Load Balancing Solution and Evaluation of F5 Content Switch Equipment

Toqeer Ahmed

Master Thesis
Computer Engineering

2006

Nr: E3381D
Abstract

The Thesis focused on hardware based Load balancing solution of web traffic through a load balancer F5 content switch. In this project, the implemented scenario for distributing HTTP-traffic load is based on different CPU usages (processing speed) of multiple member servers. Two widely used load balancing algorithms Round Robin (RR) and Ratio model (weighted Round Robin) are implemented through F5 load balancer. For evaluating the performance of F5 content switch, some experimental tests has been taken on implemented scenarios using RR and Ratio model load balancing algorithms. The performance is examined in terms of throughput (bits/sec) and Response time of member servers in a load balancing pool. From these experiments we have observed that Ratio Model load balancing algorithm is most suitable in the environment of load balancing servers with different CPU usages as it allows assigning the weight according to CPU usage both in static and dynamic load balancing of servers.
Table of Contents

1.0 Introduction............................................................................................................................... 1
  1.1 Background........................................................................................................................... 1
  1.2 Objective ............................................................................................................................... 1
  1.3 Limitations ............................................................................................................................ 1
  1.4 Work Method ....................................................................................................................... 1
  1.5 Questions for investigation ................................................................................................. 3
  1.6 Disposition ............................................................................................................................ 3

2.0 Problem Formulation ................................................................................................................ 4

3.0 System Overview ...................................................................................................................... 5

4.0 Generation of Web Requests (Literature Overview) .............................................................. 7

5.0 Load Balancing Algorithm .................................................................................................... 11
  5.1 Server Load Balancing ......................................................................................................... 11
  5.2 Why Load Balance Servers? ............................................................................................... 12
  5.3 Algorithms for Server Load Balancing ............................................................................ 12
    5.3.1 Round Robin Load Balancing Algorithm .................................................................. 12
    5.3.2 Ratio Model Load Balancing Algorithm ............................................................... 13
  5.4 Design and Implementation scenario for Load balancing Solution ................................ 14
    5.4.1 Based on Ratio Model Load Balancing Algorithm .............................................. 14
    5.4.2 Work Flow Mechanism of Ratio model ................................................................. 15
  5.5 Design and Implementation scenario for Load Balancing Solution ................................ 17
    5.5.1 Based on Round Robin Load Balancing Algorithm .............................................. 17
    5.1.2 Work Flow Mechanism ............................................................................................. 18
  5.6 Traffic Flow in Layer 7 ....................................................................................................... 19
  5.7 Queuing Model ................................................................................................................... 21
    5.7.1 Queuing Systems ......................................................................................................... 21
    5.7.2 Queuing Techniques ..................................................................................................... 23
    5.7.3 Queuing Networks ....................................................................................................... 24
  5.8 Load Balancing Device ....................................................................................................... 24
Acknowledgments

First of all I would like to thank GOD, the Creator, for giving me the ability to complete this thesis. I am very grateful for all the valuable insights, suggestions, comments, and contributions from Dr. Ernst Nordstrom, HOD Mr. Kari Bjorn and Mr. Janne Salonen for my master thesis work. They supported me during whole project and had patience when I was baffled about research issues in my project. Here I have very special thanks to Dr. Ernst Nordstrom and Mr. Kari Bjorn for all of his valuable feedback and support on technical report writing of thesis. I am very thankful to HOD Mr. Kari Bjorn who gave me the opportunity to do this project from Dept of Information Technology and Communication Helsinki stadia and as well gave me the chance to get myself familiar with very advanced and expensive hardware equipment used in project. I am really thankful for all the technical support from F5 Technical Solution support, load balancing web forums and lab assistant Mr. Tapio Wikstrom.

I would really like to thank the entire faculty of Dalarna University and Helsinki Institute of Technology, Stadia for their invaluable assistance in all the courses which they taught me during Master programme. I would like to thank my program coordinators Dr. Pascal Reybrend and Hasan Fleyeh who gave me the opportunity to pursue my Master studies in Sweden. At the end I would like to pay Special thanks to my family and friends for their kind support in prayers.
1.0 Introduction

1.1 Background
This is a Master thesis work and is compulsory part of International Master of Science in Computer Engineering Degree at Hogskolan Dalarna (Dalarna University), Sweden. The project was carried out under the exchange programme of ITCOM that is supported and funded by Department of Information Technology and Communication, Helsinki Stadia Polytechnic, Finland.

1.2 Objective
The objective of this project is to balance the load of web requests among multiple servers by using F5 content switch and evaluate the performance of F5 content switch for different implemented load balancing algorithms.

1.3 Limitations
Load balancing can be implemented through software or hardware. In software-based load balancing special software is installed on multiple servers while in hardware based load balancing, specialized switch or router is used, equipped with software to balance the load. In this project hardware based load balancing is implemented using F5 content switch. F5 load balancer works as a Local Traffic Manager (LTM) in a network that has the main task to manage the HTTP traffic. LTM of F5 can have multiple add on modules such as Intelligent compression module, fast cache module, SSL acceleration module, Layer 7 rate shaping module and Advance client Authentication module that are used to meet different requirements for network traffic management. In this project we are using LTM module of F5 content switch which is sufficient for load balancing among multiple servers (web servers) without any add on module.

1.4 Work Method
The first part in this project is to build a network for implementing Load balancing solution via F5 content switch. The purpose for building this network is to distribute the client’s web request (HTTP requests) among multiple servers. In our Network setup, we used two server machines that are connected through a network switch. Both servers have one gigabit/sec Ethernet card and Windows Server 2003 enterprise edition is installed on both servers as an operating
system. Cisco 3560 Switch is used to connect multiple servers (two or more than two) and Load balancing device (F5 content switch) in a network.

In the second step, we have configured the load balancing techniques on F5 switch for a web service that is actually hosted on two physical servers. These two servers having web server (Apache), a simple static web site is hosted separately on each. Using GUI of F5 switch, we configured a virtual server (have unique IP) and the actual IP’s of both the physical web servers (where a web site running) are assigned to the virtual server on F5. Now, the web site is mutually shared with the same virtual server’s IP address and virtual server forwards all the client’s requests to the load balancing pool members (Physical web servers). Using GUI of F5, we implement different Load balancing Algorithms (rules) for distributing the requests among two physical web servers in a load balancing pool (Pool members). Two load balancing algorithms are implemented that are Round Robin and Ratio Model (Weighted Round Robin).

In the last step of this project, we have evaluated the performance of F5 content switch using above mentioned algorithms in implemented scenario. The scenario we have implemented is based on different processing speed of pool members (servers having different processor speed) in a load balancing pool. The load balancing through F5 content switch affects the performance of each server individually in a load balancing pool. Following the test criteria for implemented load balancing scenario, we analyze the performance of each server with different load balancing algorithms that are set on F5 content switch by generating different bursts of HTTP traffic load. For generating HTTP traffic burst we used “HTTP Traffic Gen” [27] software from the workstation. The traffic generation through the software is based on deterministic approach, where the inter-arrival time between two successive requests (Data Packet) remains constant. We have evaluated the performance of F5 content switch based on throughput and response time parameter of pool members (web servers). The throughput of each server is measured by using the performance utility of F5 content switch and response time is measured by using a software named “Application Manager 7” [30].

The implemented scenario was based on different processing speed of load balancing pool members. In our network setup for traffic load distribution, all the servers were of same processing speed. So we loaded one server with extra utilities (any type of services) to make the CPU usage different from the other server in same Load Balancing pool.
1.5 Questions for investigation

We examine the performance of load balancing solution through F5 content switch by applying load balancing algorithms in our implemented scenario. We compared the performance with two load balancing algorithms, Round Robin and Ratio Model in our implemented scenario on the basis of throughput and response time parameters of pool members.

1.6 Disposition

This project is organized into six main parts. In chapter two, the problem formulation is explained. This chapter describes the basic thought of the project work and discusses the theme of the problem, which e-commerce companies are suffering in the progress of a web business. Chapter three describes the system overview of the project. Chapter four is based on literature review on web generation Chapter five describes the architecture of implemented scenario and two load balancing algorithms Round Robin and Ratio Model are also discussed in detail in this chapter. It also describes the load balancing device F5 content switch and its dealing with Layer 7 traffic. In chapter six some experiments test has been taken to validate the performance of the load balancing algorithms on proposed scenario. Chapter seven explains the results and analysis on the basis of test experiments. While chapter eight are on conclusion and recommendation for future work.
2.0 Problem Formulation

The problem in Web based services, where a large number of client’s requests may cause delay in response due to high load on a web server. This delay in web service has very high impact in the most business critical applications, where a short delay in service may cause a huge loss in business. As e-commerce based companies are responsible for providing high availability, high performance and secure solution of web business to all services for their clients.

To overcome this difficulty, multiple servers can work together to serve the larger number of clients requests in timely manner, where the traffic load is distributed among these servers. There are different Load Balancing Algorithms that can be used for distributing the traffic load. Load Balancing Algorithms are predefined techniques used for load distributions among multiple servers in different network environment. Some load balancing device that can be used to distribute the traffic load are F5 and Cisco switches, these pieces of hardware work for load balancing using predefined load balancing algorithms.

Department of Information Technology and Communication Helsinki Stadia polytechnic is suffering same kind of problem about load balancing on servers. For this issue department purchased load balancers of F5 networks. As the most common hardware of load balancing are available from F5 Networks and Cisco Systems and we have been offered F5 content switches for its implementation as a master thesis work.

We also investigate the problem that why web servers are overloaded with larger requests even there is load balancing between multiple servers and how it can be solved. We implemented a load balancing scenario that is based on different processing speed of servers. The idea behind this scenario was that several e-commerce based companies that are providing web services to clients, host the same web site on multiple servers for sharing the load and it might possible that CPU utilization (processing) of these servers are different from each other at particular time. Our implemented scenario is used for different load balancing algorithms Round Robin and Ratio Model to determine the best one in particular network environment (servers with different processing speeds).
3.0 System Overview

This section defines the physical environment of Lab where this project is implemented, its general Architecture is described in figure 3.1 below.

![Network Architecture](image)

*Figure 3.1
Network Architecture*

Three work station pc used as clients, F5 content switch BIG-IP v9 1500 series used as a load balancer which is connected with two Servers platform of Intel® Xeon™ through a 3560 series Cisco switch. All components are physically connected on LAN (Local area network).

HTTP traffic load are shared on two servers. Servers platform are Intel® Xeon™ 2.99 GHZ Processor with 1.0 GB memory and 80 Gigabyte disk. Both servers have one gigabit/sec Ethernet card. Windows server 2003 enterprise edition is installed on both servers as an operating system. Apache web server is configured on both servers. Static HTML site with different size (1KB, 10KB, 100KB, 500KB, and 1024KB) is used as testing website which is hosted on Apache web server. Web site mutually shared with same local IP address 192.168.1.182 from both web servers. F5 content switch 1500 series v9 is used as a load balancing device which is connected with servers through 3560 Cisco switch. F5 Content switch received the HTTP requests for testing website from clients and intelligently distribute their requests to servers according to the implemented load balancing algorithm. HTTP requests...
generated through these work stations with the help of a traffic generator tool named “Http Traffic Gen 1.0” or by simple Internet Explorer.

Figure 3.1 presents a typical client-server scenario; a client request goes to the destination IP address specified in the header of the request. For testing web site with a large amount of incoming HTTP traffic, the destination server can quickly become overloaded as it tries to service a large number of requests. To solve this problem, the BIG-IP® local traffic management (LTM) system of F5 content switch distributes client requests to multiple servers instead of to the specified destination IP address only.

Numbers of HTTP request which are processed through F5 content switch depends on the model of the hardware. In project, F5 1500 series v9 content switch with processing speed Single 2.5 GHz Celeron and RAM capacity 768 MB is used. Two servers of Intel® Xeon™ are used and that can be directly connected with load balancer. But in project these servers are connected with load balancer through a Cisco 3560 switch. The only function of using this Cisco switch is that, if we increase the number of servers, there are not enough network interfaces are available on F5 hardware.

Cisco 3560 switch is a device that channels incoming data from any of multiple input ports to the specific output port that will take the data toward its intended destination. On an Ethernet local area network (LAN), a switch determines from the physical device (Media Access Control or MAC) address in each incoming message frame which output port to forward it to and out of. In a wide area packet-switched network such as the Internet, a switch determines from the IP address in each packet which output port to use for the next part of its trip to the intended destination [8].
4.0 Generation of Web Requests (Literature Overview)

In this chapter we will focus on the generation of web request and method used to generate the traffic in detail. Web traffic is the stream of data that is sent and received by internet users to a web site. It is a large portion of Internet traffic that consists of packets, connection or requests. Web traffic is generating by number of processes, these processes consists of sequence of arrival. Web traffic, mostly based on two main approaches that are deterministic and stochastic.

In this project we have used “Http Traffic Gen 1.0” [27] as traffic generator software that uses deterministic traffic generation approach. HTTP Traffic Gen is a tool that generates HTTP traffic to the specified URL; need to specify HTTP request parameter packet count and interval in milliseconds. Both approaches are discussed below.

A deterministic traffic generator is derived from real simulation traces or written from scratch by IP designers. Deterministic traffic generator can generate accurate transactions in time, burst size and idle time that match the behaviour of an IP. In deterministic traffic generation model the traffic is generated by constant Bit Rate (CBR) the interval between packet generations remains constant and the size of packets may be varied in this approach depending on generation format and the measure of interest. While, a stochastic process is a random process evolving with time. More precisely, a stochastic process is a collection of random variable $X_t$ indexed by time. The traffic generator which uses stochastic approach are useful either when the IP is not fully available or when the behaviour is likely to change from one execution to the other [10] [17] [18] [32].

For a repairable system, the time of operation is not continuous. In other words, its life cycle can be described by a sequence of up and down states. The system operates until it fails, then it is repaired and returned to its original operating state. It will fail again after some random time of operation, get repaired again, and this process of failure and repair will repeat. This is called a renewal process and is defined as a sequence of independent and non-negative random variables [28] [29].

The name renewal process is derived by the fact that every time there is repetition of process "starts all over again", it renews itself. "A renewal process counts events. Suppose $X_n$ is the
duration between the \((n-1)\) st and the \(n\)th event. Thus \((X_n, n \geq 1)\) is sequence of nonnegative random variable defined on probability space \((\Omega, \mathcal{F}, P)\). If \(X_n = 0\) then the \((n-1)\) st and the \(n\)th event occur simultaneously. We assume that \((X_n, n \geq 1)\) is a sequence of independent and identically distributed (iid) random variables with the common distribution \(F\), and to avoid trivial details we suppose that \(F(0) = P(X_n = 0) < 1\). Let \(S_0, S_1, S_2, \ldots\) be the random variable defined by” [19, page 14] [20]

Where

\(F\) is common distribution and

\((X_n, n \geq 1)\) is the nonnegative random variable defined on probability space \((\Omega, \mathcal{F}, P)\).

\[
S_0 = X_0 = 0 \text{ and } S_{n+1} = S_n + X_{n+1}, \quad n \geq 0 \quad [19, \text{ page 14}]
\]

A renewal process has identical and independent inter-arrival times. When the inter-arrival times are exponentially distributed we have a Poisson renewal process, and when the inter-arrival times are Weibull distributed we have a Weibull renewal process. Both processes are particularly discussed below.

Poisson process is commonly used process in telephony network industry to find the arrival and call blocking probability. Often the arrival process of customers can be described by a Poisson process. In tele traffic theory the “customers” may be calls or packets. As, Poisson process are considered as discrete intervals e.g. calls or packets shown in figure 4.1

\[
\begin{align*}
\text{Figure 4.1[3]} \\
\text{Mathematically the Poisson process is described as counter process } N_t \text{ or } N(t). \text{ The counter tells the number of arrivals that have occurred in the interval } (0, t) \text{ or, more generally, in the interval } (t_1, t_2). \ [3] \\
& \quad \bullet \quad N(t) = \text{number of arrivals in the interval } (0, t) \\
& \quad \bullet \quad N(t_1, t_2) = \text{number of arrival in the interval } (t_1, t_2)
\end{align*}
\]
Poisson process can be expressed in three different ways. If there is only one arrival keeps occurring with the probability $\lambda \, dt$ in the extremely small time $dt$ the process of such arrival is called Poisson process when there is no change in $\lambda$ over time. Secondly, Poisson process can be consider as a renewal process whose inter-arrival times $\{tn\}$ are exponentially distributed with rate parameter $\lambda$: $P(\{tn \leq t\} = 1 - \exp(-\lambda t)$. Such probability density function is called an exponential distribution. Thirdly, in Poisson process, number of arrivals occurs in finite intervals of length $t$. These numbers of arrivals are independent in non-overlapping intervals. Such probability function is known as Poisson distribution.

Poisson process has several interesting analytical properties. First, superposition of two independent Poisson process with rate $\lambda_1$ and $\lambda_2$ is a Poisson process with sum of component rate $\lambda = \lambda_1 + \lambda_2$. Second the number of arrival in the $n$ in the interval $(0,t)$ are independently and uniformly distributed. Third, the random selection of arrivals in Poisson process with rate parameter $\lambda$ and probability $p$ is independent of others. The resulting process is called a Poisson process with rate parameter $p \lambda$. Fourth, If a Poisson process with intensity $\lambda$ is randomly split into two sub processes with probabilities $p_1$ and $p_2$, where $p_1 + p_2 = 1$, then the resulting processes are independent Poisson processes with intensities $p_1 \lambda$ and $p_2 \lambda$ [3][21][32].

A stochastic process characterize into three group of counting process that are related to the standard Weibull models are as follows.

- Weibull intensity models
- Weibull renewal process models
- Power law-Weibull renewal process

In this section Weibull renewal process models are particularly under discussion. Weibull is a stochastic renewal processes that uses number of events to occur over a time interval and is defined in sequence of independent and non-negative random variable. Weibull have been used many different applications for solving variety of problems from many different disciplines. Weibull renewal process is characterized into three main processes, as follows
When the occurrence of events with the inter event time $X_i, i=1, 2...$ are independent and identical distributed according to the two or three parameter Weibull distribution is known as Ordinary renewal processes. While, In modified renewal process the distribution function for the time to first event $f_1(x)$ is different from that for the subsequent inter event times which are identical and independent random variable with distribution function $f(x)$ either $f_1(x)$ and $f(x)$ are two or three parameter Weibull distributions. Note that when $f_1(x) = f(x)$ the model reduces to the ordinary renewal process model. In Alternating renewal process the odd numbered inter event times are a sequence of independent and identically distribution random variable with a distribution function $f_1(x)$ the even numbered inter event times are another sequence of independent and identically distributed random variable with a distribution function $f_2(x)$, either $f_1(x)$ and/or $f_2(x)$ is the two or three parameter Weibull distribution.

Weibull distribution is general distribution that is useful for describing failure time data. Its distribution is a continuous distribution that was published by a Swedish professor Waloddi Weibull in 1939. This theory was used to explain the well known but unexplained facts that the relative strength of specimen decreases with the increase dimensions and that it's bending strength is larger than its tensile strength. This theory was based on the assumption that the strength is a stochastic quantity, which has to be specified by a distribution function with one or more parameters [24] [25].
5.0 Load Balancing Algorithm

Chapter five describes in detail about the architecture of implemented scenario and load balancing algorithms which are implemented through Load balancer. It also describes the load balancing device F5 content switch, its dealing with Layer 7 traffic and traffic generation method on server’s as well queuing system.

5.1 Server Load Balancing

“Server load balancing is the process of distributing service requests across a group of servers”. [3] Load balancing is especially important for networks where it is difficult to predict the number of HTTP requests that will be issued to a specific web server. Busy Web sites typically employ two or more Web servers in a load balancing method. If one server starts to get down, requests are forwarded to another server with more capacity. Server Load Balancing can apply all kind of application servers for example file servers, web servers, database servers, mail servers, print servers etc. Server load balancing become very important in networks because it provide scalability, reliability and high performance of applications over all networks.

The two major categories of load balancing implementations are:

“Software-based load balancing” Software-based load balancing consists of special software that is installed on the servers in a load-balanced pool. Servers which are involved in load balancing known as load balancing pool. The software dispatches or accepts requests from the client to the servers, based on different algorithms. The algorithms can be a simple round-robin algorithm or a much more complicated algorithm that considers server affinity. For example, Microsoft Network Load Balancing is a load balancing software for Web farms, and Microsoft Component Load Balancing is a load balancing software for application farms” [5].

“Hardware-based load balancing” Hardware-based load balancing consists of a specialized switch or router with software to give it load balancing functionality. This solution integrates switching and load balancing into a single device, which reduces the amount of extra hardware that is required to implement load balancing. Combining the two functions, however, also makes the device more difficult to troubleshoot”. [5] Hardware based balancing is implemented in this project.
5.2 Why Load Balance Servers?
The main contribution over the past few years has been the sudden increase in internet use and the resulting need for better, faster and more reliable web sites while early users of the internet came to often expect a slow and frustrating experience when browsing the web sites. Web hosting customers are expecting their sites to be up and running 24x7 with quick response. They also expect Scalable servers that transparently meet growing user demand. Deploying ever-larger web servers to meet the customer expectation is an expensive, short-term approach, doomed to eventual failure [1] [4]. To provide scalability, reliability, operability to clients, intelligent load balancing among servers is needed.

5.3 Algorithms for Server Load Balancing
The highest performance of a server is achieved when the processing power of server is used intelligently. To achieve the maximum performance of servers, several load balancing algorithms can be used. The load-balancing algorithm known as the fastest keeps track of the length of time that a machine takes to fulfil a request and refers HTTP traffic to the server with the fastest response rate. The BIG-IP load balancer device offers a wide variety of load balancing algorithms, which are especially useful for load balancing security devices such as Firewalls, VPNs, or IDS systems. Advanced algorithms, such as Predictive, Observed, and Dynamic Ratio, take into account one or more dynamic factors, such as current connection count. For load balancing devices that significantly differ in processing speed, memory, and connection types, these algorithms provide a better and more uniform utilization of resources [3] [6].

The most common load balancing algorithms are Round Robin, Least connection Node, Least Connection Members. Because of their different techniques used for load balancing, not a single algorithm can be suitable for all network environments. For this project two load balancing algorithms Ratio Model and Round Robin are implemented through F5 load balancer. Their descriptions are as following.

5.3.1 Round Robin Load Balancing Algorithm
Round-Robin Load Balancing Algorithm is used as a default content switching algorithm. Round-robin load balancing treats all servers as equal, similar to rotary DNS but without propagation delays or caching issues.
A round-robin algorithm distributes the load equally to each server, regardless of the current number of connections or the response time. Round-robin is suitable when the servers in the load balancing pool have equal processing capabilities; otherwise, some servers may receive more requests than they can process while others are using only part of their resources. Round Robin algorithm is not successful where each server will receive the same number of connections over time irrelevant of how fast it is able to process and deal with them. [1] [5]

If there are \( n \) processes in the ready queue and the time slice is \( q \), then each process ideally would get \( \frac{1}{n} \) of the CPU time in chunks of \( q \) time units, and each process would wait no longer than \( nq \) time units until its next quantum. A more realistic formula would be \( \frac{n(q + o)}{n(q + o)} \), where \( o \) is the context switch overhead. So, for practical purposes, it is desirable that the context switch be negligible compared to the time slice. The performance of the Round-Robin algorithm depends heavily on the size of the quantum. If the quantum is very large, the Round-Robin algorithm is similar to the First-Come, First-Served algorithm. If the quantum is very small, the Round-Robin approach is called processor sharing". Another important parameter in processor scheduling is the Mean Completion Time, which is calculated as

\[
\text{Mean Completion Time} = \frac{\sum \text{Time for Jobs to Complete}}{\text{Number of Completed Jobs}}
\]  

[11]

5.3.2 Ratio Model Load Balancing Algorithm

Ratio Model load balancing Algorithm is advanced form of Round Robin load Balancing algorithm and can consider as dynamic load balancing algorithm. This is quite similar with weighted Round Robin load balancing algorithm. Weight-based load balancing improves on the round-robin algorithm by taking into account a pre-assigned weight for each server. Through using Load balancer configuration utility we can assign each server in the load balancing pool a numerical weight e.g. 1 and 100. Each server can be assigned a weight, an integer value that indicates the processing capacity. Servers with higher weights receive new connections first than those with fewer weights, and servers with higher weights get more connections than those with fewer weights and servers with equal weights get equal connections. For example, the real servers, A, B and C, have the weights, 4, 3, 2 respectively, a good scheduling sequence will be
AABABCABC in a scheduling period (mod sum (Wi)). In the implementation of the weighted round-robin scheduling, a scheduling sequence will be generated according to the server weights after the rules of Virtual Server are modified. The network connections are directed to the different real servers based on the scheduling sequence in a round-robin manner. [10] [16]

HTTP Requests are directed to the different servers according to weight assigned; Virtual server distributes each request sequentially around the pool of real servers but gives more jobs to servers with greater capacity. Weights that are assigned to servers are according to their different processing capacities how statically and dynamically HTTP request can serve through servers with their processing capacity. Faster machine processed with high weights than slower processing speed machine.

5.4 Design and Implementation scenario for Load balancing Solution

5.4.1 Based on Ratio Model Load Balancing Algorithm

An extensive number of dynamic load balancing algorithms for HTTP traffic have been designed which are able to distribute traffic load according to network requirement. [4] Ratio Model load balancing algorithm is one of them. In order to provide the quick response to clients through two web servers and to achieve the optimal load balancing between two web servers on the bases of different CPU usage using F5 content switch as a load balancing device, a simple scenario is implemented which is shown in figure 5.1 below. Ratio model Load balancing algorithm is implemented as load balancing method. HTTP traffic load distribution between two servers is on the bases of CPU utilization of servers.

HTTP traffic load are shared on two servers. Servers platform are Intel® Xeon™ 2.99 GHz Processor with 1.0 GB memory and 80 Gigabyte disk. Apache web server is configured on both servers. Five static HTML sites with size of 1KB, 10 KB, 100KB, 500KB and 1MB are used as testing websites which are hosted on Apache web server. Web site mutually shared with same local IP address 192.168.1.182 from both web servers. F5 content switch 1500 series v9 is used as a load balancing device. Content switch received the HTTP requests for testing website from clients and intelligently distribute their requests to servers according to Ratio model load balancing algorithm. For this scenario three work stations pc used as clients. HTTP requests
generated through these work stations with the help of a traffic generator tool or by simple internet explorer.

**Figure 5.1:**
Network Architecture for Ratio Model Load Balancing Algorithm

### 5.4.2 Work Flow Mechanism of Ratio model

This implemented scenario is designed on the basis of CPU utilization of servers and traffic load statically distributing between servers. As above figure 5.1 shows that there are two servers named as server A and server B. Hardware specifications of both servers are similar but their CPU Usage are different from each other. Server B has higher CPU Usage because of several other applications are running on it. So their response time to client’s requests is obviously different from each others. Response time of Sever A will faster than server B. According to Ratio model load balancing algorithm here need to specify a ratio of traffic load on both servers. So traffic load is distributed according with two ratio one (2:1). Typical or specific weight (ratio) assigning to Ratio Model load balancing Algorithm depends on several parameters e.g.
processing speed of servers, Length of IP queue. In our experiments we selected these ratio or weights for traffic distribution specifically according to CPU of servers.

For example if there are three HTTP requests from all clients to servers, load balancing device distribute the load according to specified ratio. So first two requests will forward on server A (as its CPU Usage is less) while remaining one request will forward on server B (As its CPU Usage is higher). Following is the work flow diagram of implemented Ratio Model load balancing algorithm, which describes in detail how HTTP request are processed.

![Flow chart diagram for Ratio Model Load Balancing Algorithm](image)

**Figure 5.2:**

*Flow chart diagram for Ratio Model Load Balancing Algorithm*

Above flow chart in figure 5.2 describes in detail how HTTP traffic processed according to Ratio Model load balancing algorithm. At initial stage once the condition become true request forward to desired server for processing according to specific ratio. Administrator sets the desire ratio of traffic according to CPU usage of each server. This process continues at unlimited number of requests. If number of incoming request is one it will check the previous two requests, if the previous two request are processed at server A then this request forward to server B other wise to server A.
5.5 Design and Implementation scenario for Load Balancing Solution

5.5.1 Based on Round Robin Load Balancing Algorithm

Round Robin is very simple load distribution metric and is very effective for sharing load evenly among object servers. Each web server has its own file system, IP address, hostname and its own interface to world. For example if there is assigned four IP address to any web site so according to Round Robin load balancing policy that incoming traffic will go (fairly evenly) to all four address over time. Some implementations do this automatically, circulating through all available IP addresses for a given host name. [1][2] To check the performance of proposed scenario based on CPU utilization using Ratio Model load balancing algorithm is compared with Round Robin load balancing algorithm.

General architecture of implemented scenario is presented in figure 5.3 below for Round Robin Load balancing algorithm is exactly same as it used for Ratio Model Load Balancing algorithm.

![Figure 5.3](image_url)

**Figure 5.3**

Network Architecture for Round Robin Load Balancing Algorithm
5.1.2 Work Flow Mechanism

Round Robin load balancing algorithm is implemented as load balancing algorithm for proposed scenario. Round Robin is the simplest distribution metric and is often used as a default for content switching configuration. In which requests are assigned according to the numbers of the servers and load is equally distributed among servers. A Host virtual server named Test1 is configured with load balancing pool. Two servers or nodes named server A and server B as shown in figure 5.3 above are the members of this load balancing pool. HTTP type of profile and ICMP type of health monitors are used to check the status of node either it is up or down. Below is the work flow mechanism of this scenario.

For example if there are three HTTP requests from all clients to servers, load balancing device distributes the load equally according to numbers of servers. As its name suggests incoming HTTP requests will be forward on a round robin bases, the first request forwarding on server A while second on server B then third request processed again on server A.
5.6 Traffic Flow in Layer 7

Most server load balancing topologies can be broadly grouped into Layer 2, 3, 4 and 7. Above scenario is implemented through layer 7 content switch, following is the traffic flow mechanism in layer 7.

- “Terminate the TCP from the client by completing the three ways TCP Handshake.
- Buffer the incoming packets containing the user data. This might not necessarily be as simple as buffering the first frame as, within HTTP for example, the information may not be contained in the first packet.
- Parse these packets for the required Layer 7 information, such as URL HTTP headers, FTP control commands, or DNS requests.
- Make a load balancing decision based on the information found in the user request.
- Open a new TCP connection from the content switch to the appropriate user request.
- Forward the buffer request packets on to the real server.
- For all subsequent packets in the session the content switch must splice
- These two separate TCP connections together by altering information such as TCP ports and sequence number”. [1 page 113]

Following figure 5.5 shows traffic flow mechanism in layer 7.

![Traffic Flow in Layer 7](image-url)
The process of dealing with layer 7 traffic inspections, be it for Server load balancing, firewall load balancing, web cache redirection or any other application is inherently different from layer 4. Layer 7 protocols such as HTTP operate so that certain information is only available at certain time during the session. Following figure shows a simplified HTTP session, demonstrated any useful layer 7 information which may be needed for a server load balancing decision do not appear until at least the fourth packet of session. The consequence of this behaviour has major impact on the performance of many content switches. If the decision making information is not available until the fourth packet of a TCP session, the content switch must be able to store and buffer these packets until the relevant information arrives. [1]

Figure 5.6
Layer 7 inspection in content switch [1]
5.7 Queuing Model

A queue can simply be defined as a waiting line of objects that wait for a service. These objects can be for instances customers at a store waiting for a cashier, a patient at a medical clinic, and packets of data in a networks or requests from terminals to a server.

The occurrence of queues in our daily living is very common. This is because when one has a set of objects (people, packets...) that it wants to be served at the same times. It is not possible in most cases to have servers for all these object to server them all at once. Thus when the number of objects that need to be served is more than the number of available servers at the time of requesting a service, we must have a queue or queues. To help to visualize the queuing model, a pictorial representation of servers and queues are introduced in below figure 5.7. The server is resented in circle and their associated waiting lines as rectangles. The arrow indicates the flow of incoming and outgoing transaction.

![Queuing model](image)

Pictorially, the representation of an analytical model might be as that shown in figure above. Here the transactions enter the system, possibly queue for server to become available, obtain service and then exit the system [12].

5.7.1 Queuing Systems

We characterize queuing systems by:

- Arrival process \( A(t) = P(\text{inter-arrival time} \leq t) \)
- Service process \( B(x) = P(\text{service time} \leq x) \)
- Storage capacity available for waiting customers
- The number of servers/customers available
- The different kinds of arriving customers (big jobs, small jobs, . . . )
- Queuing discipline used: FCFS, LCFS, priority,
• Defections, balking, bribing, . . . [31]

**Queuing systems notation (Kendall’s Notation)**

The Kendall notation describes a single queuing system using the notation 
A/B/m/k/l where:

- A is the inter-arrival time distribution of customers
- B is the service time distribution
- m is the number of parallel servers
- k is the limit on the customers in this system
- l is the population size

If the population size or the limit on the queue length is not specified then they are assumed to be infinite.

Typical values for A; B include:

- M : exponential distribution
- Er : r stage Erlangian distribution
- D : Deterministic
- G : General

**Examples:**

- M/M/1: exponential inter-arrival, exponential service, single server
- M/Er/1: exponential inter-arrival, r stage Erlang service, single server
- M/G/1: exponential inter-arrival, general service time, single server
- M/M/K/K: exponential inter-arrival, exponential service, K servers and at most K customers present [31].

**M/G/1:**

This is a single server model of queuing uses Poisson distribution with mean ‘l’ for the inter-arrival time. Service time for the arrival uses any distribution with mean ‘m’ and with standard deviation's. In M/G/1 the service time for each arrival is random.

**M/M/1:**

This model uses the Poisson process with exponential inter arrival time and the exponential service time using single server.
5.7.2 Queuing Techniques

Queuing techniques generally provides different queue levels and handling for different classes of traffic. For this project traffic distribution is based on CPU usage of servers. When load balancer received HTTP traffic load. These traffic requests placed in queue for their process. Virtual servers acts as bridge for these traffic, allows these according to load balancing algorithm policy. Several queuing techniques are available for study. Some of them which are consider are as commonly used are discusses below

- **Weighted Fair Queuing**
  WFQ is commonly used, flow based queuing algorithm. Different traffic flows are queued to prevent bandwidth starvation. A flow is composed of all packets with the same source address/port on and destination address/port combination. Weighted Fair Queuing (WFQ) is a packet scheduling technique allowing guaranteed bandwidth services. The purpose of WFQ is to let several sessions share the same link. WFQ is an approximation of Generalized Processor Sharing which, as the name suggest, is a generalization of Processor Sharing (PS). In PS each session has a separate FIFO queue. At any given time the N active sessions (the ones with non-empty queues) are serviced simultaneously, each at a rate of 1/N:th of the link speed. [14][15]

- **Class based Weighted Fair Queuing**
  CBWFQ is an enhancement to the WFQ algorithm that includes user defined traffic classes. Traffic classes can be defined based on protocol, port, Access Control, input queues. The bandwidth is provided to the class when congestion occurs. The queue is the maximum number of packets that allowed in the class based queue. If the queue fills up then packets are dropped.

- **Weighted Random Early Detection**
  WRED is a little different from other queuing schemes. Instead of trying to deal with congestion after it occurs, WRED tries to detect congestion before it happens and then avoid it. WRED tries to avoid the congestion by randomly discarding selected packets before the queues fill up. Ideally TCP packets are dropped, when a TCP packet is dropped, the protocol will slow down and retransmit. For UDP or RTP flows that do not retransmit. WRED usually relies on IP
precedence bits to provide a weighting scheme to help decide which packets to drop. The higher the priority, the lower a packet's chance dropped. [14]

5.7.3 Queuing Networks

We can classify queuing networks as either Closed networks: in which a fixed set of jobs circulate between nodes, but no new jobs are introduced and no jobs leave the system e.g. a computer system with a fixed number of terminals attached to it or as Open networks: in which jobs may enter and leave the system. Open networks may be further classified as feed-forward if a single job visits each node at most once. [31]

5.8 Load Balancing Device

F5 content switch BIG-IP v 9 is used as a load balancing device for this project.

5.8.1 Introducing the BIG-IP® system

“The BIG-IP® system is a port-based, multilayer switch that supports virtual local area network (VLAN) technology. Because hosts within a VLAN can communicate at the data-link layer (Layer 2), a BIG-IP system reduces the need for routers and IP routing on the network. This in turn reduces equipment costs and boosts overall network performance. At the same time, the BIG-IP system’s multilayer capabilities enable the system to process traffic at other OSI layers. The BIG-IP system can perform IP routing at Layer 3, as well as manage TCP, UDP, and other application traffic at Layers 4 through 7.” [6, page 19] BIG-IP system offers several modules like

- BIG-IP Local Traffic Manager
- BIG-IP Global Traffic Manager
- BIG-IP Link Controller

Module which is used in project is BIG-IP Local Traffic Manager.

5.8.2 BIG-IP® Local Traffic Manager

“The BIG-IP system includes local traffic management features that help to make the most of network resources. Using the powerful Configuration utility, we can customize the way that the BIG-IP system processes specific types of protocol and application traffic. By using features such as virtual servers, pools, and profiles, it ensures that traffic passing through the BIG-IP system is processed quickly and efficiently, while meeting all of security needs.” [6, page 19]

The BIG-IP® local traffic management (LTM) system is specifically designed to manage local
network traffic. Local traffic management refers to the process of managing network traffic that comes into or goes out of a local area network (LAN), including an intranet. “The most common feature of local traffic manager is to understand the nature of traffic load and intelligently redirect the incoming traffic, for the purpose of intelligently tuning the load on network servers. However, tuning server load is not the only type of local traffic management. The LTM system includes a variety of features that perform functions such as inspecting and transforming header and content data, managing SSL certificate-based authentication, and compressing HTTP responses. In so doing, the LTM system not only directs traffic to the appropriate server resource, but also enhances network security and frees up server resources by performing tasks that web servers typically perform.” [6, page 20]. Detail configuration and specification of V9 1500 series F5 content switch are in Appendix B.

5.9 Software used for HTTP- Requests

For this project Http Traffic Gen software is used. Http Traffic Gen is a tool that generates http traffic to the specified URL. Traffic Gen uses deterministic traffic generation approach, inter arrival time between two requests are constant. As parameter for traffic burst here need to specify HTTP Request Parameters packet Count and Interval in milliseconds between two packets [27].
6.0 Experiments

6.1 Introduction

This section describes the experiments, aimed for testing and verifying the performance of load balancing algorithms through implemented scenario on F5 content switch. We explain not only the experimental architecture but also workload used to perform the measurement of throughput and response time of two servers. Server response time and throughput are two important performance metrics that are measured through different performance utility.

Throughput means the measurement of amount of data that individual server can send in particular time, in other words the rate of flow of data in Kbps, Mbps or Gbps. Here the other parameter that we measured is the response time of each server for the same burst of HTTP traffic. Response time means the measurement of time in ms for response of each server to a client request.

In our experiments, throughput is measured through F5 content switch performance utility while response time of server is measured through Applications Manager 7 software. Traffic load for these experiments are generated through Traffic Generator software. This requires HTTP request parameters packet count and time interval in milliseconds for a specified IP address. In our experimental setup, for each test scenario 10,000 HTTP requests have been sent with the time interval of 1ms, 3ms, 5ms, 7ms and 10ms. Virtual server which is configured on content switch receives these burst of web requests and then forward them on pool members according to selected load balancing algorithm where it processed on the bases of first come first server. Throughput and response time at server are separately measured against individual burst of traffic on each server.

These experiments were originally designed for the purpose of testing the implemented load balancing solution using F5 content switch. These experiments points out how statically traffic load is distributed among server and how to measure the performance of load balancing pool members on the basis of CPU usage of servers (processing speed) in context of throughput and response time for the client requests. A total two experimental test were performed, all tests are separately described.
In experiments five different static HTML websites with different sizes (1KB, 10KB, 100KB, 500KB, 1024KB) are tested, that are hosted on Apache web Server on each node (hosting server) in a load balancing pool. Three workstation pc are used to generate HTTP requests as a client. Testing web site is accessible from each client with same IP (virtual IP on F5 content switch) address. While creating a load balancing pool for scenario two servers named “A” and “B” are assigned to the load balancing pool. Servers platform are Intel® Xeon™ 2.99 GHZ Processor with 1.0 GB memory and 80 Gigabyte disk with Windows Server 2003 Enterprise Edition. These servers are also known as pool members. Load balancing pool contains servers to which HTTP requests can be sent for processing. These pool members are associated with a virtual server named “Test1” have specific virtual IP address. Now, the address (pool members IP address) of testing website is same as of the virtual server on content switch. Virtual server is configured through load balancing configuration utility. This virtual server receives the HTTP traffic requests which are generated through clients and then directs these traffic requests to members of the load balancing pool according to the policy of the implemented load balancing algorithm.

Apache web server is configured on both Nodes (servers), the two pool members or nodes with IP address 10.0.1.2 represents server “A” while IP address 10.0.1.3 represents server “B” shown in figure 6.1. Testing Web site mutually shared and accessible with same virtual IP address 192.168.1.182 from both servers. An ICMP (internet control message protocol) type of
a health monitor is used for this experiments which simply determines whether the status of server is up or down.

6.2 Experimental Setup Details

In this section, setup details of all the test scenarios and their differences are presented and discussed.

6.2.1 Test Scenario 1

As implemented scenario is based on CPU utilization of servers, so before making an experiment we observed the CPU usages of both servers. Server “A” and Server “B” both are having same CPU usage (Processing speed) in our first test scenario, Testing web sites of five different sizes (1KB, 10KB, 100KB, 500KB, 1024KB) are used in this experiment. HTTP requests are generated through client workstations using Traffic Gen software. The HTTP requests have been sent to virtual server with traffic burst of 10,000 with time interval of 1ms, 3ms, 5ms, 7ms and 10ms one by one.

Initially Round Robin load balancing algorithm is configured on virtual server and traffic distribution on both nodes (servers) is according to Round Robin load balancing mechanism. Web requests are distributed on two servers “A” and “B” so according to Round Robin load balancing algorithm half requests are forward on server “A” remaining half requests are processed on server “B” simultaneously. Throughput (bits/sec) and response time on each server are separately measured.

After performing this step, Ratio Model load balancing algorithm is configured on virtual server and traffic distribution on both nodes is according to Ratio Model load balancing mechanism. While making configuration of virtual server according to Ratio Model Load balancing Algorithm here need to specify the ratio of traffic distribution so 2:1 is set for this test scenario. Traffic load is distributed on two servers “A” and “B”. Server “A” is served with ratio 2 while other server is served with ratio 1 of the total requests. Throughput (bits/sec) and response time on each server are separately measured.

All results shown are average values from 3 experiments. Average recorded data of these 3 experiments are presented as tabular form in Appendix A.
6.2.2 Test Scenario 2

The experimental environment for test scenario 2 is same as of test scenario 1, the only different is in term of CPU utilization, and CPU usage of both nodes is different from each other. Server "A" is running with CPU usage of 30 %, while server "B" is running with CPU usage of 70% thus having different processing speed. Average recorded data of these 3 experiments are presented as tabular form in Appendix A.
7.0 Results Analysis

In the previous section the experimental setup on each scenario were described In addition the main purpose of test scenario were also discussed. In this section the recorded results of both tests are presented, at the end of this section both test scenario are investigated and analyzed collectively.

7.1 Test Scenario1 Results

HTTP-requests results on the basis of throughput and Response time at both servers are collectively investigated and analyzed. Results of test scenario 1 are presented in two steps.

Step1:

Step1 is based on Round Robin load balancing algorithm, Results of this algorithm is discussed in this step. As virtual server is configured with Round Robin load balancing configuration settings, burst of HTTP requests is received on virtual server with the time interval of 1ms, 3ms, 5ms, 7ms, 10ms through clients. HTTP-traffic distribution on both servers “A” and “B” through virtual server is with equal numbers, so according to Round Robin load balancing algorithm half requests are forward on server “A” remaining half requests are processed on server “B” simultaneously.

Throughput and Response time results are collected at both servers separately. Following figure 7.1 and 7.2 represents the graph between size of the website and the throughput at server “A” and server ”B” while figure 7.3 and 7.4 represents the graph between size of the website and the Response time at server “A” and server “B”.
Round Robin Load Balancing Algorithm
Throughput results at server A

Figure 7.1

Round Robin Load Balancing Algorithm
Throughput results at server B

Figure 7.2

Round Robin Load Balancing Algorithm
Response Time results at server A

Figure 7.3

Round Robin Load Balancing Algorithm
Response Time results at server B

Figure 7.4

Step2:
In Step 2 of test scenario 1 result are based on Ratio Model load balancing algorithm, virtual server is configured with Ratio Model load balancing configuration settings. HTTP requests are received on virtual server with the time interval of 1ms, 3ms, 5ms 7ms and 10 ms through clients. While making configuration of virtual server according to Ratio Model Load balancing Algorithm here need to specify the ratio of traffic distribution so 2:1 is set for this test. HTTP requests are processed with 2:1, as traffic load is distributed on two servers “A” and “B”. So server “A” is served with ratio 2 while other server is served with ratio of 1. Throughput and Response time results of both servers are separately measured.
Throughput and Response time results are collected at both servers separately. Following figure 7.5 and 7.6 represents the graph between size of the website and the throughput at server “A” and server "B" while figure 7.7 and 7.8 represents the graph between size of the website and the Response time at server “A” and server “B”.

![Figure 7.5](image1)

![Figure 7.6](image2)

![Figure 7.7](image3)

![Figure 7.8](image4)

### 7.2 Test Scenario 2 Results

Experimental setup for test 2 is exactly same as of test 1. The only difference is at CPU utilization of servers. Server "A" is running with CPU usage of 30 %, while server "B" is running with CPU usage of 70% thus having different processing speed. Results of test scenario 2 are presented in two steps.

**Step 1:**
Step 1 presents the results of Round Robin load balancing algorithm. Following figure 7.9 and 7.10 represents the graph between size of the website and the throughput at server “A” and server “B” while figure 7.11 and 7.12 represents the graph between size of the website and the Response time at server “A” and server “B”.

Step 2:
Step 2 presents the results of Ratio Model load balancing algorithm. Following figure 7.13 and 7.14 represents the graph between size of the website and the throughput at server “A” and server “B” while figure 7.15 and 7.16 represents the graph between size of the website and the Response time at server “A” and server “B”.

Högskolan Dalarna
Röda vagen 3,
781 88 Borlange

Tel: 023 778 000
Fax: 023 778 050
ULR: http://www.du.se
7.3 Analysis on All Test Scenarios

The extreme analyses on raw statistics of results of all test scenarios are as follows.

- We analysed from the experiments that the throughput and response time varies for the websites of different sizes and also for different burst of traffic. And the results are showing that throughput for the bigger sized website is higher as compared to the website with smaller size where as the response time is also higher for the big sized website. These results are relatively simple and it happens in all situations of the web services but the important thing is to analysis that how these varying results affect our load balancing solution. The result affects the load balancing solution in improving the throughput and response time for particular website, where we can have decisions for taking different number of servers in a load balancing pool to gain the required performance of the web service.

Throughput and response time results are set on the server’s side at the run time of web service. For each single client’s request a web service shows some value of throughput and response time on the server side that keep on changing the values for varying client’s requests to a web service in particular time interval. Apache web server used in this project is a multi threaded server for Windows operating system that means a single process can handle all the client’s requests at the same time. The web server when online, always ready for serving the clients requests, and whenever it gets any single request it responses to the client. The web server becomes busy with increasing number of requests, results in increase of response time on the server side. The response time on the server side is the time in which a single client will be responded with the other multiple clients concurrently but as the request comes in a queue the arrival time of requests on the server can be different. For explaining the throughput and response time result in more detail we consider a result e.g. for 10000 burst of traffic in 1ms of time interval, load balancing pool with two servers assigned to distribute the traffic load via F5 content switch. The results of the throughput and response time on each server having web site of size 1KB where the traffic load is distributed equally using Round Robin algorithm on F5 switch.
Web site size: 1KB
Burst of request: 10000
Time interval between requests: 1ms
Total time for the burst: 10 s
The rate of data transfer on each server side (Throughput): 0.9 Mbit/s
The time on server side for response to multiple clients (Response time): 12 ms

So, in 12 ms each client will be responded with the amount of data in KB that is nearly equal to the size of the website.

- The results are showing very minor increase in data transfer rate and server response time for increasing time interval for the burst of HTTP traffic. Here we observed some limitations of the HTTP traffic generator used in this project that it can have the minimum time interval of 1 ms in between two requests and the inter arrival time between two successive requests remains constant. But in the real time web services, increasing number of client’s requests in particular time interval or the continuous requests in a very short time interval (0.001 ms, 0.0001ms) have much impact on the performance of the web service. So that’s why change in the burst of traffic didn’t worked much for observing the performance of the load balancing solution in this project.

- As from these experiments the change in the size of the website highly effects the results in throughput and response time where as the results are less effected by increase in the number of Http requests. We observed that throughput becomes constant at the value of 10.2 Mb/sec with two servers in a load balancing pool and it does not increase even there is increase in the size of the website e.g. for 10,000 burst of request in 1ms interval, the server with 1MB size of website shows the maximum throughput of 10.2Mb/s and this result never comes higher than 10.2Mb/s for bigger size of websites with 2MB, 3MB etc. It requires more servers to be configured in a load balancing pool to maintain the throughput and response time values for bigger size of websites. So we have seen the maximum throughput of 10.2 Mb/sec using two servers in a load balancing pool.
• We also observed from the results of all the experiments that the server with higher CPU utilization (lower processing speed) showing lower data transfer rate than the server with lower CPU utilization (high processing speed) where as the response time of the server with lower CPU utilization is better than the response time of the server with higher CPU utilization. For the more efficient load balancing solution in the environment where there are servers of different processing speed, we could assign comparatively less load to the node of low processing speed. Using two load balancing algorithms, we observed that the result for the round robin are quite better where the servers have equal processing speed, where as the ratio model is showing good results for servers with different processing speed by assigning different ratio on each server for traffic distribution i.e. using RR algorithm, 10000 burst of traffic in 1ms interval each web server of same processing speed has the same response time of 732 ms, where as the response time is showing much difference with different processing speed of servers in load balancing pool. By using Ratio model algorithm these results are much better in the sense that there is not much difference in the response time of servers of different processing speeds as the weights are assigned according to their processing speed.

• During all experiments CPU utilization of both servers do not remain constant it varies with the processing on HTTP-request.

• We have checked the performance of F5 content switch with high number of requests in different bursts (20000, 50000, 100000 etc) of HTTP traffic and examined that all HTTP requests from clients are served through F5 content switch none of HTTP request was dropped while taking experiments.

• For these experiments where CPU usages of server was high, data transfer rate become slower and also response time was higher but we didn’t suffer any bottleneck on both server and load balancer end, at least in our experiments.

• Bottleneck at pool members and F5 content switch will come when the number of web requests per unit time exceeds the limits or the size of the website is very big, the
servers will not respond to clients in timely manner. In our experiments, servers were still transferring data to clients with some delay and low data rate for the very high HTTP traffic load. Bottleneck at server depends on servers processing speed that varies with hardware performance (processing speed) of servers.

8.0 Conclusion and Future work

The heavy utilization of websites is often out of balance, resulting in the low-performance resources being overloaded with multiple requests and services become down which creates irritation for end user. The server load balancing is the solution to achieve the high availability and high performance of websites. F5 content switch is a server load balancer that supports emerging applications and used to evaluate the reliability and performance operation of layer7 traffic. Load balancing solution using F5 content switch provides the high scalability for distributing the HTTP requests among servers, the server nodes in a load balancing pool can be increased with the need of business requirements. It is concluded from conducted experiments that Ratio Model load balancing algorithm is most suitable in the environment of load balancing servers with different CPU utilization as it allows assigning the weight according to CPU usage.

Dynamic load balancing can consider as a future work of this project, as traffic load keeps increasing with growing number of users it requires the more dynamic solution for load balancing among servers that continuously detects the load on multiple servers and takes the decision to handle this load dynamically.
Reference

[9] A Load Balancing Algorithm for High Speed Intrusion Detection LU Sheng, GONG Jian, RUI Suying Department of Computer Science and Engineering, Southeast University, Nanjing, China Eastern China (North) Network Centre of CERNET (210096) 86-25-3614718 {shlu, jgong, syrui}@njnet.edu.cn
[28] Stochastic load balancing and related problems by Ashish Goel etal , publish 1999
[31] Computer Systems Modelling R. J. Gibbens Computer Laboratory Michaelmas Term 2001
Appendix A

Test-1
Round Robin Load Balancing Algorithm
Results: Throughputs and Response time
Server A = Server B (CPU utilization)

<table>
<thead>
<tr>
<th>Time Interval</th>
<th>Web Site Size</th>
<th>Throughput (Mbits/sec) at server A</th>
<th>Response Time (ms) at server A</th>
<th>Throughput (Mbits/sec) at server B</th>
<th>Response Time (ms) at server B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ms</td>
<td>1 KB</td>
<td>0.9</td>
<td>12</td>
<td>0.9</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>4.8</td>
<td>19</td>
<td>4.8</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>8.2</td>
<td>96</td>
<td>8.2</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>8.9</td>
<td>439</td>
<td>8.9</td>
<td>439</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>10</td>
<td>732</td>
<td>10</td>
<td>732</td>
</tr>
<tr>
<td>3 ms</td>
<td>1 KB</td>
<td>0.9</td>
<td>12</td>
<td>0.9</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>4.8</td>
<td>18</td>
<td>4.8</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>8.2</td>
<td>96</td>
<td>8.2</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>8.9</td>
<td>435</td>
<td>8.9</td>
<td>435</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>10</td>
<td>730</td>
<td>10.1</td>
<td>730</td>
</tr>
<tr>
<td>5 ms</td>
<td>1 KB</td>
<td>0.9</td>
<td>12</td>
<td>0.9</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>5.0</td>
<td>18</td>
<td>5.0</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>8.3</td>
<td>94</td>
<td>8.3</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>8.9</td>
<td>435</td>
<td>8.9</td>
<td>435</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>10.1</td>
<td>730</td>
<td>10.1</td>
<td>730</td>
</tr>
<tr>
<td>7 ms</td>
<td>1 KB</td>
<td>0.9</td>
<td>12</td>
<td>0.9</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>5.1</td>
<td>16</td>
<td>5.1</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>8.3</td>
<td>94</td>
<td>8.5</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>9.0</td>
<td>432</td>
<td>9.0</td>
<td>432</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>10.2</td>
<td>730</td>
<td>10.2</td>
<td>726</td>
</tr>
<tr>
<td>10 ms</td>
<td>1 KB</td>
<td>0.9</td>
<td>12</td>
<td>0.9</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>5.1</td>
<td>16</td>
<td>5.1</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>8.5</td>
<td>93</td>
<td>8.5</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>9.0</td>
<td>432</td>
<td>9.0</td>
<td>432</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>10.2</td>
<td>726</td>
<td>10.2</td>
<td>725</td>
</tr>
</tbody>
</table>
Ratio Model Load Balancing Algorithm

Results: Throughputs and Response time

HTTP-Requests with 2:1 (6700, 3300)

<table>
<thead>
<tr>
<th>Time Interval</th>
<th>Web Size</th>
<th>Throughput (Mbits/sec) at server A</th>
<th>Response Time (ms) at server A</th>
<th>Throughput (Mbits/sec) at server B</th>
<th>Response Time (ms) at server B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ms</td>
<td>1 KB</td>
<td>0.4</td>
<td>14</td>
<td>0.9</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>4.4</td>
<td>21</td>
<td>5.0</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>7.7</td>
<td>102</td>
<td>8.2</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>8.4</td>
<td>468</td>
<td>9.0</td>
<td>430</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>9.5</td>
<td>772</td>
<td>10.0</td>
<td>732</td>
</tr>
<tr>
<td>3 ms</td>
<td>1 KB</td>
<td>0.6</td>
<td>14</td>
<td>0.9</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>4.4</td>
<td>21</td>
<td>5.0</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>7.7</td>
<td>102</td>
<td>8.2</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>8.4</td>
<td>468</td>
<td>9.0</td>
<td>430</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>9.5</td>
<td>772</td>
<td>10.0</td>
<td>732</td>
</tr>
<tr>
<td>5 ms</td>
<td>1 KB</td>
<td>0.6</td>
<td>14</td>
<td>0.9</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>4.4</td>
<td>21</td>
<td>5.1</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>7.8</td>
<td>102</td>
<td>8.3</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>8.5</td>
<td>464</td>
<td>9.2</td>
<td>428</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>9.6</td>
<td>768</td>
<td>10.2</td>
<td>732</td>
</tr>
<tr>
<td>7 ms</td>
<td>1 KB</td>
<td>0.7</td>
<td>13</td>
<td>1.0</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>4.4</td>
<td>20</td>
<td>5.1</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>7.8</td>
<td>100</td>
<td>8.6</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>8.5</td>
<td>464</td>
<td>9.3</td>
<td>428</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>9.6</td>
<td>768</td>
<td>10.2</td>
<td>730</td>
</tr>
<tr>
<td>10 ms</td>
<td>1 KB</td>
<td>0.7</td>
<td>13</td>
<td>1.0</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>4.5</td>
<td>20</td>
<td>5.1</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>7.8</td>
<td>100</td>
<td>8.6</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>8.5</td>
<td>461</td>
<td>9.2</td>
<td>428</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>9.6</td>
<td>768</td>
<td>10.2</td>
<td>730</td>
</tr>
</tbody>
</table>
Test-2

Round Robin Load Balancing Algorithm

Results: Throughputs and Response time

Server A CPU utilization is not equal to Server B

<table>
<thead>
<tr>
<th>Time Interval</th>
<th>Web Site Size</th>
<th>Throughput (Mbits/sec) at server A</th>
<th>Response Time (ms) at server A</th>
<th>Throughput (Mbits/sec) at server B</th>
<th>Response Time (ms) at server B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ms</td>
<td>1 KB</td>
<td>0.9</td>
<td>12</td>
<td>0.6</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>4.9</td>
<td>19</td>
<td>4.5</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>8.2</td>
<td>97</td>
<td>7.8</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>8.9</td>
<td>438</td>
<td>8.5</td>
<td>464</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>10.0</td>
<td>730</td>
<td>9.4</td>
<td>770</td>
</tr>
<tr>
<td>3 ms</td>
<td>1 KB</td>
<td>0.9</td>
<td>12</td>
<td>0.6</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>4.9</td>
<td>19</td>
<td>4.5</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>8.2</td>
<td>97</td>
<td>7.8</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>8.9</td>
<td>438</td>
<td>8.5</td>
<td>464</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>10.0</td>
<td>730</td>
<td>9.4</td>
<td>770</td>
</tr>
<tr>
<td>5 ms</td>
<td>1 KB</td>
<td>0.9</td>
<td>12</td>
<td>0.7</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>5.0</td>
<td>19</td>
<td>4.5</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>8.3</td>
<td>95</td>
<td>7.8</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>8.9</td>
<td>436</td>
<td>8.5</td>
<td>462</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>10.1</td>
<td>728</td>
<td>9.5</td>
<td>768</td>
</tr>
<tr>
<td>7 ms</td>
<td>1 KB</td>
<td>0.9</td>
<td>11</td>
<td>0.7</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>5.0</td>
<td>18</td>
<td>4.5</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>8.3</td>
<td>95</td>
<td>7.9</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>9.0</td>
<td>436</td>
<td>8.6</td>
<td>462</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>10.1</td>
<td>728</td>
<td>9.5</td>
<td>766</td>
</tr>
<tr>
<td>10 ms</td>
<td>1 KB</td>
<td>1.0</td>
<td>11</td>
<td>0.7</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>5.0</td>
<td>18</td>
<td>4.6</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>8.3</td>
<td>95</td>
<td>7.9</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>9.0</td>
<td>436</td>
<td>8.6</td>
<td>462</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>10.1</td>
<td>726</td>
<td>9.5</td>
<td>766</td>
</tr>
</tbody>
</table>
Ratio Model Load Balancing Algorithm

Results: Throughputs and Response time
HTTP-Requests with 2:1 (6700, 3300)

<table>
<thead>
<tr>
<th>Time Interval</th>
<th>Web Site Size</th>
<th>Throughput (Mbits/sec) at server A</th>
<th>Response Time (ms) at server A</th>
<th>Throughput (Mbits/sec) at server B</th>
<th>Response Time (ms) at server B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ms</td>
<td>1 KB</td>
<td>0.8</td>
<td>13</td>
<td>0.9</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>4.7</td>
<td>20</td>
<td>4.8</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>8.3</td>
<td>94</td>
<td>8.2</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>9.0</td>
<td>436</td>
<td>8.8</td>
<td>438</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>10.1</td>
<td>730</td>
<td>9.9</td>
<td>740</td>
</tr>
<tr>
<td>3 ms</td>
<td>1 KB</td>
<td>0.8</td>
<td>13</td>
<td>0.1</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>4.7</td>
<td>20</td>
<td>4.8</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>8.3</td>
<td>94</td>
<td>8.2</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>9.0</td>
<td>436</td>
<td>8.8</td>
<td>438</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>10.1</td>
<td>730</td>
<td>9.9</td>
<td>736</td>
</tr>
<tr>
<td>5 ms</td>
<td>1 KB</td>
<td>0.8</td>
<td>12</td>
<td>0.9</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>4.7</td>
<td>20</td>
<td>4.7</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>8.4</td>
<td>94</td>
<td>8.2</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>9.0</td>
<td>432</td>
<td>8.8</td>
<td>436</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>10.1</td>
<td>730</td>
<td>9.9</td>
<td>736</td>
</tr>
<tr>
<td>7 ms</td>
<td>1 KB</td>
<td>0.9</td>
<td>12</td>
<td>0.9</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>4.7</td>
<td>19</td>
<td>4.8</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>8.5</td>
<td>92</td>
<td>8.4</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>9.1</td>
<td>432</td>
<td>9.0</td>
<td>430</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>10.2</td>
<td>728</td>
<td>9.9</td>
<td>732</td>
</tr>
<tr>
<td>10 ms</td>
<td>1 KB</td>
<td>0.9</td>
<td>12</td>
<td>0.9</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>10 KB</td>
<td>4.8</td>
<td>19</td>
<td>4.8</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>100 KB</td>
<td>8.5</td>
<td>92</td>
<td>8.4</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>500 KB</td>
<td>9.1</td>
<td>428</td>
<td>9.0</td>
<td>430</td>
</tr>
<tr>
<td></td>
<td>1024 KB</td>
<td>10.2</td>
<td>726</td>
<td>10.0</td>
<td>432</td>
</tr>
</tbody>
</table>
Appendix B

Local Traffic Manager Configuration:

After set up base network and authorizing the administrative access to the Local Traffic Manager system, at least a default VLAN assignment for each interface, the next step is to configure a network for managing traffic targeted to internal servers. At the heart of the LTM system are virtual servers and load balancing pools. Virtual servers receive incoming traffic; perform basic source IP and destination IP address translation, and direct traffic to servers, which are grouped together in load balancing pools. [6, page 25]

To implement the proposed emulation model LTM utility is used for configurations, with this utility a set of desired objects are created and configured according to the instruction provided by configuration help file of content switch. Each object has a set of configuration settings that can use as it is or can change to desire needs. These objects are:

- **Virtual servers**

  “Virtual servers receive requests and distribute them to pool members” [6, page 25]. One host virtual server is created with name test1 for this scenario as shown in figure-4 below. The IP address that assigned to a host virtual server is 192.168.1.182 which matches the IP address that DNS associates with the testing web site domain name. Virtual server just act as a bridge manipulate the traffic and forward the traffic load to desired servers.

While creating the virtual server here need to specify which kind of server is required. Two types of virtual server options are available through LTM.

1. Network virtual server
2. Host virtual server

![Figure 4: Virtual Server](image)
• **Network virtual server**

“NVS is a virtual server whose IP address has no bits set in the host portion of the IP address (that is, the host portion of its IP address is 0). There are two kinds of network virtual servers: those that direct client traffic based on a range of destination IP addresses, and those that direct client traffic based on specific destination IP addresses that the LTM system does not recognize”. [6, page 39]

• **Host virtual server**

“HVS represents a specific site, such as an Internet web site or an FTP site, and it load balances traffic targeted to content servers that are members of a pool. The IP address that assigned to a host virtual server should match the IP address that DNS associates with the site's domain name. When the LTM system receives a connection request for that site, the LTM system recognizes that the client's destination IP address matches the IP address of the virtual server, and subsequently forwards the client request to one of the content servers that the virtual server load balances.”[6, page 39]

**Nodes**

“Nodes represent server IP addresses on the network that can enable and disable.” (6, page 25) Two nodes named T1 and T2 are shown in figure-5 IP address 10.0.1.2 represents server A while IP address 10.0.1.3 represents server B which are created and configured for this scenario. Nodes are that objects of network devices to which a BIG-IP® local traffic management (LTM) system passes traffic. “A primary feature of nodes is their association with health monitors. Like pool members, nodes can be associated with health monitors as a way to determine server status. However, a health monitor for a pool member reports the status of a service running on the device, whereas a health monitor associated with a node reports status of the device itself” [6, page 57].
Load balancing pools:

“Load balancing pools contain servers to which requests can be sent for processing” [6, page 25]. While creating a load balancing pool for this scenario T1 and T2 assigned as servers (known as pool members) to the load balancing pool, and then associated this pool with a virtual server named “Test1” in the LTM system. The LTM system then directs traffic coming into the virtual server to a member of that pool.

“In a typical client-server scenario, a client request goes to the destination IP address specified in the header of the request. For sites with a large amount of incoming traffic, the destination server can quickly become overloaded as it tries to service a large number of requests. To solve this problem, the BIG-IP® local traffic management (LTM) system distributes client requests to multiple servers instead of to the specified destination IP address only”. [6, page 67]
Application-type profiles

"Application-type profiles contain settings that define the behaviour of various traffic types, such as TCP, HTTP, and SSL". [6, page 25]. HTTP Profile is used for this project.

Monitors

Monitors track the current health or performance of pool members. An ICMP type of monitor is used to track the health of the server which simply determines whether the status of server is up or down.

1500 platform & Specification

This platform is designed for the best performance. This switch can manage all the capabilities of F5 Networks traffic management software. The 1500 platform figure 8 shows the power of two SFP (LC connector type) and four (10/100/1000) interfaces, with SSL processing available as an additional add-on through the software license.

![Figure 7](image.png)

1500 specifications:

- Weight 19 lbs. (per unit)
- Processor Single 2.5 GHz Celeron
- Network Interface 4x10/100/1000
- 2xFiber Gigabit Ethernet interface (SFP GBIC)
- 1000BASE-SX - 850nm (LC Connector, 6 modules included)
- 1000BASE-LX - 1310nm (LC Connector, optional)
Name: Toqeer Ahmed
Registration Number: E3381D

- 1x10/100 Ethernet Management port
- Hard Drive Capacity 80 GB hard drive
- RAM 768 MB (expandable to 2 GB)
- Power supply 300W 100/240 +/- 10% VAC AUTO Switching" [6].

Högskolan Dalarna
Röda vägen 3,
781 88 Borlange

Tel: 023 778 000
Fax: 023 778 050
ULR: http://www.du.se