iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** V **Month of publication:** May 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Network Load Balancing Using Distributed Algorithm

Prof. S. G. Anantwar[1], Miss. Ujjwala Kharkar[2]
[1,2]*Information Technology,S.G.B.A.U. Amravati, Maharashtra, India*

Abstract:-Network Load Balancing (NLB) is a clustering technology offered by Microsoft as part of all Windows 2000 Server and Windows Server 2003 family operating systems. NLB uses a distributed algorithm to load balance network traffic across a number of hosts, helping to enhance the scalability and availability of mission critical, IP-based services, such as Web, Virtual Private Networking, streaming media, terminal services, proxy and so on. It also provides high availability by detecting host failures and automatically redistributing traffic to operational hosts.The performance impact of Network Load Balancing can be measured in four key areas: CPU overhead on the cluster hosts, which is the CPU percentage required to analyse and filter network packets (lower is better).Response time to clients, which increases with the non-overlapped portion of CPU overhead, calledlatency (lower is better). Throughput to clients, which increases with additional client traffic that the cluster can handle prior to saturating the cluster hosts (higher is better).Switch occupancy, which increases with additional client traffic (lower is better) and must not adversely affect port bandwidth.

I. INTRODUCTION

Network Load Balancing (NLB) is a clustering technology offered by Microsoft as part of all Windows 2000 Server and Windows Server 2003 family operating systems. NLB uses a distributed algorithm to load balance network traffic across a number of hosts, helping to enhance the scalability and availability of mission critical, IP-based services, such as Web, Virtual Private Networking, streaming media, terminal services, proxy and so on. It also provides high availability by detecting host failures and automatically redistributing traffic to operational hosts. Network Load Balancing servers (also called *hosts*) in a cluster communicate among themselves to provide key benefits, including:

A. Scalability

Scalability is the measure of how well a computer, service, or application can grow to meet increasing performance demands. For NLB clusters, scalability is the ability to incrementally add one or more systems to an existing cluster when the overall load of the cluster exceeds its capabilities. To support scalability, NLB can do the following:Balance load requests across the NLB cluster for individual TCP/IP services.Support up to 32 computers in a single cluster.Balance multiple server load requests (from either the same client or from several clients) across multiple hosts in the cluster.Support the ability to add hosts to the NLB cluster as the load goes up, without bringing the cluster down.Support the ability to remove hosts from the cluster when the load goes down.Enable high performance and low overhead through fully pipelined implementation. Pipelining allows requests to be sent to the NLB cluster without waiting for response to the previously sent one.

B. High-availability

A highly available system reliably provides an acceptable level of service with minimal downtime. To provide high availability, NLB includes built-in features that can automatically:Detect and recover from a cluster host that fails or goes offline. Balance the network load when hosts are added or removed. Recover and redistribute the workload within ten second.

For example, the clustered hosts in Figure 1 below work together to service network traffic from the Internet. Each server runs a copy of an IP-based service, such as Internet Information Services 5.0 (IIS), and Network Load Balancing distributes the networking workload among them. This speeds up normal processing so that Internet clients see faster turnaround on their requests. For added system availability, the back-end application (a database, for example) may operate on a two-node cluster running Cluster service.

II. DISTRIBUTION OF CLUSTER TRAFFIC

Network Load Balancing uses layer-two broadcast or multicast to simultaneously distribute incoming network traffic to all cluster

hosts. In its default *unicast* mode of operation, Network Load Balancing reassigns the station address ("MAC" address) of the network adapter for which it is enabled (called the *cluster adapter*), and all cluster hosts are assigned the same MAC address. Incoming packets are thereby received by all cluster hosts and passed up to the Network Load Balancing driver for filtering. To insure uniqueness, the MAC address is derived from the cluster's primary IP address entered in the Network Load Balancing Properties dialog box. entry and then reloading the adapter's driver; the operating system does not have to be restarted.

Network Load Balancing's unicast mode has the side effect of disabling communication between cluster hosts using the cluster adapters. Since outgoing packets for another cluster host are sent to the same MAC address as the sender, these packets are looped back within the sender by the network stack and never reach the wire. This limitation can be avoided by adding a second network adapter card to each cluster host. In this configuration, Network Load Balancing is bound to the network adapter on the subnet that receives incoming client requests, and the other adapter is typically placed on a separate, local subnet for communication between cluster hosts and with back-end file and database servers. Network Load Balancing only uses the cluster adapter for its heartbeat and remote control traffic.

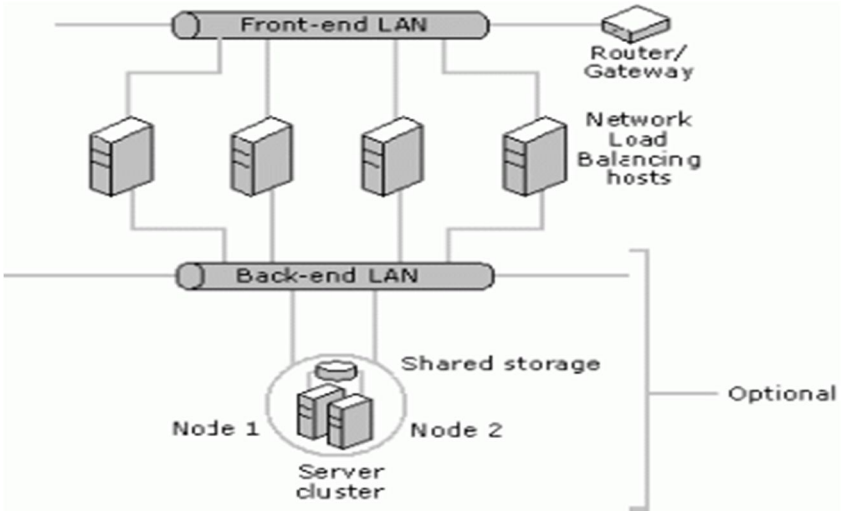


Figure 1.1: A four-host cluster works as a single virtual server to handle network traffic. Each host runs its own copy of the server with Network Load Balancing distributing the work among the four hosts.

III. NETWORK LOAD BALANCING ARCHITECTURE

To maximize throughput and high availability, Network Load Balancing uses a fully distributed software architecture. An identical copy of the Network Load Balancing driver runs in parallel on each cluster host. The drivers arrange for all cluster hosts on a single subnet to concurrently detect incoming network traffic for the cluster's primary IP address (and for additional IP addresses on multihomed hosts). On each cluster host, the driver acts as a filter between the network adapter's driver and the TCP/IP stack, allowing a portion of the incoming network traffic to be received by the host. By this means incoming client requests are partitioned and load-balanced among the cluster hosts.Network Load Balancing runs as a network driver logically situated beneath higher-level application protocols, such as HTTP and FTP. Figure 2 below shows the implementation of Network Load Balancing as an intermediate driver in the Windows 2000 network stack.

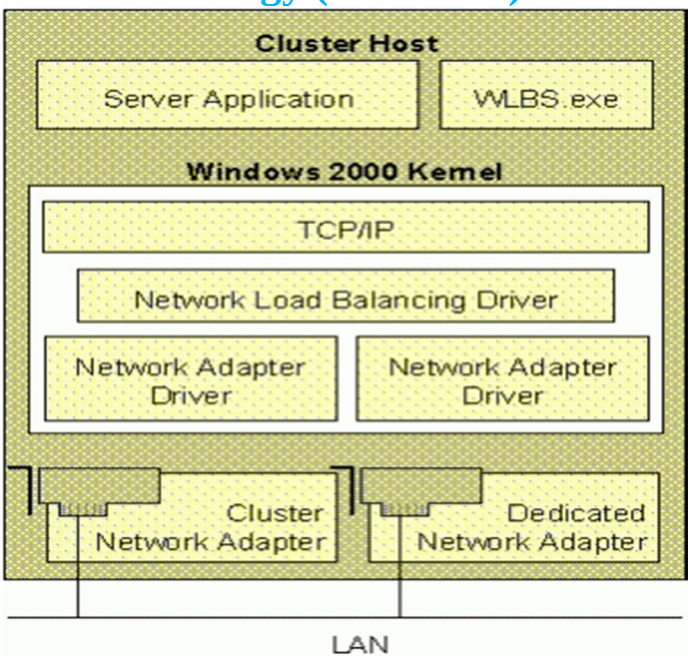


Figure 3.1: Network Load Balancing runs as an intermediate driver between the TCP/IP protocol and network adapter drivers within the Windows 2000 protocol stack Note that although two network adapters are shown, only one adapter is needed to use Network Load Balancing**.**

This architecture maximizes throughput by using the broadcast subnet to deliver incoming network traffic to all cluster hosts and by eliminating the need to route incoming packets to individual cluster hosts. Since filtering unwanted packets is faster than routing packets (which involves receiving, examining, rewriting, and resending), Network Load Balancing delivers higher network throughput than dispatcher-based solutions. As network and server speeds grow, its throughput also grows proportionally, thus eliminating any dependency on a particular hardware routing implementation. For example, Network Load Balancing has demonstrated 250 megabits per second (Mbps) throughput on Gigabit networks.Another key advantage to Network Load Balancing's fully distributed architecture is the enhanced availability resulting from (*N*-1)-way failover in a cluster with N hosts. In contrast, dispatcher-based solutions create an inherent single point of failure that must be eliminated using a redundant dispatcher that provides only 1-way failover. This offers a less robust failover solution than does a fully distributed architecture.Network Load Balancing's architecture takes advantage of the subnet's hub and/or switch architecture to simultaneously deliver incoming network traffic to all cluster hosts. However, this approach increases the burden on switches by occupying additional port bandwidth.

IV. NETWORK LOAD BALANCING PERFORMANCE

The performance impact of Network Load Balancing can be measured in four key areas:

A. *CPU overhead* on the cluster hosts, which is the CPU percentage required to analyze and filter network packets (lower is better).
B. *Response time* to clients, which increases with the non-overlapped portion of CPU overhead, called *latency* (lower is better).
C. *Throughput* to clients, which increases with additional client traffic that the cluster can handle prior to saturating the cluster hosts (higher is better).
D. *Switch occupancy* which increases with additional client traffic (lower is better) and must not adversely affect port bandwidth.In addition, Network Load Balancing's scalability determines how its performance improves as hosts are added to the cluster. Scalable performance requires that CPU overhead and latency not grow faster than the number of hosts.

V. LOAD BALANCING ALGORITHM

Network Load Balancing employs a fully distributed filtering algorithm to map incoming clients to the cluster hosts. This algorithm was chosen to enable cluster hosts to make a load-balancing decision independently and quickly for each incoming packet. It was optimized to deliver statistically even load balance for a large client population making numerous, relatively small requests, such as those typically made to Web servers. When the client population is small and/or the client connections produce widely varying loads on the server, Network Load Balancing's load balancing algorithm is less effective. However, the simplicity and speed of its algorithm allows it to deliver very high performance, including both high throughput and low response time, in a wide range of useful client/server applications.

Network Load Balancing load-balances incoming client requests by directing a selected percentage of new requests to each cluster host; the load percentage is set in the Network Load Balancing Properties dialog box for each port range to be load-balanced. The algorithm does not respond to changes in the load on each cluster host (such as the CPU load or memory usage). However, the mapping is modified when the cluster membership changes, and load percentages are renormalized accordingly.

When inspecting an arriving packet, all hosts simultaneously perform a statistical mapping to quickly determine which host should handle the packet. The mapping uses a randomization function that calculates a host priority based on the client's IP address, port, and other state information maintained to optimize load balance. The corresponding host forwards the packet up the network stack to TCP/IP, and the other cluster hosts discard it. The mapping does not vary unless the membership of cluster hosts changes, ensuring that a given client's IP address and port will always map to the same cluster host. However, the particular cluster host to which the client's IP address and port map cannot be predetermined since the randomization function takes into account the current and past cluster's membership to minimize remappings.

VI. INSTALLING NETWORK LOAD BALANCING

To use Network Load Balancing (NLB), a computer must have only TCP/IP on the adapter on which NLB is installed. Do not add any other protocols (for example, IPX) to this adapter. NLB can load balance any application or service that uses TCP/IP as its network protocol and is associated with a specific Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) port.

To install and configure NLB, you must use an account that is listed in the Administrators group on each host. If you are not using an account in the Administrators group as you install and configure each host, you will be prompted to provide the logon credentials for such an account. To set up an account that NLB Manager will use by default: in NLB Manager, expand the Options menu, and then click Credentials. We recommend that this account not be used for any other purpose.

A. To open the Add Features Wizard and install NLB

1) Click Start, click Administrative Tools, and then click Server Manager.
2) In the Features Summary area of the Server Manager main window, click Add Features.

—or—

B. In the Customize this server area of the Initial Configuration Tasks window, click Add Features.
1) In the Add Features Wizard, select the Network Load Balancing check box.
2) Click Install.
3) Alternatively, you can install NLB by typing the following command:

servermanagercmd.exe -install nlb

C. After NLB is installed, you can configure an NLB cluster by using Network Load Balancing (NLB) Manager.

The load-balancing algorithm assumes that client IP addresses and port numbers (when client affinity is not enabled) are statistically independent. This assumption can break down if a server-side firewall is used that proxies client addresses with one IP address and, at the same time, client affinity is enabled. In this case, all client requests will be handled by one cluster host and load balancing is defeated. However, if client affinity is not enabled, the distribution of client ports within the firewall usually provides good load balance.

In general, the quality of load balance is statistically determined by the number of clients making requests. This behavior is analogous to coin tosses where the two sides of the coin correspond to the number of cluster hosts (thus, in this analogy, two), and the number of tosses corresponds to the number of client requests. The load distribution improves as the number of client requests increases just as the fraction of coin tosses resulting in "heads" approaches 1/2 with an increasing number of tosses. As a rule of thumb, with client affinity set, there must be many more clients than cluster hosts to begin to observe even load balance.

As the statistical nature of the client population fluctuates, the evenness of load balance can be observed to vary slightly over time. It is important to note that achieving precisely identical load balance on each cluster host imposes a performance penalty (throughput and response time) due to the overhead required to measure and react to load changes. This performance penalty must be weighed against the benefit of maximizing the use of cluster resources (principally CPU and memory). In any case, excess cluster resources must be maintained to absorb the client load in case of failover. Network Load Balancing takes the approach of using a very simple but powerful load-balancing algorithm that delivers the highest possible performance and availability.

Network Load Balancing's client affinity settings are implemented by modifying the statistical mapping algorithm's input data. When client affinity is selected in the Network Load Balancing Properties dialog box, the client's port information is not used as part of the mapping. Hence, all requests from the same client always map to the same host within the cluster. Note that this constraint has no timeout value (as is often the case in dispatcher-based implementations) and persists until there is a change in cluster membership.

VII. HOW NETWORK LOAD BALANCING WORKS

Network Load Balancing scales the performance of a server-based program, such as a Web server, by distributing its client requests among multiple servers within the cluster. With Network Load Balancing, each incoming IP packet is received by each host, but only accepted by the intended recipient. The cluster hosts concurrently respond to different client requests, even multiple requests from the same client. For example, a Web browser may obtain the various images within a single Web page from different hosts in a load-balanced cluster. This speeds up processing and shortens the response time to clients.

Each Network Load Balancing host can specify the load percentage that it will handle, or the load can be equally distributed among all of the hosts. Using these load percentages, each Network Load Balancing server selects and handles a portion of the workload. Clients are statistically distributed among cluster hosts so that each server receives its percentage of incoming requests. This load balance dynamically changes when hosts enter or leave the cluster. In this version, the load balance does not change in response to varying server loads (such as CPU or memory usage).

Network Load Balancing cluster servers emit a heartbeat message to other hosts in the cluster, and listen for the heartbeat of other hosts. If a server in a cluster fails, the remaining hosts adjust and redistribute the workload while maintaining continuous service to their clients. Although existing connections to an offline host are lost, the Internet services nevertheless remain continuously available.

VIII. ADVANTAGES OF NETWORK LOAD BALANCING

Network Load Balancing is superior to other software solutions such as round robin DNS (RRDNS), which distributes workload among multiple servers but does not provide a mechanism for server availability. If a server within the host fails, RRDNS, unlike Network Load Balancing, will continue to send it work until a network administrator detects the failure and removes the server from the DNS address list. This results in service disruption for clients. Network Load Balancing also has advantages over other load

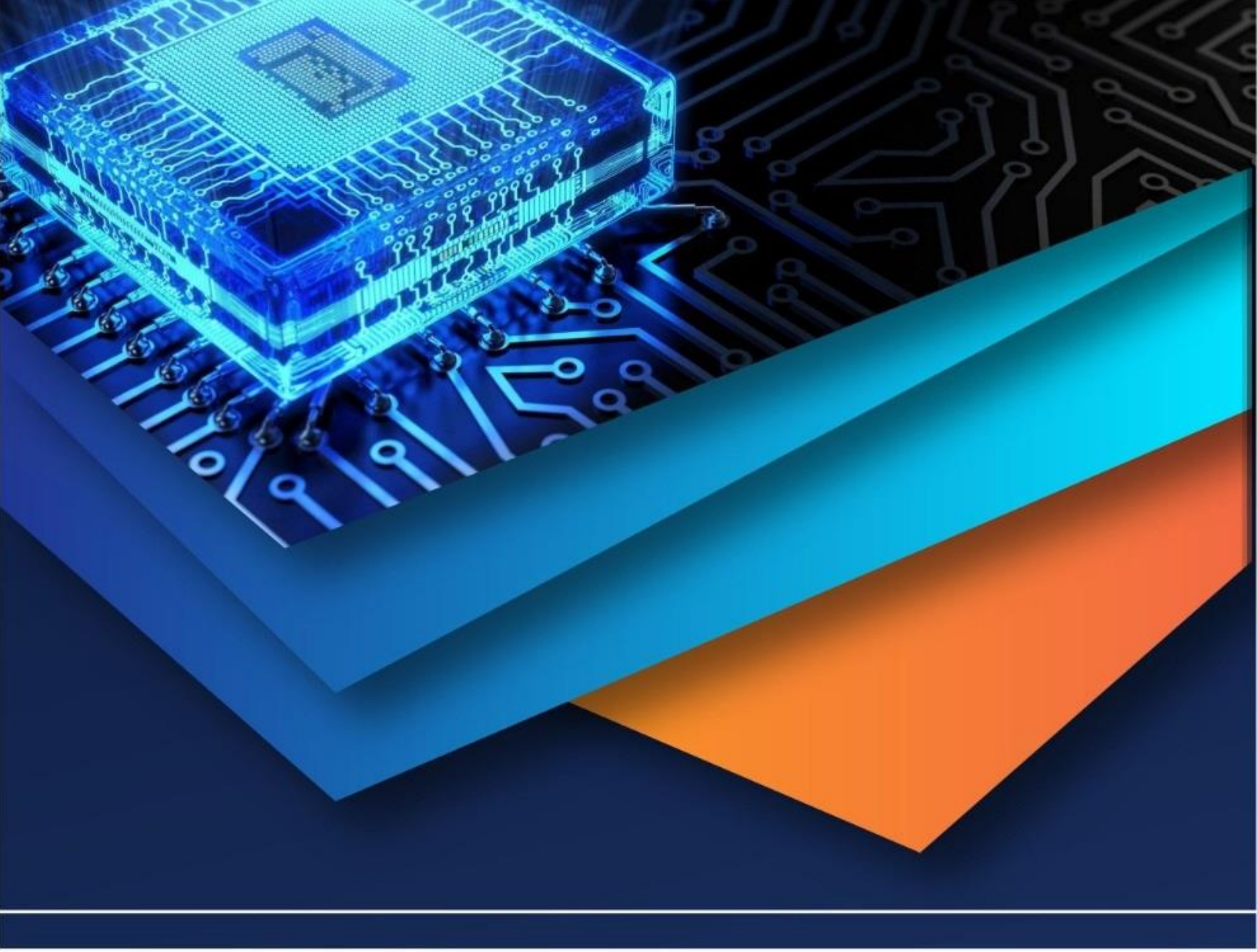
balancing solutions—both hardware- and software-based—that introduce single points of failure or performance bottlenecks by using a centralized dispatcher. Because Network Load Balancing has no proprietary hardware requirements, any industry-standard compatible computer can be used. This provides significant cost savings when compared to proprietary hardware load balancing solutions.

IX. CONCLUSION

Internet technology has been widely embraced, serving as the foundation for delivering enterprise-wide and frequently mission-critical applications such as Web, streaming media, and VPN servers. As an integral part of Windows 2000 Advanced Server and Datacenter Server, Network Load Balancing provides an ideal, cost-effective solution for enhancing the scalability and high availability of these applications in both Internet and intranet contexts. Network Load Balancing lets system administrators build clusters with up to 32 hosts among which it load-balances incoming client requests. Clients are unable to distinguish the cluster from a single server, and server programs are not aware that they are running in a cluster. Network Load Balancing uses a fully distributed algorithm to partition client workload among the hosts.

REFERENCES

[1] S. Chandra, C.S. Ellis, and A. Vahdat, “Differentiated Multimedia Web Services Using Quality Aware Transcoding,” Proc. INFO- COM 2000-19th Ann. Joint Conf. IEEE Computer and Comm. Soc., Mar. 2000.
[2] R. Keller, S. Choi, M. Dasen, D. Decasper, G. Fankhauser, and B. Platter, “An Active Router Architecture for Multicast Video Distribution,” Proc. IEEE INFOCOM, 2000.
[3] G. Welling, M. Ott, and S. Mathur, “A Cluster-Based Active Router Architecture,” IEEE Micro, vol. 21, no. 1, Jan./Feb. 2001.
[4] M. Ott, G. Welling, S. Mathur, D. Reininger, and R. Izmailov, “The Journey Active Network Model,” IEEE J. Selected Areas in Comm., vol. 19, no. 3, pp. 527-537, Mar. 2001.
[5] A. Fox, S. Gribble, E. Brewer, and E. Amir, “Adapting to Network and Client Variability via On-Demand Dynamic Distillation,” Proc. Seventh Int’l Conf. Architecture Support for Programming Language and Operating Systems (ASPLOS-VII), 1996.
[6] A. Fox, S.D. Gribble, and Y. Chawathe, “Adapting to Network and Client Variation Using Active Proxies: Lessons and Perspectives,” IEEE Personal Comm. on Adaptation, special issue, 1998.
[7] E. Amir, S. McCanne, and R. Katz, “An Active Service Framework and Its Application to Real-Time Multimedia Transcoding,” Proc. ACM SIGCOMM Symp., Sept. 1998.
[8] B.A. Shirazi, A.R. Hurson, and K.M. Kavi, Scheduling and Load Balancing in Parallel and Distributed Systems. CS Press, 1995.
[9] M. Satyanarayanan, “Scalable, Secure, and Highly Available Distributed File Access,” Computer, May 1990.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY