EVALUATION OF LOAD BALANCING ALGORITHMS AND INTERNET TRAFFIC MODELING FOR PERFORMANCE ANALYSIS

by

Arthur L. Blais

B.A., California State University, Fullerton, 1982

A thesis submitted to the Graduate Faculty of the

University of Colorado at Colorado Springs

in partial fulfillment of the

requirements for the degree of

Master of Science

Department of Computer Science

2000

This thesis for the Master of Science degree by

Arthur L. Blais

has been approved for the

Department of Computer Science

by

_____

C. Edward Chow, Chair

_____

Charles M. Shub

_____

Richard S. Wiener

_____

Date

Abstract

This thesis presents research and study of load balancing algorithms and the analysis of the performance of each algorithm in varying conditions. The research also covers a study of the characteristics of Internet traffic and its statistical properties. The network workload models that were implemented in the simulation program were derived from the many works already published within the Internet community. These workload models were successfully implemented and statistical proof is given that they exhibit characteristics similar to the workloads found on the Internet. Finally, this thesis compares and contrasts the differences between stateless server selection methods and state-base selection methods with the different algorithms studied.

Acknowledgements

First I would like to thank Dr. Chow for his advice, assistance and for guiding my research in directions that I had not considered. Second, I would like to thank Dr. Shub and Dr. Wiener for participating on my thesis committee and for their instruction in the courses that I had with them.

Next, I would like to thank the management at Ford Microelectronics, Inc. for their financial support and also to Jeff Maschler for proof reading the first draft.

Finally, a very big thank you goes to my wife and family whose patience and love helped motivate me over the last few years to complete my studies.

CONTENTS

CHAPTER

TABLES

Table

FIGURES

Figure

CHAPTER I

INTRODUCTION

The research and results of the topic of Load Balancing and Internet traffic modeling are presented in this thesis.

Load Balancing is a form of system performance evaluation, analysis and optimization, which attempts to distribute a number of logical processes across a network of processing elements. There have been many algorithms and techniques that have been developed and studied for improving system performance. In early research in the field of computer science, the main focus for improving performance was to develop algorithms and techniques to optimize the use of systems with limited and expensive resources for scientific computing and information systems. Later, there was an emphasis on how to network groups of computers or workstations and then share the resources among workgroups. More recently there has been a tremendous increase in the popularity of the Internet as a system for sharing and gathering information. The use of the Internet has been increasing at a tremendous rate and there always has been a concern among those in the Internet community that enough resources will be available to provide the expected quality of service that is received by its users.

The process of “balancing”, “sharing”, “scheduling” or “distributing” work using a network of computers or a system of multiple processing elements is a widely studied sub-

ject. This paper will focus on recent works that have been written regarding today's computing environments.

In this thesis, we will look at the art of load balancing and how it can be applied to distributed networks and more specifically the Internet. Initially, the focus of the research for this thesis was on development and analysis of algorithms that minimized the amount of messaging or probing that is required for determining the current workload of a set of processing elements, such as a group of replicated web servers. These algorithms were to compare stochastic based methods of estimating server workloads, with more intrusive methods of messaging and probing. During the process of developing the simulator to be used for evaluating the algorithms in this study, tasks related to modeling network workloads, and the workloads related to the Internet in particular, were identified to be crucial to the research in this area and as a result of this effort, network modeling has become a significant portion of this thesis.

The thesis is divided into the following chapters. Chapter 2 will discuss the concepts and research related to the subject of Load Balancing. Chapter 3 will discuss the issues related to modeling network traffic and in particular Internet traffic. Chapter 4 describes the Network Simulator used to evaluate the load balancing algorithms. Chapter 5 is the Experimental Design of the network simulations, and Chapter 6 will discuss the results of the experimental simulations. Finally, Chapter 7 will discuss the conclusions based upon the experiments performed in this study.

Finally, in addition to the goals already mentioned, it is the hope of the author that this research can be used as a reference for the continued study in the areas of network performance evaluation, modeling and simulation.

CHAPTER II

LOAD BALANCING

Load balancing is probably the most commonly used term for describing a class of processes that attempt to optimize system performance. System performance is optimized by attempting to best utilize a group of processing elements, typically a CPU, or storage elements, such as memory of disk, or some other resource that are interconnected in a distributed network. The process of Load Balancing may also be known as Load Sharing, Load Distribution, Parallel Programming, Concurrent Programming, and Control Scheduling. Although these processes can be quite different in their purpose, the processes all have a common goal. This goal is to allocate logical processes evenly across multiple processors, or a distributed network of processing elements, so that collectively all the logical processes are executed in the most efficient manner possible. Examples of some of the approaches for achieving this goal are: keeping idle systems busy, low execution latency (fast execution time), fast response time, maximizing job throughput, executing jobs in parallel and distribution of jobs to specialized systems. More generally, whenever a processing element becomes idle and there are logical processes waiting for servers, the system should attempt to place any new process or processes waiting for service on an idle server and not on a busy one.

One area that has received a lot of research is the dynamic load balancing of processes by migrating processes from busy servers to less busy servers. The issues of

dynamic load balancing along with the basic load balancing concepts will be addressed in this chapter.

The rest of this chapter is divided into the following sections, the Types of Load Balancing, Selection Methods, Related Works and finally, Applications of Load Balancing.

2.1 Types of Load Balancing

There are basically two types of load balancing, static load balancing and dynamic load balancing. The following two sections will discuss these topics.

2.1.1 Static Load Balancing

Static load balancing is the simplest form of the two types. Static load balancing is the selection and placement of a logical process on some processing element located on a distributed network. The selection of the processing element for some logical process is based upon some weighting factor for that process, or some kind of workload characterization of the processing elements or of the network topology, possibly both. The process of selecting a processing element for a logical process requesting service may be done with two possible methods, a stateless method or a state-based method. With a stateless method the selection of a processing element is done without regard to any knowledge of the system state. A state-based method of selecting a processing element implies that the selection of a processing element requires knowledge of the system state, either globally, or locally. Global knowledge implies that the state of the all the components of the system is known, and local knowledge implies that only partial knowledge is known. If the state

of the system is needed then some kind of messaging or probing of network resources among the requesting processes, agents, or processing elements is needed to determine their availability. Once a processing element is selected for a logical process, the logical process is executed on the selected processing element for the duration of the logical process lifetime.

Two examples of stateless placement techniques for selecting a processing element for logical processes are round robin and random placement. Round robin placement selects the next processing element from a predefined list. Random placement selects an element randomly from a set of processing elements.

Examples of state-based process placement techniques include greedy algorithms and stochastic selection processes. Greedy processes typically try to find the processing element with the lightest load or the best response time, therefore requiring the process to have some knowledge of the workload of each processing element or the state of the network topology between the client and server. Another example is the selection of a processing element using a randomly selected subset of a set of processing elements and then selecting the processor in the subset with the lowest load. Studies by Mitzenmacher (1997) and Dahlin (1998) researched the random subset selection technique in great detail. Finally, stochastic selection techniques may be used to select a processing element based upon the probability distribution of the server loads. This technique may select any server, but the lighter the load the higher the probability of selecting the server. These techniques are described in detail in Chapter 4.

2.1.2 Dynamic Load Balancing

Dynamic load balancing is the initial selection and placement of a logical process on some processing element in a distributed network and then at some point in time there may be a decision made to move a process to some other processing element. The initial selection and placement is done with the same method as static load balancing. But at some point in time during the execution of the logical process, based on some decision criteria, a process may be preempted and migrated to another processing element somewhere else on the network. Generally a process is migrated to another processor if the migration cost or overhead is less than some predetermined metric.

This raises a number of issues regarding dynamic load balancing when the system chooses to move a process. Some of these issues are:

- Which process or processes are candidates for migration?
- Which processing element is the best target for process migration?
- How do you preserve the process state when it is migrated?
- Is the system homogeneous or does it have heterogeneous systems?
- What is the overhead cost of migration?
- When is the decision for migration made?
- Who makes the decision to migrate; the system, server or process?

These issues and others have been the subject of extensive studies. For detailed discussions regarding the issues of dynamic load balancing and migration, see Eagar, Lazowska and Zahorjan (1986), Leland and Ott (1986), Eagar, Lazowska and Zahorjan (1988), Shub (1990), Purohit, Eagar and Bunt (1992), Hailperin (1993), Von Bank, Shub

and Sebesta (1994), Harchol-Balter (1996), and Harchol-Balter and Downey (1997). Later in this chapter a brief review of some of these studies is made.

2.2 Methods for Selecting Processing Elements

This section will discuss four methods for determining the workload of a distributed network: static placement, probing, messaging and finally stochastic methods. These methods are discussed in the following sections.

2.2.1 Static Placement

With static placement, client ambivalence occurs because a processing element is selected without regard to the current state of the processing elements within a distributed network. System state is measured with metrics such as server workloads or available bandwidth. Even though the client wants optimal performance, the client may have a predetermined processing element selected for placing all of its logical processes. Selection may occur geographically, or the client may be ignorant of the set of available processing elements and only knows of one. The client may also rely on some agent to select a processing element and the agent selects a server without any knowledge of the system workload. The method used to select a processing element may be random, round robin or geographic. This kind of strategy does not scale well and provides no alternative for placing the clients logical process on a system with better performance.

2.2.2 Probing

Probing is a strategy by which a client or an agent for a client attempts to determine the load of a system or processing element by sending messages to each processing element it is interested in, and then measuring the response time of the return messages, or by receiving a message from a processing element as a result of the probe. There are several factors that could effect measuring the response time of a probe. The first factor is the available bandwidth of the path between the client and the processing element. The link with the lowest available bandwidth (or throughput) is known as the bottleneck link of the path and consequently this will be the limiting performance factor that the client will receive. Other factors that affect response time are the workloads of the processors, bridges and routers. It may be difficult to determine the location of a system bottleneck, since the causes of system bottlenecks may be a result by one or more of the following conditions: slow or congested networks links, or slow or busy processing elements, bridges or routers. One problem with a probing strategy is that the probes may cause excessive network overhead, which can unfortunately, significantly impact the performance of the processing elements or the network. Determining where bottlenecks occur may be an important factor in improving the performance between the client and the processing element, especially if there are ways to select alternative routes or other processing elements that are capable of providing service.

An example of how to measure available bandwidth and discovering bottleneck links can be found in Carter and Crovella (1996a) using two tools they have developed. Available bandwidth is affected by two factors, the bottleneck link (the link with the lowest capacity) and the congestion caused by the traffic competing for the link capacity on

the path between the client and server. The tool called **bprobe** provides an estimate of the maximum possible (uncongested) bandwidth along a path, and a tool called **cprobe,** which estimates the current congestion along a path. The combination of these two measurements can give you an estimate of the available bandwidth for the application between the client and server. With the knowledge of which links have the best available bandwidth (at the time it is measured) the application may be able to choose a path that will deliver the best response time by avoiding other paths with congestion.

In other research by Carter and Crovella (1996b), they demonstrate how an application can use **bprobe** and **cprobe** to allow an application to dynamically select a server from a group of replicated servers to improve round-trip latency (minimize response time) by avoiding servers who paths are along congested links.

In Zhang (1999) and Chow (1999), they discuss possible ways to avoid probing problems with a set of load balancing agents that share load information about the available bandwidth on network links or the workloads of known processing elements. A client can request the best available processing element from the agent for the client to use. Therefore making the client more intelligent in selecting a processing element that is lightly loaded and improving performance.

2.2.3 Messaging

Messaging is another way a processing element can communicate its workload to the system. A client may register with the processing elements that it wishes to have workload messages sent to it from the processing elements. These messages may be in the

form of a multicast message where the processing elements periodically send messages out on the network to the group of interested clients.

In a dynamic load balancing environment a processing element that is heavily loaded over some predefined threshold may send messages to other processing elements asking for help. If a suitable processing element is available, then new logical processes can be processed there, and possibly, if there is a logical process on the busy processor that can be migrated it may choose to migrate that processes to another processor. Also, lightly loaded processors may send messages to all the servers saying that it is available for receiving new logical processes or for processes to be migrated from heavily loaded processors.

2.2.4 Stochastic Methods

The last strategy is to develop a method for predicting the load of a processing element or network segment based on historical data. Stochastic approaches can reduce system overhead by selection processing elements or network links that have a high probability of being lightly loaded or have lower loads relative to other elements. The historical load data can be communicated to the system during periods when the expected loads are low. With the updated data, the system can then adjust its models with the current trends and use this data for future predictions of system workloads. The advantages of a stochastic approach is lower system overhead by avoiding periodic probing or messaging, especially during periods when the workloads are high. A drawback may be the accuracy of the predictive model that is used for selecting a processing element and for adjusting to changing workloads.

2.3 Related Works

In this section we will look at a number of related research papers, which can be classified into two categories, load balancing techniques and workload characterization.

2.3.1 Load Balancing

Eager, Lazowska, and Zhahorjan (1986) called their research “adaptive load sharing”. They defined the goal of load sharing is to improve performance by redistributing the workload from heavily loaded processors to idle of lightly loaded ones. In their research they analyzed various policies that performed adaptive load sharing. The complexity of these policies varied in how they acquired and used system state information. Simple policies collected very small amounts of system state information and when combined with simple use of the data, yielded dramatic performance improvements. In fact, the performance improvements were close to the more complex schemes they studied. The complex policies acquire and use more system state information with the expectation to take full advantage of the processing power of the system. With the complex policies there is the cost of greater overhead in gathering data in the hope of improving performance, but there is a possibility of poor decisions being made from inaccurate information that could possibly be collected.

Research by Leland and Ott (1986) used load balancing to improve response times seen by the users. Their research quantifies the benefits with using two different load balance schemes on a network on interconnected homogenous processors. The two schemes are called initial placement and process migration. With static processor assignment, the initial placement scheme for processor selection is predetermined and independent of the

current state of the system. Once a process is placed on a processor it remains there for the duration of the life of the process. With dynamic processor assignment, the initial placement of a process on a processor is determined by the current state of the system. If at some point in time the processor is too busy, a migration scheme may be employed to reassign (migrate) a process to a different processor (with a lower load) during the execution of that process. They used CPU time as a way of determining which processes to migrate. The more CPU time a process uses the greater the chance that the process may be migrated to another processor. Their conclusions were that, “with a sufficiently integrated local scheduling policy dynamic assignment will significantly improve response times”. This conclusion is related to the processes that account for the most CPU and disk demand, without adversely affecting the many processes that require little of either. These conclusions however are based upon trace-based workload models, which may make their conclusions invalid for application beyond the scope of the environment that the traces were collected in.

Purohit, Eager and Bunt (1992) defined load sharing as a technique that attempts to improve performance by moving work from congested system processors to lightly loaded processors. They introduce the concept of priority load sharing, a technique that attempts to improve overall system performance by moving jobs from congested system processors to lightly loaded processors. Their goals are to keep all the processors as busy as possible, reduce average processing time and to make optimal use of system resources. They do this by assigning each new job to a priority queue. The system then attempts balance the priority queues in the whole system. They looked at two policy types, static and adaptive. Static load sharing does not react to the instantaneous changes in the state of the system.

A simple estimate based on the average behavior of the system is used to determine which processor to use. With adaptive (dynamic) load sharing the system reacts to the instantaneous changes in the state of the system and makes decisions to migrate a process to a lightly loaded system. This is more complex and it requires a mechanism by which the current state of the system is collected. Adaptive policies also have two additional policies, a transfer policy and a placement policy. The transfer policy determines whether to process a job locally or remotely. A placement policy decides where to send a job for processing when a transfer is necessary. They concluded that policies that exchange state information prior to load exchanges outperform policies that don't exchange information.

Research by Hailperin (1993) looked at a class of distributed applications that dynamically allocated the application processes in a distributed network. He focuses on dynamic load distribution, which migrates existing objects to new processing elements. He divides dynamic load distribution into two categories, load sharing and load balancing. Load sharing is a policy in which the system attempts to avoid having idle processing elements, by placing new jobs or moving existing jobs to the idle processors. Load balancing attempts to distribute the system workload equally, based on a global average workload, among all the processing elements. He also discusses global versus local load balancing. Global load balancing collects the workloads of all the processing elements and estimates the global instantaneous average of all the workloads and then attempts to balance the load by migrating objects from overloaded processing elements to under-loaded ones. Local load balancing has no knowledge of a global system wide workload average. With local load balancing, work is transferred between two neighboring processors. The transfers are repeated among neighboring processors until a system-wide load balance can be achieved.

His thesis solves migration problems with greedy methods and also the use of time series analysis, which models how past workloads are relevant in predicting current workloads. Finally, Hailperin's thesis covers in great detail the taxonomy of load distribution problems.

Research by Harchol-Balter (1996) and Harchol-Balter and Downey (1997) explored dynamic load balancing by analyzing the lifetime distributions of processes. They defined CPU load balancing as migrating processes across the network from processors with high loads to processors with lower loads. The goal is to reduce the average completion time of the processes and improve the utilization of the processors. Load balancing may be done explicitly by the user or implicitly by the system. Process migration is concerned with two policies, migration and location. The migration policy determines when migrations are to occur and which processes can be migrated. The location policy determines which processor is selected for a potential process migration. Choosing a target processor with the shortest CPU run queue is considered a simple and effective way to choose a processor. They concluded however that in comparison to the factors related to migration policies, location policies are insignificant.

They had two main questions they wanted to answer:

1. Is preemptive migration worthwhile, given the additional cost associated with migrating an active process?
2. Which active processes, if any are worth migrating?

To answer these questions, they analyzed the distribution of process lifetimes and the cost to migrate a process. A process is a candidate for migration if its CPU age is greater than some migration cost. Although both preemptive and non-preemptive migra-

tion policies showed improvements, their results show that preemptive migration policies outperformed non-preemptive. The most interesting information found from the research is that they concluded that the probability of a process over one second in age using more than *T* seconds of total CPU time is 1/*T.* Also the probability that a process with age *T* seconds uses at least an additional *T* seconds about 50% of the time, thus the median remaining lifetime of a process is equal to its current age. This information is very useful for system administration.

Research by Bestavros, Crovella, Liu, and Martin (1997), presents a decentralized approach to redirecting incoming requests to servers called Distributed Packet Rewriting. This approach allows all potential servers for client requests to either choose to provide service or redirect the request to a server with fewer connections. This method allows for better scalability and fault tolerance over centralized (single server) redirection strategies. Server load information is shared among the servers by periodic multicast messages.

In Aversa and Bestavros (1999), they describe load balancing web servers using Distributed Packet Rewriting as a technique for distributing web requests across a set of replicated web servers.

2.3.2 Workload Characterization

This section will cover the major research papers that provided detailed information regarding network modeling and modeling Internet traffic. The research papers focus on two areas or study, workload characterization and self-similarity. A detailed discussion of workload characterization and self-similarity is done in Chapter 3, Network Traffic Modeling.

Paxson and Floyd (1997) present a detailed discussion of the difficulties of simulating the Internet. The difficulties are caused by the fact that the Internet is a large rapidly growing and changing environment. Earlier work by Paxson and Floyd (1994, 1995) analyzed wide-area traffic patterns for various protocols and provided evidence that traditional traffic models were poor at modeling network traffic by developing new analytic models, which were better at modeling the high variability they discovered in their traces of Ethernet traffic.

Research analyzing Internet traces found similar characteristics to Ethernet traffic in Internet traffic. Papers by Arlitt and Williamson (1995, 1996, 1997) analyzed the characteristics of web server workloads and developed synthetic workload models to represent these workloads. Research by Cunha, Bestavros and Crovella (1995) analyzed the characteristics found in Internet client traces, and Barford and Crovella (1997) developed analytical models for generating Internet workloads.

Leland, Taqqu, Willinger and Wilson (1994) and Willinger, Taqqu, Sherman, and Wilson (1997) provide evidence that Ethernet traffic is statistically self-similar. Also Paxson and Floyd (1995) determined that Poisson distributions make poor workload models because of the evidence of self-similarity in their analysis of Ethernet traffic. Crovella and Bestavros (1995) look for evidence and possible causes of self-similarity in World Wide Web traffic. Park, Kim and Crovella (1996, 1997) analyze the performance impacts caused by the main factors contributing to the statistical self-similarity that is being observed. Finally, Crovella and Taqqu (1999) estimate the index of the tail in the heavy-tailed distributions found in the Internet traces.

2.3.3 Application

Sun Microsystems Inc. (1996, 1997, 1998) also promotes the use of a CPU farm as a method of executing many jobs in parallel. Sun has a tool call Solstice™ Job Scheduler that can be used to submit jobs requests for execution on a processing element.

At Ford Microelectronics Inc. (FMI) load balancing is used to execute a set of programming tools that do simulations for integrated circuit design. A product called Load Balancer[1] is used to distribute the simulations across the local area network at FMI. The Load Balancer software product allows engineers at FMI to summit job requests to a request queue that is managed by a centralize management process which selects a processing element for executing the job request. The processing elements are homogeneous systems[2] that are either in a pool of compute servers or the workstations used by the engineers on their desktops. This allows the engineers to submit job requests to the system and receive transparent execution of their requests on some processing element on the local area network.

---

1. Load Balancer is a product of Unison Software.

2. The systems are homogenous in the sense that they have the same processor architecture, operating system and execution binaries, however the processors have different clock speeds and memory configurations. With the exception of these performance factors, the execution of simulations should be transparent to the engineers.

CHAPTER III

NETWORK TRAFFIC MODELING

This chapter will present and discuss the network traffic models that characterize network traffic. More specifically this chapter will look at modeling Internet traffic and the models that are used to simulate Internet traffic for the experiments discussed later in this thesis.

The first part of this chapter will present the nature of Internet traffic and the factors that affect the performance perceived by the users. Next, this chapter will present the differences between trace-based models and analytic models. Then, this chapter will cover the characteristics of network traffic and finally, the traffic models and distributions.

3.1 Nature of Internet Traffic

There are a number of research papers that have analyzed the characteristics of traffic on the Internet. They have found that the Internet has highly variable demands in the form of request inter-arrival rates, the size of the files being transferred and popularity of each file. Statistically, the distribution of these factors have a high percentage of the values in the lower ranged but with heavy-tails. We will also see that the variations of these factors have long range dependence and are considered statistically self-similar.

Traditionally request inter-arrival rates were modeled with Poisson distributions. Recently a number of research papers reached a different conclusion, suggesting that

request inter-arrival rates for Ethernet traffic and Internet traffic are poorly modeled by Poisson distributions. In Paxson and Floyd (1995), Arlitt and Williamson (1997) and Barford and Crovella (1997), they made the following conclusions:

- Request inter-arrival times are best modeled by Weibull and Pareto distributions.
- File size distributions have high variance with heavy tails. The best distribution for file sizes is lognormal for the body of the distribution and a Pareto distribution for the tail.
- The popularity of particular files also has an effect on performance. The more frequently accessed files are typically cached. Zipf's law can be used to model the distributions of the popular files that are stored in cache.

3.2 Traffic Models

There are two basic ways for modeling network traffic, trace-based models and analytic models. The two models are compared in the following sections.

3.2.1 Trace-based Models

Trace-based models use actual data traces to generate simulation workloads. The main advantage of using trace-based models is that they are easy to use and implement. The disadvantages are that the traces model only the workload at the time the trace was made. This makes the model difficult to vary or change. Also this model may not reliably identify the causes of system behavior. This model works well if you wish to simulate a specific set of conditions.

3.2.2 Analytic Models

Analytic models use mathematical models to generate simulation workloads. Mathematical models are capable of generating different workloads by varying or changing the workload characteristics. They can however, be difficult to construct. First you need to identify the important workload characteristics to model. Then the characteristics need to be empirically measured and finally the best mathematical model needs to be chosen and fitted to the measured data. Despite these difficulties, if you need to vary or change the conditions of the simulations, then analytic models provide far more flexibility than trace-based models.

3.3 Network Traffic Characteristics

This section on network traffic characteristics identifies the main factors that affect the performance of the network that is being studied in this paper. The factors chosen for this study are the characteristics of the clients, servers and the overall network.

3.3.1 Client Characteristics

The client has two factors that can be characterized, the request size and request rate.

The size of client requests is relatively insignificant when compared to the size of the requested data that can be returned from Internet web servers. The section on server characteristics shows that requested file sizes can be very large.

Client request rates can be broken down into two categories, sleep time and active time. These times can be described as ON/OFF times. Client sleep time occurs when the

client is not actively making requests on the network is considered an OFF time. When the client wakes up it is called the active or ON time. The active time is broken down into two more categories, inactive off time, also called think time and active off time. Inactive off time is the time between when a client wakes up and makes its first request or the time between the completion of one request and the action taken to make a new request. Active off time occurs when there are multiple embedded references in a HTTP request. An HTTP request may have zero or more additional references embedded in it to other objects. These objects can be other web pages, images, video files or audio files. The embedded references are downloaded with the original request and the active off time is the time between the completion of one embedded reference and beginning of another.

3.3.2 Server Characteristics

There are many factors that affect the performance characteristics of a server. These factors are CPU type, operating system, memory, caches, disks and other peripherals, and bus speed. These factors will not be analyzed in any detail since all the processing elements are homogenous in this study and their performance will be held constant. For detailed information about system performance, see Hennessy and Patterson (1990) or similar text.

There are however three factors that can affect the system performance as perceived by the client. These factors are the number of simultaneous connections that a server can service, the request queue, and the sizes of the files that are requested.

The number of connections can affect queue lengths and the amount of time a request may wait in the queue. In this study it is assumed that network link has enough

capacity to handle the maximum throughput of all the connections. It is also assumed that the maximum throughput for each connection is bounded by the throughput of the client link, which is considered the bottleneck link. The maximum queue length is assumed to be infinite, but in this study it will never be longer than the maximum number of clients, since each client will make at most one request at time. Intuitively, the more connections that are available the shorter the average queue length and therefore the shorter the queue wait times will be.

File size distributions significantly impact the performance of the Internet. Most of the files requested are less than 10 kilobytes in size, and many are less than 1 kilobyte. There are however a small percentage of the files served that are very large in size. These large files can significantly impact system performance This leads to two different distributions, a distribution for files less than 10 kilobytes in size called the body and a distribution for files greater than 10 kilobytes, which represents the tail of the distribution. The body of the distribution is best modeled with a lognormal statistical distribution and the tail of the distributions is modeled with a Pareto distribution. The determination of the distributions and the associated research will be discussed in detail in Section 3.4.

3.3.3 Network Characteristics

This section will discuss the issues related to the performance and quality of service that is perceived in the Internet and network topology in general. Because of the complexities and numerous issues that affect the performance of the Internet, this research will not attempt to model these characteristics. For the purposes of simulations the network topology and performance characteristics will be held constant, between the client

and server. This may not be a realistic assumption but for the purposes of evaluation the performance of the load balancing algorithms, it is sufficient. The remainder of this section will address the issues that need to be considered when modeling Internet traffic.

When modeling a network, the relationship of the network components and the topology needs to be considered. The components of a network basically include bridges and routers, and the links that connects them.

When a request is made, it must traverse a path, which consists of a number of links and routers to reach its destination. Once a request reaches its destination, a response must traverse a path (not necessarily the same path as the request) back to the requesting client. This asymmetry can have an adverse affect on the response time a client receives. Each link can have a different capacity than the others along the path. The link with the smallest total capacity is considered the bottleneck link. Even more critical than the capacity of the bottleneck link is the link with the smallest available capacity. The link with the largest total capacity may be handling so many packets that its available capacity is less than the available capacity of the smallest link, therefore making it the bottleneck link.

The bridges and routers can affect response times when they are busy. Two factors can affect their performance, effective throughput and buffer space. If a bridge or router is receiving packets faster than it can deliver them on another link, the packets must wait in a buffer until they can be delivered. The time in the queue is added to the over all latency of the response time. Even worse, the buffer may become full and it will ignore any incoming packets until there is room available in the buffers. This will cause even longer delays because the sender will eventually have to retransmit the packet. Also, large transmission

latencies can also cause the sender to assume that packets are lost if the acknowledgements are not received before some timeout. The server's response to timeouts is to resend the packets, resulting in duplicating the packet already sent and causing unnecessary traffic in the network.

3.4 Model Data and Distributions

This section will describe the analytical models and the associated distributions that are used for simulating Internet traffic. Two groups of workload models will be looked at, the client models and the server models. As mentioned in the previous section a model of network topology was not developed for this simulator. All factors associated with the characteristics are assumed to be constant.

3.4.1 Client Workload Models

Four analytical models are required to model the characteristics of client workloads. The characteristics that are modeled are sleep time, inactive off time, active off time and the number of embedded references in a HTTP request. A description of the characteristics and the mathematical model to generate the appropriate workload for each characteristic are described in the following sections.

3.4.1.1 Sleep Time

One of the factors in modeling network traffic is modeling the total hourly request rates made by all the clients. During each hour of the day the request rates per hour vary from low rates during the early morning hours, and the busy hours, which typically occurs

at midday. Sometimes you will see a two-humped curve representing two peak busy periods that can occur during the day. The first busy period is late morning and the other busy period is early afternoon. Figure 1 shows a typical distribution of the request rates per hour that were approximated from the following papers. The Internet distribution is approximated from data in Arlitt and Williamson (1997) and the Telnet distribution is approximated from Paxson and Floyd (1995).

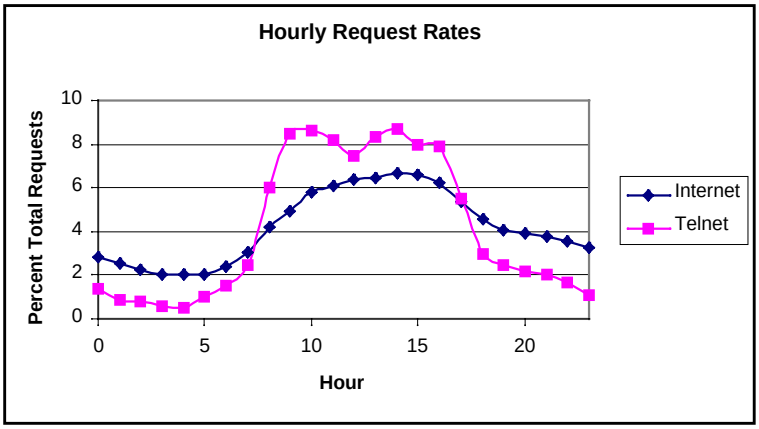

Figure 1: Hourly Request Rates

Client sleep time is the time when the client is not making any requests on the network. The characterization of client sleep time is done by defining when a client is awake and when it is asleep.

Using the data from the Internet distribution in Figure 1, a model that varies the request rate per hour can be made by changing the number of clients making requests. The percent total ranges from a low of 2% to a high of almost 8%. An easy way to model this distribution is to assume that each 1% of the total request rate is approximately one client process. Therefore it is assumed that for each set of eight clients, the distribution is fairly modeled by the number of clients equal to the percent total per hour. For example at Hour 0, approximately 3% of the total Internet requests are made, which can therefore

be represented by three clients. Approximately 7-8% of the total requests are made at Hour 14 therefore eight clients can be used to generate requests for that hour.

3.4.1.2 Inactive Off Time

Inactive off time is the time that occurs between requests. This is also referred to as client think time. Using the research from Barford and Crovella (1997) they concluded that inactive off times are modeled best with a Pareto Distribution with a shape parameter of 1.5 and a lower bound of 1.0. The inactive off time distribution is shown in Figure 2.

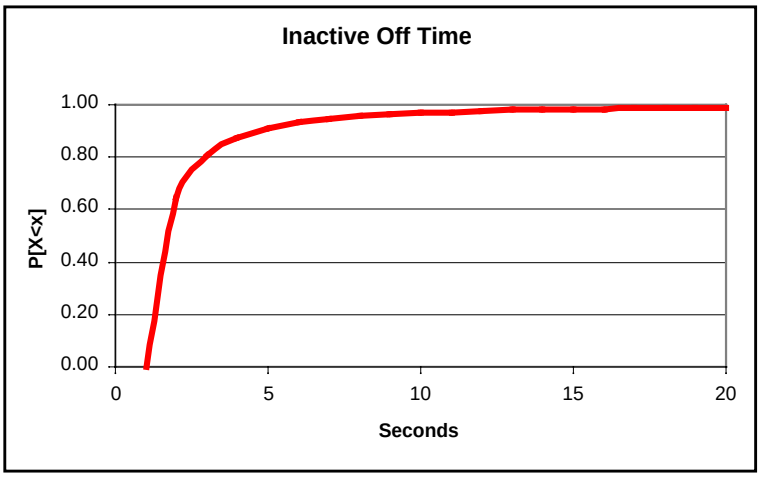

Figure 2: Inactive Off Time Distribution

3.4.1.3 Active Off Time

Active off time is the time that occurs between each embedded reference inside a HTTP request. This is the time between the completion time of the last object received and the time the client starts receiving the next object. Using the research from Barford and Crovella (1997) they concluded that active off times are modeled best with a Weibull distribution with a scale parameter of 1.46 and a shape parameter of 0.382. The active off time distribution is shown in Figure 3.

.

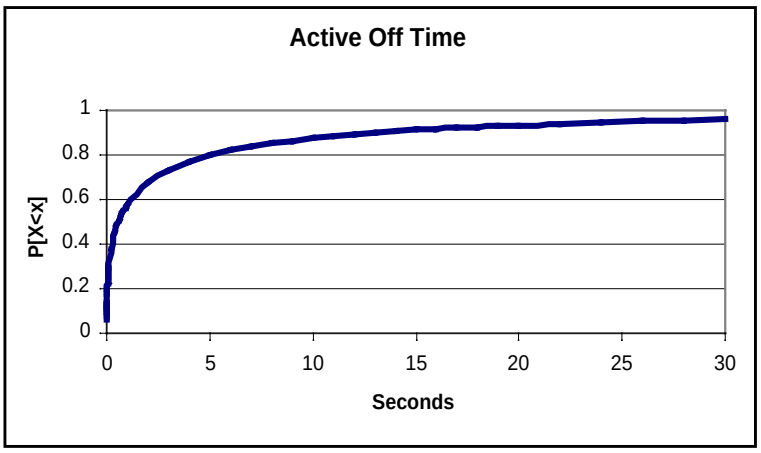

Figure 3: Active Off Time Distribution

3.4.1.4 Embedded References

When an HTTP request is made, the document requested may have one or more references to other objects. The references to theses objects are called embedded references. Barford and Crovella (1997) concluded that number of embedded references in each HTTP request is modeled with a Pareto distribution a shape parameter of 2.43 and a lower bound of 1.0. The distribution is shown in Figure 4.

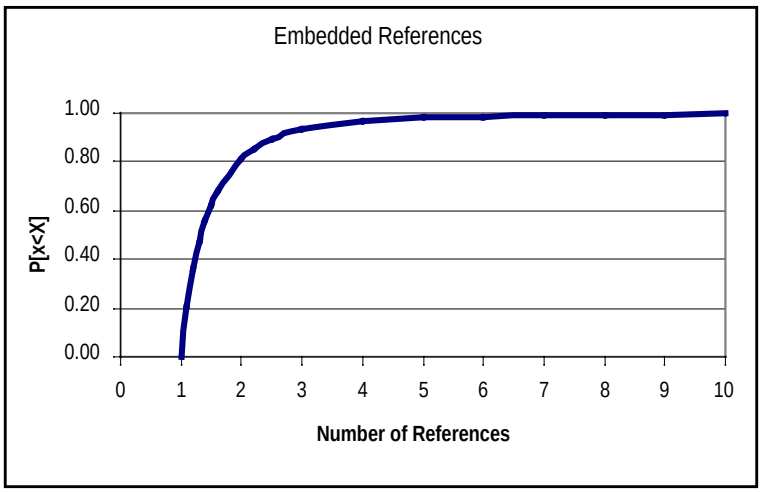

Figure 4: Distribution for the Number of Embedded References

3.4.2 Server Workload Models

Server workloads are characterized by the distribution of the files that are requested by the client. Analyzing the data on file sizes show that the majority of the files requested from Web servers are less than 10,000 bytes and a large proportion of those are less than 1,000 bytes. Table 1 shows the file size frequency distributions from various studies. Specweb99 (1999) collected empirical data by analyzing the log files from selected popular web servers at NCSA, HP, HAL Computers and a Comics Web site. Arlitt and Williamson (1997) collected empirical data from university web servers at University of Waterloo, University of Calgary, and University of Saskatchewan, and from NASA, ClarkNet and NCSA. Cunha, Bestavros, and Crovella (1995) and Crovella, Frangioso and Harchol-Balter (1999) collected empirical data from web servers at Boston University.

File Size	Specweb99 (1999)	Arlitt, Williamson (1997)	Cunha, Bestavros, Crovella (1995)	Crovella, Frangioso, Harchol-Balter (1999)
< 1 KB	35%	-	62%	23%
1-10 KB	50%	76%	31%	70%
10 - 100 KB	14%	23%	6	6%
> 100 KB	1%	<1%	1%	1%

Table 1: File Size Distribution

Table 1 fails to show the impact of files sizes greater than 100 kilobytes. In Arlitt and Williamson (1997), even though file sizes of greater than 100 kilobytes accounted for less than 1 percent of the total requests in their measurements, those files accounted for 11 percent of the total bytes transferred. Similar observations were also made in Paxson and

Floyd (1995), Cunha, Bestavros, and Crovella (1995), Barford and Crovella (1997), and Crovella, Frangiosa and Harchol-Balter (1999).

In modeling the distribution of file sizes, the distribution was broken into two distinct pieces, the body and the tail. Using the results from Barford and Crovella (1997), and Crovella, Frangiosa and Harchol-Balter (1999) the distribution body includes approximately 93 percent of all the distribution with file sizes less than 10 kilobytes. This part of the distribution is modeled with a lognormal distribution. The tail contains the remaining 7 percent of the total distribution with file sizes greater than 10 kilobytes. This part of the distribution is modeled with a Pareto distribution. The two distributions are discussed in more detail in the next two sections.

3.4.2.1 File Size Distribution - Body

The research from Barford and Crovella (1997) concluded that the body of the distribution represents the files less than 133 kilobytes. Later, Crovella, Frangiosa and Harchol-Balter (1999) concluded that the body of the distribution represents the files less than

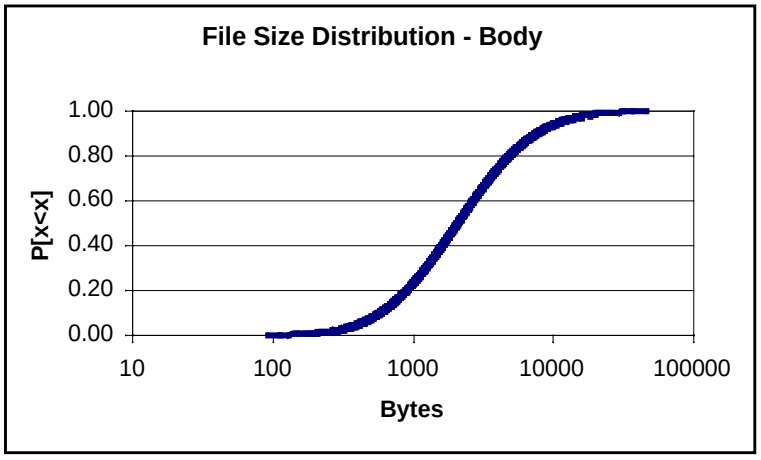

Figure 5: File Size Distribution – Body

9,020 bytes. Both studies found that the file size distribution of the body is best modeled with a lognormal distribution. Using the results from the more recent study the file size distribution body is shown in Figure 5.

3.4.2.2 File Size Distribution – Tail

Barford and Crovella (1997) concluded that the files which are larger than 133 kilobytes represent the tail of the distribution. Later, Crovella, Frangiosa and Harchol-Balter (1999) concluded that the files that are larger than 9,020 bytes represent the tail of the distribution. The distribution is represented with a Pareto distribution with a scaling factor of 1.0 and lower bound of 9,020. Figure 6 displays the distribution for the tail of the file size.

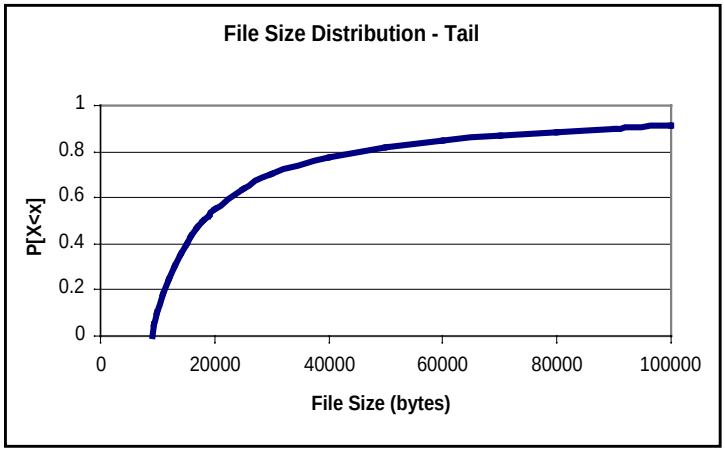

Figure 6: File Size Distribution - Tail

3.5 Self-similarity

In the preceding sections, analytical models were developed to characterize work-loads from empirical studies that are statistically self-similar. This section covers the basic concepts of statistical self-similarity.

Self-similarity can be best described with the concept of fractals. Fractals have the characteristics of looking the same regardless of the size scale that the fractal is being viewed. The empirical studies done on Internet traffic has concluded that the variation in traffic characteristics shows the same behavior regardless of the time scale that the data is viewed. The variation of the traffic workloads is cause by the bursty behavior of request inter-arrival times, variation of file sizes, file caching, and network performance problems related to congestion and bottleneck bandwidth. The burstiness looks the same over varying time scales and it has no natural length of time. In fact the variation may be infinite.

Statistically, self-similarity has three properties, long range dependence, slowly decaying variance, and it is defined by a power law. A process has long range dependence if its autocorrelation function *r(k)* is non-summable, that is $\sum r(k) = \infty$. A slowly decaying variance implies that the variance of the sample mean decreases more slowly than the reciprocal of the sample size.

For detailed information on the self-similarity found in network traffic, see Leland, Taqqu, Willinger and Wilson (1994), Paxson and Floyd (1995), Crovella and Bestavros (1995), Park, Kim and Crovella (1996, 1997), Willinger, Taqqu, Sherman and Wilson (1997). In Chapter 4 there is verification of self-similarity generated by the system simulator.

CHAPTER IV

NETWORK SIMULATOR

The network simulator designed for this thesis is implemented using a simple model to test the algorithms being researched. The main focus of this thesis is to research methods of estimating server workloads and not primarily bottleneck analysis. Therefore the simulator has been designed for this purpose. Assuming that bottlenecks occur somewhere within the network topology, the network topology is generalized by considering that all throughput is constant. Therefore, the simulator has been simplified to include all network delays within the server response time calculations.

The model in the simulator consists of four main entities, a client, a server, a load balancing manager and a request. The entities interact with each other in the following manner. The clients make periodic requests for documents of unknown sizes and wait for a response to each request. Each request is submitted to the client's load balancing manager for selection of a server. Then the client's load balancing manager selects the server that will be used to process the request and forwards the request to the selected server. When the server receives the request it is placed into a queue (First Come First Served). The request waits in the queue until a client connection becomes available. When a connection is available the server processes the request at the head of the queue and sends the request to the requesting client.

There are two different classes of entities in the simulator, simulation entities and simulator entities. The simulation entities are the entities that are being modeled for this study. The simulator entities are the entities that allow the simulator to function and provide the actions on the simulation entities. A detailed description of each entity is provided in the following sections.

4.1 Simulation Entities

This section describes the entities that are modeled in the simulator and the entity attributes. The entities that are modeled are the client, server, load balance manager and request. A description of each entity is in the following sections.

4.1.1 Clients

The client entity makes periodic requests for an object from a group of replicated servers. The client has the following primary attributes, a unique identifier, the client's load balancing manager, and the current request. The client also has the following statistical attributes, the total request count, the total execution time of all the requests and the total number of bytes transferred. Finally the client has attributes that affect its workload characterization, these attribute are wakeup time, sleep time, and a uniform random number generator. The client request rate was discussed in the previous chapter on Network Traffic Modeling.

4.1.2 Load Balance Managers

The load balance manager selects a server for processing a client request. The load balance manager has the following attributes for tracking the selections it makes. The load balance manager attributes are a unique identifier, a server count (the total number of servers the load balance manager can select from) and an array that stores the current number of request submitted to each server. The load balance manager selects a server for processing request based upon an algorithm that is chosen for the manager to use. The algorithms are discussed later in this chapter.

4.1.3 Servers

Servers have the following primary attributes, a unique identifier, a request count, the maximum client connection count, the processor power[1], the connection bandwidth, a request queue and a table for storing the file sizes of the distribution body.

The request queue is a First Come First Served queue for placing incoming requests. The requests wait for service in the queue until they reach the head of the queue and a client connection becomes available. When a connection becomes available, the server removes the request at the head of the queue and processes the request. When the server first receives a request, it selects a file size for the request. The file sizes are stored in a table of file size distributions used for determining the request size. This table is probability distribution of potential file sizes and the distribution was discussed in detail in the last chapter on Network Traffic Modeling. The server also keeps a set of statistical

1. Processor power was designed into the system to provide for a means to vary the performance of each processor relative to each other. In the experiments in this thesis the servers, are all identically replicated, therefore the value was set to 1.

attributes for tracking the total number of requests processed, the total bytes transferred and queue statistics (wait time, and queue length).

4.1.4 Requests

The request entity contains the following attributes, a unique identifier, the requesting client, the assigned server, the request size, the request time, the server receive time, the server start time and the server complete time. With these time stamps, the response time, queue wait time and execution time can be calculated for each request.

Each request proceeds through a series of events where state of the each entity in the system is changed appropriately as each event occurs. The event types that occur on each request are, **New Request**, **Request Server**, **Document Request**, **Document Send**, and **Document Done**.

The first event for a request is a **New Request** event. An idle client will create this event with the event time base on when the next client request interval will occur. The client request interval is determined by the inactive off time or client think time models. When the **New Request** event occurs, a **Request Server** event is scheduled by placing the event back into the event queue at the current event time. The **Request Server** event is executed by requesting the client's load balancing manager to select a server. Once a server is selected, a **Document Request** event is scheduled at the current event time. The **Document Request** event places the request into the selected server's request queue. If the server is idle (a connection is available), then a **Document Send** event is immediately scheduled at the current time plus a delay time based upon the server workload, or else the request will wait in the server queue until the server executes a **Document Done** event.

The **Document Done** event updates the client statistics and the server statistics and if the server request queue has a request waiting at the head of the queue, then it will remove the request and schedule a **Document Send** event for that request. Figure 7 shows the states a single event flows through.

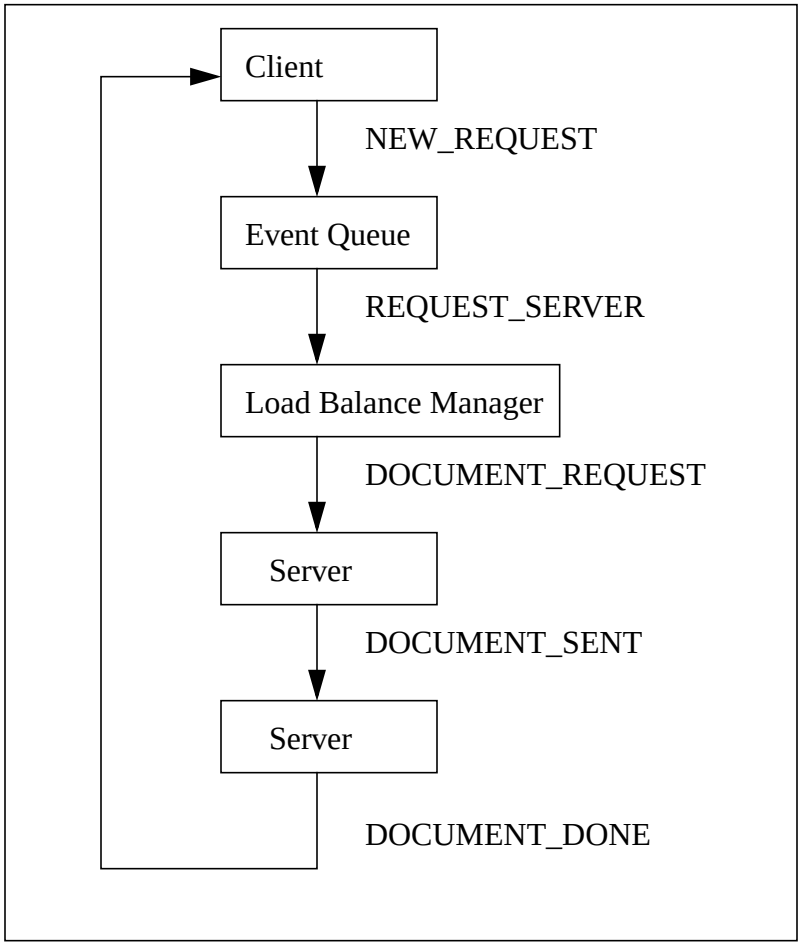

Figure 7: Request Event Types

4.2 Simulator Entities

The simulator also contains the following entities, a system clock, an event, and an event queue.

The simulator is a discrete event simulator, which uses an event scan time management technique. The simulator has a system clock which keeps the current time, an event object that schedules and executes events, and an event queue. The entities and their attributes are discussed in the following sections.

4.2.1 System Clock

The system clock maintains the current time of the system. It has one attribute, the current system time and two methods, a method to set the system time and a method to return the system time. As previously mentioned the simulator uses an event scan technique, which is an optimistic technique for advancing the system time. This is done by analyzing the event time for all the events in the event queue and advancing the time of the system clock to the minimum event time of all the events analyzed. Since the event queue is a priority queue, this is the event at the head of the queue. If the event time of next event to be executed is less than the current system time then an error occurred because the simulator is not designed to do rollbacks.

4.2.2 Events

An event causes some action to occur on the simulation entities. When an event is executed, it causes the system state to change by updating the state of the entities that are being acted upon. Events have the following attributes: the event time, the event type, sender, receiver, and a request.

The event time is the system time that an event will occur. The time may be at the current time or at some time in the future. The event type is the one of the five types

already discussed in the section on requests. The sender and receiver are the unique identifiers that represent a client, server or load balance manager depending upon the context of the event type. The request is a client request that is in some state based upon event type.

4.2.3 Event Queue

The event queue is a priority queue that inserts new events based on the time that the new event is to occur. If the queue already has an event scheduled for that event time, then each new event is scheduled first come first serve for that specific event time.

4.3 Algorithms

The simulator is designed to exercise the effective performance of five different algorithms that distribute the workload among a set of processors. The workload is generated using the synthetic models discussed in Chapter 3 and the simulations generate statistics on each algorithm.

The algorithms are separated into two types, stateless and state-based.

4.3.1 Stateless Server Selection

The stateless algorithms select a processing element with out any knowledge or consideration of the system state, or the workload of the selected processing element. There are two algorithms that are studied, round robin and random.

4.3.1.1 Round Robin Server Selection

The first algorithm is a round robin method. This method selects a server in linear order from a list of servers. When the end of the list is reached, it starts over at the first server in the list. This is similar to Round Robin DNS, see Emery (2000) for a study using this method.

4.3.1.2 Random Server Selection

The second method selects a server randomly using a uniform pseudo-random variable. The random variable is modulated with the number of servers, which selects a server identifier within the range of all the servers.

4.3.2 State-based Server Selection

The next set of algorithms select a processing element based upon its knowledge of the system state. This knowledge may be global or local. The age of the knowledge may also vary. Global knowledge of the system state implies that the algorithm considers the state of the whole system when selecting a processing element. Local knowledge of the system state is more limited. It implies the algorithm considers a subset of the system state when selecting a processing element.

There are three algorithms studied in this thesis, greedy, random subset, and a stochastic method.

4.3.2.1 Greedy Server Selection

Greedy server selection implies that the processing element with the lightest workload is selected. This would include not only server performance but also the best network path. Intuitively, this is obviously the most preferred method to use for selecting a processor, however for the algorithm to select the best possible processor implies it must make a choice based upon the state of the whole system at the time the selection decision is being made. This may not be possible.

The algorithm chooses a server by scanning the complete server list for the server with the lowest workload. Ties favor the server that is positioned closer to the beginning of the list. For example when the simulation selects the first server to use, it will use the server at the beginning of the list, and the next selection will select the second server in the list if the first server is still busy, and so forth, until the system reaches its full operating level. As each request is serviced and completed by the servers, the workload of each server is adjusted accordingly.

4.3.2.2 Random Subset Server Selection

Selecting a server with a random subset is a form of greedy algorithm. First the algorithm selects two or more servers randomly from the list of servers and then it selects the server with the lowest workload from the servers selected. Unlike the random selection technique discussed in the previous section that was stateless, this technique requires that the workload of each server be known in order to be able to select the best performer of the randomly chosen candidates. Mitzenmacher (1997) and Dahlin (1998) both con-

cluded that this method performs well over a large range of systems. For the simulations in this study, a subset of two servers is chosen

4.3.2.3 Stochastic Server Selection

A stochastic selection process selects a server based on a probability that the selected server has an expected load lighter than the expected load of the other servers in the system based upon the current distribution of server workloads. The lighter the server workload the higher the probability that the server will be chosen. All servers have a chance of being selected. The purpose in attempting to predict server loads is to minimize the network and server overhead by probing servers or having the servers send messages to load balance agents, or having load balancing agents sending data between other load balancing agents. The load balancing agents will at sometime need to receive data regarding the behavior of the servers it is responsible for selecting, but the overhead will be much lower.

4.4 Model Validation and Verification

The simulator model is validated with the comparison of the output data of the simulator program and the mathematical model. This section will discuss the verification of the distributions for file size, request inter-arrival rate and a section that provides proof that the simulator generates statistically self-similar workloads.

4.4.1 File Size Distributions

Figure 8 shows the verification of the file size distribution for the distribution body. The verification was done from an output file with 3,295,407 requests. A Chi-squared verification yielded a result of less than 0.001, proving that simulation model is generating the expected distribution. The result of the Chi-squared test is small because the data is generated with the lognormal distribution used as a data model. This is more a validation of the Uniform random number generator than the model, since the model is solely dependent upon the uniform distribution of random variables generating the distribution values.

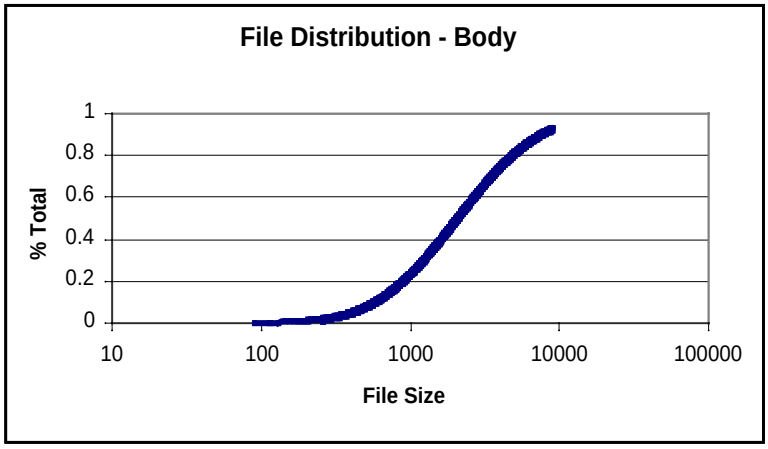

Figure 8: Verification of the Body of the File Size Distribution

Verification of the tail of the file distribution is shown in Figure 9. The Chi-squared test result is less than 0.001 proving that simulation model is generating the expected distribution. The small result from the Chi-squared test has the same dependencies as the file size distributions for the body except that here a Pareto distribution is used. Its accuracy also has the same dependency with the random variables generated.

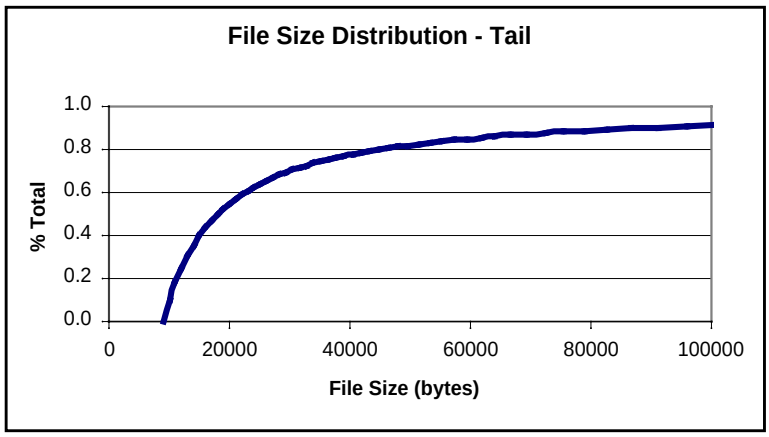

Figure 9: Verification of the Tail of the File Size Distribution

4.4.2 Request Inter-arrival Rates

Four distributions were used to generate request inter-arrival rates. The distributions modeled Hourly Request Rates (client awake time), Active Off Time, Inactive Off Time, and Embedded References. The verification of each distribution is discussed int he following sections.

4.4.2.1 Hourly Request Rates

Figure 10 shows the Hourly Request Rate distribution with plots of the approximated values and measured values. The Chi-squared verification of the measured values results in a value of 2.87. This value is small enough to verify that the ratio of client processes that are awake during each hour of the simulation are resonably close enough to the approximated data to meet the characteristics of the behavior that is being modeled.

4.4.2.2 Active Off Time

Figure 11 shows the distribution for Active Off Time. The Chi-squared test result is less than 0.001 proving that the simulation model is generating the expected distribution.

4.4.2.3 Inactive Off Time

Figure 12 shows the distribution for Inactive Off Time. The Chi-squared test result is less than 0.001 proving that simulation model is generating the expected distribution.

4.4.2.4 Number of Embedded References

Figure 13 shows the distribution for Embedded References. The Chi-squared test result is less than 0.001 proving that simulation model is generating the expected distribution. In the figure, two plots are shown. One plot is the mathematical distribution and the other is the measure output. Since the number of embedded references has to be an integer value the distribution is skewed some because of the truncation of the random variables from decimal values to integer values.

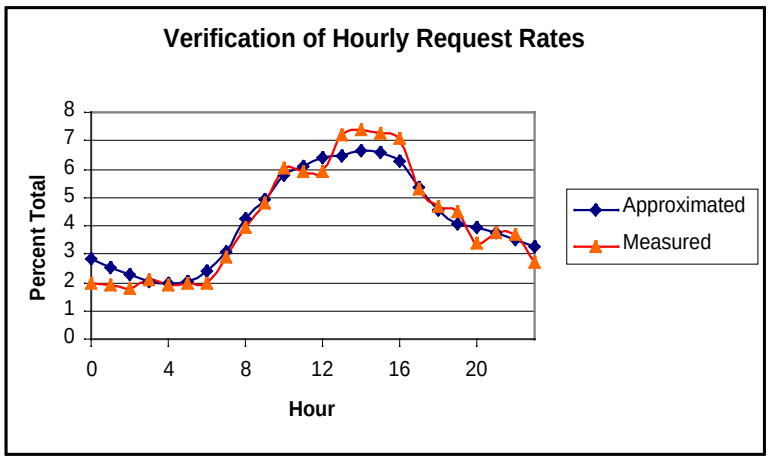

Figure 10: Hourly Request Rates

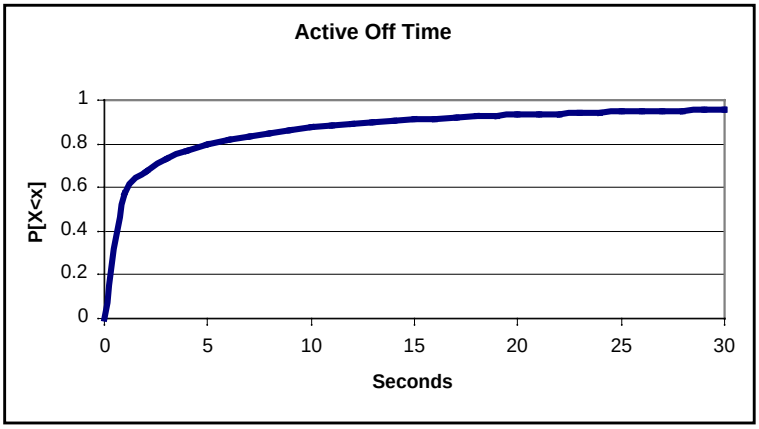

Figure 11: Active Off Time

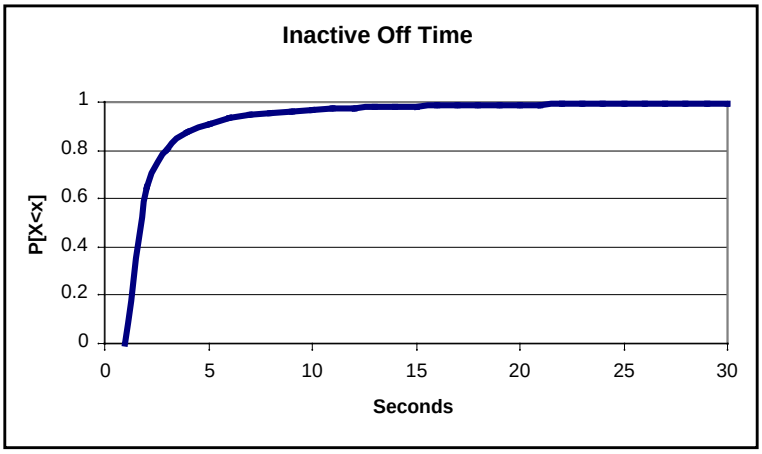

Figure 12: Inactive Off Time

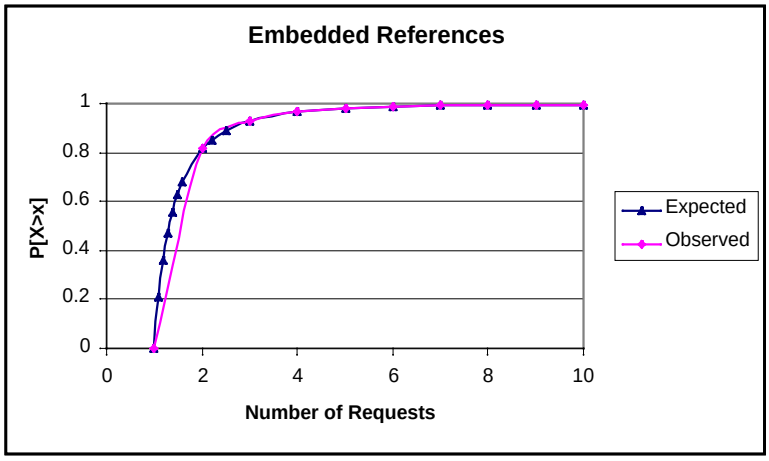

Figure 13: Embedded References

4.4.3 Self-similarity

This section analyzes the system workloads generated by the simulator for statistical self-similarity. The easiest method for verifying self-similarity is by visual inspection. Figure 14 shows the request rate for four different time scales, 1 second, 10 seconds, 100 seconds and 1000 seconds. By observation, each plot exhibits the high variability expected for each time scale, which indicates that the simulator is generating workloads that are self-similar.

There are several statistical methods for determining the level of self-similarity of a distribution. These methods are the variance-time plot, the rescaled range (R/S) plot, the periodogram, and the Whittle estimator. The first three methods are graphical plots for showing the degree of self-similarity. See Leland, Taqqu, Willinger and Wilson (1994), Paxson and Floyd (1995), Crovella and Bestavros (1995), and Park, Kim and Crovella (1996) for examples of these methods.

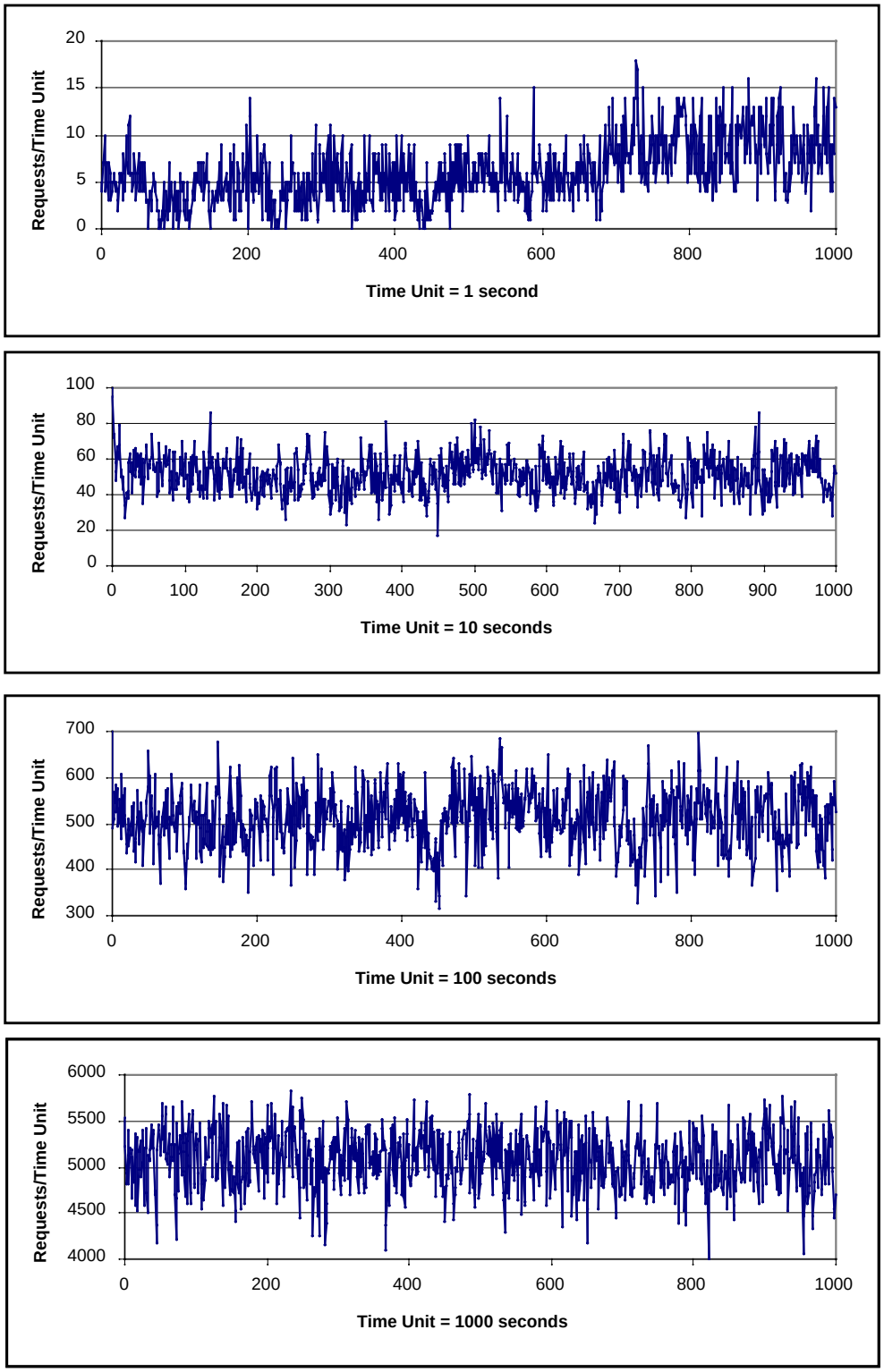

Figure 14: Request Rate per Time Scale for 1, 10, 100 and 1000 seconds

The variance-time plot calculates the degree of self-similarity by comparing the slope of the data to the slop of 1/M, where M is the scale being analyzed. If the slope of the plot is greater than –1 then the data is considered to have self-similar characteristics. A parameter call the Hurst parameter is calculated from the slope of the plot with $H = 1 - \hat{\beta}/2$, where β represents the slope of the plot. The data series is considered self-similar when $1/2 < H < 1$. As $H \to 1$, the degree of self-similarity and long-range dependence increases.

In Figure 15, the data series for the simulated data is plotted using variance-time plot. The plot has a slope -0.61, and the resulting value for $H = 0.7$. Therefore it can be concluded that the experimental data is self-similar.

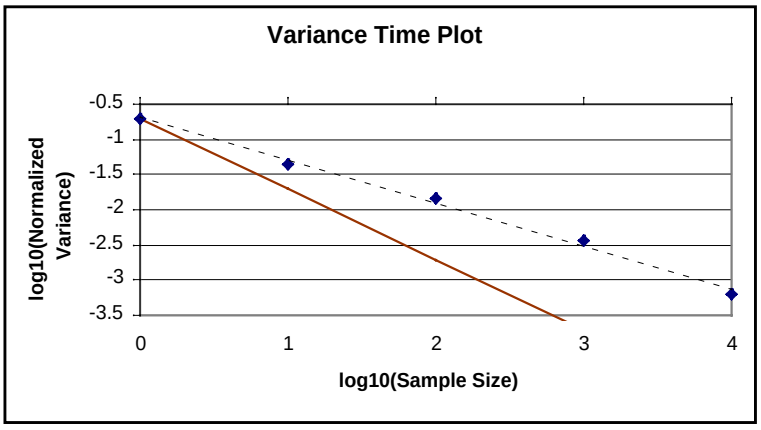

Figure 15: Variance Time Plot

CHAPTER V

EXPERIMENTAL DESIGN

This chapter will describe the experimental design. The experimental design is concerned with identifying the main factors that need to be imputed into the simulations and the different levels that each factor should characterize. These issues are discussed in the following sections.

5.1 Experimental Factors

The factors that were identified in the experimental design for the simulation experiments for evaluating the algorithms in this thesis are listed here:

1. The number of clients[1].
2. The number of load balance managers.
3. The number of servers.
4. The number of client connections for each server.

5.2 Factor Values

The four factors can multiply into a large number of experiments. Initially the design considered a configuration of five clients, three load balance managers, three serv-

1. Barford and Crovella (1997) use the term User Equivalents, which roughly corresponds to the workload created by a population of a known number of users. Each User Equivalent, which represents a population of users, as an ON/OFF process that makes requests and then is idle for some period of time.

ers and two connections. This will require the analysis of 90 experiments for each of the five algorithms for a total of 450 experiments. However it was discovered that two additional client configurations would be useful, since the experimental data showed that some of the algorithms performed better at higher loads that it did a low loads. So, additional experiments were performed to find the intersection of the algorithms. This resulted in the performance of 126 experiments for each algorithm for a total of 630 experiments. Finally, a set of adversarial experiments was performed with combinations of random stateless load balance manager working with the three state-base algorithms using only the two configurations with eight servers. This required the performance of an additional 280 experiments.

The experiments that were chosen were more as a matter of personal choice for comparing the performance of the various algorithms, as opposed to attempting to achieve any particular result. Also, there is a huge array of combinations that can be performed; therefore, a representative set of experiments were performed to satisfy the goals of this thesis.

Client workloads were varied with seven different workloads. The workloads were modeled with 8, 16, 32, 64, 128, 256 and 512 clients generating requests for files.

The servers were configured in combinations of 2, 4, or 8 servers processing requests for the clients. The server configurations were also configured with 1 and 4 simultaneous connections. The configuration with one connection is intended to cause server bottlenecks for collecting a baseline set of performance data. The servers configured with four connections are designed to avoid single connection bottlenecks but not to have so many connections that it doesn’t force some jobs to wait in the server queues.

Finally, the load balance managers were done in two sets of experiments. The first set of experiments had the same type of load balance manager in configurations 1, 2 and 4. This allowed for the analysis of the load balance managers with global versus local system state information. The second set of experiments included adversarial combinations of the state-based load balance managers being tested with random placement (stateless) load balance managers. The experiments are call adversarial because the random placement load balance manager makes selections without regard to the system state. This makes the environment for the state-based load balance managers less cooperative than the environment of the first set of experiments. Experiments with two and four load balance managers were done with combinations of two load balance managers consisting of one load balance manager and one adversary, and four load balance managers, consisting of one load balance manager with three adversaries, two load balance managers with two adversaries, and three load balance managers with one adversary. These experiments will allow for the analysis of how each algorithm performs in various hostile conditions.

CHAPTER VI

SIMULATION EXPERIMENTS AND RESULTS

The simulation experiments and the results are presented here in this chapter. First the simulation experiments will be discussed and then the final part of this chapter will discuss the experimental result.

6.1 Simulation Experiments

Experiments were performed using the simulator by varying the factors discussed in the previous Chapter 5. Each series of experiments is shown in the following sections.

6.1.1 Series 1 – Servers with One Client Connection

A series of experiments was performed with a single client connection per server.

- For each Load Balance Algorithm {round robin, random, greedy, subset, stochastic}
    - For each set of {2, 4, 8} servers with a single connection
        - For each set {1, 2, 4} of load balance managers
            - Run the simulation with each set {8, 16, 32, 64, 128, 256, 512} of clients

6.1.2 Series 2 - Servers with Four Client Connections

A series of experiments was performed with four client connections per server.

- For each Load Balance Algorithm {round robin, random, greedy, subset, stochastic}.
    - For each set of {2, 4, 8} servers with four client connections
        - For each set {1, 2, 4} of load balance managers
            - Run the simulation with each set {8, 16, 32, 64, 128, 256, 512} clients.

6.1.3 Series 3 - Servers with One Client Connection and Adversaries

A series of experiments was performed with a single client connection per server and a combination of load balance managers with configurations of random and one of the following state-based algorithms, greedy, subset and stochastic. The ratio of state-based managers to greedy managers is denoted as <total load balance managers, number of adversaries>.

- For each Load Balance Algorithm {greedy, subset, stochastic}
    - For each set of {2, 4, 8} servers with a single client connection
        - For each set {<2.1>, <4.1>, <4.2>, <4.3>} of load balance managers
            - Run the simulation with each set {8, 16, 32, 64, 128, 256, 512} of clients.

6.1.4 Series 4 - Servers with Four Client Connections and Adversaries

A series of experiments was performed with a single client connection per server and a combination of load balance managers with configurations of random and one of the following state-based algorithms, greedy, subset and stochastic.

- For each Load Balance Algorithm {greedy, subset, stochastic}
    - For each set of {2, 4, 8} servers with a four client connections
        - For each set {<2.1>, <4.1>, <4.2>, <4.2>} of load balance managers
            - Run the simulation with each set {8, 16, 32, 64, 128, 256, 512} of clients.

6.2 Experimental Results

The initial analysis of the simulation data showed that the results of the experiments between the different server configurations were very similar. The difference in magnitude of the server utilization was inversely proportional to the magnitude of difference in the number of servers. Therefore, since the results of the experiments are so close for each server group, only one set of the data will be presented. The data set that was chosen is the data set for eight servers.

1 Load Balance Manager

Clients	Round Robin		Random		Greedy		Subset		Stochastic	
	Util	Resp	Util	Resp	Util	Resp	Util	Resp	Util	Resp
8	0.11	2.66	0.11	2.76	0.16	0.10	0.14	0.81	0.12	2.19
16	0.17	5.72	0.16	6.03	0.31	0.27	0.24	2.01	0.19	4.10
32	0.23	11.29	0.22	11.62	0.55	1.11	0.37	4.55	0.31	6.86
64	0.28	21.96	0.28	22.25	0.80	3.80	0.51	9.61	0.46	11.42
128	0.34	41.13	0.34	41.34	0.95	10.68	0.63	19.15	0.64	18.94
256	0.40	73.41	0.40	73.86	1.00	25.96	0.74	37.00	0.80	33.78
512	0.48	127.40	0.47	128.70	1.00	58.00	0.84	69.80	0.92	63.70

2 Load Balance Managers

Clients	Round Robin		Random		Greedy		Subset		Stochastic	
	Util	Resp	Util	Resp	Util	Resp	Util	Resp	Util	Resp
8	0.11	2.76	0.11	2.76	0.14	0.94	0.13	1.28	0.11	2.49
16	0.17	5.83	0.16	6.03	0.27	1.08	0.23	2.42	0.18	4.74
32	0.23	11.39	0.22	11.62	0.50	1.76	0.36	4.93	0.28	8.08
64	0.28	22.08	0.28	22.25	0.77	4.26	0.50	9.95	0.41	13.21
128	0.34	41.13	0.34	41.34	0.94	10.95	0.63	19.43	0.58	21.27
256	0.40	73.86	0.40	73.86	1.00	25.99	0.74	37.23	0.76	36.07
512	0.48	127.50	0.47	128.70	1.00	58.00	0.84	70.00	0.90	65.50

4 Load Balance Managers

Clients	Round Robin		Random		Greedy		Subset		Stochastic	
	Util	Resp	Util	Resp	Util	Resp	Util	Resp	Util	Resp
8	0.11	2.97	0.11	2.76	0.12	2.07	0.12	1.91	0.11	2.67
16	0.16	6.05	0.16	6.03	0.23	2.62	0.21	3.18	0.17	5.40
32	0.22	11.67	0.22	11.62	0.43	3.11	0.34	5.58	0.26	9.35
64	0.28	22.21	0.28	22.25	0.71	5.18	0.48	10.52	0.37	15.34
128	0.34	41.19	0.34	41.34	0.91	11.42	0.61	19.96	0.52	24.36
256	0.40	73.46	0.40	73.86	1.00	26.09	0.73	37.74	0.70	39.36
512	0.48	127.10	0.47	128.80	1.00	58.00	0.84	70.30	0.86	68.70

Table 2: Experimental Results - Servers with one connection

6.2.1 Servers with One Connection

First the data from the single connection servers will be analyzed. Table 2 shows the experimental results for servers with one connection. Under each load balance algorithm in the heading is two columns. The first column is server utilization and the second column is the response time. Three sets of experimental data are shown. The sets

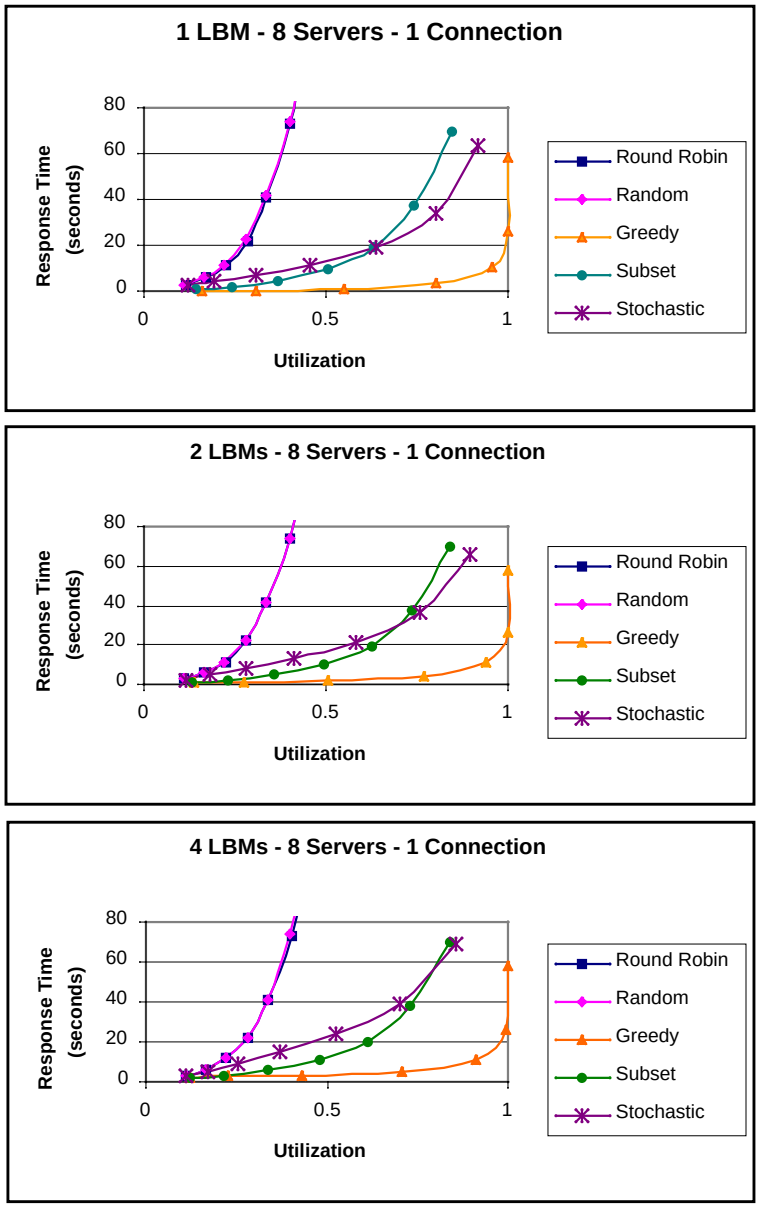

Figure 16: Load Balance Algorithm Performance – Single Connection

represent the experiments with one load balance manager, two load balance managers and four load balance managers.

As mentioned earlier in the experimental design, the single connection servers were designed to cause bottlenecks within the server causing requests to wait in the server queue. This was done to observe the impact of server bottlenecks on client response times. The results of the three server configurations are shown in Figure 16. In analyzing the data, the stateless algorithms (round robin and random) performed the worst. The two algorithms have nearly the same performance and appear as the same line in the plots. The greedy algorithm performed the best. This should be expected since it will always choose the server with the lightest load. The random subset algorithm and stochastic algorithm have mixed results when they are compared. Both of the algorithms performed better that the stateless algorithms and not a well as the greedy algorithm. In lower server workloads the random subset algorithm outperforms the stochastic algorithm, but with higher server loads the stochastic algorithm performs better. By analyzing the plots, this occurs between 60%-80% server utilization.

Figure 17 shows the comparison of the number of load balance managers for each algorithm. The round robin, random, greedy and subset algorithms have almost the same performance regardless of the number of load balance managers placing requests. The stateless algorithms, round robin and random, do not keep track of the system state therefore it appears that they are not affected by lack of system state as the system adds more load balance managers. The greedy and subset algorithm the difference between 1, 2 and 4 load balance managers is almost the same but the variation is a little greater than the stateless techniques. The good performance by the greedy and subset algorithms is

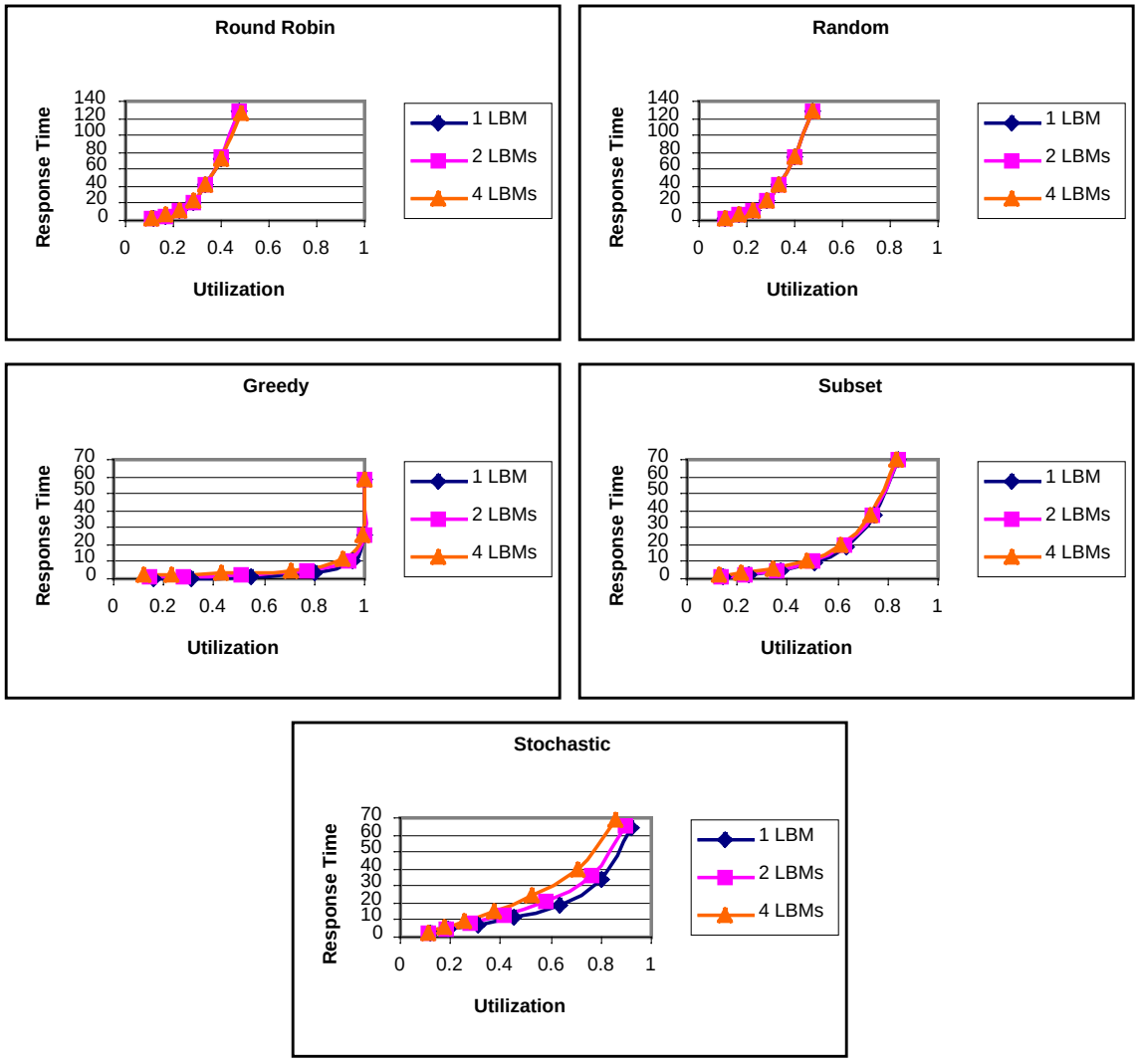

Figure 17: Load Balance Algorithm System Performance

probably because they attempt to select the server with the lowest load, therefore minimizing the affects of local knowledge versus global knowledge as the number of load balance managers changes. The stochastic algorithm appears to be the most sensitive to changes in the number of load balance managers, with higher variations in response time as the number of load balance managers increase. The cause of this is because the stochas-

tic algorithm is a combination of random and greedy selection techniques, which selects a server based upon a probability distribution of the server workloads.

With the exception of the stochastic algorithm, the variation of each algorithm appears to not be significant. This implies that for this set of experiments the difference in system performance between the load balance managers having global workload knowledge and local workload knowledge was very little. Intuitively, it would seem that the more load balance managers the system has, the lower the system performance, something that this set of experiments did not show.

1 Load Balance Manager

Clients	Round Robin		Random		Greedy		Subset		Stochastic	
	Util	Resp	Util	Resp	Util	Resp	Util	Resp	Util	Resp
8	0.04	0.11	0.04	0.12	0.04	0.10	0.04	0.10	0.04	0.11
16	0.08	0.12	0.08	0.14	0.08	0.10	0.08	0.11	0.08	0.12
32	0.16	0.17	0.16	0.20	0.16	0.12	0.16	0.14	0.16	0.15
64	0.30	0.55	0.30	0.63	0.32	0.18	0.32	0.21	0.32	0.28
128	0.47	2.49	0.47	2.60	0.61	0.61	0.59	0.79	0.57	1.04
256	0.60	7.45	0.59	7.55	0.86	3.46	0.84	3.66	0.82	3.88
512	0.67	18.04	0.67	18.18	1.00	10.32	0.98	10.60	0.97	10.82

2 Load Balance Managers

Clients	Round Robin		Random		Greedy		Subset		Stochastic	
	Util	Resp	Util	Resp	Util	Resp	Util	Resp	Util	Resp
8	0.04	0.11	0.04	0.12	0.04	0.11	0.04	0.11	0.04	0.11
16	0.08	0.13	0.08	0.14	0.08	0.12	0.08	0.12	0.08	0.13
32	0.16	0.18	0.16	0.20	0.16	0.14	0.16	0.15	0.16	0.16
64	0.30	0.57	0.30	0.63	0.32	0.20	0.32	0.24	0.32	0.31
128	0.47	2.50	0.47	2.60	0.61	0.64	0.59	0.87	0.57	1.11
256	0.60	7.37	0.60	7.38	0.86	3.38	0.83	3.63	0.81	3.85
512	0.68	17.61	0.67	17.84	0.99	10.12	0.97	10.47	0.96	10.67

4 Load Balance Managers

Clients	Round Robin		Random		Greedy		Subset		Stochastic	
	Util	Resp	Util	Resp	Util	Resp	Util	Resp	Util	Resp
8	0.04	0.11	0.04	0.12	0.04	0.11	0.04	0.11	0.04	0.11
16	0.08	0.13	0.08	0.14	0.08	0.13	0.08	0.13	0.08	0.13
32	0.16	0.19	0.16	0.20	0.16	0.16	0.16	0.16	0.16	0.18
64	0.30	0.57	0.30	0.63	0.32	0.22	0.32	0.28	0.31	0.36
128	0.47	2.52	0.47	2.60	0.60	0.70	0.58	0.97	0.56	1.23
256	0.60	7.20	0.60	7.38	0.85	3.41	0.82	3.74	0.80	3.99
512	0.67	17.67	0.67	17.84	0.99	10.16	0.97	10.58	0.95	10.80

Table 3: Experimental Results - Servers with four connections

6.2.2 Servers with Four Connections

Next we will look at the results for eight servers with four client connections. Table 3 shows the experimental results for servers with four connections. Under each load balance algorithm in the heading is two columns. The first column is server utilization and the second column is the response time. Three sets of experimental data are shown. The sets represent the experiments with one load balance manager, two load balance managers and four load balance managers.

This configuration is less susceptible to server bottlenecks, which is evident by observing the lower variation in the results. As with the single connection discipline the round robin and random algorithms perform the same. The plots of the three state-based

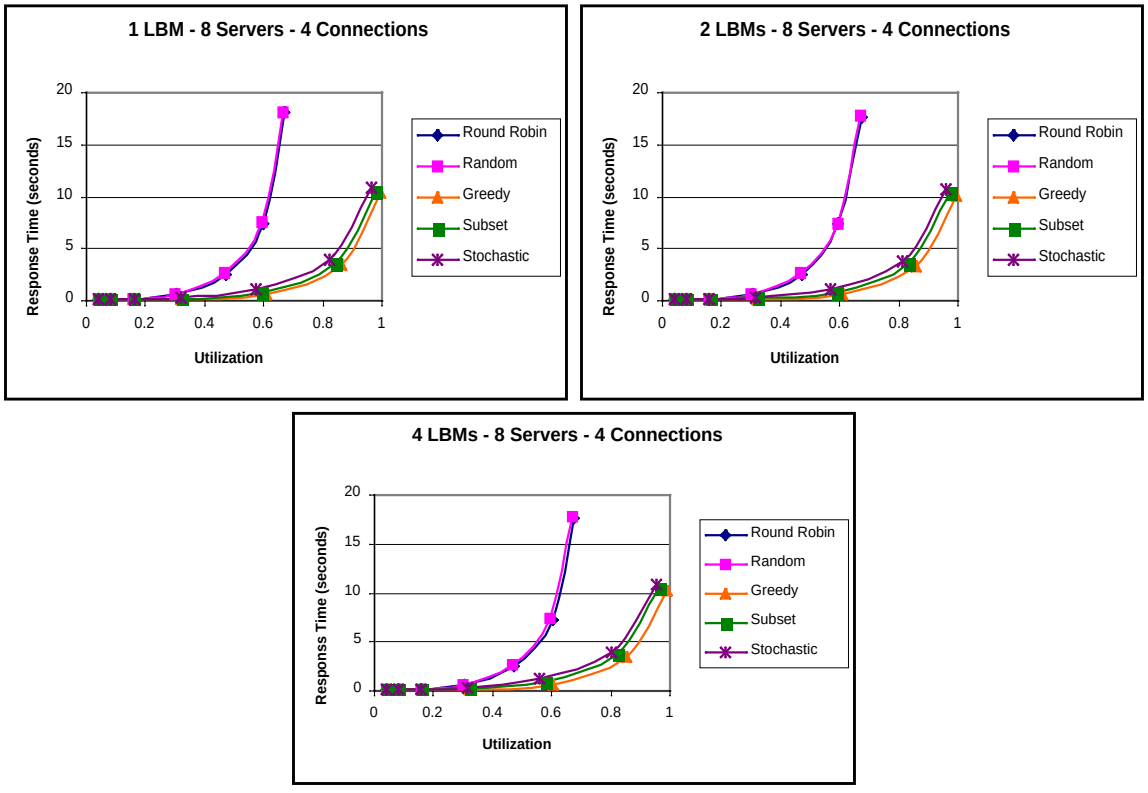

Figure 18: Load Balance Algorithm Performance - Four Connections

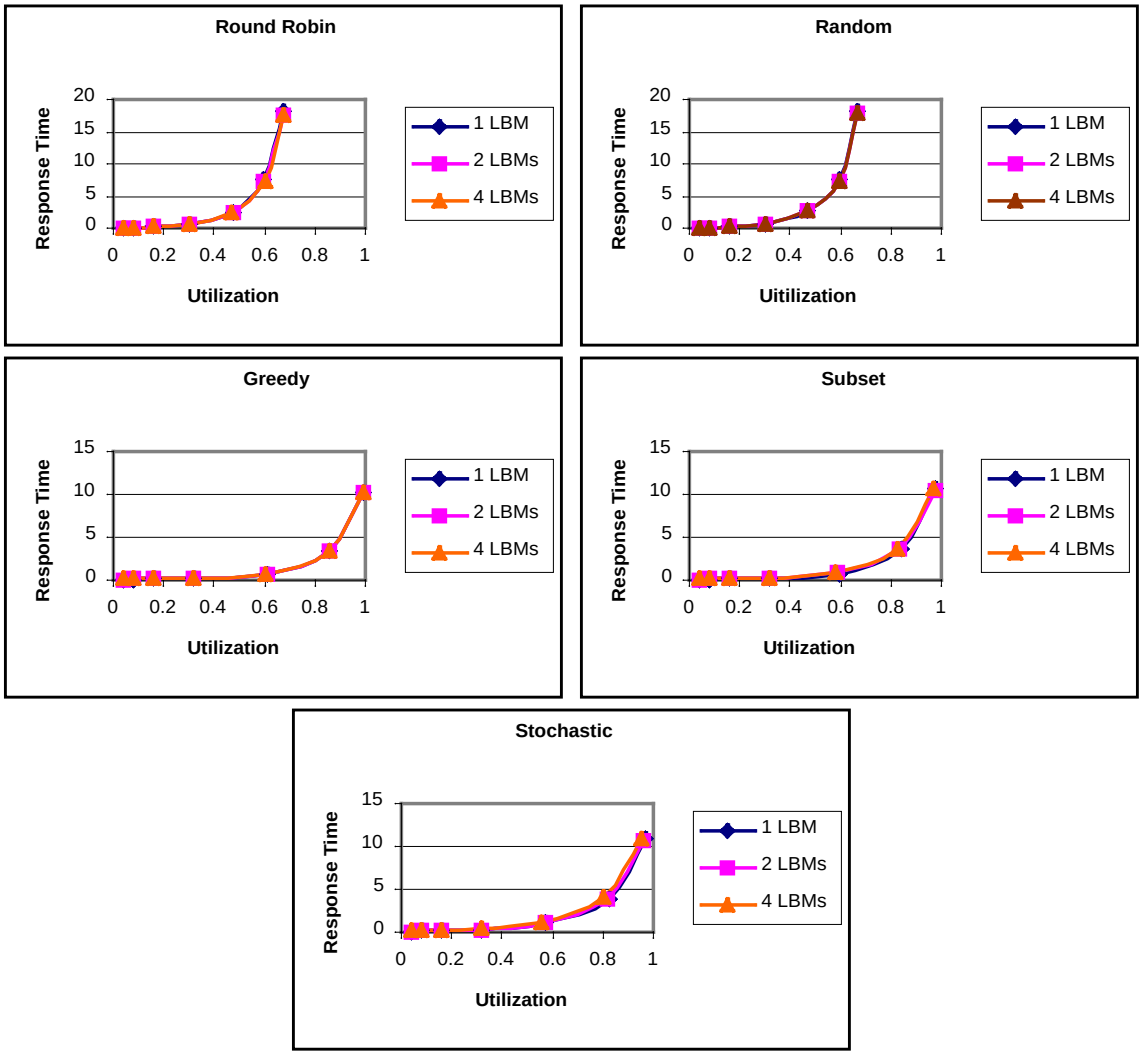

Figure 19: Load Balance Algorithm System Performance

algorithms are nearly parallel at all workloads. The greedy algorithm has the best performance, next is the random subset and then the stochastic method. Figure 18 shows the plots for each configuration.

The analysis of global and local load balance manager performance is shown in Figure 19. The plots show very little variance for each configuration of load balance managers. As with the single connection results, round robin, random, greedy and subset algorithms show almost no variation at each level. The stochastic algorithms shows some

variation in performance, but the variation appears to be insignificant. With the exception of the stochastic algorithm the results regarding global versus local knowledge of the system state the results are basically the same as the results of the single connection server experiments.

Clients	Greedy		Subset		Stochastic	
No Adversary						
	Util	Resp	Util	Resp	Util	Resp
8	0.12	2.07	0.12	1.91	0.11	2.67
16	0.23	2.62	0.21	3.18	0.17	5.40
32	0.43	3.11	0.34	5.58	0.26	9.35
64	0.71	5.18	0.48	10.52	0.37	15.34
128	0.91	11.42	0.61	19.96	0.52	24.36
256	1.00	26.09	0.73	37.74	0.70	39.36
512	1.00	58.00	0.84	70.30	0.86	67.80
1 Adversary						
8	0.12	2.18	0.12	2.06	0.11	2.74
16	0.21	3.21	0.20	3.73	0.17	5.69
32	0.38	4.04	0.32	6.32	0.25	9.81
64	0.64	6.12	0.45	11.42	0.34	16.76
128	0.87	11.95	0.57	21.22	0.49	26.08
256	0.98	29.93	0.70	39.01	0.67	40.88
512	1.00	56.92	0.81	72.03	0.84	69.70
2 Adversaries						
8	0.12	2.25	0.12	2.22	0.11	2.76
16	0.20	3.95	0.19	4.38	0.17	5.81
32	0.34	5.48	0.29	7.37	0.24	10.52
64	0.56	8.02	0.41	13.05	0.33	17.85
128	0.80	13.56	0.53	23.87	0.45	28.85
256	0.95	27.13	0.65	42.45	0.61	45.17
512	1.00	57.01	0.76	77.17	0.78	74.89
3 Adversaries						
8	0.12	2.50	0.12	2.46	0.11	2.81
16	0.18	4.85	0.18	5.08	0.16	5.93
32	0.28	7.77	0.26	9.03	0.23	11.10
64	0.43	11.96	0.35	16.10	0.31	19.40
128	0.64	18.31	0.46	28.10	0.40	32.80
256	0.85	30.90	0.57	49.20	0.53	53.30
512	0.97	58.98	0.68	86.00	0.69	84.40

Table 4: Experimental Results - Servers with one connection

6.2.3 Load Balancing with Adversaries

In this section we will look at the effects that adversarial load balance managers have on system performance. Table 4 shows the experimental results for servers with one connections. Under each load balance algorithm in the heading is two columns. The first column is server utilization and the second column is the response time. Four sets of experimental data are shown. The sets represent the experiments with zero to three adversarial load balance managers.

In this set of simulations the three state-base algorithms are competing for servers with one or more adversarial load balance manager. The adversary that was chosen is the random load balancing algorithm. Both the round robin and random algorithms perform equally making the choice based upon performance irrelevant. Therefore the random

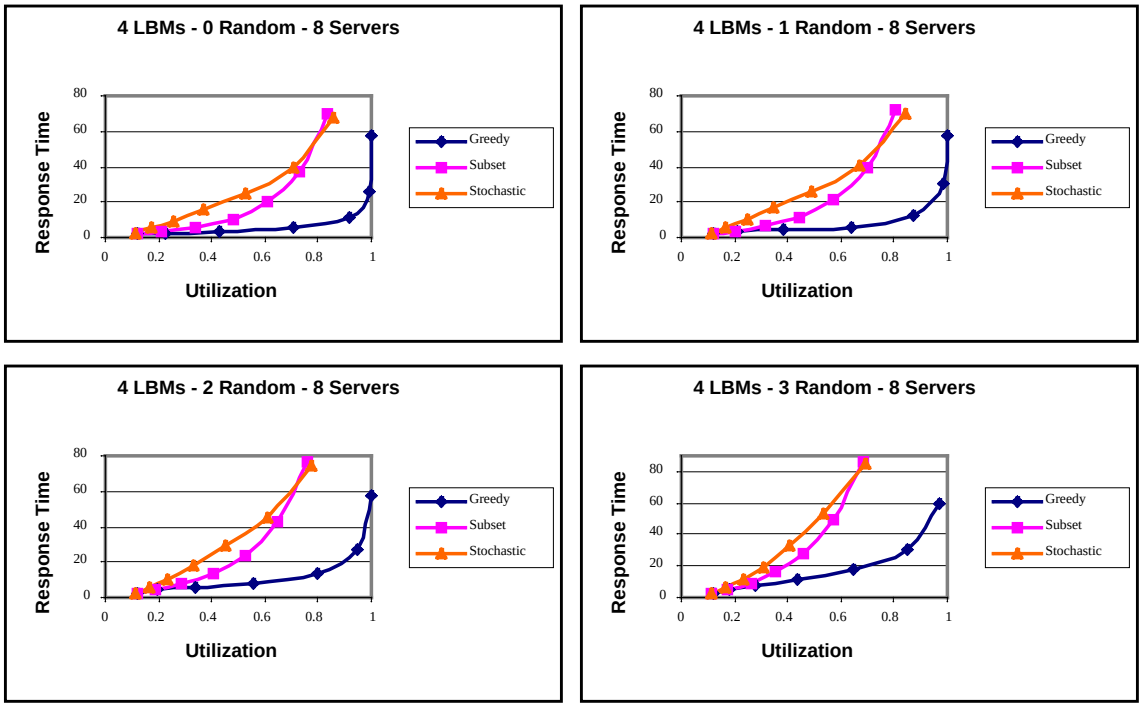

Figure 20: Results of Adversarial Load Balancing - Servers with one connection

algorithm is chosen because it is the easiest to implement. Figure 20 shows the results for four load balance managers, with zero to three adversaries competing for service. The greedy algorithm shows the best performance. The subset algorithm performs better than the stochastic algorithm at low server workloads and stochastic algorithm outperforms the subset algorithm at high server workloads.

Figure 21 shows the plots of each algorithm based upon the algorithm's performance versus each level of adversary. As you can see by the plots, the number of adversaries significantly impacts the performance of each algorithm as the response time increase with the addition of each adversary. But event though the adversaries have this impact on performance all the state-based algorithms improve performance when compared to the performance using only a stateless algorithm such as the plot showing the load balance manager as all random.

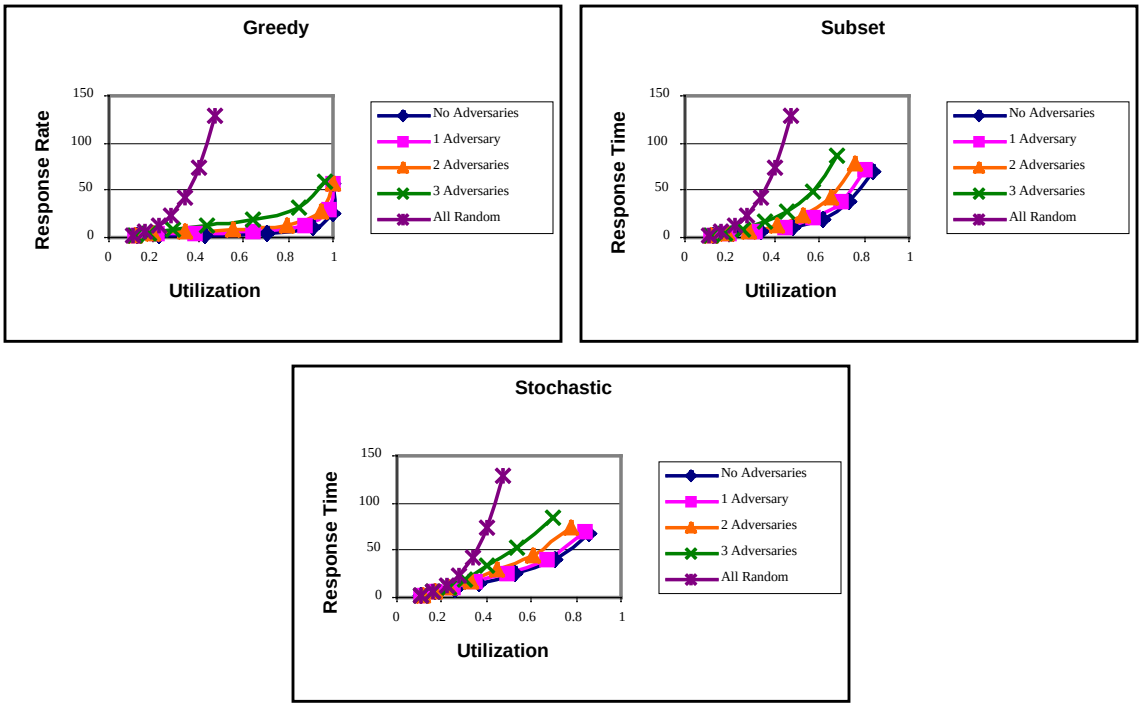

Figure 21: Comparison of each Load Balance Algorithm vs. Adversaries

Finally, we will look at the performance of load balancing with adversaries using servers with four client connections. Table 5 shows the experimental results for servers with one connections. Under each load balance algorithm in the heading is two columns. The first column is server utilization and the second column is the response time. Four sets of experimental data are shown. The sets represent the experiments with zero to three adversarial load balance managers.

Clients	Greedy		Subset		Stochastic	
No Adversary						
	Util	Resp	Util	Resp	Util	Resp
8	0.04	0.11	0.04	0.11	0.04	0.11
16	0.08	0.13	0.08	0.13	0.08	0.13
32	0.16	0.16	0.16	0.16	0.16	0.18
64	0.32	0.22	0.32	0.28	0.31	0.36
128	0.60	0.70	0.58	0.97	0.56	1.23
256	0.85	3.41	0.82	3.74	0.80	3.99
512	0.99	10.16	0.97	10.58	0.95	10.80
1 Adversary						
8	0.04	0.11	0.04	0.11	0.04	0.12
16	0.08	0.13	0.08	0.13	0.08	0.13
32	0.16	0.17	0.16	0.17	0.16	0.18
64	0.31	0.27	0.31	0.32	0.31	0.40
128	0.58	0.87	0.56	1.15	0.54	1.44
256	0.84	3.55	0.80	4.02	0.78	4.33
512	0.98	10.33	0.95	10.90	0.93	11.16
2 Adversaries						
8	0.04	0.11	0.04	0.11	0.04	0.12
16	0.08	0.13	0.08	0.13	0.08	0.13
32	0.16	0.17	0.16	0.18	0.16	0.19
64	0.31	0.34	0.31	0.40	0.31	0.45
128	0.56	1.18	0.53	1.47	0.52	1.69
256	0.81	3.92	0.77	4.46	0.73	4.91
512	0.96	10.73	0.91	11.51	0.90	11.81
3 Adversaries						
8	0.04	0.12	0.04	0.12	0.04	0.12
16	0.08	0.13	0.08	0.13	0.08	0.14
32	0.16	0.18	0.16	0.18	0.16	0.19
64	0.31	0.44	0.30	0.47	0.30	0.50
128	0.52	1.72	0.51	1.86	0.49	2.05
256	0.73	4.91	0.70	5.44	0.68	5.81
512	0.90	11.71	0.84	12.90	0.82	13.40

Table 5: Experimental Results - Servers with four connections

Figures 22 and 23 show the plots comparing the performance of each algorithm. These plots show basically the same results as the previous experiments with single server connections.

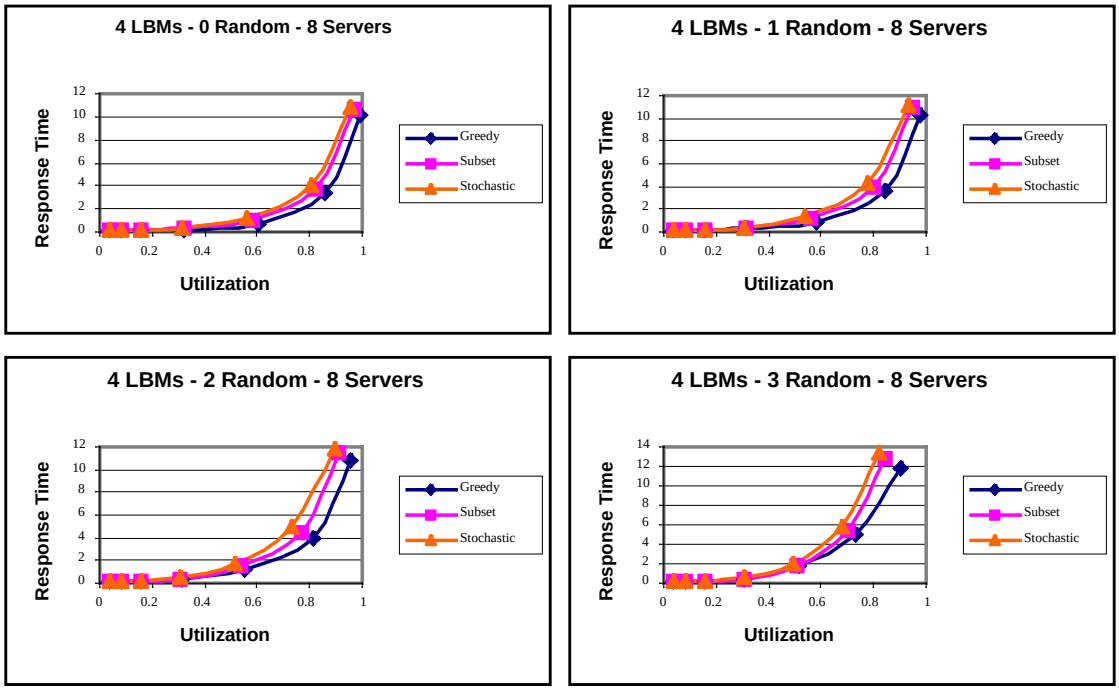

Figure 22: Results of Adversarial Load Balancing with 4 connections.

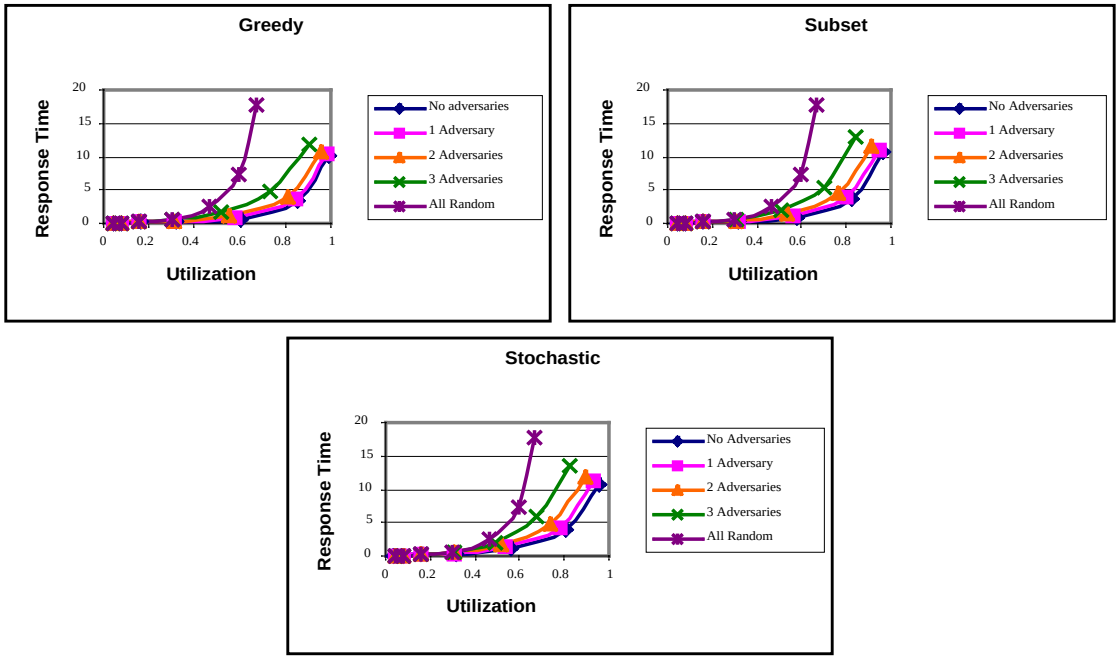

Figure 23: Comparison of each Load Balance Algorithm vs. Adversaries

CHAPTER VII

CONCLUSIONS

In this thesis, the subjects of Load Balancing and Internet Workload Modeling were presented for the purpose of doing performance analysis of different load balancing algorithms in different environments. Chapter 2 contained the important concepts related to load balancing and the discussion of recent research in that area. In Chapter 3, the details of modeling the characteristics of Internet traffic were discussed. These characteristics were shown to have high statistical variability caused by request inter-arrival rates, file size distributions, server workloads and network workloads. It was also shown that the high variability of these characteristics are statistically self-similar. Self-similarity is a fractal-like behavior where the high variations in workloads exist over a wide range of time scales. Chapter 4 contained the implementation details of the workload models developed from researching this thesis. At the end of Chapter 4, the verification and validation of these models is shown. Chapter 5 described the experimental design and Chapter 6 provided an analysis of the results.

The final conclusion in this thesis is that client response times can be reduced by using an agent, referred to here as a load balance manager that can obtain some knowledge of the system state for selecting processing elements to service requests. In comparison of stateless versus state-based load balance managers, the experimental results showed up to 27 times improvement in response time can be achieved by state-based algorithms avoid-

ing servers that are system bottlenecks over stateless algorithms. In all experiments, both stateless algorithms, round robin and random performed the same. Therefore a random algorithm would be the best choice to use because it should be the easiest to implement. Comparing the state-based algorithms the greedy algorithm showed the best performance in all cases, with the subset algorithm performing better than the stochastic algorithms at low server utilization, and the stochastic algorithms out performing the subset algorithm at higher server utilization. The conclusion that can be drawn here is use the greedy algorithms if it is possible. The results of the experiments in this study are inconclusive for selecting either of the subset algorithm or the stochastic algorithm over the other.

This thesis also showed how to model Internet traffic workloads that can be used for additional research to develop algorithms and protocols to be used in the Internet and distributed networks.

REFERENCES

Abdulla, Ghaleb, Ali H. Nayfeh and Edward A. Fox, *Modeling Correlated Proxy Web Traffic Using Fourier Analysis*, Virginia Polytechnic Institute and State University, 1997

Almeida, Virgilio, Azer Bestavros, Mark Crovella, and Adrianna de Oliveira, *Characterizing Reference Locality in the WWW*, Department of Computer Science, Boston University, Technical Paper TR-96-11, 1996

Almeida, Jussara, Virgilio Almeida and David J. Yates, *Measuring the Behavior of a World-Wide Web Server*, Computer Science Department, Boston University, Technical Report CS 96-025, October 29, 1996

Alrefaei, Mahmoud H. and Sigrun Andradottir, *Discrete Stochastic Optimization via a Modification of the Stochastic Ruler Method*, Proceedings of the 1996 Winter Simulation Conference, 1996

Alrefaei, Mahmoud H. and Sigrun Andradottir, *Accelerating the Convergence of the Stochastic Ruler Method Discrete Stochastic Optimization*, Proceedings of the 1997 Winter Simulation Conference, 1997

Altman, Eitan, and Shaler Stidham Jr., *Optimally of Monotonic Policies for Two-Action Markovian Decision Processes, with Applications to Control of Queues with Delayed Information*, March 8, 1994, revised, February 27, 1995 and June 15, 1995

Arlitt, Martin F., Ying Chen, Remi J. Gurski and Carey L. Williamson, *Traffic Modeling in the ATM-TM Telesim Project: Design, Implementation, and Performance Evaluation*, Department of Computer Science, University of Saskatchewan, Technical

Papel 95-6, 1995

Arlitt, Martin F. and Carey L. Williamson, *A Synthetic Workload Model for Internet Mosaic Traffic*, Department of Computer Science, University of Saskatchewan, Technical Paper 95-8, In *Proceedings of the 1995 Summer Computer Simulation Conference*, 1995

Arlitt, Martin F. and Carey L. Williamson, *Web Server Characterization: The Search for Invariants*, Department of Computer Science, University of Saskatchewan, In Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, pages 126-137, 1996

Arlitt, Martin F. and Carey L. Williamson, *Internet Web Servers: Workload Characterization and Performance Implications*, In IEEE/ACM Transactions on Networking, Volume 5, Number 5, pages 631-645, October 1997

Aversa, Luis and Azer Bestavros, *Load Balancing a Cluster of Web Servers Using Distributed Packet Rewriting*, Technical Report 99-001, Computer Science Department, Boston University, 1999

Avidor, Adi, Yossi Azar and Jiri Sgall, *Ancient and new algorithms for load balancing in the Lp norm*, Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 426-425, 1998

Avril, Herve and Carl Tropper, *The Dynamic Load Balancing of Clustered Time Warp for Logic Simulation*, Proceedings of the Tenth Workshop on Parallel Computing and Distributed Simulation, pages 20-27, IEEE, 1996

Balakrishnan, Hari, Srinivasan Seshan, Mark Stemm, and Randy H. Katz, *Analyzing Stability in Wide-Area Network Performance*, Proceedings of the 1997 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, Pages 2 – 12, 1997

Barford, Paul, and Mark Crovella, *Generating Representative Web Workloads for Network and Server Performance Evaluation,* Computer Science Department, Boston Uni-

versity, Technical Paper BU-CS-97-006, Revised December 31, 1997

Barford, Paul and Mark Crovella, *Measuring Web Performance in the Wide Area*, Technical Report BU-CS-99-004, Computer Science Department, Boston University, April 26, 1999

Barker, Ralph, *Optimizing Internet Bandwidth*, Performance Computing, Volume 18, No. 2, pages 29-33, January 2000

Bartal, Yair, Amos Fiat, Howard Karloff and Rakesh Vohra, *New Algorithms for an Ancient Scheduling Problem*, Proceedings of the Twenty forth Annual ACM Symposium on Theory of Computing, pages 51-58, 1992

Berenbrink, Petra, Tom Friedetzky and Angelika Steger, *Randomized and Adversarial Load Balancing (Extended Edition)*, Proceedings of the Eleventh Annual ACM Symposium on Parallel Algorithms and Architectures, Pages 175-184, 1999

Bestavros, Azer, Mark Crovella, Jan Liu, and David Martin, *Distributed Packet Rewriting and its Application to Scalable Server Architectures*, Computer Science Department, Boston University, 1997

Bestavros, Azer, Naomi Katagai, and Jorge M. Londono, *Admission Control and Scheduling for High-Performance WWW Servers*, BU-CS 97-015, Computer Science Department, Boston University, 1997, revised May 1998

Bhattacharjee, Samrat, Mostafa H. Ammar, Ellen W. Zegura, Viren Shah, and Zongming Fei, *Application-Layer Anycasting*, Networking and Telecommunications Group, College of Computing, Georgia Institute of Technology, Atlanta Ga, Technical Paper GIT-CC-96/25, 1996

Boukerche, Azzedine and Sajal K. Das, *Dynamic Load Balancing Strategies for Conservative Parallel Simulations*, Proceedings of the 1997 Workshop on Parallel and Distributed Simulation, pages 20-28, IEEE, 1997

Cahn, Robert S., *Wide Area Network Design, Concepts and Tools for Optimization*, Morgan Kaufmann Publishers, Inc. 1998

Carter, Robert L. and Mark E. Crovella, *Measuring Bottleneck Link Speed in Packet-Switched Networks*, BU-CS-96-006, Computer Science Department, Boston University, 1996

Carter, Robert L. and Mark E. Crovella, *Dynamic Server Selection using Bandwidth Probing in Wide-Area Networks,* BU-CS-96-007, Computer Science Department, Boston University, 1996

Cheng, Stephen, Kevin Lai and Mary Baker, *Analysis of HTTP/1.1 Performance on a Wireless Network,* Computer Systems Laboratory, Departments of Electrical Engineering and Computer Science, Stanford University, Stanford Ca., Technical Report CSL-TR-99-778, February 1999

Cho, Seung Ho, and Sang Young Han, *Dynamic Load Sharing Algorithm with a Weighted Load Representation*, Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing, page 385, 1995

Chow, C. Edward, *Dynamic Server Selection, One Candidate Architecture*, CS 622 Class Notes Page 59, University of Colorado, Colorado Springs, March 3, 1999

Cohen, Edith, and Haim Kaplin, *Exploiting Regularities in Web Traffic Patterns*, STOC'99, Atlanta Ga. USA, ACM 1999

Cormen, Thomas H., Charles E. Leiserson and Ronald L.Rivest, *Introduction to Algorithms*, The MIT Press, 1990, 1997

Crovella, Mark E. and Azer Bestavros, *Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes*, In IEEE/ACM Transactions on Networking, Volume 5, Number 6, pages 835-846, December 1997, also in Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pages 160-169, May 1996

Crovella Mark E. and Mor Harchol-Balter, *Task Assignment in a Distributed System: Improving Performance by Unbalancing Load*, Technical Paper BUCS-TR-1997-018, Department of Computer Science, Boston University, 1997

Crovella, Mark E., Robert Frangioso, and Mor Harchol-Balter, *Connection Scheduling in Web Servers*, Department of Computer Science, Boston University and Laboratory for Computer Science, MIT, Technical Paper BUCS-TR-99-003, April 10, 1999

Crovella, Mark E. and Murad S. Taqqu, *Estimating the Heavy Tail Index from Scaling Properties*, In Methodology and Computing in Applied Probability, Volume 1, Number 1, 1999

Cunha, Carlos, Azer Bestavros and Mark E. Crovella, *Characteristics of WWW Client-bases Traces*, Computer Science Department, Boston University, Technical Paper BU-CS-95-010, July 18, 1995

Daduna, Hans, *Busy Periods for Subnetworks in Stochastic Networks: Mean Value Analyses*, Journal of the Association for Computing Machinery, Vol. 35, No. 3, pages 668-674, 1988

Dahlin, Michael, *Interpreting Stale Load Information*, UTCS Technical Report TR98-20, Department of Computer Sciences, University of Texas at Austin, 1998

Das, Sajal K, Daniel J. Harvey and Rupak Biswas, *Dynamic Load Balancing for Adaptive Meshes using Symmetric Broadcast Networks*, Proceedings of the 1998 International Conference on Supercomputing, pages 417-424, ACM 1998

Decker, Rick, *Data Structures*, Prentice-Hall, Inc. 1989

Deelman, Ewa and Boleslaw K. Szymanski, *Dynamic Load Balancing in Parallel Discrete Event Simulation Spatially Explicit Problems*, Proceedings of the 12th Workshop on Parallel and Distributed Simulation, pages 46-53, IEEE, 1998

Dudewicz, Edward J. and Zaven A. Karian, *Modern Design and Analysis of Discrete-event Computer Simulations*, IEEE Computer Society Press, Washington DC, 1985

Duska, Bradley M., David Marwood and Michael J. Feeley, *The Measured Access Characteristics of World-Wide-Web Client Proxy Caches*, Technical Report TR-97-16, Department of Computer Science, University of British Columbia, Proceedings of

the USENIX Symposium in Internet Technologies and Systems, December 1997

Eager, Derek L., Edward D. Lazowska, and John Zahorjan, *Adaptive Load Sharing in Homogeneous Distributed Systems*, IEEE Transactions on Software Engineering Vol. SE-12, No. 5, May 1986

Eager, Derek L., Edward D. Lazowska, and John Zahorjan, *The Limited Performance Benefits of Migrating Active Processes for Load Sharing*, Proceedings of the 1988 ACM SIGMETRICS Conference on Measurement and Modeling of Computer System, Vol. 16, No. 1, May 1988

Elling, Volker, and Karsten Schwan, *Min-cut methods for Mapping Dataflow Graphs*, GIT-CC-99-05, Georgia Institute of Technology, 1999

Emery, Shawn, *Dynamic Load Balancing of Virtual Web Servers*, Master Thesis, University of Colorado, Colorado Spring, 2000

Faloutsos, Michalis, Petros Faloutsos and Christos Faloutsos, *On Power-Law Relationships of the Internet Topology*, Proceedings of ACM SIGCOMM '99 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Pages 251 – 262, 1999

Fei, Zongming, Samrat Bhattacharjee, Ellen W. Zegura and Mostafa H. Ammar, *A Novel Server Selection Technique for Improving the Response Time of a Replicated Service*, Networking and Telecommunications Group, College of Computing, Georgia Institute of Technology, Atlanta Ga, Technical Paper GIT-CC-97-24, 1997

Flaherty, J. E., R. M. Loy, P. C. Scully, M. S. Shepard, B. K. Szymanski, J. D. Teresco and L. H. Ziantz, *Load Balancing and Communication Optimization for Parallel Adaptive Finite Element Methods*, Rensselaer Polytechnic Institute, Proceedings of the 17th International Conference of the Chilean Computer Science Society (SCCC'97), Institute of Electrical and Electronics Engineers, Inc. 1997

Flanagan, David, *Java in a Nutshell, A Desktop Quick Reference*, O'Reilly & Associates Inc., Sebastopol, Ca, Second Edition, May 1997

Floyd, Sally and Kevin Fall, *Promoting the Use of End-to-End Congestion Control in the Internet*, IEEE/ACM Transactions on Networking, Vol. 7, No. 4, August 1999

Fox, G. C. and W. Furmanski, *Load Balancing Loosely Synchronous Problems with a Neural Network*, ACM 1988

Ghosh, Bhaskar and S. Muthukrishnan, *Dynamic Load Balancing in Parallel and Distributed Networks by Random Matchings (Extended Abstract)*, Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architecture, pages 226-235, 1994

Gualti, Sandeep, Jacob Barhen and S. Sitharama Iyengar, *The Pebble Crunching Model for Load Balancing in Concurrent Hypercube Ensembles*, Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications, Vol. 1, pages 189-199, 1998

Gullickson, Maria L. and Ann L. Chervenak, *Websnatcher: Customized WWW Prefetching*, January 1998

Hailperin, Max, *Load Balancing using Time Series Analysis for Soft Real Time Systems with Statistically Periodic Loads*, Phd. Dissertation, Stanford University, 1993

Hamdi, Mounir and Chi-Kin Lee, *Dynamic Load Balancing of Data Parallel Applications on a Distributed Network*, Proceedings of the 9th ACM International Conference on Supercomputing, pages 170-179, 1995

Hao, Fang, and Ellen Zegura, *On Scalable QoS Routing: Performance Evaluation of Topology Aggregation*, GIT-CC-99-04, Georgia Institute of Technology, 1999

Harchol-Balter, Mor, *Network Analysis Without Exponentially Assumptions*, Phd. Thesis, University of California, Berkeley, 1996

Harchol-Balter, Mor and Allen B. Downey, *Exploiting Process Lifetime Distributions for Dynamic Load Balancing*, University of California, Berkeley, ACM Transactions on Computer Systems, Vol. 15, No. 3, pages 253-285, August 1997

Harchol-Balter, Mor, Tom Leighton and Daniel Lewin, *Resource Discovery in Distributed*

Networks, PDOC'99, Atlanta, Ga, ACM 1999

Harinarayan, Venkatesh, and Leonard Klienrock, *Load Sharing In Limited Access Distributed Systems*, Proceedings of the 1991 ACM SIGMETRICS Conference on Measurement and Modeling Computer Systems, pages 21-30, 1991

Harold, Elliotte Rusty, *Java Network Programming*, O'Reilly & Associates, Inc. Sebastopol, Ca., First Edition, February 1997

Helsel, Brett, *Adding Intelegence to Standard DNS*, Performance Computing, Volume 18, No. 2, pages 19-21, January 2000

Hennessy, John L. and David A. Patterson, *Computer Architecture, A Quantitative Approach*, Morgan Kaufmann Publishers Inc. San Mateo, CA, 1990

Herrin II, Eric H. and Raphael Finkel, *An Implementation of Service Rebalancing, Technical Report Number 191-91*, University of Kentucky, 1991

Herrin II, Eric H. and Raphael Finkel, *Service Rebalancing, Technical Report Number 235-93*, Department of Computer Science, University of Kentucky, May 12,1993

Higgins, Kelly Jackson, *Load-Balancing Helps Redefine Encyclopedia Britannica*, Network Computing, August 23, 1999

Hui, Chi-Chung and Samuel T. Chanson, *Efficient Load Balancing in Interconnected LANs Using Group Communication*, Proceedings of the 17th International Conference on Distributed Computing Systems (ICDCS'97), Institute of Electrical and Electronics Engineers, Inc. 1997

Hummel, Susan Flynn, Jeanette Schmidt, R. N. Uma and Joel Wein, *Load-Sharing in Heterogeneous Systems via Weighted Factoring*, Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures, pages 318-328, 1996

Jain, Raj, *The Art of Computer System Performance Analysis, Techniques for Experimental Design, Measurement, Simulation, and Modeling*, John Wiley & Sons, Inc., 1991

Kasaraneni Jagadeesh, Theodore Johnson and Paul Avery, *Load Balancing in a Distrib-*

uted Processing System for High-Energy Physics (UFMulti), Proceedings of the 1995 ACM Symposium on Applied Computing, pages 172-181, 1995

Kim, Yonghwan, and San-qi Li, *Performance Analysis of Data Packet Discarding in ATM Networks*, IEEE/ACM Transactions on Networking, Vol. 7, No. 2, pages 216-227, April 1999

Knuth, Donald E, *The Art of Computer Programming - Volume 1*, Addison-Wesley, Reading, Mass., Third Edition, 1998

Kong, Keith and Dipak Ghosal, *Mitigating Server-Side Congestion in the Internet Through Pseudoserving*, IEEE/ACM Transactions on Networking, Vol. 7, No. 4, pages 530-544, August 1999

Krupczak, Bobby, Ken Calvert and Mostafa Ammar, *Implementing Protocols in Java: the Price of Portability*, GIT-CC-97-21, Georgia Institute of Technology, August 1, 1997

Lee, Yen-Jen and David H. C. Du, *Adaptive Load Sharing and Scheduling Schemes for Distributed Continuous Media Delivery*, Department of Computer Science and Engineering, University of Minnesota

Leland, Will E. and Teunis J. Ott, *Load-balancing Heuristics*, Bell Communications Research, Proceeding of the Joint Conference on Computer Performance Modeling, Measurement and Evaluation, pages 54-69, 1986

Leland, Will E., Murad S. Taqqu, Walter Willinger and Daniel V. Wilson, *On the Self-Similar Nature of Ethernet Traffic (Extended Version)*, IEEE/ACM Transactions on Networking, Volume 2, Number 1, pages 1-15, 1994

Martin, John C., *Introduction to Languages and the Theory of Computation*, McGraw-Hill, Inc., 1991

Medina, Alberto, Ibrahim Matta and John Byers, *On the Origin of Power Laws in Internet Topologies*, Computer Science Department, Boston University, Technical Paper 2000-004, January 21, 2000

Mitzenmacher, Michael, *On the Analysis of Randomized Load Balancing Schemes, Extended Abstract*, Digital Systems Research Center, Palo Alto, Ca., Proceedings of the 9th Annual Symposium on Parallel Algorithms and Architectures, 1997, pages 292-301, Newport, Rhode Island, USA, ACM 1997

Mitzenmacher, Michael, *How Useful is Old Information? (Extended Extract)*, Digital Systems Research Center, Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, Pages 83 – 91, 1997

Muthukrishnam, S. and Rajmohan Rajaraman, *An Adversarial Model for Distributed Dynamic Load Balancing*, Proceedings of the tenth annual ACM Symposium on Parallel Algorithms and Architectures, pages 47-54, 1998

Nelms, Sarah, *Using Java™ Servlets in Web Server Load Balancing*, CS 622, Distributed Networks Semester Project, University of Colorado, Colorado Spring, May 1999

Nicol, David M. and Philip Heidelberger, *A Comparative Study of Parallel Algorithms for Simulating Continuous Time Markov Chains*, ACM Transactions on Modeling and Computer Simulation, Vol. 5, No. 4, pages, 326-354, October 1995

Park, Kihong, Gatae Kim and Mark Crovella, On the relationship between file sizes, transport protocols, and self-similar network traffic, Technical Paper BU-CS-96-016, Computer Science Department, Boston University, August 7, 1996

Park, Kihong, Gatae Kim and Mark Crovella, *On the Effect of Traffic Self-similarity on Network Performance*, In Proceedings of the 1997 SPIE International Conference of Performance and Control of Network Systems, 1997

Paxson, Vern, *Empirically Derived Analytic Models of Wide-Area TCP Connections*, IEEE/ACM Transactions on Networking, Vol. 2, No. 4, August 1994

Paxson, Vern, and Sally Floyd, *Wide-Area Traffic: The Failure of Poisson Modeling*, Lawrence Berkeley Laboratory and EECS Division, University of California, Berkeley. July 18, 1995

Paxson, Vern and Sally Floyd, *Why We Don't Know How to Simulate the Internet*, Net-

work Research Group, Lawrence Berkeley National Laboratory, University of California, Berkeley, In Proceedings of the 1997 Winter Simulation Conference, 1997

Paxson, Vern, *End-to-End Internet Packet Dynamics*, IEEE/ACM Transactions on Networking, Vol. 7, No. 3, June 1999

Petrosky, Mary, *Ethernet Gets its Priorities Straight*, Performance Computing, Volume 18, No. 2, Pages 54-56, January 2000

Pooch, Udo W. and James A. Wall, *Discrete Event Simulation, A Practical Approach*, CRC Press, Inc., 1993

Purohit, Shashi K., Derek L. Eager and Richard B. Bunt, *An Evaluation of Priority Load Sharing*, Department of Computational Science, University of Saskatchewan, Saskatoon, Canada, 1992

Ross, Keith, W. and David D. Yao, *Optimal Load Balancing and Scheduling in a Distributed Computer System*, Journal of the Association for Computing Machinery, Vol. 38, No. 3, pages 676-690, July 1991

Ross, Kenneth A., Charles R. B. Wright, *Discrete Mathematics*, Prentice-Hall, Englewood NJ, Third Edition, 1992

Rudolph, Larry, Miriam Slivkin-Allalouf and Eli Upfal, *A Simple Load Balancing Scheme for Task Allocation in Parallel Machines*, ACM 1991

Romanow, Allyn and Sally Floyd, *Dynamics of TCP Traffic over ATM Networks*, 1994

Sebesta, Robert W., *Concepts of Programming Languages*, Second Edition, The Benjamin/Cummings Publishing Company, Inc. 1993

Sedgewick, Robert, *Algorithms in C*, Addison-Wesley Publishing Company, Inc. 1990

Seshan, Srinivasan, Mark Stemm and Randy H. Katz, *SPAND: Shared Passive Network Performance Discovery*, IBM T. J. Watson Research Center, Computer Science Division, University of California at Berkeley, Technical Paper 97-967

Shin, Kang G. and Chao-Ju Hou, *Evaluation of Load Sharing in HARTS while Considering Message Routing and Broadcasting (Extended Abstract)*, Proceedings of the

1993 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, pages 270-271, 1993

Shub, Charles M., *On Determining the Distribution of Software Response Times*, Proceedings of the 1985 Winter Simulation Conference, Pages 405-410, ACM 1985

Shub, Charles M., *Native Code Process-Originated Migration In A Heterogeneous Environment*, Proceedings of the 1990 ACM Conference on Cooperation, Pages 266 – 270, 1990

Srinivasan Seshan, Mark Stemm and Randy H. Katz, *SPAND: Shared Passive Network Performance Discovery*, IBM T. J. Watson Research Center, Yorktown Heights, NY, Computer Science Division, University of California at Berkeley, 1999

Standard Performance Evaluation Corporation, *SPECweb96*, http://www.spec.org/osg/web96, 1996

Standard Performance Evaluation Corporation, *The Workload for the SPECweb96 Benchmark*, http://www.specbench.org/osg/web96/workload.html, 1999

Standard Performance Evaluation Corporation, *SPECweb99*, http://www.specbench.org/osg/web99, 1999

Stone, Richard, *WAN Load Balancing Algorithm Analysis*, CS 622, Distributed Networks Semester Project, University of Colorado, Colorado Springs, May 5, 1999

Stubbs, Daniel F., and Neil W. Webre, *Data Structures with Abstract Data Types and Modula-2*, Brooks/Cole Publishing Company, Wadsworth, Inc., Belmont CA, 1987

Sun Microsystems Inc., *Solstice™ Job Scheduler Pro™ A Technical White Paper*, 1996

Sun Microsystems Inc., *Solstice™ Job Scheduler Pro™ Just the Facts*, 1997

Sun Microsystems Inc., *Maximizing Productivity: Compute Farms for EDA, Technical White Paper*, 1998

Tanenbaum, Andrew S., *Modern Operating Systems*, Prentice-Hall, Inc. Englewood Cliffs, NJ, 1992

Tanenbaum, Andrew S., *Computer Networks*, Prentice-Hall, Inc. Englewood Cliffs, NJ, Third Edition, 1996

Tantawi, Asser N. and Don Towsley, *Optimal Static Load Balancing in Distributed Computer Systems*, Journal of the Association for Computing Machinery, Vol. 32, No. 2 pages 445-465, April 1985

Terry, Chris and William *LeFebvre, Bearing the Load*, Performance Computing, Volume 18, No. 2, pages 50-53, January 2000

Van Cruyningen, Ike, *E-Business Everlasting*, Intelligent Enterprise, pages, 28-37, October 26, 1999

Vellanki, Vivekanand and Ann L. Chervenak, *Prefetching without Hints: A Cost-Benefit Analysis for Predicted Accesses*, GIT-CC-99-07, College of Computing, Georgia Institute of Technology, 1999

Viles, Charles L. and James C. French, *Availability and Latency of World Wide Web Information Servers*, University of Virginia, The USENIX Association, *Computing Systems*, Volume 8, No. 1, Winter 1995

Von Bank, David G., Charles M. Shub, and Robert W. Sebesta, *A Unified Model of Pointwise Equivalence of Procedural Computations*, ACM Transactions on Programming Languages and Systems, Vol. 16, No. 6, Pages 1842-1874, November 1994

Wagner, David B. and Edward D. Lazowska, *Parallel Simulation of Queue Networks: Limitations and Potentials*, Performance Evaluation Review, Vol. 17, #1, May 1989

Wall, Larry, Tom Christiansen and Randal L. Schwartz, *Programming Perl*, O'Reilly & Associates, Inc. Sebastopol, Ca., Second Edition, September 1996

Ware, Peter P., Thomas W. Page, Jr., and Barry L. Nelson, *Automatic Modeling of File System Workloads Using Two-Level Arrival Processes*, Ohio State University, Northwestern University, ACM 1998

Weiss, Mark Allen, *Data Structures & Problem Solving Using Java*, Addison Wesley

Longman, Inc., 1998

Willick, Darryl L. and Derek L. Eager, *An Analytical Model of Multistage Interconnection Networks*, Proceedings of the ACM Conference on Measurement and Modeling of Computer Systems, pages 192-202, 1990

Willinger, Walter, Murad S. Taqqu, Robert Sherman, and Daniel V. Wilson, *Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level*, IEEE/ACM Transactions on Networking, Volume 5, Number 1, pages 71-86, 1997

Wilson, Linda F. and David M. Nicol, *Automated Load Balancing in SPEEDES*, Proceedings of the 1995 Winter Simulation Conference, 1995

Wilson, Linda F. and David M. Nicol, *Experiments in Automated Load Balancing*, Proceedings of the Tenth Workshop on Parallel and Distributed Simulation, pages 4-11, IEEE, 1996

Wilson, Linda F. and Wei Shen, *Experiments in Load Migration and Dynamic Load Balancing in SPEEDES*, Proceedings of the 1998 Winter Simulation Conference, pages 483-490, 1998

Yerxa, Gregory, *Sharing the Load Across the Web*, Network Computing, Volume 10, No. 25, December 13, 1999

Zhang, Yingfang, *Wide Web Load Balancing Algorithm Design*, Master Thesis, University of Colorado, Colorado Springs, 1999

Zhang, Yu, Jia Wang and Srinivasan Keshav, *The Implication of Network Performance on Service Quality*, Department of Computer Science, Cornell University, July 5, 1999

APPENDIX

SIMULATION SOFTWARE

The simulation software is located in http://www.cs.uccs.edu/~chow/pub/master/alblais/sim. Java JDK1.2 is needed to run the simulator. The simulator can be executed with the following command:

java Netsim <execution time> <report time>

where execution time is the length in seconds to run the simulations, and the report time is the time interval for reporting the progress.

Input Files

The input files that need to be configured to do a simulation run are:

lbm.dat - the load balance manager configuration

server.dat - the server configuration

client.dat - the client configuration

Load Balance Manager Configuration File

The lbm.dat file defines the type of load balancing algorithm that is to be used for a simulation run. Line one contains a value representing the number of load balance manag-

ers. Each additional line contains two values with each line representing a load balance manager. The first value is a unique identifier and the second value is the load balance discipline, either the default or a random adversary. If the second value is zero then the load balance manager uses the default discipline, or else if it is one then the load balance manager uses a random placement discipline.

Example lbm.dat file:

```
4
0 0
1 0
2 0
3 0
```

Server Configuration File

The server.dat file defines the type of server that is used for a simulation run. The first line contains a value for the number of servers to be configured. For each server there is a line with a server identifier, the server power, and the number of client connections.

Example server.dat file:

```
4
0 1 1
1 1 1
2 1 1
3 1 1
```

Client Configuration File

The client.dat file contains the client information that is used for running a simulation. The first line contains the number of clients. For each client a line contains four values. The first value is an unique identifier, the second value is the identifier of the client's

load balance manager, the third value is the wake up time for the client, and the fourth value is the client sleep time.

Example client.dat file:

```
8
0 0 0 24
1 0 0 24
2 0 7 24
3 0 8 22
4 0 9 19
5 0 10 17
6 0 13 16
7 0 13 16
```

Output Files

The simulator generates four output files and the simulation report to the standard output. The four output files are:

clients.out - the client report interval status

servers.out - the servers report interval status

requests.out - simulation times for each complete request

event.out - unexecuted events at the end of a simulation run.

Client Output File

The client output file (clients.out) contains the client identifier, the hour of day, the report time, the total number of request, the number of request in the last interval, the total number bytes received, the current number of byte received in the last interval, the average

response time for the simulation, and the average response time during the current report interval.

Example clients.out file:

```
0       1       3600    195     195     1204191 1204191 6175    13.2028 13.2028
1       1       3600    175     175     658064  658064  3760    15.7474 15.7474
2       1       3600    173     173     1006875 1006875 5820    13.8675 13.8675
3       1       3600    174     174     744408  744408  4278    15.4982 15.4982
4       1       3600    158     158     695964  695964  4404    17.0347 17.0347
5       1       3600    166     166     801411  801411  4827    16.5923 16.5923
6       1       3600    155     155     1028942 1028942 6638    13.6916 13.6916
7       1       3600    169     169     641773  641773  3797    15.2645 15.2645
```

Server Output File

The server output file (servers.out) contains the server identifier, the hour of day, report time, the total number of requests received, the number of requests received during the last report interval, the total kilobytes transferred, the total kilobyte transferred during the last report interval, average queue length for the simulation, the average queue length for the last report interval, the total average utilization, and the utilization for the last report interval.

Example servers.out file:

```
0    1    3600    674     674    2,873     2,873   0   0.0000   0.1996  0.1996
1    1    3600    612     612    3,651     3,651   0   0.0000   0.2536  0.2536
```

Requests Output File

The request output file (requests.out) contain the following information for completed requests. The client identifier, the request number, the server identifier, the request time, the server receive time, the server start time, the complete time and the request file size.

Example requests.out file:

```
28      28      0       1003    1003    1103    1692    2824
21      21      0       1035    1035    1792    2284    2425
20      20      0       1048    1048    2384    2517    953
13      13      1       1016    1016    1116    2525    6182
```

Event Output File

The event output file (event.out) contains the event time, the event type, the request number, the sender identifier, and the receiver identifier.

Example event.out file:

```
Event Time: 3000 Type: DOCUMENT_DONE Request: 3295 Client: 84 Server: 0
Event Time: 3005 Type: DOCUMENT_DONE Request: 3294 Client: 53 Server: 1
```

Program Files

The program files are listed below.

```
-rw-r--r--   1 blais        8568 Mar 29 10:35 Client.java
-rw-r--r--   1 blais         789 Feb  8 10:03 Clock.java
-rw-r--r--   1 blais        1485 Feb  8 13:09 Event.java
-r--r--r--   1 blais        2258 Sep 29  1999 List.java
-r--r--r--   1 blais         615 Sep 29  1999 ListNode.java
-rw-r--r--   1 blais       17910 Mar 29 07:27 Netsim.java
-rw-r--r--   1 blais         439 Nov  4 10:01 Pareto.java
-rw-r--r--   1 blais        2578 Jan  4 10:36 Queue.java
-rw-r--r--   1 blais        3355 Feb 21 08:58 Request.java
-rw-r--r--   1 blais       17531 Feb 25 08:35 Server.java
-rw-r--r--   1 blais         589 Nov  4 11:22 Weibull.java
-rw-r--r--   1 blais        1335 Mar 29 09:09 LoadBalanceManager.java
```

The main program file is `Netsim.java`, this is the file that executes simulation events. `Netsim.java` uses two other files for running the simulation, `Clock.java` and `Event.java`. The `Clock.java` file is the simulation clock and the

`Event.java` file contains the data structures for executing the next event stored in that object.

The simulation objects have their data structures defined in `Client.java`, `Server.java`, `LoadBalanceManager.java` and `Request.java`.

The workload distributions are generated with `Pareto.java`, and `Weibull.java`.

Program utilities are implemented in `List.java`, `ListNode.java`, and `Queue.java`.

Simulation Model Verification

The verification data of the simulation models is listed in the following sections for the distributions representing file size body and tail, hourly request rate, active off time, inactive off time and embedded references

File Size Distribution Body

File Size Distribution (body) Chi Squared verification of simulation output.

Column 1: file size.

Column 2: total count of each file size by the simulator

Column 3: the percent total of all files generated

Column 4: cumulative total of the percentages in column 3

Column 5: expected cumulative percentage

Column 6: Chi Squared calculation of the cumulative total

Column 7: Chi Squared calculation of probability of each file size (0.001)

File Size	Count	Pct Total	Cumulative	Expected Pct.	Chi Sq.(cdf)	Chi Sq (pdf)
92	3335	0.001012015	0.0010217251	0.001	0.0000004720	0.0000001444
114	3275	0.000993807	0.0020155325	0.002	0.0000001206	0.0000000383
131	3278	0.000994718	0.0030102503	0.003	0.0000000350	0.0000000279
144	3310	0.001004428	0.0040146786	0.004	0.0000000539	0.0000000196
155	3315	0.001005946	0.0050206242	0.005	0.0000000851	0.0000000353
166	3292	0.000998966	0.0060195903	0.006	0.0000000640	0.0000000011
175	3306	0.001003214	0.0070228048	0.007	0.0000000743	0.0000000103
184	3407	0.001033863	0.0080566680	0.008	0.0000004014	0.0000011467
192	3331	0.001010801	0.0090674688	0.009	0.0000005058	0.0000001167
200	3350	0.001016566	0.0100840351	0.01	0.0000007062	0.0000002744
207	3208	0.000973476	0.0110575113	0.011	0.0000003007	0.0000007035
214	3327	0.001009587	0.0120670982	0.012	0.0000003752	0.0000000919

221	3282	0.000995932	0.0130630298	0.013	0.0000003056	0.0000000166
227	3331	0.001010801	0.0140738306	0.014	0.0000003894	0.0000001167
234	3275	0.000993807	0.0150676381	0.015	0.0000003050	0.0000000383
240	3324	0.001008677	0.0160763147	0.016	0.0000003640	0.0000000753
246	3222	0.000977724	0.0170540392	0.017	0.0000001718	0.0000004962
251	3348	0.001015959	0.0180699986	0.018	0.0000002722	0.0000002547
257	3300	0.001001394	0.0190713924	0.019	0.0000002683	0.0000000019
263	3282	0.000995932	0.0200673240	0.02	0.0000002266	0.0000000166
268	3397	0.001030829	0.0210981527	0.021	0.0000004588	0.0000009504
273	3351	0.00101687	0.0221150225	0.022	0.0000006014	0.0000002846
278	3256	0.000988042	0.0231030644	0.023	0.0000004618	0.0000001430
283	3343	0.001014442	0.0241175066	0.024	0.0000005753	0.0000002086
288	3279	0.000995021	0.0251125278	0.025	0.0000005065	0.0000000248
293	3389	0.001028401	0.0261409289	0.026	0.0000007639	0.0000008066
298	3274	0.000993504	0.0271344329	0.027	0.0000006693	0.0000000422
303	3267	0.00099138	0.0281258127	0.028	0.0000005653	0.0000000743
308	3294	0.000999573	0.0291253857	0.029	0.0000005421	0.0000000002
312	3339	0.001013228	0.0301386141	0.03	0.0000006405	0.0000001750
317	3326	0.001009284	0.0311478977	0.031	0.0000007056	0.0000000862
321	3225	0.000978635	0.0321265325	0.032	0.0000005003	0.0000004565
326	3245	0.000984704	0.0331112363	0.033	0.0000003750	0.0000002340
330	3332	0.001011104	0.0341223406	0.034	0.0000004402	0.0000001233
335	3234	0.000981366	0.0351037065	0.035	0.0000003073	0.0000003472
339	3362	0.001020208	0.0361239143	0.036	0.0000004265	0.0000004084
343	3299	0.00100109	0.0371250046	0.037	0.0000004223	0.0000000012
348	3311	0.001004732	0.0381297363	0.038	0.0000004429	0.0000000224
352	3285	0.000996842	0.0391265783	0.039	0.0000004108	0.0000000100
356	3344	0.001014746	0.0401413240	0.04	0.0000004993	0.0000002174
360	3293	0.00099927	0.0411405936	0.041	0.0000004821	0.0000000005
364	3317	0.001006552	0.0421471460	0.042	0.0000005155	0.0000000429
368	3394	0.001029918	0.0431770643	0.043	0.0000007291	0.0000008951
372	3285	0.000996842	0.0441739063	0.044	0.0000006873	0.0000000100
376	3327	0.001009587	0.0451834933	0.045	0.0000007482	0.0000000919
380	3252	0.000986828	0.0461703213	0.046	0.0000006306	0.0000001735
384	3301	0.001001697	0.0471720185	0.047	0.0000006296	0.0000000029
388	3271	0.000992594	0.0481646121	0.048	0.0000005645	0.0000000549
392	3284	0.000996539	0.0491611507	0.049	0.0000005300	0.0000000120
396	3317	0.001006552	0.0501677031	0.05	0.0000005625	0.0000000429
400	3267	0.00099138	0.0511590829	0.051	0.0000004962	0.0000000743
403	3251	0.000986525	0.0521456075	0.052	0.0000004077	0.0000001816
407	3353	0.001017477	0.0531630843	0.053	0.0000005018	0.0000003054
411	3243	0.000984097	0.0541471812	0.054	0.0000004012	0.0000002529
415	3296	0.00100018	0.0551473612	0.055	0.0000003948	0.0000000000
419	3314	0.001005642	0.0561530033	0.056	0.0000004180	0.0000000318
422	3277	0.000994414	0.0571474176	0.057	0.0000003813	0.0000000312
426	3172	0.000962552	0.0581099694	0.058	0.0000002085	0.0000014024
430	3273	0.000993201	0.0591031700	0.059	0.0000001804	0.0000000462
433	3294	0.000999573	0.0601027430	0.06	0.0000001759	0.0000000002
437	3215	0.0009756	0.0610783433	0.061	0.0000001006	0.0000005953
441	3364	0.001020815	0.0620991580	0.062	0.0000001586	0.0000004333
444	3267	0.00099138	0.0630905378	0.063	0.0000001301	0.0000000743
448	3271	0.000992594	0.0640831315	0.064	0.0000001080	0.0000000549
451	3397	0.001030829	0.0651139601	0.065	0.0000001998	0.0000009504
455	3329	0.001010194	0.0661241540	0.066	0.0000002335	0.0000001039
458	3320	0.001007463	0.0671316168	0.067	0.0000002586	0.0000000557
462	3322	0.00100807	0.0681396865	0.068	0.0000002869	0.0000000651
465	3361	0.001019904	0.0691595909	0.069	0.0000003691	0.0000003962
469	3341	0.001013835	0.0701734262	0.07	0.0000004297	0.0000001914
472	3355	0.001018084	0.0711915099	0.071	0.0000005166	0.0000003270
476	3298	0.001000787	0.0721922967	0.072	0.0000005136	0.0000000006
479	3345	0.001015049	0.0732073459	0.073	0.0000005889	0.0000002265
483	3210	0.000974083	0.0741814289	0.074	0.0000004448	0.0000006717
486	3286	0.000997145	0.0751785743	0.075	0.0000004252	0.0000000081
490	3265	0.000990773	0.0761693472	0.076	0.0000003773	0.0000000851
493	3245	0.000984704	0.0771540511	0.077	0.0000003082	0.0000002340
497	3331	0.001010801	0.0781648519	0.078	0.0000003484	0.0000001167
500	3354	0.00101778	0.0791826321	0.079	0.0000004222	0.0000003161
503	3325	0.00100898	0.0801916121	0.08	0.0000004589	0.0000000806
507	3362	0.001020208	0.0812118200	0.081	0.0000005539	0.0000004084
510	3222	0.000977724	0.0821895444	0.082	0.0000004381	0.0000004962
514	3235	0.000981669	0.0831712138	0.083	0.0000003532	0.0000003360
517	3273	0.000993201	0.0841644143	0.084	0.0000003218	0.0000000462
520	3249	0.000985918	0.0851503320	0.085	0.0000002659	0.0000001983
524	3252	0.000986828	0.0861371600	0.086	0.0000002188	0.0000001735
527	3302	0.001002001	0.0871391607	0.087	0.0000002226	0.0000000040
530	3381	0.001025973	0.0881651341	0.088	0.0000003099	0.0000006746
534	3322	0.00100807	0.0891732038	0.089	0.0000003371	0.0000000651
537	3274	0.000993504	0.0901667078	0.09	0.0000003088	0.0000000422

540	3228	0.000979545	0.0911462530	0.091	0.0000002351	0.0000004184
544	3153	0.000956786	0.0921030392	0.092	0.0000001154	0.0000018674
547	3215	0.0009756	0.0930786395	0.093	0.0000000665	0.0000005953
550	3330	0.001010497	0.0940891368	0.094	0.0000000845	0.0000001102
554	3380	0.00102567	0.0951148068	0.095	0.0000001387	0.0000006589
557	3266	0.000991076	0.0961058831	0.096	0.0000001168	0.0000000796
560	3285	0.000996842	0.0971027251	0.097	0.0000001088	0.0000000100
563	3306	0.001003214	0.0981059396	0.098	0.0000001145	0.0000000103
567	3243	0.000984097	0.0990900365	0.099	0.0000000819	0.0000002529
570	3265	0.000990773	0.1000808094	0.1	0.0000000653	0.0000000851
573	3218	0.000976511	0.1010573201	0.101	0.0000000325	0.0000005518
576	3163	0.000959821	0.1020171408	0.102	0.0000000029	0.0000016144
580	3245	0.000984704	0.1030018447	0.103	0.0000000000	0.0000002340
583	3348	0.001015959	0.1040178042	0.104	0.0000000030	0.0000002547
586	3413	0.001035684	0.1050534881	0.105	0.0000000272	0.0000012733
589	3406	0.00103356	0.1060870478	0.106	0.0000000715	0.0000011263
593	3327	0.001009587	0.1070966348	0.107	0.0000000873	0.0000000919
596	3289	0.000998056	0.1080946906	0.108	0.0000000830	0.0000000038
599	3280	0.000995325	0.1090900153	0.109	0.0000000743	0.0000000219
602	3343	0.001014442	0.1101044575	0.11	0.0000000992	0.0000002086
605	3275	0.000993807	0.1110982649	0.111	0.0000000870	0.0000000383
609	3316	0.001006249	0.1121045139	0.112	0.0000000975	0.0000000391
612	3308	0.001003821	0.1131083353	0.113	0.0000001039	0.0000000146
615	3312	0.001005035	0.1141133705	0.114	0.0000001127	0.0000000254
618	3231	0.000980456	0.1150938260	0.115	0.0000000766	0.0000003820
621	3321	0.001007766	0.1161015923	0.116	0.0000000890	0.0000000603
625	3301	0.001001697	0.1171032895	0.117	0.0000000912	0.0000000029
628	3361	0.001019904	0.1181231939	0.118	0.0000001286	0.0000003962
631	3300	0.001001394	0.1191245876	0.119	0.0000001304	0.0000000019
634	3289	0.000998056	0.1201226434	0.12	0.0000001253	0.0000000038
637	3332	0.001011104	0.1211337477	0.121	0.0000001478	0.0000001233
640	3280	0.000995325	0.1221290724	0.122	0.0000001366	0.0000000219
644	3381	0.001025973	0.1231550458	0.123	0.0000001954	0.0000006746
647	3255	0.000987738	0.1241427842	0.124	0.0000001644	0.0000001503
650	3284	0.000996539	0.1251393227	0.125	0.0000001553	0.0000000120
653	3265	0.000990773	0.1261300956	0.126	0.0000001343	0.0000000851
656	3194	0.000969228	0.1270993234	0.127	0.0000000777	0.0000009469
659	3316	0.001006249	0.1281055724	0.128	0.0000000871	0.0000000391
663	3394	0.001029918	0.1291354907	0.129	0.0000001423	0.0000008951
666	3347	0.001015656	0.1301511467	0.13	0.0000001757	0.0000002451
669	3343	0.001014442	0.1311655890	0.131	0.0000002093	0.0000002086
672	3284	0.000996539	0.1321621275	0.132	0.0000001991	0.0000000120
675	3291	0.000998663	0.1331607902	0.133	0.0000001944	0.0000000018
678	3345	0.001015049	0.1341758393	0.134	0.0000002307	0.0000002265
682	3280	0.000995325	0.1351711640	0.135	0.0000002170	0.0000000219
685	3330	0.001010497	0.1361816613	0.136	0.0000002427	0.0000001102
688	3258	0.000988649	0.1371703101	0.137	0.0000002117	0.0000001289
691	3261	0.000989559	0.1381598692	0.138	0.0000001852	0.0000001090
694	3345	0.001015049	0.1391749183	0.139	0.0000002201	0.0000002265
697	3310	0.001004428	0.1401793466	0.14	0.0000002298	0.0000000196
700	3310	0.001004428	0.1411837749	0.141	0.0000002395	0.0000000196
704	3212	0.00097469	0.1421584648	0.142	0.0000001768	0.0000006406
707	3347	0.001015656	0.1431741208	0.143	0.0000002120	0.0000002451
710	3352	0.001017173	0.1441912941	0.144	0.0000002541	0.0000002949
713	3345	0.001015049	0.1452063433	0.145	0.0000002936	0.0000002265
716	3283	0.000996235	0.1462025783	0.146	0.0000002811	0.0000000142
719	3297	0.001000483	0.1472030617	0.147	0.0000002805	0.0000000002
722	3252	0.000986828	0.1481898897	0.148	0.0000002436	0.0000001735
725	3237	0.000982276	0.1491721660	0.149	0.0000001989	0.0000003141
729	3353	0.001017477	0.1501896427	0.15	0.0000002398	0.0000003054
732	3295	0.000999876	0.1511895192	0.151	0.0000002379	0.0000000000
735	3301	0.001001697	0.1521912164	0.152	0.0000002406	0.0000000029
738	3281	0.000995628	0.1531868446	0.153	0.0000002282	0.0000000191
741	3458	0.001049339	0.1542361839	0.154	0.0000003622	0.0000024344
744	3389	0.001028401	0.1552645849	0.155	0.0000004516	0.0000008066
747	3362	0.001020208	0.1562847927	0.156	0.0000005199	0.0000004084
751	3276	0.000994111	0.1572789036	0.157	0.0000004955	0.0000000347
754	3288	0.000997752	0.1582766560	0.158	0.0000004844	0.0000000051
757	3472	0.001053588	0.1593302436	0.159	0.0000006859	0.0000028716
760	3322	0.00100807	0.1603383133	0.16	0.0000007153	0.0000000651
763	3300	0.001001394	0.1613397071	0.161	0.0000007168	0.0000000019
766	3339	0.001013228	0.1623529355	0.162	0.0000007689	0.0000001750
769	3227	0.000979242	0.1633321772	0.163	0.0000006769	0.0000004309
772	3274	0.000993504	0.1643256812	0.164	0.0000006468	0.0000000422
776	3279	0.000995021	0.1653207024	0.165	0.0000006233	0.0000000248
779	3263	0.000990166	0.1663108684	0.166	0.0000005822	0.0000000967
782	3391	0.001029008	0.1673398764	0.167	0.0000006917	0.0000008415
785	3297	0.001000483	0.1683403598	0.168	0.0000006896	0.0000000002

788	3325	0.00100898	0.1693493399	0.169	0.0000007221	0.0000000806
791	3266	0.000991076	0.1703404162	0.17	0.0000006817	0.0000000796
794	3301	0.001001697	0.1713421134	0.171	0.0000006845	0.0000000029
798	3296	0.00100018	0.1723422934	0.172	0.0000006812	0.0000000000
801	3294	0.000999573	0.1733418664	0.173	0.0000006756	0.0000000002
804	3272	0.000992897	0.1743347635	0.174	0.0000006441	0.0000000505
807	3271	0.000992594	0.1753273571	0.175	0.0000006124	0.0000000549
810	3279	0.000995021	0.1763223784	0.176	0.0000005905	0.0000000248
813	3310	0.001004428	0.1773268067	0.177	0.0000006034	0.0000000196
816	3207	0.000973173	0.1782999793	0.178	0.0000005055	0.0000007197
819	3148	0.000955269	0.1792552483	0.179	0.0000003640	0.0000020009
823	3212	0.00097469	0.1802299382	0.18	0.0000002937	0.0000006406
826	3300	0.001001394	0.1812313320	0.181	0.0000002957	0.0000000019
829	3341	0.001013835	0.1822451673	0.182	0.0000003303	0.0000001914
832	3268	0.000991683	0.1832368506	0.183	0.0000003065	0.0000000692
835	3269	0.000991987	0.1842288373	0.184	0.0000002846	0.0000000642
838	3316	0.001006249	0.1852350863	0.185	0.0000002987	0.0000000391
842	3287	0.000997449	0.1862325352	0.186	0.0000002907	0.0000000065
845	3304	0.001002608	0.1872351427	0.187	0.0000002957	0.0000000068
848	3367	0.001021725	0.1882568678	0.188	0.0000003510	0.0000004720
851	3287	0.000997449	0.1892543167	0.189	0.0000003422	0.0000000065
854	3319	0.001007159	0.1902614760	0.19	0.0000003598	0.0000000513
857	3343	0.001014442	0.1912759183	0.191	0.0000003986	0.0000002086
860	3402	0.001032346	0.1923082642	0.192	0.0000004949	0.0000010463
864	3310	0.001004428	0.1933126925	0.193	0.0000005066	0.0000000196
867	3207	0.000973173	0.1942858651	0.194	0.0000004212	0.0000007197
870	3348	0.001015959	0.1953018246	0.195	0.0000004672	0.0000002547
873	3272	0.000992897	0.1962947217	0.196	0.0000004432	0.0000000505
876	3213	0.000974993	0.1972697151	0.197	0.0000003693	0.0000006253
879	3248	0.000985614	0.1982553293	0.198	0.0000003293	0.0000002070
883	3394	0.001029918	0.1992852476	0.199	0.0000004089	0.0000008951
886	3252	0.000986828	0.2002720756	0.2	0.0000003701	0.0000001735
889	3214	0.000975297	0.2012473725	0.201	0.0000003044	0.0000006102
892	3110	0.000943738	0.2021911102	0.202	0.0000001808	0.0000031654
895	3261	0.000989559	0.2031806693	0.203	0.0000001608	0.0000001090
898	3255	0.000987738	0.2041684077	0.204	0.0000001390	0.0000001503
902	3285	0.000996842	0.2051652497	0.205	0.0000001332	0.0000000100
905	3400	0.001031739	0.2061969887	0.206	0.0000001884	0.0000010074
908	3338	0.001012925	0.2072099137	0.207	0.0000002129	0.0000001671
911	3250	0.000986221	0.2081961348	0.208	0.0000001849	0.0000001899
914	3339	0.001013228	0.2092093632	0.209	0.0000002097	0.0000001750
918	3257	0.000988345	0.2101977085	0.21	0.0000001861	0.0000001358
921	3327	0.001009587	0.2112072955	0.211	0.0000002037	0.0000000919
924	3220	0.000977118	0.2121844130	0.212	0.0000001604	0.0000005236
927	3169	0.000961641	0.2131460545	0.213	0.0000001001	0.0000014714
930	3333	0.001011408	0.2141574622	0.214	0.0000001159	0.0000001301
934	3149	0.000955572	0.2151130346	0.215	0.0000000594	0.0000019738
937	3352	0.001017173	0.2161302079	0.216	0.0000000785	0.0000002949
940	3385	0.001027187	0.2171573951	0.217	0.0000001142	0.0000007391
943	3188	0.000967407	0.2181248022	0.218	0.0000000714	0.0000010623
946	3185	0.000966497	0.2190912989	0.219	0.0000000381	0.0000011225
950	3265	0.000990773	0.2200820718	0.22	0.0000000306	0.0000000851
953	3369	0.001022332	0.2211044038	0.221	0.0000000493	0.0000004987
956	3367	0.001021725	0.2221261289	0.222	0.0000000717	0.0000004720
959	3280	0.000995325	0.2231214536	0.223	0.0000000661	0.0000000219
962	3359	0.001019297	0.2241407511	0.224	0.0000000884	0.0000003724
966	3407	0.001033863	0.2251746142	0.225	0.0000001355	0.0000011467
969	3272	0.000992897	0.2261675113	0.226	0.0000001242	0.0000000505
972	3291	0.000998663	0.2271661740	0.227	0.0000001216	0.0000000018
975	3340	0.001013532	0.2281797059	0.228	0.0000001416	0.0000001831
979	3177	0.000964069	0.2291437750	0.229	0.0000000903	0.0000012910
982	3315	0.001005946	0.2301497205	0.23	0.0000000975	0.0000000353
985	3340	0.001013532	0.2311632524	0.231	0.0000001154	0.0000001831
988	3305	0.001002911	0.2321661634	0.232	0.0000001190	0.0000000085
992	3225	0.000978635	0.2331447982	0.233	0.0000000900	0.0000004565
995	3277	0.000994414	0.2341392125	0.234	0.0000000828	0.0000000312
998	3381	0.001025973	0.2351651860	0.235	0.0000001161	0.0000006746
1001	3242	0.000983794	0.2361489795	0.236	0.0000000940	0.0000002627
1005	3245	0.000984704	0.2371336833	0.237	0.0000000754	0.0000002340
1008	3384	0.001026884	0.2381605671	0.238	0.0000001083	0.0000007227
1011	3333	0.001011408	0.2391719748	0.239	0.0000001237	0.0000001301
1014	3273	0.000993201	0.2401651753	0.24	0.0000001137	0.0000000462
1018	3274	0.000993504	0.2411586793	0.241	0.0000001045	0.0000000422
1021	3251	0.000986525	0.2421452039	0.242	0.0000000871	0.0000001816
1024	3273	0.000993201	0.2431384045	0.243	0.0000000788	0.0000000462
1027	3230	0.000980152	0.2441185565	0.244	0.0000000576	0.0000003939
1031	3253	0.000987131	0.2451056880	0.245	0.0000000456	0.0000001656
1034	3289	0.000998056	0.2461037438	0.246	0.0000000438	0.0000000038

1037	3232	0.000980759	0.2470845028	0.247	0.0000000289	0.0000003702
1041	3362	0.001020208	0.2481047106	0.248	0.0000000442	0.0000004084
1044	3386	0.001027491	0.2491322013	0.249	0.0000000702	0.0000007557
1047	3251	0.000986525	0.2501187259	0.25	0.0000000564	0.0000001816
1051	3351	0.00101687	0.2511355957	0.251	0.0000000733	0.0000002846
1054	3345	0.001015049	0.2521506448	0.252	0.0000000901	0.0000002265
1057	3429	0.001040539	0.2531911840	0.253	0.0000001445	0.0000016434
1060	3230	0.000980152	0.2541713360	0.254	0.0000001156	0.0000003939
1064	3297	0.001000483	0.2551718194	0.255	0.0000001158	0.0000000002
1067	3299	0.00100109	0.2561729097	0.256	0.0000001168	0.0000000012
1070	3324	0.001008677	0.2571815864	0.257	0.0000001283	0.0000000753
1074	3329	0.001010194	0.2581917803	0.258	0.0000001426	0.0000001039
1077	3196	0.000969835	0.2591616149	0.259	0.0000001008	0.0000009099
1080	3256	0.000988042	0.2601496568	0.26	0.0000000861	0.0000001430
1084	3348	0.001015959	0.2611656163	0.261	0.0000001051	0.0000002547
1087	3231	0.000980456	0.2621460718	0.262	0.0000000814	0.0000003820
1090	3300	0.001001394	0.2631474655	0.263	0.0000000827	0.0000000019
1094	3268	0.000991683	0.2641391488	0.264	0.0000000733	0.0000000692
1097	3318	0.001006856	0.2651460047	0.265	0.0000000804	0.0000000470
1100	3258	0.000988649	0.2661346535	0.266	0.0000000682	0.0000001289
1104	3272	0.000992897	0.2671275506	0.267	0.0000000609	0.0000000505
1107	3287	0.000997449	0.2681249994	0.268	0.0000000583	0.0000000065
1111	3366	0.001021422	0.2691464211	0.269	0.0000000797	0.0000004589
1114	3288	0.000997752	0.2701441734	0.27	0.0000000770	0.0000000051
1117	3282	0.000995932	0.2711401050	0.271	0.0000000724	0.0000000166
1121	3234	0.000981366	0.2721214709	0.272	0.0000000542	0.0000003472
1124	3236	0.000981973	0.2731034437	0.273	0.0000000392	0.0000003250
1127	3347	0.001015656	0.2741190997	0.274	0.0000000518	0.0000002451
1131	3265	0.000990773	0.2751098726	0.275	0.0000000439	0.0000000851
1134	3342	0.001014139	0.2761240114	0.276	0.0000000557	0.0000001999
1138	3298	0.001000787	0.2771247982	0.277	0.0000000562	0.0000000006
1141	3236	0.000981973	0.2781067710	0.278	0.0000000410	0.0000003250
1145	3343	0.001014442	0.2791212133	0.279	0.0000000527	0.0000002086
1148	3301	0.001001697	0.2801229105	0.28	0.0000000540	0.0000000029
1151	3343	0.001014442	0.2811373527	0.281	0.0000000671	0.0000002086
1155	3291	0.000998663	0.2821360154	0.282	0.0000000656	0.0000000018
1158	3230	0.000980152	0.2831161674	0.283	0.0000000477	0.0000003939
1162	3267	0.00099138	0.2841075473	0.284	0.0000000407	0.0000000743
1165	3323	0.001008373	0.2851159204	0.285	0.0000000471	0.0000000701
1168	3322	0.00100807	0.2861239901	0.286	0.0000000538	0.0000000651
1172	3290	0.000998359	0.2871223494	0.287	0.0000000522	0.0000000027
1175	3337	0.001012622	0.2881349709	0.288	0.0000000633	0.0000001593
1179	3267	0.00099138	0.2891263507	0.289	0.0000000552	0.0000000743
1182	3241	0.00098349	0.2901098408	0.29	0.0000000416	0.0000002726
1186	3419	0.001037505	0.2911473454	0.291	0.0000000746	0.0000014066
1189	3232	0.000980759	0.2921281044	0.292	0.0000000562	0.0000003702
1193	3305	0.001002911	0.2931310154	0.293	0.0000000586	0.0000000085
1196	3385	0.001027187	0.2941582026	0.294	0.0000000851	0.0000007391
1200	3315	0.001005946	0.2951641482	0.295	0.0000000913	0.0000000353
1203	3397	0.001030829	0.2961949768	0.296	0.0000001284	0.0000009504
1207	3336	0.001012318	0.2972072949	0.297	0.0000001447	0.0000001517
1210	3324	0.001008677	0.2982159715	0.298	0.0000001565	0.0000000753
1214	3306	0.001003214	0.2992191860	0.299	0.0000001607	0.0000000103
1217	3274	0.000993504	0.3002126900	0.3	0.0000001508	0.0000000422
1221	3287	0.000997449	0.3012101388	0.301	0.0000001467	0.0000000065
1224	3417	0.001036898	0.3022470366	0.302	0.0000002021	0.0000013614
1228	3228	0.000979545	0.3032265817	0.303	0.0000001694	0.0000004184
1231	3291	0.000998663	0.3042252444	0.304	0.0000001669	0.0000000018
1235	3283	0.000996235	0.3052214795	0.305	0.0000001608	0.0000000142
1238	3398	0.001031132	0.3062526116	0.306	0.0000002085	0.0000009692
1242	3296	0.00100018	0.3072527915	0.307	0.0000002082	0.0000000000
1245	3279	0.000995021	0.3082478128	0.308	0.0000001994	0.0000000248
1249	3260	0.000989256	0.3092370684	0.309	0.0000001819	0.0000001154
1252	3297	0.001000483	0.3102375518	0.31	0.0000001820	0.0000000002
1256	3270	0.00099229	0.3112298420	0.311	0.0000001699	0.0000000594
1260	3247	0.000985311	0.3122151528	0.312	0.0000001484	0.0000002158
1263	3355	0.001018084	0.3132332364	0.313	0.0000001738	0.0000003270
1267	3345	0.001015049	0.3142482856	0.314	0.0000001963	0.0000002265
1270	3317	0.001006552	0.3152548380	0.315	0.0000002062	0.0000000429
1274	3238	0.00098258	0.3162374177	0.316	0.0000001784	0.0000003035
1277	3370	0.001022635	0.3172600532	0.317	0.0000002133	0.0000005124
1281	3250	0.000986221	0.3182462743	0.318	0.0000001907	0.0000001899
1285	3316	0.001006249	0.3192525233	0.319	0.0000001999	0.0000000391
1288	3332	0.001011104	0.3202636275	0.32	0.0000002172	0.0000001233
1292	3269	0.000991987	0.3212556143	0.321	0.0000002035	0.0000000642
1296	3308	0.001003821	0.3222594356	0.322	0.0000002090	0.0000000146
1299	3419	0.001037505	0.3232969403	0.323	0.0000002730	0.0000014066
1303	3344	0.001014746	0.3243116859	0.324	0.0000002998	0.0000002174

1306	3349	0.001016263	0.3253279489	0.325	0.0000003309	0.0000002645
1310	3242	0.000983794	0.3263117424	0.326	0.0000002981	0.0000002627
1314	3217	0.000976207	0.3272879496	0.327	0.0000002536	0.0000005661
1317	3281	0.000995628	0.3282835777	0.328	0.0000002452	0.0000000191
1321	3169	0.000961641	0.3292452192	0.329	0.0000001828	0.0000014714
1325	3277	0.000994414	0.3302396335	0.33	0.0000001740	0.0000000312
1328	3379	0.001025367	0.3312650000	0.331	0.0000002122	0.0000006435
1332	3247	0.000985311	0.3322503108	0.332	0.0000001887	0.0000002158
1336	3284	0.000996539	0.3332468493	0.333	0.0000001830	0.0000000120
1339	3367	0.001021725	0.3342685744	0.334	0.0000002160	0.0000004720
1343	3289	0.000998056	0.3352666302	0.335	0.0000002122	0.0000000038
1347	3284	0.000996539	0.3362631687	0.336	0.0000002061	0.0000000120
1350	3312	0.001005035	0.3372682039	0.337	0.0000002135	0.0000000254
1354	3300	0.001001394	0.3382695977	0.338	0.0000002150	0.0000000019
1358	3245	0.000984704	0.3392543015	0.339	0.0000001908	0.0000002340
1362	3339	0.001013228	0.3402675299	0.34	0.0000002105	0.0000001750
1365	3249	0.000985918	0.3412534476	0.341	0.0000001884	0.0000001983
1369	3350	0.001016566	0.3422700140	0.342	0.0000002132	0.0000002744
1373	3334	0.001011711	0.3432817251	0.343	0.0000002314	0.0000001372
1377	3409	0.00103447	0.3443161952	0.344	0.0000002906	0.0000011882
1380	3311	0.001004732	0.3453209270	0.345	0.0000002985	0.0000000224
1384	3279	0.000995021	0.3463159482	0.346	0.0000002885	0.0000000248
1388	3209	0.00097378	0.3472897278	0.347	0.0000002419	0.0000006875
1392	3248	0.000985614	0.3482753420	0.348	0.0000002179	0.0000002070
1395	3308	0.001003821	0.3492791634	0.349	0.0000002233	0.0000000146
1399	3277	0.000994414	0.3502735777	0.35	0.0000002138	0.0000000312
1403	3374	0.001023849	0.3512974270	0.351	0.0000002520	0.0000005688
1407	3325	0.00100898	0.3523064071	0.352	0.0000002667	0.0000000806
1410	3277	0.000994414	0.3533008214	0.353	0.0000002564	0.0000000312
1414	3296	0.00100018	0.3543010014	0.354	0.0000002559	0.0000000000
1418	3281	0.000995628	0.3552966295	0.355	0.0000002479	0.0000000191
1422	3295	0.000999876	0.3562965060	0.356	0.0000002470	0.0000000000
1426	3327	0.001009587	0.3573060930	0.357	0.0000002624	0.0000000919
1430	3272	0.000992897	0.3582989901	0.358	0.0000002497	0.0000000505
1433	3184	0.000966193	0.3592651833	0.359	0.0000001959	0.0000011429
1437	3323	0.001008373	0.3602735565	0.36	0.0000002079	0.0000000701
1441	3222	0.000977724	0.3612512809	0.361	0.0000001749	0.0000004962
1445	3244	0.0009844	0.3622356814	0.362	0.0000001534	0.0000002433
1449	3221	0.000977421	0.3632131024	0.363	0.0000001251	0.0000005098
1453	3171	0.000962248	0.3641753507	0.364	0.0000000845	0.0000014252
1457	3248	0.000985614	0.3651609649	0.365	0.0000000710	0.0000002070
1460	3209	0.00097378	0.3661347445	0.366	0.0000000496	0.0000006875
1464	3303	0.001002304	0.3671370486	0.367	0.0000000512	0.0000000053
1468	3342	0.001014139	0.3681511874	0.368	0.0000000621	0.0000001999
1472	3211	0.000974386	0.3691255739	0.369	0.0000000427	0.0000006561
1476	3190	0.000968014	0.3700935878	0.37	0.0000000237	0.0000010231
1480	3243	0.000984097	0.3710776848	0.371	0.0000000163	0.0000002529
1484	3303	0.001002304	0.3720799889	0.372	0.0000000172	0.0000000053
1488	3317	0.001006552	0.3730865414	0.373	0.0000000201	0.0000000429
1492	3265	0.000990773	0.3740773143	0.374	0.0000000160	0.0000000851
1496	3202	0.000971655	0.3750489697	0.375	0.0000000064	0.0000008034
1500	3328	0.00100989	0.3760588601	0.376	0.0000000092	0.0000000978
1504	3314	0.001005642	0.3770645022	0.377	0.0000000110	0.0000000318
1508	3373	0.001023546	0.3780880480	0.378	0.0000000205	0.0000005544
1512	3240	0.000983187	0.3790712346	0.379	0.0000000134	0.0000002827
1516	3167	0.000961035	0.3800322692	0.38	0.0000000027	0.0000015183
1520	3332	0.001011104	0.3810433734	0.381	0.0000000049	0.0000001233
1524	3305	0.001002911	0.3820462844	0.382	0.0000000056	0.0000000085
1528	3407	0.001033863	0.3830801476	0.383	0.0000000168	0.0000011467
1532	3296	0.00100018	0.3840803276	0.384	0.0000000168	0.0000000000
1536	3319	0.001007159	0.3850874869	0.385	0.0000000199	0.0000000513
1540	3315	0.001005946	0.3860934325	0.386	0.0000000226	0.0000000353
1544	3419	0.001037505	0.3871309371	0.387	0.0000000443	0.0000014066
1548	3323	0.001008373	0.3881393103	0.388	0.0000000500	0.0000000701
1552	3361	0.001019904	0.3891592146	0.389	0.0000000652	0.0000003962
1556	3252	0.000986828	0.3901460427	0.39	0.0000000547	0.0000001735
1560	3278	0.000994718	0.3911407605	0.391	0.0000000507	0.0000000279
1564	3362	0.001020208	0.3921609683	0.392	0.0000000661	0.0000004084
1568	3317	0.001006552	0.3931675207	0.393	0.0000000714	0.0000000429
1572	3207	0.000973173	0.3941406934	0.394	0.0000000502	0.0000007197
1576	3427	0.001039932	0.3951806256	0.395	0.0000000826	0.0000015946
1580	3332	0.001011104	0.3961917299	0.396	0.0000000928	0.0000001233
1584	3362	0.001020208	0.3972119377	0.397	0.0000001131	0.0000004084
1589	3300	0.001001394	0.3982133315	0.398	0.0000001143	0.0000000019
1593	3369	0.001022332	0.3992356635	0.399	0.0000001392	0.0000004987
1597	3309	0.001004125	0.4002397883	0.4	0.0000001437	0.0000000170
1601	3355	0.001018084	0.4012578719	0.401	0.0000001658	0.0000003270
1605	3233	0.000981062	0.4022389344	0.402	0.0000001420	0.0000003586

1609	3272	0.000992897	0.4032318315	0.403	0.0000001334	0.0000000505
1613	3268	0.000991683	0.4042235147	0.404	0.0000001237	0.0000000692
1618	3292	0.000998966	0.4052224809	0.405	0.0000001222	0.0000000011
1622	3208	0.000973476	0.4061959570	0.406	0.0000000946	0.0000007035
1626	3245	0.000984704	0.4071806608	0.407	0.0000000802	0.0000002340
1630	3294	0.000999573	0.4081802339	0.408	0.0000000796	0.0000000002
1634	3307	0.001003518	0.4091837518	0.409	0.0000000826	0.0000000124
1639	3310	0.001004428	0.4101881801	0.41	0.0000000864	0.0000000196
1643	3324	0.001008677	0.4111968567	0.411	0.0000000943	0.0000000753
1647	3353	0.001017477	0.4122143335	0.412	0.0000001115	0.0000003054
1651	3273	0.000993201	0.4132075340	0.413	0.0000001043	0.0000000462
1656	3356	0.001018387	0.4142259211	0.414	0.0000001233	0.0000003381
1660	3337	0.001012622	0.4152385426	0.415	0.0000001371	0.0000001593
1664	3334	0.001011711	0.4162502538	0.416	0.0000001505	0.0000001372
1668	3313	0.001005339	0.4172555924	0.417	0.0000001567	0.0000000285
1673	3329	0.001010194	0.4182657863	0.418	0.0000001690	0.0000001039
1677	3233	0.000981062	0.4192468487	0.419	0.0000001454	0.0000003586
1681	3247	0.000985311	0.4202321595	0.42	0.0000001283	0.0000002158
1686	3297	0.001000483	0.4212326429	0.421	0.0000001286	0.0000000002
1690	3308	0.001003821	0.4222364643	0.422	0.0000001325	0.0000000146
1694	3289	0.000998056	0.4232345200	0.423	0.0000001300	0.0000000038
1699	3121	0.000947076	0.4241815958	0.424	0.0000000778	0.0000028010
1703	3283	0.000996235	0.4251778308	0.425	0.0000000744	0.0000000142
1707	3312	0.001005035	0.4261828660	0.426	0.0000000785	0.0000000254
1712	3240	0.000983187	0.4271660526	0.427	0.0000000646	0.0000002827
1716	3337	0.001012622	0.4281786741	0.428	0.0000000746	0.0000001593
1720	3222	0.000977724	0.4291563986	0.429	0.0000000570	0.0000004962
1725	3324	0.001008677	0.4301650752	0.43	0.0000000634	0.0000000753
1729	3245	0.000984704	0.4311497791	0.431	0.0000000521	0.0000002340
1734	3295	0.000999876	0.4321496556	0.432	0.0000000518	0.0000000000
1738	3211	0.000974386	0.4331240420	0.433	0.0000000355	0.0000006561
1742	3259	0.000988952	0.4341129942	0.434	0.0000000294	0.0000001221
1747	3241	0.00098349	0.4350964843	0.435	0.0000000214	0.0000002726
1751	3344	0.001014746	0.4361112300	0.436	0.0000000284	0.0000002174
1756	3277	0.000994414	0.4371056443	0.437	0.0000000255	0.0000000312
1760	3245	0.000984704	0.4380903482	0.438	0.0000000186	0.0000002340
1765	3478	0.001055408	0.4391457565	0.439	0.0000000484	0.0000030701
1769	3217	0.000976207	0.4401219637	0.44	0.0000000338	0.0000005661
1774	3466	0.001051767	0.4411737306	0.441	0.0000000684	0.0000026798
1778	3312	0.001005035	0.4421787658	0.442	0.0000000723	0.0000000254
1783	3285	0.000996842	0.4431756077	0.443	0.0000000696	0.0000000100
1787	3250	0.000986221	0.4441618289	0.444	0.0000000590	0.0000001899
1792	3296	0.00100018	0.4451620088	0.445	0.0000000590	0.0000000000
1796	3275	0.000993807	0.4461558163	0.446	0.0000000544	0.0000000383
1801	3293	0.00099927	0.4471550859	0.447	0.0000000538	0.0000000005
1806	3127	0.000948896	0.4481039823	0.448	0.0000000241	0.0000026116
1810	3369	0.001022332	0.4491263143	0.449	0.0000000355	0.0000004987
1815	3340	0.001013532	0.4501398462	0.45	0.0000000435	0.0000001831
1819	3297	0.001000483	0.4511403296	0.451	0.0000000437	0.0000000002
1824	3284	0.000996539	0.4521368681	0.452	0.0000000414	0.0000000120
1828	3287	0.000997449	0.4531343169	0.453	0.0000000398	0.0000000065
1833	3390	0.001028704	0.4541630214	0.454	0.0000000585	0.0000008239
1838	3296	0.00100018	0.4551632014	0.455	0.0000000585	0.0000000000
1842	3257	0.000988345	0.4561515467	0.456	0.0000000504	0.0000001358
1847	3368	0.001022029	0.4571735752	0.457	0.0000000659	0.0000004853
1852	3301	0.001001697	0.4581752724	0.458	0.0000000671	0.0000000029
1856	3205	0.000972566	0.4591478382	0.459	0.0000000476	0.0000007526
1861	3292	0.000998966	0.4601468043	0.46	0.0000000469	0.0000000011
1866	3275	0.000993807	0.4611406118	0.461	0.0000000429	0.0000000383
1871	3396	0.001030525	0.4621711370	0.462	0.0000000634	0.0000009318
1875	3304	0.001002608	0.4631737445	0.463	0.0000000652	0.0000000068
1880	3295	0.000999876	0.4641736210	0.464	0.0000000650	0.0000000000
1885	3221	0.000977421	0.4651510420	0.465	0.0000000491	0.0000005098
1889	3361	0.001019904	0.4661709464	0.466	0.0000000627	0.0000003962
1894	3253	0.000987131	0.4671580779	0.467	0.0000000535	0.0000001656
1899	3340	0.001013532	0.4681716098	0.468	0.0000000629	0.0000001831
1904	3300	0.001001394	0.4691730035	0.469	0.0000000638	0.0000000019
1909	3256	0.000988042	0.4701610454	0.47	0.0000000552	0.0000001430
1913	3257	0.000988345	0.4711493907	0.471	0.0000000474	0.0000001358
1918	3261	0.000989559	0.4721389498	0.472	0.0000000409	0.0000001090
1923	3185	0.000966497	0.4731054465	0.473	0.0000000235	0.0000011225
1928	3371	0.001022939	0.4741283854	0.474	0.0000000348	0.0000005262
1933	3232	0.000980759	0.4751091443	0.475	0.0000000251	0.0000003702
1938	3349	0.001016263	0.4761254073	0.476	0.0000000330	0.0000002645
1943	3253	0.000987131	0.4771125388	0.477	0.0000000266	0.0000001656
1947	3209	0.00097378	0.4780863183	0.478	0.0000000156	0.0000006875
1952	3334	0.001011711	0.4790980295	0.479	0.0000000201	0.0000001372
1957	3293	0.00099927	0.4800972991	0.48	0.0000000197	0.0000000005

1962	3317	0.001006552	0.4811038515	0.481	0.0000000224	0.0000000429
1967	3197	0.000970138	0.4820739896	0.482	0.0000000114	0.0000008917
1972	3167	0.000961035	0.4830350242	0.483	0.0000000025	0.0000015183
1977	3346	0.001015353	0.4840503768	0.484	0.0000000052	0.0000002357
1982	3281	0.000995628	0.4850460049	0.485	0.0000000044	0.0000000191
1987	3271	0.000992594	0.4860385986	0.486	0.0000000031	0.0000000549
1992	3317	0.001006552	0.4870451510	0.487	0.0000000042	0.0000000429
1997	3243	0.000984097	0.4880292480	0.488	0.0000000018	0.0000002529
2002	3309	0.001004125	0.4890333728	0.489	0.0000000023	0.0000000170
2007	3275	0.000993807	0.4900271803	0.49	0.0000000015	0.0000000383
2012	3248	0.000985614	0.4910127945	0.491	0.0000000003	0.0000002070
2017	3320	0.001007463	0.4920202573	0.492	0.0000000008	0.0000000557
2022	3199	0.000970745	0.4929910023	0.493	0.0000000002	0.0000008559
2027	3306	0.001003214	0.4939942168	0.494	0.0000000001	0.0000000103
2032	3179	0.000964676	0.4949588928	0.495	0.0000000034	0.0000012478
2037	3323	0.001008373	0.4959672660	0.496	0.0000000022	0.0000000701
2043	3339	0.001013228	0.4969804944	0.497	0.0000000008	0.0000001750
2048	3426	0.001039629	0.4980201232	0.498	0.0000000008	0.0000015704
2053	3239	0.000982883	0.4990030063	0.499	0.0000000000	0.0000002930
2058	3317	0.001006552	0.5000095588	0.5	0.0000000002	0.0000000429
2063	3275	0.000993807	0.5010033662	0.501	0.0000000000	0.0000000383
2068	3274	0.000993504	0.5019968702	0.502	0.0000000000	0.0000000422
2074	3292	0.000998966	0.5029958363	0.503	0.0000000000	0.0000000011
2079	3272	0.000992897	0.5039887334	0.504	0.0000000003	0.0000000505
2084	3333	0.001011408	0.5050001411	0.505	0.0000000000	0.0000001301
2089	3281	0.000995628	0.5059957693	0.506	0.0000000000	0.0000000191
2095	3305	0.001002911	0.5069986803	0.507	0.0000000000	0.0000000085
2100	3275	0.000993807	0.5079924877	0.508	0.0000000001	0.0000000383
2105	3260	0.000989256	0.5089817434	0.509	0.0000000007	0.0000001154
2110	3283	0.000996235	0.5099779784	0.51	0.0000000010	0.0000000142
2116	3290	0.000998359	0.5109763377	0.511	0.0000000011	0.0000000027
2121	3282	0.000995932	0.5119722693	0.512	0.0000000015	0.0000000166
2126	3303	0.001002304	0.5129745734	0.513	0.0000000013	0.0000000053
2132	3222	0.000977724	0.5139522978	0.514	0.0000000044	0.0000004962
2137	3323	0.001008373	0.5149606710	0.515	0.0000000030	0.0000000701
2142	3416	0.001036594	0.5159972653	0.516	0.0000000000	0.0000013391
2148	3447	0.001046001	0.5170432666	0.517	0.0000000036	0.0000021161
2153	3273	0.000993201	0.5180364671	0.518	0.0000000026	0.0000000462
2159	3391	0.001029008	0.5190654751	0.519	0.0000000083	0.0000008415
2164	3338	0.001012925	0.5200784000	0.52	0.0000000118	0.0000001671
2170	3314	0.001005642	0.5210840421	0.521	0.0000000136	0.0000000318
2175	3256	0.000988042	0.5220720840	0.522	0.0000000100	0.0000001430
2180	3260	0.000989256	0.5230613396	0.523	0.0000000072	0.0000001154
2186	3267	0.00099138	0.5240527194	0.524	0.0000000053	0.0000000743
2191	3240	0.000983187	0.5250359060	0.525	0.0000000025	0.0000002827
2197	3291	0.000998663	0.5260345687	0.526	0.0000000023	0.0000000018
2202	3359	0.001019297	0.5270538662	0.527	0.0000000055	0.0000003724
2208	3255	0.000987738	0.5280416046	0.528	0.0000000033	0.0000001503
2214	3316	0.001006249	0.5290478536	0.529	0.0000000043	0.0000000391
2219	3345	0.001015049	0.5300629027	0.53	0.0000000075	0.0000002265
2225	3213	0.000974993	0.5310378961	0.531	0.0000000027	0.0000006253
2230	3228	0.000979545	0.5320174412	0.532	0.0000000006	0.0000004184
2236	3250	0.000986221	0.5330036624	0.533	0.0000000000	0.0000001899
2242	3249	0.000985918	0.5339895800	0.534	0.0000000002	0.0000001983
2247	3259	0.000988952	0.5349785322	0.535	0.0000000009	0.0000001221
2253	3287	0.000997449	0.5359759811	0.536	0.0000000011	0.0000000065
2259	3275	0.000993807	0.5369697886	0.537	0.0000000017	0.0000000383
2264	3344	0.001014746	0.5379845342	0.538	0.0000000004	0.0000002174
2270	3212	0.00097469	0.5389592242	0.539	0.0000000031	0.0000006406
2276	3260	0.000989256	0.5399484798	0.54	0.0000000049	0.0000001154
2282	3302	0.001002001	0.5409504805	0.541	0.0000000045	0.0000000040
2287	3299	0.00100109	0.5419515708	0.542	0.0000000043	0.0000000012
2293	3318	0.001006856	0.5429584267	0.543	0.0000000032	0.0000000470
2299	3318	0.001006856	0.5439652826	0.544	0.0000000022	0.0000000470
2305	3207	0.000973173	0.5449384552	0.545	0.0000000070	0.0000007197
2311	3304	0.001002608	0.5459410628	0.546	0.0000000064	0.0000000068
2316	3296	0.00100018	0.5469412428	0.547	0.0000000063	0.0000000000
2322	3365	0.001021118	0.5479623609	0.548	0.0000000026	0.0000004460
2328	3377	0.00102476	0.5489871206	0.549	0.0000000003	0.0000006130
2334	3245	0.000984704	0.5499718244	0.55	0.0000000014	0.0000002340
2340	3261	0.000989559	0.5509613835	0.551	0.0000000027	0.0000001090
2346	3393	0.001029615	0.5519909984	0.552	0.0000000001	0.0000008770
2352	3305	0.001002911	0.5529939094	0.553	0.0000000001	0.0000000085
2358	3332	0.001011104	0.5540050136	0.554	0.0000000000	0.0000001233
2364	3334	0.001011711	0.5550167248	0.555	0.0000000005	0.0000001372
2370	3280	0.000995325	0.5560120495	0.556	0.0000000003	0.0000000219
2376	3331	0.001010801	0.5570228503	0.557	0.0000000009	0.0000001167
2382	3295	0.000999876	0.5580227268	0.558	0.0000000009	0.0000000000

2388	3219	0.000976814	0.5589995409	0.559	0.0000000000	0.0000005376
2394	3227	0.000979242	0.5599787826	0.56	0.0000000008	0.0000004309
2400	3279	0.000995021	0.5609738038	0.561	0.0000000012	0.0000000248
2406	3363	0.001020511	0.5619943151	0.562	0.0000000001	0.0000004207
2412	3204	0.000972262	0.5629665774	0.563	0.0000000020	0.0000007694
2418	3264	0.000990469	0.5639570469	0.564	0.0000000033	0.0000000908
2425	3357	0.001018691	0.5649757374	0.565	0.0000000010	0.0000003493
2431	3224	0.000978331	0.5659540688	0.566	0.0000000037	0.0000004695
2437	3307	0.001003518	0.5669575867	0.567	0.0000000032	0.0000000124
2443	3207	0.000973173	0.5679307594	0.568	0.0000000084	0.0000007197
2449	3243	0.000984097	0.5689148563	0.569	0.0000000127	0.0000002529
2456	3344	0.001014746	0.5699296020	0.57	0.0000000087	0.0000002174
2462	3295	0.000999876	0.5709294785	0.571	0.0000000087	0.0000000000
2468	3371	0.001022939	0.5719524174	0.572	0.0000000040	0.0000005262
2475	3368	0.001022029	0.5729744459	0.573	0.0000000011	0.0000004853
2481	3340	0.001013532	0.5739879778	0.574	0.0000000003	0.0000001831
2487	3245	0.000984704	0.5749726817	0.575	0.0000000013	0.0000002340
2494	3261	0.000989559	0.5759622408	0.576	0.0000000025	0.0000001090
2500	3243	0.000984097	0.5769463377	0.577	0.0000000050	0.0000002529
2506	3286	0.000997145	0.5779434832	0.578	0.0000000055	0.0000000081
2513	3315	0.001005946	0.5789494287	0.579	0.0000000044	0.0000000353
2519	3316	0.001006249	0.5799556777	0.58	0.0000000034	0.0000000391
2526	3325	0.00100898	0.5809646578	0.581	0.0000000021	0.0000000806
2532	3280	0.000995325	0.5819599825	0.582	0.0000000028	0.0000000219
2539	3302	0.001002001	0.5829619831	0.583	0.0000000025	0.0000000040
2545	3343	0.001014442	0.5839764254	0.584	0.0000000010	0.0000002086
2552	3215	0.0009756	0.5849520257	0.585	0.0000000039	0.0000005953
2558	3273	0.000993201	0.5859452262	0.586	0.0000000051	0.0000000462
2565	3397	0.001030829	0.5869760549	0.587	0.0000000010	0.0000009504
2571	3265	0.000990773	0.5879668278	0.588	0.0000000019	0.0000000851
2578	3387	0.001027794	0.5889946219	0.589	0.0000000000	0.0000007725
2585	3236	0.000981973	0.5899765947	0.59	0.0000000009	0.0000003250
2591	3239	0.000982883	0.5909594778	0.591	0.0000000028	0.0000002930
2598	3283	0.000996235	0.5919557129	0.592	0.0000000033	0.0000000142
2605	3312	0.001005035	0.5929607481	0.593	0.0000000026	0.0000000254
2612	3264	0.000990469	0.5939512176	0.594	0.0000000040	0.0000000908
2618	3419	0.001037505	0.5949887222	0.595	0.0000000002	0.0000014066
2625	3244	0.0009844	0.5959731226	0.596	0.0000000012	0.0000002433
2632	3284	0.000996539	0.5969696611	0.597	0.0000000015	0.0000000120
2639	3264	0.000990469	0.5979601306	0.598	0.0000000027	0.0000000908
2646	3322	0.00100807	0.5989682003	0.599	0.0000000017	0.0000000651
2652	3299	0.00100109	0.5999692906	0.6	0.0000000016	0.0000000012
2659	3203	0.000971959	0.6009412494	0.601	0.0000000057	0.0000007863
2666	3222	0.000977724	0.6019189739	0.602	0.0000000109	0.0000004962
2673	3295	0.000999876	0.6029188504	0.603	0.0000000109	0.0000000000
2680	3295	0.000999876	0.6039187269	0.604	0.0000000109	0.0000000000
2687	3368	0.001022029	0.6049407554	0.605	0.0000000058	0.0000004853
2694	3239	0.000982883	0.6059236386	0.606	0.0000000096	0.0000002930
2701	3259	0.000988952	0.6069125908	0.607	0.0000000126	0.0000001221
2708	3274	0.000993504	0.6079060948	0.608	0.0000000145	0.0000000422
2715	3284	0.000996539	0.6089026333	0.609	0.0000000156	0.0000000120
2722	3416	0.001036594	0.6099392275	0.61	0.0000000061	0.0000013391
2729	3337	0.001012622	0.6109518490	0.611	0.0000000038	0.0000001593
2737	3278	0.000994718	0.6119465668	0.612	0.0000000047	0.0000000279
2744	3297	0.001000483	0.6129470502	0.613	0.0000000046	0.0000000002
2751	3331	0.001010801	0.6139578510	0.614	0.0000000029	0.0000001167
2758	3280	0.000995325	0.6149531757	0.615	0.0000000036	0.0000000219
2765	3250	0.000986221	0.6159393969	0.616	0.0000000060	0.0000001899
2773	3386	0.001027491	0.6169668875	0.617	0.0000000018	0.0000007557
2780	3317	0.001006552	0.6179734400	0.618	0.0000000011	0.0000000429
2787	3224	0.000978331	0.6189517714	0.619	0.0000000038	0.0000004695
2795	3324	0.001008677	0.6199604480	0.62	0.0000000025	0.0000000753
2802	3200	0.000971048	0.6209314965	0.621	0.0000000076	0.0000008382
2809	3384	0.001026884	0.6219583803	0.622	0.0000000028	0.0000007227
2817	3303	0.001002304	0.6229606844	0.623	0.0000000025	0.0000000053
2824	3267	0.00099138	0.6239520642	0.624	0.0000000037	0.0000000743
2832	3236	0.000981973	0.6249340370	0.625	0.0000000070	0.0000003250
2839	3324	0.001008677	0.6259427136	0.626	0.0000000052	0.0000000753
2847	3353	0.001017477	0.6269601903	0.627	0.0000000025	0.0000003054
2854	3289	0.000998056	0.6279582461	0.628	0.0000000028	0.0000000038
2862	3216	0.000975904	0.6289341499	0.629	0.0000000069	0.0000005806
2869	3316	0.001006249	0.6299403989	0.63	0.0000000056	0.0000000391
2877	3326	0.001009284	0.6309496824	0.631	0.0000000040	0.0000000862
2885	3246	0.000985007	0.6319346897	0.632	0.0000000067	0.0000002248
2892	3290	0.000998359	0.6329330489	0.633	0.0000000071	0.0000000027
2900	3251	0.000986525	0.6339195735	0.634	0.0000000102	0.0000001816
2908	3268	0.000991683	0.6349112568	0.635	0.0000000124	0.0000000692
2915	3346	0.001015353	0.6359266094	0.636	0.0000000085	0.0000002357

2923	3301	0.001001697	0.6369283066	0.637	0.0000000081	0.0000000029
2931	3360	0.001019601	0.6379479075	0.638	0.0000000043	0.0000003842
2939	3305	0.001002911	0.6389508185	0.639	0.0000000038	0.0000000085
2947	3265	0.000990773	0.6399415914	0.64	0.0000000053	0.0000000851
2955	3270	0.00099229	0.6409338816	0.641	0.0000000068	0.0000000594
2963	3296	0.00100018	0.6419340616	0.642	0.0000000068	0.0000000000
2971	3297	0.001000483	0.6429345450	0.643	0.0000000067	0.0000000002
2979	3368	0.001022029	0.6439565735	0.644	0.0000000029	0.0000004853
2987	3313	0.001005339	0.6449619121	0.645	0.0000000022	0.0000000285
2995	3336	0.001012318	0.6459742302	0.646	0.0000000010	0.0000001517
3003	3391	0.001029008	0.6470032381	0.647	0.0000000000	0.0000008415
3011	3350	0.001016566	0.6480198045	0.648	0.0000000006	0.0000002744
3019	3360	0.001019601	0.6490394055	0.649	0.0000000024	0.0000003842
3027	3344	0.001014746	0.6500541511	0.65	0.0000000045	0.0000002174
3035	3290	0.000998359	0.6510525104	0.651	0.0000000042	0.0000000027
3044	3207	0.000973173	0.6520256830	0.652	0.0000000010	0.0000007197
3052	3353	0.001017477	0.6530431598	0.653	0.0000000029	0.0000003054
3060	3291	0.000998663	0.6540418225	0.654	0.0000000027	0.0000000018
3068	3241	0.00098349	0.6550253125	0.655	0.0000000010	0.0000002726
3077	3357	0.001018691	0.6560440031	0.656	0.0000000030	0.0000003493
3085	3250	0.000986221	0.6570302242	0.657	0.0000000014	0.0000001899
3094	3321	0.001007766	0.6580379905	0.658	0.0000000022	0.0000000603
3102	3227	0.000979242	0.6590172322	0.659	0.0000000005	0.0000004309
3111	3269	0.000991987	0.6600092189	0.66	0.0000000001	0.0000000642
3119	3289	0.000998056	0.6610072747	0.661	0.0000000001	0.0000000038
3128	3345	0.001015049	0.6620223238	0.662	0.0000000008	0.0000002265
3136	3241	0.00098349	0.6630058138	0.663	0.0000000001	0.0000002726
3145	3396	0.001030525	0.6640363391	0.664	0.0000000020	0.0000009318
3153	3320	0.001007463	0.6650438019	0.665	0.0000000029	0.0000000557
3162	3326	0.001009284	0.6660530854	0.666	0.0000000042	0.0000000862
3171	3307	0.001003518	0.6670566033	0.667	0.0000000048	0.0000000124
3180	3279	0.000995021	0.6680516246	0.668	0.0000000040	0.0000000248
3188	3275	0.000993807	0.6690454320	0.669	0.0000000031	0.0000000383
3197	3192	0.000968621	0.6700140529	0.67	0.0000000003	0.0000009846
3206	3287	0.000997449	0.6710115018	0.671	0.0000000002	0.0000000065
3215	3343	0.001014442	0.6720259440	0.672	0.0000000010	0.0000002086
3224	3266	0.000991076	0.6730170204	0.673	0.0000000004	0.0000000796
3233	3335	0.001012015	0.6740290350	0.674	0.0000000013	0.0000001444
3242	3367	0.001021725	0.6750507600	0.675	0.0000000038	0.0000004720
3251	3237	0.000982276	0.6760330363	0.676	0.0000000016	0.0000003141
3260	3263	0.000990166	0.6770232023	0.677	0.0000000008	0.0000000967
3269	3348	0.001015959	0.6780391618	0.678	0.0000000023	0.0000002547
3278	3324	0.001008677	0.6790478384	0.679	0.0000000034	0.0000000753
3287	3298	0.001000787	0.6800486253	0.68	0.0000000035	0.0000000006
3297	3353	0.001017477	0.6810661020	0.681	0.0000000064	0.0000003054
3306	3296	0.00100018	0.6820662819	0.682	0.0000000064	0.0000000000
3315	3376	0.001024456	0.6830907381	0.683	0.0000000121	0.0000005981
3325	3295	0.000999876	0.6840906146	0.684	0.0000000120	0.0000000000
3334	3217	0.000976207	0.6850668218	0.685	0.0000000065	0.0000005661
3343	3392	0.001029311	0.6860961332	0.686	0.0000000135	0.0000008592
3353	3256	0.000988042	0.6870841750	0.687	0.0000000103	0.0000001430
3362	3322	0.00100807	0.6880922448	0.688	0.0000000124	0.0000000651
3372	3240	0.000983187	0.6890754314	0.689	0.0000000083	0.0000002827
3381	3363	0.001020511	0.6900959426	0.69	0.0000000133	0.0000004207
3391	3241	0.00098349	0.6910794327	0.691	0.0000000091	0.0000002726
3401	3239	0.000982883	0.6920623158	0.692	0.0000000056	0.0000002930
3410	3381	0.001025973	0.6930882892	0.693	0.0000000112	0.0000006746
3420	3296	0.00100018	0.6940884692	0.694	0.0000000113	0.0000000000
3430	3350	0.001016566	0.6951050356	0.695	0.0000000159	0.0000002744
3440	3363	0.001020511	0.6961255469	0.696	0.0000000226	0.0000004207
3450	3308	0.001003821	0.6971293682	0.697	0.0000000240	0.0000000146
3460	3316	0.001006249	0.6981356172	0.698	0.0000000263	0.0000000391
3469	3292	0.000998966	0.6991345834	0.699	0.0000000259	0.0000000011
3479	3317	0.001006552	0.7001411358	0.7	0.0000000285	0.0000000429
3490	3470	0.001052981	0.7011941165	0.701	0.0000000538	0.0000028070
3500	3283	0.000996235	0.7021903516	0.702	0.0000000516	0.0000000142
3510	3304	0.001002608	0.7031929592	0.703	0.0000000530	0.0000000068
3520	3323	0.001008373	0.7042013323	0.704	0.0000000576	0.0000000701
3530	3221	0.000977421	0.7051787533	0.705	0.0000000453	0.0000005098
3540	3200	0.000971048	0.7061498018	0.706	0.0000000318	0.0000008382
3551	3282	0.000995932	0.7071457334	0.707	0.0000000300	0.0000000166
3561	3273	0.000993201	0.7081389340	0.708	0.0000000273	0.0000000462
3571	3288	0.000997752	0.7091366863	0.709	0.0000000264	0.0000000051
3582	3348	0.001015959	0.7101526458	0.71	0.0000000328	0.0000002547
3592	3290	0.000998359	0.7111510050	0.711	0.0000000321	0.0000000027
3603	3272	0.000992897	0.7121439021	0.712	0.0000000291	0.0000000505
3614	3286	0.000997145	0.7131410475	0.713	0.0000000279	0.0000000081
3624	3239	0.000982883	0.7141239307	0.714	0.0000000215	0.0000002930

3635	3308	0.001003821	0.7151277521	0.715	0.0000000228	0.0000000146
3646	3359	0.001019297	0.7161470495	0.716	0.0000000302	0.0000003724
3656	3207	0.000973173	0.7171202222	0.717	0.0000000202	0.0000007197
3667	3379	0.001025367	0.7181455887	0.718	0.0000000295	0.0000006435
3678	3231	0.000980456	0.7191260442	0.719	0.0000000221	0.0000003820
3689	3176	0.000963766	0.7200898098	0.72	0.0000000112	0.0000013129
3700	3243	0.000984097	0.7210739068	0.721	0.0000000076	0.0000002529
3711	3373	0.001023546	0.7220974526	0.722	0.0000000132	0.0000005544
3722	3351	0.00101687	0.7231143224	0.723	0.0000000181	0.0000002846
3733	3314	0.001005642	0.7241199645	0.724	0.0000000199	0.0000000318
3745	3276	0.000994111	0.7251140754	0.725	0.0000000179	0.0000000347
3756	3270	0.00099229	0.7261063656	0.726	0.0000000156	0.0000000594
3767	3308	0.001003821	0.7271101870	0.727	0.0000000167	0.0000000146
3779	3260	0.000989256	0.7280994426	0.728	0.0000000136	0.0000001154
3790	3311	0.001004732	0.7291041744	0.729	0.0000000149	0.0000000224
3802	3328	0.00100989	0.7301140648	0.73	0.0000000178	0.0000000978
3813	3192	0.000968621	0.7310826857	0.731	0.0000000094	0.0000009846
3825	3271	0.000992594	0.7320752793	0.732	0.0000000077	0.0000000549
3836	3263	0.000990166	0.7330654453	0.733	0.0000000058	0.0000000967
3848	3305	0.001002911	0.7340683564	0.734	0.0000000064	0.0000000085
3860	3229	0.000979849	0.7350482050	0.735	0.0000000032	0.0000004061
3872	3307	0.001003518	0.7360517229	0.736	0.0000000036	0.0000000124
3884	3202	0.000971655	0.7370233783	0.737	0.0000000007	0.0000008034
3895	3304	0.001002608	0.7380259859	0.738	0.0000000009	0.0000000068
3907	3315	0.001005946	0.7390319314	0.739	0.0000000014	0.0000000353
3920	3386	0.001027491	0.7400594221	0.74	0.0000000048	0.0000007557
3932	3373	0.001023546	0.7410829679	0.741	0.0000000093	0.0000005544
3944	3250	0.000986221	0.7420691890	0.742	0.0000000065	0.0000001899
3956	3339	0.001013228	0.7430824174	0.743	0.0000000091	0.0000001750
3968	3310	0.001004428	0.7440868457	0.744	0.0000000101	0.0000000196
3981	3313	0.001005339	0.7450921844	0.745	0.0000000114	0.0000000285
3993	3358	0.001018994	0.7461111784	0.746	0.0000000166	0.0000003608
4006	3283	0.000996235	0.7471074134	0.747	0.0000000154	0.0000000142
4018	3304	0.001002608	0.7481100210	0.748	0.0000000162	0.0000000068
4031	3363	0.001020511	0.7491305323	0.749	0.0000000227	0.0000004207
4044	3313	0.001005339	0.7501358709	0.75	0.0000000246	0.0000000285
4056	3243	0.000984097	0.7511199679	0.751	0.0000000192	0.0000002529
4069	3309	0.001004125	0.7521240927	0.752	0.0000000205	0.0000000170
4082	3320	0.001007463	0.7531315555	0.753	0.0000000230	0.0000000557
4095	3275	0.000993807	0.7541253630	0.754	0.0000000208	0.0000000383
4108	3405	0.001033256	0.7551586193	0.755	0.0000000333	0.0000011060
4121	3235	0.000981669	0.7561402886	0.756	0.0000000260	0.0000003360
4135	3433	0.001041753	0.7571820416	0.757	0.0000000438	0.0000017433
4148	3300	0.001001394	0.7581834353	0.758	0.0000000444	0.0000000019
4161	3331	0.001010801	0.7591942361	0.759	0.0000000497	0.0000001167
4175	3218	0.000976511	0.7601707467	0.76	0.0000000384	0.0000005518
4188	3174	0.000963159	0.7611339055	0.761	0.0000000236	0.0000013573
4202	3279	0.000995021	0.7621289267	0.762	0.0000000218	0.0000000248
4215	3252	0.000986828	0.7631157547	0.763	0.0000000176	0.0000001735
4229	3277	0.000994414	0.7641101691	0.764	0.0000000159	0.0000000312
4243	3363	0.001020511	0.7651306804	0.765	0.0000000223	0.0000004207
4257	3298	0.001000787	0.7661314672	0.766	0.0000000226	0.0000000006
4271	3312	0.001005035	0.7671365024	0.767	0.0000000243	0.0000000254
4285	3277	0.000994414	0.7681309168	0.768	0.0000000223	0.0000000312
4299	3203	0.000971959	0.7691028756	0.769	0.0000000138	0.0000007863
4313	3219	0.000976814	0.7700796897	0.77	0.0000000082	0.0000005376
4327	3218	0.000976511	0.7710562003	0.771	0.0000000041	0.0000005518
4341	3210	0.000974083	0.7720302834	0.772	0.0000000012	0.0000006717
4356	3372	0.001023242	0.7730535257	0.773	0.0000000037	0.0000005402
4370	3313	0.001005339	0.7740588644	0.774	0.0000000045	0.0000000285
4385	3389	0.001028401	0.7750872654	0.775	0.0000000098	0.0000008066
4400	3233	0.000981062	0.7760683278	0.776	0.0000000060	0.0000003586
4414	3247	0.000985311	0.7770536386	0.777	0.0000000037	0.0000002158
4429	3271	0.000992594	0.7780462322	0.778	0.0000000027	0.0000000549
4444	3258	0.000988649	0.7790348810	0.779	0.0000000016	0.0000001289
4459	3284	0.000996539	0.7800314195	0.78	0.0000000013	0.0000000120
4474	3213	0.000974993	0.7810064129	0.781	0.0000000001	0.0000006253
4490	3380	0.00102567	0.7820320828	0.782	0.0000000013	0.0000006589
4505	3306	0.001003214	0.7830352973	0.783	0.0000000016	0.0000000103
4520	3301	0.001001697	0.7840369945	0.784	0.0000000017	0.0000000029
4536	3279	0.000995021	0.7850320158	0.785	0.0000000013	0.0000000248
4551	3423	0.001038718	0.7860707342	0.786	0.0000000064	0.0000014991
4567	3271	0.000992594	0.7870633278	0.787	0.0000000051	0.0000000549
4583	3341	0.001013835	0.7880771632	0.788	0.0000000076	0.0000001914
4599	3180	0.000964979	0.7890421426	0.789	0.0000000023	0.0000012264
4615	3362	0.001020208	0.7900623504	0.79	0.0000000049	0.0000004084
4631	3239	0.000982883	0.7910452336	0.791	0.0000000026	0.0000002930
4647	3330	0.001010497	0.7920557309	0.792	0.0000000039	0.0000001102

4663	3282	0.000995932	0.7930516625	0.793	0.0000000034	0.0000000166
4680	3330	0.001010497	0.7940621598	0.794	0.0000000049	0.0000001102
4696	3380	0.00102567	0.7950878298	0.795	0.0000000097	0.0000006589
4713	3366	0.001021422	0.7961092515	0.796	0.0000000150	0.0000004589
4729	3293	0.00099927	0.7971085210	0.797	0.0000000148	0.0000000005
4746	3351	0.00101687	0.7981253909	0.798	0.0000000197	0.0000002846
4763	3231	0.000980456	0.7991058464	0.799	0.0000000140	0.0000003820
4780	3358	0.001018994	0.8001248404	0.8	0.0000000195	0.0000003608
4797	3336	0.001012318	0.8011371585	0.801	0.0000000235	0.0000001517
4815	3342	0.001014139	0.8021512972	0.802	0.0000000285	0.0000001999
4832	3337	0.001012622	0.8031639188	0.803	0.0000000335	0.0000001593
4850	3336	0.001012318	0.8041762368	0.804	0.0000000386	0.0000001517
4867	3262	0.000989863	0.8051660994	0.805	0.0000000343	0.0000001028
4885	3330	0.001010497	0.8061765967	0.806	0.0000000387	0.0000001102
4903	3253	0.000987131	0.8071637282	0.807	0.0000000332	0.0000001656
4921	3282	0.000995932	0.8081596598	0.808	0.0000000315	0.0000000166
4939	3280	0.000995325	0.8091549845	0.809	0.0000000297	0.0000000219
4957	3318	0.001006856	0.8101618404	0.81	0.0000000323	0.0000000470
4975	3293	0.00099927	0.8111611100	0.811	0.0000000320	0.0000000005
4994	3343	0.001014442	0.8121755522	0.812	0.0000000380	0.0000002086
5013	3242	0.000983794	0.8131593457	0.813	0.0000000312	0.0000002627
5031	3270	0.00099229	0.8141516359	0.814	0.0000000282	0.0000000594
5050	3334	0.001011711	0.8151633470	0.815	0.0000000327	0.0000001372
5069	3248	0.000985614	0.8161489613	0.816	0.0000000272	0.0000002070
5088	3199	0.000970745	0.8171197063	0.817	0.0000000175	0.0000008559
5108	3260	0.000989256	0.8181089620	0.818	0.0000000145	0.0000001154
5127	3215	0.0009756	0.8190845622	0.819	0.0000000087	0.0000005953
5147	3258	0.000988649	0.8200732110	0.82	0.0000000065	0.0000001289
5166	3330	0.001010497	0.8210837083	0.821	0.0000000085	0.0000001102
5186	3418	0.001037201	0.8221209095	0.822	0.0000000178	0.0000013839
5206	3288	0.000997752	0.8231186618	0.823	0.0000000171	0.0000000051
5226	3248	0.000985614	0.8241042760	0.824	0.0000000132	0.0000002070
5247	3240	0.000983187	0.8250874626	0.825	0.0000000093	0.0000002827
5267	3326	0.001009284	0.8260967462	0.826	0.0000000113	0.0000000862
5288	3303	0.001002304	0.8270990503	0.827	0.0000000119	0.0000000053
5308	3317	0.001006552	0.8281056027	0.828	0.0000000135	0.0000000429
5329	3240	0.000983187	0.8290887893	0.829	0.0000000095	0.0000002827
5350	3241	0.00098349	0.8300722794	0.83	0.0000000063	0.0000002726
5372	3331	0.001010801	0.8310830802	0.831	0.0000000083	0.0000001167
5393	3258	0.000988649	0.8320717289	0.832	0.0000000062	0.0000001289
5415	3218	0.000976511	0.8330482396	0.833	0.0000000028	0.0000005518
5436	3296	0.00100018	0.8340484195	0.834	0.0000000028	0.0000000000
5458	3264	0.000990469	0.8350388890	0.835	0.0000000018	0.0000000908
5480	3238	0.00098258	0.8360214687	0.836	0.0000000006	0.0000003035
5503	3308	0.001003821	0.8370252900	0.837	0.0000000008	0.0000000146
5525	3138	0.000952234	0.8379775245	0.838	0.0000000006	0.0000022815
5548	3266	0.000991076	0.8389686008	0.839	0.0000000012	0.0000000796
5571	3321	0.001007766	0.8399763671	0.84	0.0000000007	0.0000000603
5594	3209	0.00097378	0.8409501467	0.841	0.0000000030	0.0000006875
5617	3363	0.001020511	0.8419706579	0.842	0.0000000010	0.0000004207
5640	3203	0.000971959	0.8429426168	0.843	0.0000000039	0.0000007863
5664	3277	0.000994414	0.8439370311	0.844	0.0000000047	0.0000000312
5688	3318	0.001006856	0.8449438871	0.845	0.0000000037	0.0000000470
5712	3362	0.001020208	0.8459640949	0.846	0.0000000015	0.0000004084
5736	3225	0.000978635	0.8469427297	0.847	0.0000000039	0.0000004565
5760	3122	0.000947379	0.8478901089	0.848	0.0000000142	0.0000027690
5785	3305	0.001002911	0.8488930199	0.849	0.0000000135	0.0000000085
5810	3308	0.001003821	0.8498968413	0.85	0.0000000125	0.0000000146
5835	3289	0.000998056	0.8508948970	0.851	0.0000000130	0.0000000038
5860	3323	0.001008373	0.8519032702	0.852	0.0000000110	0.0000000701
5886	3324	0.001008677	0.8529119468	0.853	0.0000000091	0.0000000753
5911	3256	0.000988042	0.8538999887	0.854	0.0000000117	0.0000001430
5937	3297	0.001000483	0.8549004721	0.855	0.0000000116	0.0000000002
5964	3297	0.001000483	0.8559009555	0.856	0.0000000115	0.0000000002
5990	3273	0.000993201	0.8568941560	0.857	0.0000000131	0.0000000462
6017	3280	0.000995325	0.8578894807	0.858	0.0000000142	0.0000000219
6044	3400	0.001031739	0.8589212197	0.859	0.0000000072	0.0000010074
6071	3263	0.000990166	0.8599113858	0.86	0.0000000091	0.0000000967
6098	3331	0.001010801	0.8609221865	0.861	0.0000000070	0.0000001167
6126	3287	0.000997449	0.8619196354	0.862	0.0000000075	0.0000000065
6154	3270	0.00099229	0.8629119256	0.863	0.0000000090	0.0000000594
6182	3243	0.000984097	0.8638960226	0.864	0.0000000125	0.0000002529
6211	3312	0.001005035	0.8649010577	0.865	0.0000000113	0.0000000254
6239	3322	0.00100807	0.8659091275	0.866	0.0000000095	0.0000000651
6268	3293	0.00099927	0.8669083971	0.867	0.0000000097	0.0000000005
6298	3359	0.001019297	0.8679276945	0.868	0.0000000060	0.0000003724
6327	3389	0.001028401	0.8689560956	0.869	0.0000000022	0.0000008066
6357	3318	0.001006856	0.8699629515	0.87	0.0000000016	0.0000000470

6388	3422	0.001038415	0.8710013664	0.871	0.0000000000	0.0000014757
6418	3225	0.000978635	0.8719800013	0.872	0.0000000005	0.0000004565
6449	3177	0.000964069	0.8729440703	0.873	0.0000000036	0.0000012910
6480	3288	0.000997752	0.8739418227	0.874	0.0000000039	0.0000000051
6512	3286	0.000997145	0.8749389681	0.875	0.0000000043	0.0000000081
6543	3198	0.000970442	0.8759094097	0.876	0.0000000094	0.0000008737
6576	3336	0.001012318	0.8769217277	0.877	0.0000000070	0.0000001517
6608	3351	0.00101687	0.8779385976	0.878	0.0000000043	0.0000002846
6641	3393	0.001029615	0.8789682124	0.879	0.0000000011	0.0000008770
6674	3329	0.001010194	0.8799784063	0.88	0.0000000005	0.0000001039
6708	3185	0.000966497	0.8809449030	0.881	0.0000000034	0.0000011225
6742	3350	0.001016566	0.8819614694	0.882	0.0000000017	0.0000002744
6776	3295	0.000999876	0.8829613459	0.883	0.0000000017	0.0000000000
6811	3225	0.000978635	0.8839399807	0.884	0.0000000041	0.0000004565
6846	3253	0.000987131	0.8849271122	0.885	0.0000000060	0.0000001656
6881	3271	0.000992594	0.8859197058	0.886	0.0000000073	0.0000000549
6917	3345	0.001015049	0.8869347549	0.887	0.0000000048	0.0000002265
6954	3335	0.001012015	0.8879467695	0.888	0.0000000032	0.0000001444
6990	3231	0.000980456	0.8889272251	0.889	0.0000000060	0.0000003820
7028	3282	0.000995932	0.8899231567	0.89	0.0000000066	0.0000000166
7065	3337	0.001012622	0.8909357782	0.891	0.0000000046	0.0000001593
7103	3305	0.001002911	0.8919386892	0.892	0.0000000042	0.0000000085
7142	3285	0.000996842	0.8929355312	0.893	0.0000000047	0.0000000100
7181	3219	0.000976814	0.8939123453	0.894	0.0000000086	0.0000005376
7221	3250	0.000986221	0.8948985664	0.895	0.0000000115	0.0000001899
7261	3204	0.000972262	0.8958708287	0.896	0.0000000186	0.0000007694
7301	3185	0.000966497	0.8968373254	0.897	0.0000000295	0.0000011225
7342	3226	0.000978938	0.8978162637	0.898	0.0000000376	0.0000004436
7384	3342	0.001014139	0.8988304024	0.899	0.0000000320	0.0000001999
7426	3289	0.000998056	0.8998284582	0.9	0.0000000327	0.0000000038
7468	3286	0.000997145	0.9008256036	0.901	0.0000000338	0.0000000081
7512	3328	0.00100989	0.9018354941	0.902	0.0000000300	0.0000000978
7555	3339	0.001013228	0.9028487225	0.903	0.0000000253	0.0000001750
7600	3365	0.001021118	0.9038698407	0.904	0.0000000187	0.0000004460
7645	3209	0.00097378	0.9048436202	0.905	0.0000000270	0.0000006875
7690	3340	0.001013532	0.9058571521	0.906	0.0000000225	0.0000001831
7736	3248	0.000985614	0.9068427663	0.907	0.0000000273	0.0000002070
7783	3272	0.000992897	0.9078356634	0.908	0.0000000297	0.0000000505
7831	3321	0.001007766	0.9088434297	0.909	0.0000000270	0.0000000603
7879	3398	0.001031132	0.9098745618	0.91	0.0000000173	0.0000009692
7928	3324	0.001008677	0.9108832384	0.911	0.0000000150	0.0000000753
7978	3315	0.001005946	0.9118891839	0.912	0.0000000135	0.0000000353
8028	3256	0.000988042	0.9128772258	0.913	0.0000000165	0.0000001430
8079	3315	0.001005946	0.9138831713	0.914	0.0000000149	0.0000000353
8131	3250	0.000986221	0.9148693925	0.915	0.0000000186	0.0000001899
8184	3319	0.001007159	0.9158765518	0.916	0.0000000166	0.0000000513
8237	3262	0.000989863	0.9168664144	0.917	0.0000000195	0.0000001028
8292	3319	0.001007159	0.9178735737	0.918	0.0000000174	0.0000000513
8347	3249	0.000985918	0.9188594914	0.919	0.0000000215	0.0000001983
8403	3302	0.001002001	0.9198614921	0.92	0.0000000209	0.0000000040
8460	3402	0.001032346	0.9208938380	0.921	0.0000000122	0.0000010463
8518	3383	0.00102658	0.9219204183	0.922	0.0000000069	0.0000007065
8577	3271	0.000992594	0.9229130120	0.923	0.0000000082	0.0000000549
8637	3380	0.00102567	0.9239386819	0.924	0.0000000041	0.0000006589
8698	3286	0.000997145	0.9249358274	0.925	0.0000000045	0.0000000081
8760	3249	0.000985918	0.9259217450	0.926	0.0000000066	0.0000001983
8823	3268	0.000991683	0.9269134283	0.927	0.0000000081	0.0000000692
8888	3363	0.001020511	0.9279339396	0.928	0.0000000047	0.0000004207
8953	3298	0.001000787	0.9289347264	0.929	0.0000000046	0.0000000006
9020	3266	0.000991076	0.9299258028	0.93	0.0000000059	0.0000000796
10000	230923	0.070074197	1.0000000000	1	0.0000000000	0.0000000786
totals	3295407				0.0001031470	0.0002746239

File Size Distribution - Tail

File Size Distribution (tail) Chi Squared verification of simulation output for selected file sizes.

Column 1: file size.

Column 2: cumulative total of the percentages

Column 3: expected cumulative percentage

Column 4: Chi Squared calculation of the cumulative total

File Size	Cumulative	Expected	Chi Sq
9021	0.000125583	0.000110852	0.0000019575
10021	0.099916422	0.099890231	0.0000000069
11021	0.181675277	0.181562472	0.0000000701
12021	0.249498751	0.249646452	0.0000000874
13021	0.307154333	0.307272867	0.0000000457
14021	0.355993123	0.356679267	0.0000013199
15021	0.399146036	0.399507356	0.0000003268
16021	0.437336255	0.436988952	0.0000002760
17022	0.470187032	0.470097521	0.0000000170
18023	0.499590773	0.49952838	0.0000000078
19026	0.52567306	0.52591191	0.0000001085
20033	0.549356279	0.549742924	0.0000002719
21042	0.571541163	0.571333523	0.0000000755
22054	0.591227379	0.5910039	0.0000000845
23071	0.608990876	0.609032985	0.0000000029
24101	0.625333986	0.62574167	0.0000002656
25131	0.640438588	0.641080737	0.0000006432
26166	0.654607813	0.655277841	0.0000006851
27232	0.668270376	0.668772033	0.0000003763
28295	0.681014884	0.681215763	0.0000000592
29386	0.693040537	0.693051113	0.0000000002
30501	0.704273719	0.704271991	0.0000000000
31636	0.715047873	0.71488178	0.0000000386
32785	0.725085851	0.72487418	0.0000000618
33961	0.734868333	0.734401225	0.0000002971
35173	0.743957943	0.743553294	0.0000002202
36407	0.752857013	0.752245447	0.0000004972
37704	0.761180134	0.760768088	0.0000002232
39060	0.769061549	0.769073221	0.0000000002
40451	0.776968946	0.777014165	0.0000000026
41831	0.784495265	0.784370443	0.0000000199
43263	0.791753095	0.791507755	0.0000000760
44773	0.798790073	0.798539298	0.0000000788
46337	0.805601867	0.805339146	0.0000000857
47993	0.812227452	0.812055925	0.0000000362
49642	0.818818394	0.818299021	0.0000003296
51505	0.825192813	0.824871372	0.0000001253
53316	0.831342049	0.830820017	0.0000003280
55306	0.837266102	0.836907388	0.0000001538
57354	0.843164172	0.842731109	0.0000002225

59542	0.849010276	0.848510295	0.0000002946
61758	0.854873703	0.853946047	0.0000010077
64069	0.860433998	0.859214285	0.0000017315
66686	0.865894692	0.864739226	0.0000015439
69371	0.871355387	0.869974485	0.0000021919
72423	0.876746794	0.87545393	0.0000019093
75480	0.881865384	0.880498145	0.0000021230
78857	0.887118217	0.885615735	0.0000025490
82550	0.892228145	0.890732889	0.0000025101
86576	0.897130212	0.895814082	0.0000019337
91202	0.902162193	0.90109866	0.0000012552
95962	0.906938677	0.90600446	0.0000009633
101158	0.911702169	0.910832559	0.0000008303
106727	0.916435349	0.915485304	0.0000009859
113357	0.921138215	0.920428381	0.0000005474
120628	0.925810768	0.925224658	0.0000003713
128570	0.930453008	0.929843665	0.0000003993
137792	0.935043283	0.934539015	0.0000002721
147785	0.939603244	0.938965389	0.0000004333
159835	0.944093919	0.943566803	0.0000002945
173579	0.948571602	0.948035189	0.0000003035
190457	0.953036294	0.952640229	0.0000001647
209905	0.95745335	0.957028179	0.0000001889
234471	0.961874738	0.961530424	0.0000001233
263666	0.966283133	0.965790053	0.0000002517
302704	0.970665547	0.970201913	0.0000002216
351957	0.975060951	0.974371869	0.0000004873
430180	0.97940439	0.979032033	0.0000001416
544332	0.983765151	0.983429231	0.0000001147
755384	0.98810426	0.988059053	0.0000000021
1218701	0.992443369	0.992598677	0.0000000243
2861463	0.996773816	0.996847766	0.0000000055
			0.0000356613

Hourly Request Rate

Hourly request rate Chi-squared verification.

Column 1: Hour of the day

Column 2: Approximated rate

Column 3: Measured output data.

Column 4:Chi-squared verification

Hour	Approximation	Measured	Chi Squared
0	2.816666667	1.971774268	0.005990381
1	2.533333333	1.882524636	0.124479704
2	2.25	1.800700574	0.060016952
3	2.033333333	2.071591057	0.02661541
4	2	1.909942123	0.00256264
5	2.033333333	1.986768207	0.007487897
6	2.416666667	1.976915047	0.076474215
7	3.066666667	2.86855457	0.387247367
8	4.216666667	3.942120543	0.431005429
9	4.95	4.812768552	0.205216364
10	5.766666667	6.010570012	0.157789875
11	6.083333333	5.917321997	0.000870329
12	6.383333333	5.918749991	0.034020872
13	6.45	7.171100826	0.043756058
14	6.666666667	7.37401879	0.038168073
15	6.583333333	7.28177037	0.094964581
16	6.25	7.10241431	0.170328015
17	5.333333333	5.266299482	0.586808907
18	4.533333333	4.705240596	0.118508686
19	4.066666667	4.520029759	0.10027295
20	3.916666667	3.401053574	0.092948176
21	3.783333333	3.723780243	0.038626735
22	3.516666667	3.697219553	0.012197924
23	3.283333333	2.68677092	0.052173138
			2.868530677

Active Off Time

Active off time Chi-squared verification.

Column 1: Seconds of off time

Column 2: Sample count from simulations.

Column 3: Probability density function of measured output.

Column 4: Cumulative density function of measured output.

Column 5: Expected value (cdf).

Column 6: Chi-squared verification.

Seconds	Count	pdf	cdf	Expected	Chi-squared
0	41368	0.575883287	0.575883287	0.579117101	0.0000180578
1	7030	0.097864521	0.673747807	0.676237277	0.0000091646
2	4101	0.057089957	0.730837765	0.731974133	0.0000017642
3	2677	0.037266476	0.76810424	0.769986404	0.0000046008
4	2051	0.028551939	0.79665618	0.798180661	0.0000029117
5	1534	0.02135479	0.81801097	0.820183264	0.0000057534
6	1261	0.017554361	0.835565331	0.837956101	0.0000068211
7	1115	0.015521898	0.851087229	0.852676882	0.0000029636
8	921	0.012821227	0.863908456	0.86510552	0.0000016564
9	713	0.009925662	0.873834118	0.875759203	0.0000042317
10	648	0.009020798	0.882854916	0.885004699	0.0000052221
11	601	0.008366512	0.891221427	0.893110762	0.0000039968
12	491	0.006835203	0.898056631	0.900279656	0.0000054892
13	474	0.006598547	0.904655177	0.90666703	0.0000044642
14	388	0.005401342	0.910056519	0.912394937	0.0000059932
15	372	0.005178606	0.915235125	0.917560643	0.0000058939
16	377	0.005248211	0.920483337	0.922242752	0.0000033565
17	310	0.004315505	0.924798842	0.926505576	0.0000031440
18	249	0.003466325	0.928265167	0.9304023	0.0000049090
19	268	0.003730824	0.931995991	0.933977341	0.0000042033
20	213	0.00296517	0.934961161	0.937268113	0.0000056782
21	237	0.003299273	0.938260434	0.940306377	0.0000044516
22	208	0.002895565	0.941155999	0.943119289	0.0000040870
23	182	0.002533619	0.943689618	0.945730217	0.0000044030
24	167	0.002324804	0.946014422	0.948159388	0.0000048524
25	159	0.002213437	0.948227859	0.950424406	0.0000050765
26	159	0.002213437	0.950441295	0.952540669	0.0000046270
27	149	0.002074227	0.952515522	0.954521706	0.0000042165
28	134	0.001865412	0.954380934	0.956379459	0.0000041763
29	130	0.001809728	0.956190662	0.958124505	0.0000039032
30	132	0.00183757	0.958028232	0.959766255	0.0000031474
31	108	0.001503466	0.959531698	0.961313104	0.0000033011
32	102	0.00141994	0.960951639	0.96277257	0.0000034440
33	90	0.001252889	0.962204527	0.964151405	0.0000039313
34	102	0.00141994	0.963624468	0.965455691	0.0000034734
35	94	0.001308573	0.96493304	0.96669092	0.0000031966
36	99	0.001378178	0.966311218	0.967862064	0.0000024850

37	96	0.001336415	0.967647632	0.96897364	0.0000018146
38	70	0.000974469	0.968622101	0.970029753	0.0000020427
39	57	0.000793496	0.969415597	0.971034148	0.0000026979
40	79	0.001099758	0.970515355	0.971990246	0.0000022380
41	73	0.001016232	0.971531587	0.97290118	0.0000019280
42	56	0.000779575	0.972311162	0.973769823	0.0000021850
43	76	0.001057995	0.973369157	0.974598814	0.0000015515
44	68	0.000946627	0.974315784	0.975390584	0.0000011843
45	55	0.000765654	0.975081438	0.976147373	0.0000011640
46	65	0.000904864	0.975986302	0.97687125	0.0000008017
47	52	0.000723891	0.976710193	0.977564127	0.0000007459
48	51	0.00070997	0.977420163	0.978227777	0.0000006668
49	61	0.00084918	0.978269343	0.978863843	0.0000003611
50	48	0.000668207	0.978937551	0.979473851	0.0000002936
51	49	0.000682128	0.979619679	0.980059221	0.0000001971
52	37	0.000515076	0.980134755	0.980621274	0.0000002414
53	38	0.000528997	0.980663753	0.981161241	0.0000002522
54	43	0.000598602	0.981262355	0.981680273	0.0000001779
55	38	0.000528997	0.981791352	0.982179441	0.0000001533
56	45	0.000626444	0.982417797	0.982659752	0.0000000596
57	32	0.000445472	0.982863268	0.983122144	0.0000000682
58	30	0.00041763	0.983280898	0.9835675	0.0000000835
59	23	0.000320183	0.98360108	0.983996645	0.0000001590
60	25	0.000348025	0.983949105	0.984410356	0.0000002161
61	27	0.000375867	0.984324972	0.984809361	0.0000002383
62	29	0.000403709	0.98472868	0.985194347	0.0000002201
63	24	0.000334104	0.985062784	0.98556596	0.0000002569
64	24	0.000334104	0.985396887	0.985924807	0.0000002827
65	23	0.000320183	0.98571707	0.986271463	0.0000003116
66	19	0.000264499	0.985981569	0.986606468	0.0000003958
67	25	0.000348025	0.986329593	0.986930334	0.0000003657
68	15	0.000208815	0.986538408	0.987243544	0.0000005036
69	29	0.000403709	0.986942117	0.987546555	0.0000003700
70	24	0.000334104	0.98727622	0.987839798	0.0000003215
71	20	0.00027842	0.98755464	0.988123685	0.0000003277
72	19	0.000264499	0.987819139	0.988398601	0.0000003397
73	16	0.000222736	0.988041874	0.988664916	0.0000003926
74	21	0.000292341	0.988334215	0.988922979	0.0000003505
75	28	0.000389788	0.988724003	0.98917312	0.0000002039
76	22	0.000306262	0.989030264	0.989415655	0.0000001501
77	12	0.000167052	0.989197316	0.989650882	0.0000002079
78	11	0.000153131	0.989350447	0.989879087	0.0000002823
79	20	0.00027842	0.989628867	0.990100539	0.0000002247
80	19	0.000264499	0.989893365	0.990315498	0.0000001799
81	13	0.000180973	0.990074338	0.990524208	0.0000002043
82	11	0.000153131	0.990227469	0.990726903	0.0000002518
83	22	0.000306262	0.990533731	0.990923806	0.0000001536
84	14	0.000194894	0.990728624	0.991115131	0.0000001507
85	12	0.000167052	0.990895676	0.99130108	0.0000001658
86	13	0.000180973	0.991076649	0.991481848	0.0000001656
87	14	0.000194894	0.991271543	0.991657619	0.0000001503
88	13	0.000180973	0.991452516	0.991828571	0.0000001426
89	11	0.000153131	0.991605646	0.991994873	0.0000001527
90	10	0.00013921	0.991744856	0.992156686	0.0000001709
91	12	0.000167052	0.991911908	0.992314167	0.0000001631
92	8	0.000111368	0.992023276	0.992467463	0.0000001988
93	8	0.000111368	0.992134644	0.992616716	0.0000002341
94	12	0.000167052	0.992301696	0.992762063	0.0000002135
95	10	0.00013921	0.992440905	0.992903633	0.0000002156
96	16	0.000222736	0.992663641	0.993041553	0.0000001438
97	13	0.000180973	0.992844614	0.993175941	0.0000001105
98	9	0.000125289	0.992969903	0.993306914	0.0000001143
99	7	9.74469E-05	0.99306735	0.993434581	0.0000001358
100	7	9.74469E-05	0.993164797	0.993559049	0.0000001564
101	9	0.000125289	0.993290086	0.993680419	0.0000001533

102	9	0.000125289	0.993415374	0.993798789	0.0000001479
103	9	0.000125289	0.993540663	0.993914255	0.0000001404
104	4	5.56839E-05	0.993596347	0.994026905	0.0000001865
105	6	8.35259E-05	0.993679873	0.994136827	0.0000002100
106	6	8.35259E-05	0.993763399	0.994244104	0.0000002324
107	13	0.000180973	0.993944372	0.994348818	0.0000001645
108	10	0.00013921	0.994083582	0.994451045	0.0000001358
109	6	8.35259E-05	0.994167108	0.994550861	0.0000001481
110	14	0.000194894	0.994362001	0.994648337	0.0000000824
111	8	0.000111368	0.994473369	0.994743543	0.0000000734
112	7	9.74469E-05	0.994570816	0.994836545	0.0000000710
113	7	9.74469E-05	0.994668263	0.994927408	0.0000000675
114	4	5.56839E-05	0.994723947	0.995016193	0.0000000858
115	4	5.56839E-05	0.994779631	0.995102961	0.0000001051
116	8	0.000111368	0.994890999	0.995187769	0.0000000885
117	10	0.00013921	0.995030209	0.995270674	0.0000000581
118	5	6.96049E-05	0.995099814	0.995351728	0.0000000638
119	2	0.000027842	0.995127655	0.995430983	0.0000000924
120	4	5.56839E-05	0.995183339	0.995508491	0.0000001062
121	8	0.000111368	0.995294707	0.995584299	0.0000000842
122	3	0.000041763	0.99533647	0.995658454	0.0000001041
123	4	5.56839E-05	0.995392154	0.995731002	0.0000001153
124	6	8.35259E-05	0.99547568	0.995801985	0.0000001069
125	6	8.35259E-05	0.995559206	0.995871447	0.0000000979
126	7	9.74469E-05	0.995656653	0.995939428	0.0000000803
127	7	9.74469E-05	0.9957541	0.996005967	0.0000000637
128	4	5.56839E-05	0.995809784	0.996071104	0.0000000686
129	3	0.000041763	0.995851547	0.996134875	0.0000000806
130	8	0.000111368	0.995962915	0.996197315	0.0000000552
131	8	0.000111368	0.996074282	0.99625846	0.0000000340
132	3	0.000041763	0.996116045	0.996318343	0.0000000411
133	7	9.74469E-05	0.996213492	0.996376997	0.0000000268
134	3	0.000041763	0.996255255	0.996434453	0.0000000322
135	7	9.74469E-05	0.996352702	0.996490741	0.0000000191
136	3	0.000041763	0.996394465	0.996545891	0.0000000230
137	1	0.000013921	0.996408386	0.996599933	0.0000000368
138	3	0.000041763	0.996450149	0.996652893	0.0000000412
139	3	0.000041763	0.996491912	0.996704799	0.0000000455
140	4	5.56839E-05	0.996547596	0.996755676	0.0000000434
141	4	5.56839E-05	0.99660328	0.99680555	0.0000000410
142	3	0.000041763	0.996645043	0.996854447	0.0000000440
143	1	0.000013921	0.996658964	0.996902388	0.0000000594
144	8	0.000111368	0.996770332	0.996949399	0.0000000322
145	5	6.96049E-05	0.996839937	0.9969955	0.0000000243
147	1	0.000013921	0.996853858	0.997085063	0.0000000536
148	3	0.000041763	0.996895621	0.997128567	0.0000000544
149	1	0.000013921	0.996909541	0.997171245	0.0000000687
150	3	0.000041763	0.996951304	0.997213117	0.0000000687
151	3	0.000041763	0.996993067	0.997254202	0.0000000684
152	5	6.96049E-05	0.997062672	0.997294518	0.0000000539
153	2	0.000027842	0.997090514	0.997334084	0.0000000595
154	1	0.000013921	0.997104435	0.997372915	0.0000000723
155	2	0.000027842	0.997132277	0.99741103	0.0000000779
156	6	8.35259E-05	0.997215803	0.997448445	0.0000000543
158	5	6.96049E-05	0.997285408	0.997521236	0.0000000558
159	5	6.96049E-05	0.997355013	0.997556643	0.0000000408
160	6	8.35259E-05	0.997438539	0.997591411	0.0000000234
161	1	0.000013921	0.99745246	0.997625554	0.0000000300
162	2	0.000027842	0.997480302	0.997659085	0.0000000320
164	3	0.000041763	0.997522065	0.997724368	0.0000000410
165	4	5.56839E-05	0.997577749	0.997756145	0.0000000319
166	3	0.000041763	0.997619512	0.997787363	0.0000000282
167	5	6.96049E-05	0.997689117	0.997818033	0.0000000167
168	1	0.000013921	0.997703038	0.997848169	0.0000000211
169	2	0.000027842	0.99773088	0.99787778	0.0000000216

170	2	0.000027842	0.997758722	0.997906878	0.0000000220
171	2	0.000027842	0.997786564	0.997935475	0.0000000222
173	2	0.000027842	0.997814405	0.997991206	0.0000000313
174	1	0.000013921	0.997828326	0.99801836	0.0000000362
175	3	0.000041763	0.997870089	0.998045053	0.0000000307
176	1	0.000013921	0.99788401	0.998071295	0.0000000351
178	3	0.000041763	0.997925773	0.998122462	0.0000000388
179	3	0.000041763	0.997967536	0.998147406	0.0000000324
181	3	0.000041763	0.998009299	0.998196056	0.0000000349
182	1	0.000013921	0.99802322	0.998219779	0.0000000387
183	2	0.000027842	0.998051062	0.998243112	0.0000000369
185	1	0.000013921	0.998064983	0.998288637	0.0000000501
186	2	0.000027842	0.998092825	0.998310845	0.0000000476
190	1	0.000013921	0.998106746	0.998396148	0.0000000839
191	2	0.000027842	0.998134588	0.998416625	0.0000000797
192	1	0.000013921	0.998148509	0.998436777	0.0000000832
193	1	0.000013921	0.99816243	0.998456609	0.0000000867
194	4	5.56839E-05	0.998218114	0.998476127	0.0000000667
195	2	0.000027842	0.998245956	0.998495338	0.0000000623
196	1	0.000013921	0.998259877	0.998514247	0.0000000648
199	3	0.000041763	0.99830164	0.998569224	0.0000000717
200	1	0.000013921	0.998315561	0.998586983	0.0000000738
201	1	0.000013921	0.998329482	0.998604469	0.0000000757
203	1	0.000013921	0.998343403	0.998638639	0.0000000873
204	2	0.000027842	0.998371245	0.998655334	0.0000000808
206	4	5.56839E-05	0.998426929	0.998687965	0.0000000682
207	1	0.000013921	0.99844085	0.998703911	0.0000000693
208	4	5.56839E-05	0.998496534	0.998719617	0.0000000498
209	2	0.000027842	0.998524376	0.998735087	0.0000000445
210	1	0.000013921	0.998538297	0.998750326	0.0000000450
211	1	0.000013921	0.998552218	0.998765337	0.0000000455
212	2	0.000027842	0.99858006	0.998780125	0.0000000401
214	4	5.56839E-05	0.998635744	0.998809047	0.0000000301
215	3	0.000041763	0.998677507	0.998823189	0.0000000212
217	1	0.000013921	0.998691428	0.998850854	0.0000000254
219	1	0.000013921	0.998705348	0.998877718	0.0000000297
221	1	0.000013921	0.998719269	0.998903808	0.0000000341
222	1	0.000013921	0.99873319	0.998916572	0.0000000337
224	1	0.000013921	0.998747111	0.998941553	0.0000000378
228	1	0.000013921	0.998761032	0.998989414	0.0000000522
229	2	0.000027842	0.998788874	0.999000959	0.0000000450
230	3	0.000041763	0.998830637	0.999012341	0.0000000330
231	1	0.000013921	0.998844558	0.999023565	0.0000000321
232	1	0.000013921	0.998858479	0.999034631	0.0000000311
234	2	0.000027842	0.998886321	0.999056303	0.0000000289
235	1	0.000013921	0.998900242	0.999066915	0.0000000278
236	2	0.000027842	0.998928084	0.999077379	0.0000000223
237	1	0.000013921	0.998942005	0.9990877	0.0000000212
239	2	0.000027842	0.998969847	0.999107918	0.0000000191
241	1	0.000013921	0.998983768	0.999127588	0.0000000207
242	2	0.000027842	0.99901161	0.999137223	0.0000000158
243	2	0.000027842	0.999039452	0.999146727	0.0000000115
245	2	0.000027842	0.999067294	0.999165353	0.0000000096
247	1	0.000013921	0.999081215	0.999183481	0.0000000105
250	2	0.000027842	0.999109057	0.999209778	0.0000000102
251	1	0.000013921	0.999122978	0.999218311	0.0000000091
252	1	0.000013921	0.999136899	0.999226732	0.0000000081
253	2	0.000027842	0.999164741	0.999235043	0.0000000049
254	1	0.000013921	0.999178662	0.999243244	0.0000000042
255	1	0.000013921	0.999192583	0.999251337	0.0000000035
256	3	0.000041763	0.999234346	0.999259325	0.0000000006
257	2	0.000027842	0.999262188	0.999267208	0.0000000000
267	1	0.000013921	0.999276109	0.999340663	0.0000000042
270	1	0.000013921	0.99929003	0.999360924	0.0000000050
274	1	0.000013921	0.999303951	0.999386772	0.0000000069

276	1	0.000013921	0.999317872	0.999399217	0.0000000066
278	1	0.000013921	0.999331793	0.999411356	0.0000000063
280	1	0.000013921	0.999345714	0.999423198	0.0000000060
281	1	0.000013921	0.999359635	0.99942901	0.0000000048
282	1	0.000013921	0.999373556	0.999434751	0.0000000037
283	1	0.000013921	0.999387477	0.999440422	0.0000000028
286	2	0.000027842	0.999415319	0.999457024	0.0000000017
288	1	0.000013921	0.99942924	0.999467761	0.0000000015
289	3	0.000041763	0.999471003	0.999473032	0.0000000000
301	1	0.000013921	0.999484924	0.999531603	0.0000000022
303	1	0.000013921	0.999498845	0.999540582	0.0000000017
304	1	0.000013921	0.999512766	0.999544993	0.0000000010
308	1	0.000013921	0.999526687	0.999562132	0.0000000013
312	1	0.000013921	0.999540608	0.999578497	0.0000000014
313	1	0.000013921	0.999554529	0.999582472	0.0000000008
315	1	0.000013921	0.99956845	0.999590288	0.0000000005
322	1	0.000013921	0.999582371	0.999616286	0.0000000012
330	1	0.000013921	0.999596291	0.999643605	0.0000000022
331	1	0.000013921	0.999610212	0.999646852	0.0000000013
333	1	0.000013921	0.999624133	0.999653241	0.0000000008
337	1	0.000013921	0.999638054	0.999665607	0.0000000008
339	1	0.000013921	0.999651975	0.999671591	0.0000000004
340	1	0.000013921	0.999665896	0.999674535	0.0000000001
341	2	0.000027842	0.999693738	0.999677447	0.0000000003
343	1	0.000013921	0.999707659	0.999683178	0.0000000006
346	1	0.000013921	0.99972158	0.999691547	0.0000000009
350	2	0.000027842	0.999749422	0.999702296	0.0000000022
352	1	0.000013921	0.999763343	0.999707502	0.0000000031
357	1	0.000013921	0.999777264	0.999720047	0.0000000033
388	1	0.000013921	0.999791185	0.999784937	0.0000000000
412	1	0.000013921	0.999805106	0.9998231	0.0000000003
415	1	0.000013921	0.999819027	0.999827282	0.0000000001
418	1	0.000013921	0.999832948	0.999831346	0.0000000000
431	1	0.000013921	0.999846869	0.9998477	0.0000000000
433	1	0.000013921	0.99986079	0.999850046	0.0000000001
448	1	0.000013921	0.999874711	0.99986634	0.0000000001
481	1	0.000013921	0.999888632	0.999895381	0.0000000000
487	1	0.000013921	0.999902553	0.999899826	0.0000000000
490	1	0.000013921	0.999916474	0.999901965	0.0000000002
585	1	0.000013921	0.999930395	0.999948575	0.0000000003
614	1	0.000013921	0.999944316	0.999957214	0.0000000002
623	1	0.000013921	0.999958237	0.999959543	0.0000000000
826	1	0.000013921	0.999972158	0.999987177	0.0000000002
937	1	0.000013921	0.999986079	0.99999264	0.0000000000
1140	1	0.000013921	1	0.999997063	0.0000000000
	71834				0.0002108865

Inactive Off Time

Inactive off time Chi-squared verification.

Column 1: Seconds of inactive off time.

Column 2: Sample count.

Column 3: Probability density function of measured output.

Column 4: Cumulative density function of measured output.

Column 5: Expected value (cdf).

Column 6: Chi-squared verification.

Seconds	Count	pdf	Measured	Expected	Chi Squared
1	33894	0.645858343	0.645858343	0.646446609	0.0000005353
2	8438	0.160788125	0.806646468	0.80754991	0.0000010107
3	3594	0.068484537	0.875131005	0.875	0.0000000196
4	1836	0.034985423	0.910116428	0.910557281	0.0000002134
5	1134	0.021608644	0.931725071	0.931958618	0.0000000585
6	731	0.013929381	0.945654452	0.946005075	0.0000001300
7	511	0.009737228	0.955391681	0.955805826	0.0000001794
8	396	0.007545876	0.962937556	0.962962963	0.0000000007
9	300	0.005716572	0.968654128	0.968377223	0.0000000792
10	216	0.004115932	0.97277006	0.972589878	0.0000000334
11	190	0.003620496	0.976390556	0.975943739	0.0000002046
12	140	0.002667734	0.97905829	0.980909911	0.0000034952
13	114	0.002172298	0.981230588	0.982786741	0.0000024640
14	92	0.001753082	0.98298367	0.982786741	0.0000000395
15	76	0.001448198	0.984431868	0.984375	0.0000000033
16	79	0.001505364	0.985937232	0.985733199	0.0000000422
17	67	0.001276701	0.987213933	0.98690543	0.0000000964
18	46	0.000876541	0.988090474	0.987925488	0.0000000276
19	47	0.000895596	0.988986071	0.98881966	0.0000000280
				0.0000086611	

Embedded References

Embedded references Chi-squared verification.

Column 1: Reference count.

Column 2: Measured cumulative distribution.

Column 3: Expected cumulative distribution.

Column 4: Chi-squared verification

References	Measured	Expected	Chi-square
1	0.814233001	0.814434554	0.0000000499
2	0.93062306	0.930721998	0.0000000105
3	0.965266015	0.965565465	0.0000000929
4	0.979733091	0.979978211	0.0000000613
5	0.987093418	0.987144397	0.0000000026
6	0.991108438	0.991160824	0.0000000028
7	0.993494693	0.99361014	0.0000000134
8	0.995168972	0.995200559	0.0000000010
9	0.996271071	0.996284648	0.0000000002
			0.0000002346