned using pseudo random number generator. Results show that the average response time of the high priority requests are improved by up to 25-28% in spite of heavy traffic loading conditions at the Web servers. The dangers of request starvation and unexpected arrival of identical request categories are not considered by the authors (Yoon-Jung Rhee et al 2002).

Amit Sharma et al (2003) suggested simple classification schemes to regulate the throughput of a particular type of request belonging to the service class using probabilistic scheduling. Lars Eggert et al (1999) suggested three simple application level mechanisms at the server side, namely, limiting process pool size, lowering process priorities and limiting transmission rates, to provide two different levels of Web services such as regular and low priority services. The performances of these mechanisms are evaluated under the combination of different workloads, heavy and light, and two levels of available bandwidth (10Mb/s and 100 Mb/s). The application level mechanisms provide better performance for different service classes even in the absence of an operating system or a network support.

Almeida et al (2000) proposed an application and kernel level approaches for improving the QoS in Web server. The application level mechanism limits the server pool sizes to the allocated requests of different categories. The second mechanism is a kernel level scheduler that allows preemption of low-level requests and assigns process priorities based on the request class. Sook-Hyun Ryu et al (2001) suggested two approaches to implement differentiated quality of Web services in Apache server. A Web server is made to include a classification process, priority queues and a scheduler in the user level approach. However, it is difficult to achieve portability with this approach. In kernel approach, a real-time scheduler has been added to the OS kernel to support prioritized user requests. Authors
(Sook-Hyun Ryu et al 2001) have not provided any information related to the implementation of the proposed approach.

Vasiliou et al (2000) designed a differentiated QoS architecture, which can be added to the Apache Web server to provide a differentiated performance to the clients of a Web site, based on the client’s identity and attributes. Huamin Chen et al (2003) proposed a Dynamic Weighted Fair Sharing (DWFS) scheduling algorithm for controlling the overloads at the Web servers. DWFS aims to avoid processing of requests that belong to sessions that are likely to be aborted in the near future. The results show that DWFS can improve server responsiveness by as high as 50% while providing QoS support through service differentiation for a class of application environment.

Nong ye et al (2005) suggested a new Web server model that uses a single queue along with scheduling rules such as Weighted Shortest Processing Time (WSPT), Apparent Tardiness Cost (ATC) and Earliest Due Date applied to production planning in the manufacturing domain to differentiate QoS for requests with different priorities. Results show that the performance of WSPT and ATC is better when the requested services exceed the capacity of a Web server.

Peter Key et al (2004) proposed an application level approach for background low priority data transfer across the wide area environment. A receive window control is used to limit the transfer rate of the application and the optimal rate for transfer is determined by detecting a change-point. Kyoung-Don Kang et al (2003) proposed a differentiated real-time data service framework for e-commerce applications according to the importance and performance guarantees in terms of deadline miss ratio. Data, which is committed within their deadlines, is guaranteed for all transactions.
Surendar Chandra et al (2000) introduced a robust mechanism for managing the network resources using application-specific characteristics of Web services. Transcoding is used to allow Web servers to customize the size of objects constituting a Web page and dynamically varying the size of multimedia objects on a per-client basis. The contribution of this work is to use informed transcoding techniques to provide differentiated service and dynamically allocate available bandwidth among different classes while delivering good quality of information content for all the clients.

Borst et al (2002) considered a system with two service classes, one of which supports elastic traffic. The link capacity is shared between the two traffic classes in accordance with the Generalized Processor Sharing (GPS) discipline. GPS based scheduling algorithm provides a flexible mechanism for service differentiation and prioritization. The transfer delay incurred by the elastic traffic flows is asymptotically equivalent to that in an isolated PS system with constant service rates. The service rate is affected by the streaming traffic through its average rate. Elastic traffic is largely immune from possible adverse traffic characteristics and performance degradation due to prioritization of the streaming traffic. The limitation of this approach is to know the type of traffic before prioritization.

Load balancing policies can be classified as centralized or decentralized. In centralized policies (El-Zoghdy et al 2002; Kameda et al 2001, 1997; Milan et al 2002; Deng et al 1997 and Berger et al 1999), a system with only one load balancing decision maker distributes the incoming requests among the servers in the cluster. The incoming jobs to the system are sent to this load balancing decision maker, which directs the jobs to appropriate processing nodes in the cluster. Centralized policies have the advantages of easy information collection about the job arrivals, departures and the natural implementation employing client-server model of the
distributed processing. The major limitation of the centralized policies are performance and reliability bottlenecks due to heavy load at the centralized job load balancing decision maker (Berger et al 1999). For this reason, the centralized approaches are not appropriate for large-scale systems. Furthermore, the failure of the load balancing decision maker will make load balancing inoperable. It appears that this policy is closely related to the overall optimal policy in that there is only one load balancing decision maker and it makes all the load balancing decisions.

In the past two decades, there has been significant research in load balancing. Several approaches have been put forth for load distribution at the server. Traditionally, the workload was distributed among the servers using a message-passing paradigm. Every server needed to periodically broadcast load information to other servers. For a total of N servers, the broadcast incurred the communication overhead as high as O(N^2) (Jianniong cao et al 2003). Equiload is a load balancing policy for clustered Web servers (Gianfranco ciardo et al 2004). It partitions the workload based on the length of jobs and assigns them to the corresponding servers. However, this method has its own drawbacks. First, the value of the workload boundaries is critical, since it affects the load seen by each server. Second, under highly variable workloads, system performance improves considerably, only if detailed information about the activity of each back-end server is available to the front-end dispatcher (Reinhardt riedl 1999).

Another technique is the LARD – a “locality-aware” scheduling policy that takes the cache contents of the back-end servers and offers significant performance improvement over existing policies (Lang fang et al 2005). Load distribution among replicated Web servers considered the following three principal strategies: a mirror-based strategy, that provides a browser with access to the particular Web server that is geographically closer
to that browser, a DNS-based strategy, that implements load distribution among Web servers by using a round robin discipline, and a QoS-based strategy that implements load distribution among Web servers.

Online client-server-based load balancing without global information modeled as a bipartite graph (Morharchol-Balter et al 2003) produces maximum throughput in a network of clients and servers. Though this approach is interesting, drawbacks persist. Every client must remain active for the next 'r' rounds. The structure of the entire graph is known to all clients and servers, and that they have access to an unlimited supply of shared random bits. In addition, it defines throughput as the number of distinct clients whose requests are satisfied, i.e., a client whose request is satisfied by two or more servers contributes to the throughput. In Join the Shortest Queue (JSQ) load balancing policy (Hwa-Chun Lin 1996), a job is assigned to the queue with the smallest number of jobs. Ties are broken by selecting one of the queues randomly with minimal number of jobs. No jockeying between the queues is permitted. This policy is applicable for systems with single-server queues and systems with multi-server queues. The average response times of the JSQ policy is perfect for the systems with single server queues and systems with multi-server queues.

A client-based approach implements the server selection on the client side. Clients can select one of the Web servers in random using an intelligent selection mechanism. Netscape Navigator browser adopts a client-based approach to access its own Website (MoseDale et al 1997). The Navigator randomly selects one of the servers and directs a user's request to it. The random selection of server cannot guarantee load balancing and availability.
A Domain Name System (DNS) server is the routing mechanism for distributed Web servers. It can select one of the Web servers to process a request by mapping the Uniform Resource Locator (URL) to the IP address of a Web server. This mapping can implement different policies concerning server selection. However, the DNS-based approach has limited control on the request routing due to the intervention of intermediate naming servers that stores the URL to IP address mapping between the clients and the DNS server. The DNS server may become a bottleneck in routing. The DNS server initially assigns the client requests to the Web servers. Then, each server may reassign a request to another server. This is a decentralized load reallocation strategy where all the servers are allowed to participate in the process of load balancing. The DNS servers evaluate network proximity and users served from the closest cluster reduce network impact on response time (Valeria Cardellini et al 2000). The serving request in the nearest server may cause an unbalanced servers and increase system impact on response time. To achieve a scalable Web system, Valeria cardellin et al (2000) proposed to integrate DNS proximity scheduling with an HTTP request redirection and the dispatching mechanism enhances the percentage of requests with guaranteed response times, thereby enhancing the QoS of geographically distributed Web sites.

The Mod_backhand scheme (Schlossnagle 2000) provides a server-based solution for the Web server that allows for seamless reallocation of HTTP requests from heavy loaded servers to the under utilized servers on a per-request basis. The Mod_backhand passes a request through a chain of decision making functions called candidacy functions. Each candidacy function reorders the set of available servers and chooses the method of request reallocation based on the resource information. Milan E. Soklic (2002) introduced a new load balancing algorithm, called diffusive load balancing which consists of two Distributed Component Object Modeling
(DCOM) software components, known as the object factory and the request acceptor. When a client needs to create an object, it contacts the object factory component on one of the servers by examining a list of all servers and selecting the one that has not been called for the longest time from that client. After the selected server is contacted, the object factory component on that server initiates a search for a granting server, which will accept the client's object creation request. The search for a granting server starts with the invocation of the request acceptor component on the selected server, which invokes request acceptors on its directly connected neighboring servers. The request acceptor forwards the request to the server with the smallest amount of workload.

Tariq Omari et al (2004) proposed a novel load balancing based on a traveling token concept for dynamic load balancing. This algorithm identified the heavily loaded and lightly loaded computer instead of probing each computer in the cluster. Qi Zhang et al (2005) focused on load balancing policies for homogeneous clustered Web servers that tune their parameters on-the-fly to adapt changes in the arrival rates and the service times of incoming requests. The proposed algorithm monitors the incoming workload and self adjusts its balancing parameters according to the changes in the operational environment, such as rapid fluctuations in the arrival rates. The Shortest Remaining Processing Time (SRPT) (Nikhil Bansal et al 2001; Mor Harchol-Balter et al 2001) scheduling policy produces minimum mean response time for any sequence of requests. However, it has been suspected by many that SRPT is a very unfair scheduling policy for large requests. SPRT penalizes the large jobs in order to help the small jobs. SPRT is not prevalent in practice because the size of a job is not known in advance.

A load balancing strategy based on “centralized decision making” schemes suffer from the obvious drawbacks of single-point of failure,
problems of caching of browsers’ requests, delay incurred in decision making at a single point, etc. Considering all such issues, a new decentralized framework using mobile agent is proposed. The decentralized policies delegates’ job distribution decisions to individual nodes. Usually each node accepts the local job arrivals and takes decisions to send them to other nodes based on its own partial information on the system load distribution. It is closely related to the optimal policy in that each job (or the user of each) optimizes its own cost (e.g., its own expected mean response time), independently of the others. Decentralized load balancing is widely used to handle the imperfect system load information (Kameda et al 2001; Milan E. Soklic 2002; Deng et al 1997; Valeria Cardellini et al 2000). Decentralized load balancing policies can be broadly characterized as sender-initiated, receiver-initiated, and symmetrically initiated. In sender-initiated policies (Kameda et al 1997; Berger 1999; Deng et al 1997), congested nodes attempt to transfer jobs to lightly loaded ones. In the receiver initiated policies lightly loaded nodes search for congested nodes from which jobs may be transferred. Many policies combine the desired features of both sender and receiver-initiated techniques, popularly known as symmetrically initiated policies.

Jianniong cao et al (2003) proposed a Mobile Agent-based Load Balancing (MALD) framework, which creates a foundation for developing decentralized load balancing schemes on distributed Web servers. MALD framework uses mobile agents similar to us for load balancing. However, the high execution overhead of Java programs results in lower throughput of the MALD scheme when the number of clients is less than 500. The MALD framework specifies three types of agents such as Server Management Agent (SMA), Load Information Agent (LIA) and Job Delivering Agent (JDA). SMA is a stationary agent that is responsible for monitoring the workload on local servers and executing job transfer policy if it is required. LIA is a mobile agent responsible for information gathering. It travels across the
network of servers, collects the load information and propagates the load information to the servers. JDA is activated by the SMA sits in an overloaded server. A JDA executes the server selection policy to select another server to receive the reallocated job.

The Tiered Service (TS) architecture provides overhead control and service differentiation for load balancing at the back-end systems (Reinhardt Riedl 1999). However, this is effective only if there is an equal load distribution at the front-end. Multi-agent computing on a cluster is widely envisioned to be a powerful paradigm for building useful distributed applications (Ka-po Chow et al 2002). Ka-Po-Chow et al (2002) proposed an algorithm by associating a credit value with each agent. The credit of an agent depends on its affinity to a machine, its current workload, its communication behavior, mobility etc. When a load imbalance occurs, the credits of all agents are examined and an agent with a lower credit value is migrated to relatively lightly loaded machines in the system.

Owe Dralle et al (1997) proposes an algorithm for distributed and parallel systems that provides optimal concurrent communication and load balancing. This algorithm can be used for determining optimal task migration paths in meta-computing environments and heterogeneous computer networks. The multiple-commodity problems in operations research can be solved using this method. Qi Zhang et al (2002) focused on load balancing policies for clustered Web servers that tune their parameters on-the-fly to adapt to the changes in the arrival rates and service times of the incoming request. The proposed scheduling policy, ADAPTLOAD, monitors the incoming workload and self-adjusts its balancing parameters such as rapid fluctuations in the arrival rates and document popularity according to changes in the operational environments. The proposed load balancing policy is based
on statistical information derived from recent workload history without requiring locality data.

Insensitive load balancing (Bonald et al 2004) is a linear combination of simple strategies for both a single customer class and several customer classes with decentralized routing decisions. This characterization leads to simple optimality. Authors (Bonald et al 2004) focused on an external routing decision where successive nodes are visited by a customer depending on the load of the previous nodes. In the classical load balancing, the given sequences of jobs of varying sizes are assigned to one of the servers in the cluster (Shirazi et al 1995). The objective is to minimize the load on the overloaded server. The initial assignment is not remaining optimal with time. A set of jobs are relocated to other servers to increase the performance of the server in the cluster. The goal is to achieve the best possible performance under a constraint that no more than 'k' jobs are relocated.

Load balancing exploits the communication facility between the servers of a distributed system, by exchanging status information and jobs between any two servers of the system in order to improve the overall performance. Cho Cho Myint et al (2005) attempted to balance the load by splitting up the processes into separate jobs and then distributes them equally among the nodes by a novel load balancing method instead of moving an entire process to a less loaded computer. There are five kinds of agents, which take responsibility to balance the workload among clusters namely Directory agent, Account agent, Threshold agent, Monitor agent and the Client agent. The Directory agent keeps track of the nodes within a cluster. The Monitor agent shows the resource condition of a node. The Account agent keeps tracks of the resources used by each executing agents. The Threshold agent controls the access to a node and migrate agent with load to be executed at other nodes.
in the cluster if the current node becomes overloaded. The Client agents carry the workload to be executed at the client nodes in cluster.

Steffen Lipperts et al (2005) proposed a mobile agent-based approach to load balancing in telecommunication networks. A decentralized approach is used for network management that is based on independent, autonomous agents with the overall system behavior being drawn from the emergence of their interaction. The Load agents and the strategy agents are used for balancing the load in the network. The Load agents operate in the lowest layer of the architecture and determine the paths offering the largest free capacity form the current node to each of the other nodes in the network and then modify the routing table accordingly, making use of its ability to migrate to the other nodes. The Strategy agents are responsible for the creation and the termination of load agents.

Asser Tantawi et al (1985) proposed two efficient algorithms that determine the optimal load on each host computer. The first algorithm is used to improve the speed of the communication network. The second algorithm is a single-point algorithm, which yields the optimal solution for given system parameters. In both algorithms, a job may be either processed at the host which it arrives or transferred to another host. The performance metric considered for both the algorithms is the *mean response time*. The main contribution of the work is a parametric analysis algorithm that allows a designer to study the effects of increasing communication times on the optimal load balancing strategy.

Minar et al (1999) proposed an algorithm for load balancing in heterogeneous, dynamic, P2P systems. This algorithm achieves minimum load imbalance compared to the total amount of load transferred. The basis of the load balancing algorithm is to store load information of the peer nodes in a
number of directories, which periodically schedule reassignments of virtual servers to achieve better balance. Thus, it reduces the distributed load balancing problem to a centralized problem at each directory. Hong Bong Kim et al (2005) proposed a load balancing scheme for centralized wireless networks using the users in the overlapping area between cells can efficiently and dynamically reconnected to one of the accessible base stations by a central control station. A simulation study was done to compare the proposed scheme with conventional load balancing strategies, showing that the proposed scheme is more efficient for optimum system resource utilization.

Workflow Management Systems (WFMS) play a very important role in constructing e-commerce environments through automating business processes and services. Li-jie Jin Fabio Casati et al (2001) introduced a workflow load index to measure the load level of workflow engines to achieve scalability in a distributed WFMS. The WFMS architecture includes a business process unit cluster and a load balancing sub-system. The load aware balancing mechanism is suited for managing the load of distributed workflow management system in order to achieve scalability and support deadline and execution priority properties for business processes with multiple processing units.

Berger et al (1999) designed an integrated load distribution load balancing algorithm which was targeted to be both efficient and scalable for dynamically structured computations. This algorithm combines an initial partitioning of the graph with application of randomized work stealing on the basis of sub-graphs to refine imbalances in the initial partitioning and to balance the additional computational work generated by runtime instantiation of sub graphs and nodes. Sivarama P. Dandamudu et al (1997) proposed the Virtual Tree Algorithm (VTA), which creates a virtual binary tree structure over the actual network topology. It uses the difference-initiated (DI)
technique for load balancing, which needs remote information for the transfer policy and no additional information for the location policy. Authors demonstrate that the introduced virtual construction keeps the exchanged messages at relatively low levels.

Chenyang Lu et al proposed the design, implementation and evaluation of an architecture to provide delay guarantees for different service classes on Web servers under HTTP 1.1. Their contribution is feedback control loop that enforces desired delays among classes through dynamic connection scheduling and process reallocation. Li et al (1999) used control theory to develop a feedback control loop to guarantee desired network packet rate in a distributed visual tracking system. Hollot et al (2001) applied control theory to analyze a congestion control algorithm on IP routers. Control theory analysis is used on computing systems and service delays on end-server systems. Rate Monotonic (RM) (Liu et al 1973, Abdelzaher et al 2000) and Earliest Deadline First (EDF) (Kar et al 1989) require complete knowledge about the task set such as resource requirements, precedence constraints, resource contention, and future arrival times. Unlike RM or EDF, the dynamic real-time systems do not require future task arrival time to be known a priori. RM and EDF can support sophisticated task set characteristics and unpredictable environments where an accurate workload model is not available.

A feedback control loop was used to control the desired CPU utilization of a Web server with admission control (Abdelzaher et al 2000; Al-Omari et al 2003). CPU utilization control can be extended to guarantee the desired absolute delay in Web servers under HTTP 1.0 protocol. This technique is not applicable to the servers under HTTP 1.1. Popular scheduling algorithms that are used in Web servers include Round Robin (RR), Earliest Deadline First (EDF) (Chenyang lu et al 2002) introduced a measure called
the server's productivity, which is defined as the ratio of the number of requests completed to the number of requests aborted within a specified time interval. Sha et al (2002) proposed *Queuing Model Based Feedback Control* architecture that provides improved mean delay regulation for Web servers under single client classes. The design includes a feed forward predictor, which keeps the system state near an equilibrium operation point in the presence of dynamic workloads. This is essentially linear.

Al-omari et al (2003) proposed an *Open Loop Dynamic Scheduling* algorithm that employs overlaps in order to provide flexibility in task execution times. Chenyang Lu et al (2003) proposed a *Feedback Control real-time Scheduling* (FCS) framework for adaptive real-time systems. It contains an FCS architecture that uses plug-ins of different real-time scheduling policies and QoS optimization algorithms. They applied a design methodology that is based on *control theory* to systematically design FCS algorithms, which satisfied desired transient and steady state performance specifications of real-time systems. The FCS algorithms provide robust transient and steady state performance guarantees for periodic and aperiodic tasks even when the task execution times vary by as much as 100% from the initial estimate. Ming Zu et al (2003) Bonomi et al (1990) proposed a hierarchical framework of reward-based tasks which consists of a mandatory part that must be accomplished before the given deadline, and an optional part that is associated with rewards for partial completion.

After careful analysis of the past research work on differentiated services at application level, load balancing and real-time scheduling, new and customized approaches are introduced at the server side to improve the effective handling of Web requests in distributed environment. Chapter 3 discusses with a customized tuning algorithm which is used for modifying the predefined weight of the queue depending on the number of previous requests
in that queue. Chapter 4 describes the Load balancing decision making using Decentralized Mobile Agents (LDMA) and Least Total Processing Time (LTPT) load balancing algorithms, their framework, operation and performance in LAN Web server environment. WLDMA, a new proposed load balancing approach for WAN environments is discussed in Chapter 5. Chapter 6 discusses about the sequential and differentiated real-time Web server architecture with open and closed loop and its implementation. Chapter 7 discusses about the Conclusion and future work.