

Fall 2012

# Transferring big data across the globe

Adam H. Villa

*University of New Hampshire, Durham*

Follow this and additional works at: <https://scholars.unh.edu/dissertation>

---

## Recommended Citation

Villa, Adam H., "Transferring big data across the globe" (2012). *Doctoral Dissertations*. 680.  
<https://scholars.unh.edu/dissertation/680>

This Dissertation is brought to you for free and open access by the Student Scholarship at University of New Hampshire Scholars' Repository. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of University of New Hampshire Scholars' Repository. For more information, please contact [nicole.hentz@unh.edu](mailto:nicole.hentz@unh.edu).

TRANSFERRING BIG DATA ACROSS THE GLOBE

BY

ADAM H. VILLA

B.A. Wheaton College, Norton, MA, 2003

DISSERTATION

Submitted to the University of New Hampshire

in Partial Fulfillment of

the Requirements for the Degree of

Doctor of Philosophy

in

Computer Science

September, 2012

UMI Number: 3533710

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.

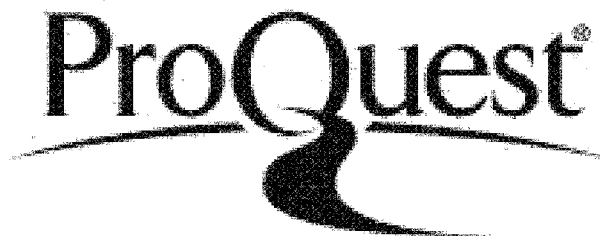


UMI 3533710

Published by ProQuest LLC 2012. Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against  
unauthorized copying under Title 17, United States Code.



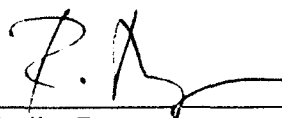
ProQuest LLC  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106-1346

This dissertation has been examined and approved.



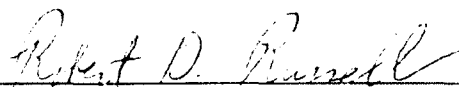
---

Dissertation director, Dr. Elizabeth Varki  
Associate Professor of Computer Science



---

Dr. Radim Bartos  
Associate Professor of Computer Science



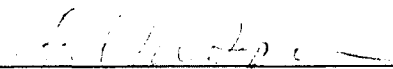
---

Dr. Robert Russell  
Associate Professor of Computer Science



---

Dr. Eleanne Solorzano  
Associate Professor of Statistics



---

Dr. Ted M. Sparr  
Professor of Computer Science

16<sup>th</sup> May 2012  
Date

This thesis is dedicated to my parents.  
I am forever grateful for their endless love, support and guidance.

143

## ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my advisor Professor Elizabeth Varki for her support of my Ph.D study and for her motivation and enthusiasm. Her guidance and encouragement helped through my graduate studies. Without her help, this dissertation would not be possible.

Besides my advisor, I would like to thank the rest of my dissertation committee: Professor Radim Bartos, Professor Eleanne Solorzano, Professor Ted Sparr, and Professor Robert Russell, for their insightful comments and guidance. I would also like to thank my professors who have supported and guided me throughout my graduate studies: Professor James Weiner, Professor Pilar de la Torre, and Professor Philip Hatcher.

My sincere thanks also go to Petr Brym and Tony Borgado for their immense assistance with my study of the campus network. I would also like to thank Mike Hagen and the Interoperability Laboratory at UNH for aiding my research throughout the years.

Without the support of my family and friends, I would not have reached this point in my life. Thank you Katie for your endless encouragement and support.

I would also like to acknowledge the following support that I received from the University of New Hampshire: Dissertation Year Fellowship (2010), College of Engineering and Physical Science Graduate Teaching Achievement Award (2007), and Summer Teaching Assistant Fellowships (2006, 2008).

## TABLE OF CONTENTS

<b>DEDICATION</b> . . . . .	iii
<b>ACKNOWLEDGEMENTS</b> . . . . .	iv
<b>LIST OF FIGURES</b> . . . . .	viii
<b>ABSTRACT</b> . . . . .	xiii
<b>CHAPTER</b>	
<b>I. Introduction</b> . . . . .	1
1.1 Research road map . . . . .	2
<b>II. Big Data: Creation and Management</b> . . . . .	7
2.1 Big Data Management . . . . .	8
2.2 Grids . . . . .	12
2.3 Grid Middleware . . . . .	14
2.4 Data Movement . . . . .	15
2.5 Replication . . . . .	17
2.6 Replica Selection . . . . .	19
<b>III. Parallel Transfer Techniques</b> . . . . .	22
3.0.1 Basic Technique . . . . .	22
3.0.2 Predictive Techniques - History Based . . . . .	23
3.0.3 Predictive Techniques - Network Weather Service and Probes . . . . .	24
3.0.4 Dynamic Techniques - Equal Request Sizes . . . . .	26
3.0.5 Dynamic Techniques - Varying Request Sizes . . . . .	27
3.0.6 Dynamic Techniques - Preemptive Measures . . . . .	29
3.0.7 Peer-to-Peer Techniques . . . . .	32
3.1 Parallel transmission techniques used on the Internet . . . . .	34

3.2	Key Concepts of Parallel Transfer Techniques . . . . .	36
<b>IV.</b>	<b>Preliminary work . . . . .</b>	<b>40</b>
4.1	Replica Traffic Manager . . . . .	40
4.2	Simulating multi-user parallel data transfers . . . . .	41
4.3	Impacting users with parallel transfers . . . . .	42
<b>V.</b>	<b>Live evaluations of parallel transfers . . . . .</b>	<b>45</b>
5.1	Experiments and Observations . . . . .	46
5.1.1	Normal Data Retrieval . . . . .	47
5.1.2	Advanced Data Retrieval . . . . .	48
5.2	Issues and Challenges . . . . .	55
5.3	Impacting other users . . . . .	57
<b>VI.</b>	<b>Examining the campus network and its user workloads . . . .</b>	<b>60</b>
6.1	Campus network configuration . . . . .	62
6.2	Bandwidth usage . . . . .	64
6.3	Internet application workloads . . . . .	67
6.4	Increasing bandwidth . . . . .	72
6.5	Summary . . . . .	76
<b>VII.</b>	<b>Feasibility of big data transfers on the campus network . . . .</b>	<b>79</b>
7.1	Campus Network . . . . .	81
7.2	Traffic on the Campus Network . . . . .	81
7.3	Impact of Big Data Transmissions . . . . .	82
7.4	Potential and Limitations . . . . .	82
<b>VIII.</b>	<b>Nice model for big data transfers . . . . .</b>	<b>85</b>
8.1	Platform . . . . .	86
8.1.1	Performance metrics . . . . .	88
8.2	Nice model . . . . .	89
8.3	Parallel model . . . . .	92
8.4	Related work: Data transfers over the Internet . . . . .	93
8.5	Summary . . . . .	95
<b>IX.</b>	<b>Evaluating the nice model for big data transfers . . . . .</b>	<b>97</b>
9.1	Transmission time (TT as TimeDiff and InitiateTime varies): . .	98
9.2	Response time (RT as TimeDiff and InitiateTime varies): . . .	119
9.3	Bandwidth differential between sender and receiver: . . . . .	119



9.4 Summary . . . . .	120
<b>X. Analysis of Evaluations . . . . .</b>	<b>125</b>
10.0.1 Nice . . . . .	126
10.0.2 Parallel . . . . .	128
10.0.3 Summary . . . . .	129
<b>XI. Conclusions and Future Work . . . . .</b>	<b>131</b>
11.1 Future Work: CargoExchange application . . . . .	132
<b>BIBLIOGRAPHY . . . . .</b>	<b>135</b>

## LIST OF FIGURES

### Figure

2.1	Size of CERN LHC experimental data sets over the past 16 months. The total disk and tape storage amounts aggregated for all tier-1 locations in the CERN grid. . . . .	9
2.2	CERN's WLCG Tiered Replica Structure (5) . . . . .	10
2.3	WLCG Tier-1 and Tier-2 Connections (84) . . . . .	11
5.1	Variations in user transfer rates when retrieving a 1MB file from a remote server over the course of several weeks. . . . .	47
5.2	Normal Data Retrieval: Service times (minutes) for each server when retrieving the entire 30GB data file independently. . . . .	48
5.3	Advanced Data Retrieval - Brute Force Technique: Service times (minutes) for each server when retrieving equal 1GB portions of the 30GB data file. . . . .	50
5.4	Advanced Data Retrieval - Performance-based Technique: Total service time (minutes) for the file transfer as the number of servers concurrently used increases. . . . .	52
5.5	Transfer rates for the fastest server connection observed, as the number of servers concurrently used increases. . . . .	53
5.6	Incremental Distributed File Retrieval: Changes in service time (minutes) as the user's retrieval capacity approaches its maximum utilization. . . . .	54
5.7	Comparison of service times (minutes) for all data retrieval techniques observed. . . . .	55

5.8	Bandwidth usage on wide area network connections before, during and after my evaluations. The shaded regions indicate the time period during my experiments. (a) - This graph shows bandwidth usage of the Internet2 connection for a two-week period. (b) - This graph shows total bandwidth usage of all WAN connections for a two-week period. (c) - This graph shows total bandwidth usage of all WAN connections for a four-month period. . . . .	58
6.1	Network layout for the university network and its connection to the shared data center in a nearby metropolitan area. . . . .	62
6.2	Changes in maximum bandwidth consumption over 12 months for all data passing through all of the university's shared Internet connections. Each semester user demand and bandwidth consumption increases. . . . .	65
6.3	Changes in the minimum, average and maximum bandwidth usage (all receiving and transmitting traffic) for a typical week during the Spring 2011 semester. . . . .	66
6.4	Changes in the minimum, average and maximum bandwidth usage (all receiving and transmitting traffic) for a typical day during the Spring 2011 semester. . . . .	67
6.5	Total amount of data transferred by the campus network each day. .	68
6.6	Most active protocols utilized on an average day. Protocol usage by types of users are shown: A) all users, B) faculty/staff users, C) student users during the daytime and D) student users during the nighttime. . . . .	69
6.7	Changes in protocol usage for the most actively used protocols on campus throughout a typical day. . . . .	71
6.8	Total amount of data transferred by each application class between October 2010 and May 2011 for all users on campus. . . . .	72
6.9	This table lists the total amount of data transferred by application class between October 2010 and May 2011 for all users on campus.	73
6.10	Changes in bandwidth usage for top applications when the students' bandwidth is increased. . . . .	74
8.1	Network utilization and bandwidth availability for each hour of a typical day. . . . .	87

8.2	Nice model . . . . .	90
8.3	Parallel model . . . . .	92
9.1	Simulator configuration map: The left side of the map represents that sending campus network and its client/server machines. The right side of the map represents the receiving campus network and its client/server machines. The staging server in the middle of the map is utilized when the sender/receiver networks have non-synchronous low demand periods. . . . .	98
9.2	Transmitting a 1 TB data set when TimeDiff = 0 (top graph) and TimeDiff = 12 (bottom graph). . . . .	99
9.3	Transmitting a 1 TB data set when TimeDiff = 0 (top graph) and TimeDiff = 3 (bottom graph). . . . .	100
9.4	Transmitting a 1 TB data set when TimeDiff = 6 (top graph) and TimeDiff = 9 (bottom graph). . . . .	101
9.5	Transmitting a 1 TB data set when TimeDiff = 12 (top graph) and TimeDiff = 15 (bottom graph). . . . .	102
9.6	Transmitting a 1 TB data set when TimeDiff = 18 (top graph) and TimeDiff = 21 (bottom graph). . . . .	103
9.7	Transmission time comparison for transmitting a 1 TB data set using the parallel model when the time zone difference is between 0-8 hours (top graph) and 9-12 hours (bottom graph). . . . .	105
9.8	Transmission time comparison for transmitting a 1 TB data set using the parallel model when the time zone difference is between 13-16 hours (top graph) and 17-23 hours (bottom graph). . . . .	106
9.9	Transmission time comparison for transmitting a 1 TB data set using the nice model when the time zone difference is between 0-12 hours (top graph) and 13-23 hours (bottom graph). . . . .	107
9.10	Percentage improvement (max 100%) in transmission time when the nice model is used instead of the parallel model for time zone differences 0-12 (top graph) and 13-23 (bottom graph). . . . .	108

9.11	Reduction in transmission time when the nice model is used instead of the parallel model for time zone differences 0-12 (top graph) and 13-23 (bottom graph). . . . .	109
9.12	Comparison of transmission times for the 1 TB data set for both nice and parallel models under all time zone differences when the request submission time is 12AM (top graph) and 3AM (bottom graph). . .	110
9.13	Comparison of transmission times for the 1 TB data set for both nice and parallel models under all time zone differences when the request submission time is 6AM (top graph) and 9AM (bottom graph). . . .	111
9.14	Comparison of transmission times for the 1 TB data set for both nice and parallel models under all time zone differences when the request submission time is 12PM (top graph) and 3PM (bottom graph). . .	112
9.15	Comparison of transmission times for the 1 TB data set for both nice and parallel models under all time zone differences when the request submission time is 6PM (top graph) and 9PM (bottom graph). . . .	113
9.16	Comparison of response times for the 1 TB data set for both nice and parallel models when the time zone differences are 0 (top graph) and 3 (bottom graph). . . . .	114
9.17	Comparison of response times for the 1 TB data set for both nice and parallel models when the time zone differences are 6 (top graph) and 9 (bottom graph). . . . .	115
9.18	Comparison of response times for the 1 TB data set for both nice and parallel models when the time zone differences are 12 (top graph) and 15 (bottom graph). . . . .	116
9.19	Comparison of response times for the 1 TB data set for both nice and parallel models when the time zone differences are 18 (top graph) and 21 (bottom graph). . . . .	117
9.20	Percentage improvement (max 100%) in response time when the nice model is used instead of the parallel model for time zone differences 0-12 (top graph) and 13-23 (bottom graph). . . . .	118
9.21	Transmitting a 1 TB data set when $\text{TimeDiff} = 0$ (top graph) and $\text{TimeDiff} = 3$ (bottom graph) where the receiver has 4 times the available bandwidth than the sender. . . . .	121

9.22	Transmitting a 1 TB data set when <code>TimeDiff</code> = 6 (top graph) and <code>TimeDiff</code> = 9 (bottom graph) where the receiver has 4 times the available bandwidth than the sender. . . . .	122
9.23	Transmitting a 1 TB data set when <code>TimeDiff</code> = 12 (top graph) and <code>TimeDiff</code> = 15 (bottom graph) where the receiver has 4 times the available bandwidth than the sender. . . . .	123
9.24	Transmitting a 1 TB data set when <code>TimeDiff</code> = 18 (top graph) and <code>TimeDiff</code> = 21 (bottom graph) where the receiver has 4 times the available bandwidth than the sender. . . . .	124

# ABSTRACT

## TRANSFERRING BIG DATA ACROSS THE GLOBE

by

Adam H. Villa

University of New Hampshire, September, 2012

Transmitting data via the Internet is a routine and common task for users today. The amount of data being transmitted by the average user has dramatically increased over the past few years. Transferring a gigabyte of data in an entire day was normal, however users are now transmitting multiple gigabytes in a single hour. With the influx of big data and massive scientific data sets that are measured in tens of petabytes, a user has the propensity to transfer even larger amounts of data. When transferring data sets of this magnitude on public or shared networks, the performance of all workloads in the system will be impacted.

This dissertation addresses the issues and challenges inherent with transferring big data over shared networks. A survey of current transfer techniques is provided and these techniques are evaluated in simulated, experimental and live environments. The main contribution of this dissertation is the development of a new, “nice” model for big data transfers, which is based on a store-and-forward methodology instead of an end-to-end approach. This nice model ensures that big data transfers only occur when there is idle bandwidth that can be repurposed for these large transfers. The

nice model improves overall performance and significantly reduces the transmission time for big data transfers. The model allows for efficient transfers regardless of time zone differences or variations in bandwidth between sender and receiver. Nice is the first model that addresses the challenges of transferring big data across the globe.



## CHAPTER I

### Introduction

Over the past several years there has been a tremendous increase in the amount of data being transferred between Internet users. Escalating usage of streaming multimedia and other Internet based applications has contributed to this surge in data transmissions. Another facet of the increase is due to the expansion of Big Data, which refers to data sets that are an order of magnitude larger than the standard file transmitted via the Internet. Big Data can range in size from hundreds of gigabytes to petabytes. Big Data creation and examples of massive data sets are given in Chapter II.

Today everything is being stored digitally. Within the past decade, everything from banking transactions to medical history has migrated to digital storage. This change from physical documents to digital files has necessitated the creation of large data sets and consequently the transfer of large amounts of data. There is no sign that the amount of data being stored or transmitted by users is steady or even decreasing. Every year average Internet users are moving more and more data through their Internet connections. Depending on the bandwidth of these connections and the size of the data sets being transmitted, the duration of transfers could potentially be measured in days or even weeks.

There exists a need for an efficient transfer technique that can move large amounts

of data quickly and easily without impacting other users or applications. This dissertation presents my work in identifying and solving this problem. The following section details the journey of my research that led me to study this problem and it highlights the difficulty in seeing this problem from the beginning.

## 1.1 Research road map

Identifying the problem of moving large amounts of data across the globe was not evident at the start of my research. Only after years of study and examination did I recognize that is this an unsolved problem that will become even more apparent as users transfers larger amounts of data. My dissertation follows the journey that I took to identify this stated problem and presents my solution to this data movement challenge. The following is a road map to my research and the chapters of my dissertation.

My research journey began by examining storage systems in grids, the newest and most popular distributed computing environment at the time. I began my study of grids by examining their usage and the software/hardware systems utilized to support their functionality. I focused my study on their storage subsystems and particularly on their use of data replication. Due to the large number of users utilizing a grid, data sets needed to be duplicated and distributed throughout the system to ensure efficient access for the users. Chapter II summarizes my findings on grid computing and my study of the replication strategies commonly utilized in these environments. This study is still applicable today since there are many grid systems actively utilized around the world. Many fundamental grid components are also part of cloud computing.

After surveying how data replicas are utilized in grid computing, I identified an issue with the user request process. Due to the distributed nature of the environment, users are able to request data from any available replica in the system regardless of

system state. The performance of a user's request is dependent on the replica selected and can vary greatly depending on how the replicas are utilized. In Section 4.1, I present my first foray into grid research, the development of the Replica Traffic Manager service. This service is designed to improve performance of replicated data requests by managing all workloads in the system. In my experimental evaluation, I find that this traffic manager provides improved performance and reliability. This study however, specifically focuses on only one component of a user's request - that of storage performance. Transferring the data over shared networks is also a major factor in servicing users' requests. This initial study opened my eyes to the challenges of moving large amounts of data.

Examining the applications and techniques utilized for transferring data in grid and cloud systems became the next focus of my research. Chapter III summarizes the specific applications used for data transfers and the various techniques proposed in recent literature for utilizing these applications to transfer data as quickly as possible. These data transmission techniques attempt to grab as much bandwidth as possible by utilizing multiple transfer streams and possibly multiple replica sources concurrently. Since these parallel downloads are inherently greedy by their nature, I conducted a study to examine the performance of a grid system when multiple users simultaneously utilize these techniques. Using a grid simulator, I was able to simulate multiple user workloads and observe overall system performance. Section 4.2 presents the details of this study and my findings, which show that uncontrolled multi-user usage of these parallel transfers can significantly impact the performance of the system. There needs to be a way to balance the usage of parallel techniques for fast data transmission and still maintain a stable environment.

In Section 4.3, I describe experiments evaluating parallel transfer techniques in a real testing environment in order to understand how these transfer techniques effect the workloads of other users in the system. This study shows that they can indeed

significantly impact the performance of other users/applications. My initial attempt to reduce the impact of parallel of transfers was to place bandwidth restrictions on these workloads when the system is under high demand. The system forces the parallel transfers to wait if they are utilizing too much bandwidth. These restrictions allowed other users' workloads to gain access to the shared network connections, which improved their performance. Surprisingly, the transfer times for the parallel downloads were only minimally impacted by the restrictions. While the study shows that placing these restrictions reduces the impact of big data transfers, this is not a viable solution, as it only prolongs the amount of time these transfers are present on the network. Other solutions needed to be investigated.

After examining the current trend in data transmission in simulated and controlled testing environments, my research continued by examining the performance of parallel transfers in a real, shared system. This study also evaluated the performance of a new parallel transfer technique that I develop, which dynamically utilizes multiple replica sources based on the bandwidth availability. Chapter V details my experiences conducting live experiments using existing parallel techniques and my dynamic retrieval technique on the UNH campus network. I found that parallel download techniques result in varied performance and have the potential for utilizing a significant portion of the shared network bandwidth for the entire campus. I identified that my dynamic technique provides the fastest transfer times and utilizes the smallest number of remote sources. This, however, was not the most significant finding of this work. The degree of impact that these experiments had on the campus network is the most surprising and important outcome of this study and led me to examine the network architecture and its performance in great detail.

Chapter VI presents the findings of my traffic study of the UNH campus network and the trends that exist in shared networks around the world. From this study, I determined that the campus network is heavily utilized by thousands of users every

day. The applications most commonly used are real time applications that are very sensitive to changes in network bandwidth. Adding large data transmissions to an already heavy workload resulted in decreased output for these applications and very angry users. My next task was to determine whether it was feasible for the campus network to support these big data transfers. I conducted a feasibility study, described in Chapter VII, to determine if and how these types of requests could be accommodated in a shared system without impacting existing workloads. I found that due to the human work-sleep schedule there are periods of low usage throughout the course of a normal day. During these periods, there is available, idle bandwidth that could be repurposed for conducting big data transfers.

Taking advantage of low demand periods is not trivial, especially since both ends of a transfer need to have the same level of available bandwidth. Due to differences in distance and in time zones changes, it is possible that there will never be a common time when both the sender and receiver have bandwidth available to accommodate big data transfers. Existing transfer applications and transfer technique do not address this problem and there is a clear need for a solution. Chapter VIII presents my “nice” model for big data transmissions across the globe. The nice model utilizes a store-and-forward approach to data transfers instead of the typical end-to-end methodology used by the existing parallel model.

In order to show that the nice model provides performance improvements over the existing parallel method, I conducted experiments using a commercial network simulator that allowed me to emulate the campus network and its workloads. My evaluations of the nice model are presented in Chapter IX and they show that the nice model delivers marked improvements in data transmission times and allows for efficient use of existing network connections during low demand periods regardless of time zone differences. Chapter X gives a theoretical analysis of my evaluations and further shows the improvements provided by the nice model.

My conclusions and future work are presented in Chapter XI. The next step for my research is to develop a system level service that utilizes the nice model and that allows users to automate big data transfers. In this chapter, I outline some of the components that would be necessary for this service to be developed and identify existing technologies that could be utilized to ensure efficient and secure data transfers.

## CHAPTER II

### Big Data: Creation and Management

Big Data is growing at a tremendous rate. Enormous data sets measured in terabytes and petabytes are being created everyday. With the growth of Internet based applications, cloud computing, and data mining, the amount of data being stored in distributed systems around the world is skyrocketing. In addition to corporate/commercial data sets, academic data are also being produced in the similarly large quantities.

Scientific experiments are creating massive amounts of data that need to be accessible to users around the world. Research areas creating this deluge of data include bioinformatics, particle physics, astronomy and environmental science (49). The size of data sets created by experiments, simulations, sensors and satellites continues to grow each year.

To give an example of the size of the data sets utilized by some of these experiments, a recent study observed a particle physics experiment (DZero) taking place at the Fermi Lab research center. While observing the DZero experiment between January 2003 and May 2005, Iamnitchi et al. (13) analyzed the data usage patterns of users. They found that 561 users processed more than 5 PB of data with 13 million file accesses to more than 1.3 million distinct data files. An individual file was requested by at most 45 different users during the entire analyzed time period (2003 to 2005).

In the DZero experiment and many like it, scientists are generating datasets with an extremely large number of data files. Entire datasets are quite popular amongst users, however the individual data files in these sets are rarely used concurrently since they are so numerous.

There are many research initiatives that have similar data demands. The most popular example today is the Large Hadron Collider at CERN. This experiment is well known and thousands of researchers in the physics and computer science fields are involved. The four experiments being conducted on the LHC generate petabytes of data annually (80; 84). One experiment, ALICE, is can generate data at the rate of 1.25 GB/s (54). Figure 2.1 illustrates the growth in the size of data sets being created and stored by CERN. This graph shows the total amount of storage (both disk and tape) utilized by all of the top-level servers in the CERN organization. The amount of data stored in the system has grown at a steady pace over the past 3 years and is expected to grow faster now that the intensity of their experiments is increasing, which will result in more data collection per second (27). Geographically dispersed researchers eagerly await access to the newest datasets as they become available. The task of providing and maintaining fast and efficient data access to these users is a major undertaking. Since the CERN experiments are so well known and many studies have been conducted on their demands and requirements, I will use the CERN LHC experiments as a motivating example throughout my research.

## 2.1 Big Data Management

To meet the computing demands of experiments like CERN's LHC, a specialized distributed computing environment is needed. Grid computing fits the needs of the LHC experiments and other similar research initiatives. In Section 2.2, I examine the definition and usage of grid computing (grids). The software architecture used to coordinate the functionality of grids is then discussed in Sections 2.3-2.5.



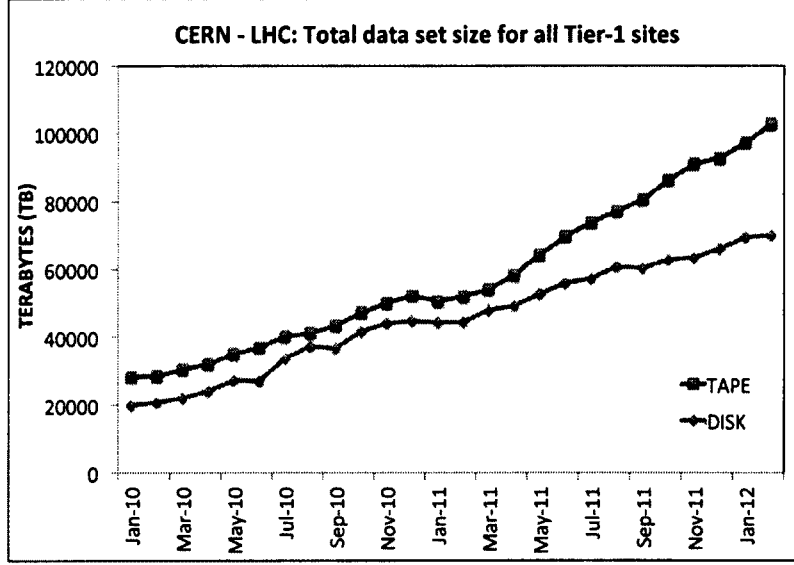


Figure 2.1: Size of CERN LHC experimental data sets over the past 16 months. The total disk and tape storage amounts aggregated for all tier-1 locations in the CERN grid.

The Worldwide LHC Computing Grid (WLCG) was created by CERN in 2001 in order to facilitate the access and dissemination of experiment data. The goal of the WLCG is to develop, build, and maintain a distributed computing infrastructure for the storage and analysis of data from LHC experiments (54). The WLCG is composed of over a hundred physical computing centers with more than 100,000 processors (5). Since the data sets produced by the LHC are extremely large and highly desired, the WLCG utilizes replication to help meet the demands of users. Copies of raw, processed, and simulated data are made at several locations throughout the grid.

The WLCG utilizes a four-tiered model for data dissemination, shown in Figure 2.2. The original raw data is acquired and stored in the Tier-0 center at CERN. This data is then forwarded in a highly controlled fashion on dedicated network connections to all Tier-1 sites. There are eleven Tier-1 sites located in Canada, Germany, Spain, France, Italy, Nordic countries, Netherlands, Taipei, United Kingdom and USA.

The role of the Tier-1 sites varies according to the particular experiment, but in general they have the main responsibility for managing the permanent data storage -

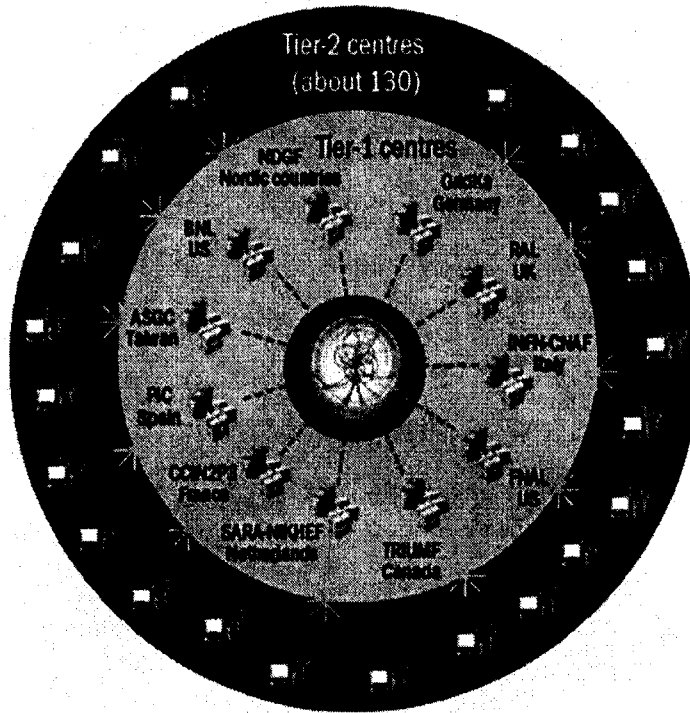


Figure 2.2: CERN's WLCG Tiered Replica Structure (5)

raw, simulated, and processed data - and providing computational capacity for processing and analysis (54). The Tier-1 centers are connected with CERN through dedicated links (Figure 2.3) to ensure high reliability and high-bandwidth data exchange, but they are also connected to many research networks and to the Internet (5). The underlying components of a Tier-1 site consist of online (disk) storage, archival (tape) storage, computing (process farms), and structured information (database) storage. Tier-1 sites are independently managed and have pledged specific levels of service to CERN. It is therefore left to the site's administrators to guarantee that these services are reliably provided.

Data from Tier-1 sites are forwarded to over 130 Tier-2 sites located around the world. The network connections between many Tier-1 and Tier-2 sites are still under development. Some of those connections are dedicated and others utilize public/shared networks. These Tier-2 sites provide widespread access to datasets for researchers. These sites also provide computational capacity and storage services for

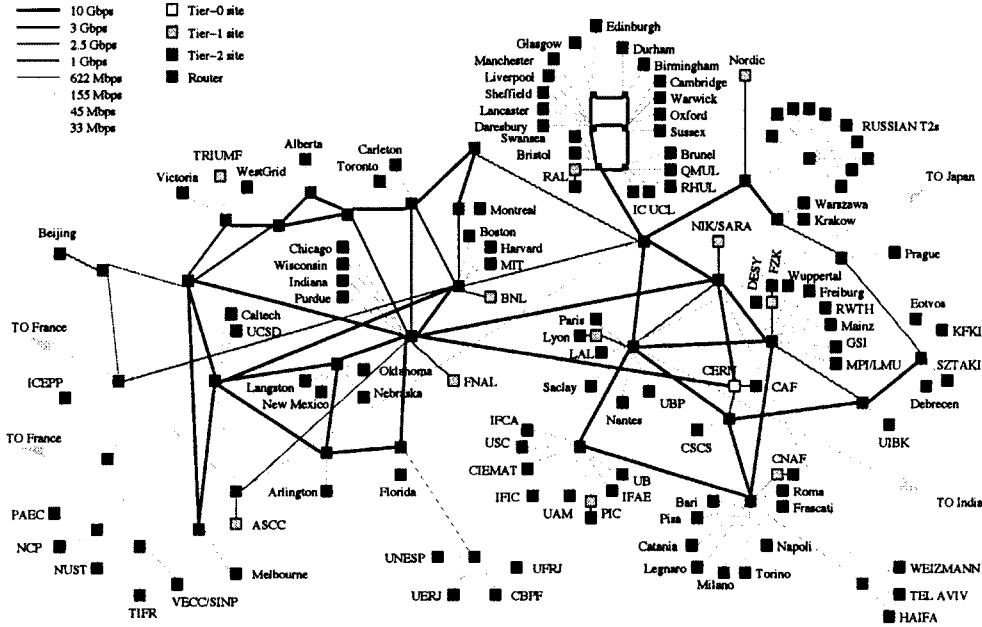


Figure 2.3: WLCG Tier-1 and Tier-2 Connections (84)

Monte Carlo event simulation and for end-user analysis. Any data generated at Tier-2 sites is forwarded back to Tier-1 centers for archival storage.

Other computing facilities in universities and laboratories are able to retrieve data from Tier-2 sites for personal processing and analysis. These sites constitute the Tier-3 centers, which are outside the scope of the controlled LCG project and are individually maintained and governed. Tier-3 sites allow researchers to retrieve, host, and analyze specific datasets of interest. Freed from the reprocessing and simulation responsibilities of Tier-1 and Tier-2 centers, these Tier-3 sites can devote their resources to their own desired analyses and are allowed more flexibility with fewer constraints (46). As there are thousands of researchers eagerly waiting for new data to analyze, many users will find less competition for time and resources at Tier-3 sites than at the Tier-2 sites.

It is important to note that users connecting to either Tier-2 or Tier-3 sites will use public, shared network connections, including the Internet. Grid traffic and normal World Wide Web traffic will both be present on these shared links. A user will also be

sharing the site that they access with multiple other users. These factors can affect the performance of the data transfer between the selected retrieval site and the user. Retrieving these large data files also places a burden on shared resources and impacts other grid and non-grid users.

When it comes to retrieving data in the WLCG, a normal user (depending on their security credentials) can access data on either Tier-2 and Tier-3 sites. The user would select a desired site and issue a request for a specific data file. Selecting a site to utilize can be a complicated task and a user's performance is dependent on the location chosen. I explore several techniques for selecting a replica site in Section 2.6.

## 2.2 Grids

Grid computing has emerged as a framework for aggregating geographically distributed, heterogeneous resources that enables secure and unified access to computing, storage and networking resources (40). Grid applications have vast datasets and/or complex computations that require secure resource sharing among geographically distributed systems. The term "Grid" was inspired by the electrical grid system, where a user can plug in an appliance to a universal socket and have instant access to power without knowing exactly where that power was generated or how it came to reach the socket (40). The vision for grids was similar. A user could simply access as much computing power as required through a common interface without concern for who was providing the resources. Currently, grids have not yet reached that level of simplicity.

Grids offer coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations (41). A virtual organization (VO) comprises a set of individuals and/or institutions having access to computers, software, data, and other resources for collaborative problem-solving or other purposes (65). A grid can also be defined as a system that coordinates resources that are not subject to centralized

control, using standard, open, general-purpose protocols and interfaces in order to deliver nontrivial qualities of service (37).

Data grids, a specialized extension of grid computing, are responsible for providing the infrastructure and services to access, transfer, and modify massive datasets stored in distributed storage resources (125). They allow users to access computational and storage resources in order to execute data-intensive applications on remote data. Data grids were originally designed with the following principles (30):

- *Mechanism neutrality*: the data grid is designed to be as independent as possible of low-level mechanisms
- *Policy neutrality*: the data grid is structured so that significant design decisions are explicitly stated and left for the user to modify or implement
- *Compatibility with grid infrastructure*: the data grid should utilize components of existing grid infrastructure such as authentication, resource management, and information services.
- *Uniformity of information infrastructure*: similar to the grid, the data grid should have access to uniform information about resource structure and state, which allows for runtime adaptation to system conditions.

The objective of a data grid system is to integrate heterogeneous data files stored in a large number of geographically distributed sites into a single virtual data management system and to provide diverse services to fit the needs of high-performance distributed and data-intensive computing (125).

CERN's Worldwide LHC Computing Grid (WLCG) is a combination of computation and data grids. It provides a distributed computing infrastructure for the storage and analysis of data from LHC experiments. In the following section, I examine the software architecture that enables grids, like the WLCG, to perform their functions.

## 2.3 Grid Middleware

The sharing of resources in a grid is facilitated and controlled by a set of services that allow resources to be discovered, accessed, allocated, monitored, and accounted for, regardless of their physical location (66). Since these services create a layer between physical resources and applications, they are often referred to as Grid Middleware. Every grid has different service requirements, therefore the architecture and grid middleware implementation of every grid can vary.

The middleware of many grids is based on the software architecture called the Globus Toolkit (38). The toolkit is a set of libraries and programs that address common problems that occur when building distributed system services and applications (8). It provides a set of infrastructure services that implement interfaces for managing computational, storage, and other resources. The Globus Toolkit provides all of these services and it is left to grid administrators to determine whether or not to include certain services in their grid implementation. These are a few of the well-known and widely used grids that deploy the Globus Toolkit: TeraGrid (4), Open Science Grid (11), EGEE (7), Worldwide LHC Computing Grid (WLCG) (5), China National Grid, UK National Grid Service (12) and NAREGI (9).

The architecture of the Globus Toolkit contains several components, each of which is responsible for different grid functions. A few of these services are (38):

- Grid Resource Allocation and Management (GRAM) - This service initiates, monitors, and manages the execution of computations on remote computers. It allows a user to specify: the quantity and type of resources needed, the data sets required for their computation, the executable application to be run, the necessary security credentials, and the job persistence requirements.
- Data access and movement - The reliable file transfer (RFT) service is provided to ensure that data is successfully transferred from one location to another.

- Replica management - This service keeps track of all replicas and their content using a replica location service (RLS) and a data replication service (DRS).
- Monitoring and Discovery - Multiple services collect and process information about the configuration and state of all resources to enable monitoring of system status.
- Security - Services establish the identity of users or services (authentication), protect communications, and determine who is allowed to perform what actions (authorization), as well as manage user credentials and maintain group membership information.

Grid middleware systems are custom designed to fit the needs of a particular grid. The components of the Globus Toolkit provide the tools for creating a basic, functional grid. Many of the detailed technical decisions and optimizations are left to grid administrators, such as replication management and replica selection. Finding the optimal settings and configurations for grid middleware components is still a work in progress and requires further study.

Many grid implementations utilize Globus components in addition to their own custom components. The WLCG's middleware, gLite, contains some Globus components as well as software developed by several research projects in the European Union, including programs from the EGEE consortium (7). In the next two sections, I will focus my examination on two general middleware services utilized in most grids: data movement and replica management.

## 2.4 Data Movement

There are several grid applications available for moving data from one location in a grid to another. The most widely-used data movement tool, which is also a component of the Globus Toolkit, is called GridFTP (18; 17). It is an extension of

the File Transfer Protocol (FTP) and was designed specifically for grid environments. GridFTP offers several features over standard FTP (19):

- Third-party control of data transfer - This allows a user to remotely monitor and control a data transfer between two other sites.
- Authentication, data integrity, data confidentiality - GridFTP supports and interfaces with grid middleware security and authentication components.
- Striped data transfer - Data can be interleaved across multiple servers and GridFTP supports the transfer of data portioned among multiple servers.
- Parallel data transfer - GridFTP supports multiple transfer streams in parallel between a single source and destination.
- Partial file transfer - GridFTP allows the user to transfer only a portion of a file rather than the entire file.
- Support for reliable and restartable data transfer.

New features were also recently released for GridFTP, such as the option to utilize the UDP protocol and pipelining (25). Pipelining allows many transfer requests to be sent to the server before any transfer completes. This technique hides the latency of transfer requests by overlapping them with data transfers. The server does not have to wait for a new request to arrive after it finishes the current request.

It is important to note that GridFTP is not the sole data transport tool used in grids. There are other mechanisms available, such as the data movement operation for the Storage Resource Broker (SRB) (98). The SRB data movement tool allows a user to access data on normal filesystems, as well as archival resources such as HPSS (97). Another mechanism to transport data across the grid is OGSA-DAI (10), which can accommodate different types of data resources, including relational and



XML databases (45). Simple FTP and HTTP file transfers are also very commonplace.

## 2.5 Replication

Replication is used in data grids to help improve access to high-demand datasets, by reducing access latency and bandwidth consumption. Replicas or copies of data file(s) are created in order to improve access performance and data integrity. In most grid implementations replicated files are read only, which eliminates problems with file updates and coherency (30). All replicas located in a grid are managed by a replica management service, a component of the Globus middleware architecture, which has several responsibilities (18; 17):

- creates new copies of a complete or partial data set
- registers new replicas in a *Replica Catalog*
- allows users and applications to query the catalog to find all existing replicas of a particular file or collections of files.

The replica management services has several components that accomplish these tasks. The *Replica Location Service (RLS)* component maintains and provides access to information about the physical location of replicas. The main task that the RLS performs is: Given a unique logical identifier (logical file name - LFN) for desired data, determine the physical locations (physical file name - PFN) of one or more copies of the data. In order to perform this task, the RLS maintains records of all logical to physical file name mappings. The physical file names are structured similar to URLs, where the access protocol, site address, and directory structure are fully specified. The Giggie Framework (30) is the basis for the RLS component in Globus.

The framework ensures that replica location data is distributed throughout the grid in order to maintain efficient access. There are two types of data repositories that the RLS uses to store replica information: local replica catalogs (LRCs) and replica location indices (RLIs). An LRC stores information about logical filenames, such as creation data, access lists and other file attributes. It also stores a map of all physical filenames that are replicas for a logical filename. An RLI maintains information about the replica catalogs and the logical file names that they contain. It can locate which LRC contains the replica file list for a given logical filename. The Giggles framework specifies how LRCs and RLIs are interconnected to construct a scalable and reliable replica location service. Studies have analyzed the effectiveness of the RLS and the replica management service and have shown that they perform well for large-scale, heavily loaded systems (26).

Creating a more sophisticated replica management service is still a work in progress and many middleware developers have left advanced features, such as replica management, to be implemented by individual grid administrators. An example of an advanced service is the selection of the “best” replica to service a user’s request based on storage and network performance predictions. Several studies examine this selection problem and develop replication selection algorithms. These are discussed in the next section.

Many replicas are manually created when needed. Several studies have also developed mechanisms for dynamically creating and deleting replicas in order to fit the demand of the grid. As the popularity of a data file increases, a dynamic replication tool would automatically create new replicas to service the increased demand. One such technique, called *Fast Spread*, creates a replica of a requested file at every node that is encountered on the data delivery path from the server to the client (100). Another technique creates replicas close to the users requesting the file in order to exploit geographical locality (62). The globus architecture does not specifically uti-

lize dynamic replication strategies and therefore they would have to be manually implemented.

When users want to retrieve a data file from a remote grid resource, they contact the replica management service to receive a listing of available replicas that contain the specified data. The users then utilize some or all of the available replicas to service their requests. The users decide which resources to utilize and to what extent. Using a data movement tool, like those described in Section 2.4, users would then initiate transfer requests on the resources that they have selected. Users can choose which data movement tool to utilize and how to configure the data transfer settings in order to achieve the best performance. Selecting the proper settings for a data transfer is not a trivial task and often requires detailed knowledge about grid resources.

## 2.6 Replica Selection

Users are able to retrieve data from any replica server that is available to them. Making an informed decision about which replica to use can affect a client's performance. Choosing a lightly loaded server over a heavily loaded server can result in dramatically different completion times for a client. Finding the most efficient replica is a difficult and complicated task.

Server selection is a task common in many computing environments and there are many generalized server selection algorithms. A number of these algorithms are driven by performance metrics, such as proximity metrics that measure proximity of servers to a client and server load metrics that measure the load of servers or network paths (94). There are advantages and disadvantages to these performance metric based algorithms, such as the issue of the freshness of metric values. Another generalized selection technique is to just select a server at a random. Mitzenmacher (81; 82) developed a technique that randomly selects a subset of available replicas and then selects the best replica based on the performance metric values

available. Several studies analyze different methods of replica selection specifically for grids. Papers (26; 95; 96; 113) present a few of these varying selection techniques.

Vazhkudai et al. (113) create a storage broker that identifies a suitable replica based on the requesting application's requirements. The broker submits classified advertisements to all available replicas listing these requirements. It is the broker's responsibility to map application requirements against the capabilities of the various storage resources. The authors designed a decentralized storage brokering strategy where every client that requests data performs the selection process, rather than a central manager. There is no central point of control and the decision-making is delegated to every client.

The replica management system for the European Data Grid utilizes a Replication Optimization Service (ROS) that selects the best replica of a data file for a given request (26). The service takes into account the location of the computing resources and network latencies. Network monitoring services provide the ROS with network latencies between various grid resources, which are then used to predict expected transfer times. The service selects the replica with the best expected transfer time to complete the request.

Rahman et al. (95; 96) describe an optimization technique that utilizes the k-Nearest Neighbor (KNN) rule. The KNN rule selects the best replica for a file by examining previous file transfer logs. When a new request arrives, all previous data is analyzed to find a subset of previous file requests that are similar to the new request, which are the k-nearest neighbors. The technique then uses these previous requests to estimate transfer times between replicas and the user. The algorithm selects the best replica based on its predictions.

These studies represent only a portion of the literature on single server selection techniques. In general, I find that there is no perfect solution to the server selection process. Users can only approximate the best server to fit their needs at the current

moment. Relying on a single server, selected by any algorithm, could possibly affect a user's data retrieval performance. There are many situations when a server's performance can degrade:

- The server could suddenly become unavailable or disconnected, which would require the user to re-initiate the server selection process.
- The server could quickly become overloaded. The number of concurrent users could utilize all of the server's available bandwidth. Multiple users could also simultaneously select the same server based on the available performance metrics, creating a herd effect (94).
- The server's transfer rate could be lower than the desired rate of the user.

The performance of the server can change at any time, which directly impacts the user's data retrieval. In the following chapter, I explore recently proposed techniques for quickly transferring large files between users and storage servers.

## CHAPTER III

### Parallel Transfer Techniques

In this chapter, I examine several data retrieval techniques developed specifically for retrieving large files in grid computing environments. The sizes of data files requested in grids are much larger than normal web data requests. It is not uncommon for a grid data file size to be measured in gigabytes or terabytes. Users want to be able to download these files as quickly as possible, by any means necessary. Since utilizing a single server can be limiting, retrieving data from multiple servers in a parallel (also known as data co-allocation) has been suggested as an alternative. In my examination, I group these recently proposed parallel transfer techniques based on how they retrieve data from various replica servers.

#### 3.0.1 Basic Technique

The basic, **brute-force**, data co-allocation technique (110) issues a request for equal sized portions of the file from all available replicas. Every replica that contains the file is utilized and each is responsible for servicing an equal amount of data. There is no consideration given to the performance of replica servers or network conditions. Many studies include this technique as a baseline for comparison with other co-allocation strategies.

The brute-force technique is not an optimal technique. It assumes that all servers

are available and will provide adequate service to the user. It also places an equally heavy burden all servers and there is no consideration for the workload it places on grid resources. When many users utilize this technique, the performance of the entire grid is affected.

### 3.0.2 Predictive Techniques - History Based

In the brute-force technique, the performance of each transfer is not analyzed. Depending on network and server workload, each transfer will have varying performance. The following algorithms take into account the performance metrics of each server interaction when dividing the workload amongst all replicas in order to minimize the transfer completion time. These papers present a few of these methods.

Vazhkudai presents a **history-based data co-allocation technique** (110; 111). He addresses the fact that each transfer between a replica and the client has varying transfer rates. This technique adjusts the amount of data retrieved from each replica by predicting the expected transfer rate for each replica. In a previous work (112), Vazhkudai and Schopf developed a series of univariate and multivariate predictors that create forecasts based on past transfer history with network and disk load data. Using this technique, the author demonstrates how historically faster servers are assigned to deliver larger portions of the file and slower servers are assigned smaller pieces. In his evaluations, he finds that the history-based technique significantly outperforms single replica usage technique and provides improved performance over a simple, brute-force technique.

Zhou et al. also develop a history-based data co-allocation technique. They develop **Replica Convoy (ReCon)** (131), a tool for retrieving data from multiple replicas simultaneously. ReCon is composed of two services: the Replica Convey Service (RCS) and the Replica Convoy Client (RCC). The RCS determines an appropriate replica convey plan for the client using decision algorithms that control how

the replicas will be used to retrieve the desired data. One group of these decision algorithms utilizes GridFTP logs to predict the network throughput for each replica server. This group includes the latest-based, mean-based and median-based techniques. As their names suggest, they utilize the past transfer history data in different ways. The latest-based technique predicts throughput based on the last completed transfers. Median and mean-based techniques utilize the median and mean values of all completed transfers. Using these predictions, the entire data file is divided into varying sized segments specifically for each replica. For example: if there were three available replicas, one replica could be assigned  $3/6$  of the data, another replica would transfer  $2/6$  and the last replica could service the remaining  $1/6$  of the data. Replicas that are predicted to deliver data faster are assigned a larger portion to service.

### 3.0.3 Predictive Techniques - Network Weather Service and Probes

Many grid environments deploy a network monitoring tool called the Network Weather Service (NWS)(124). The NWS is a distributed system that detects the network status at periodic intervals. The service utilizes a set of performance sensors to determine the condition of grid components. These sensors gather data on the latency and bandwidth of end-to-end TCP/IP performance, as well as available CPU and memory of replica servers. Using mathematical models on the data gathered by the sensors, the service creates forecasts of system conditions for given time periods.

Feng and Humphrey develop data retrieval techniques that utilize NWS predictions to specify the amount of data to be requested from replica servers (36). They develop two techniques that utilize these network forecasts: **NWS Static** and **NWS Dynamic**.

In the NWS Static algorithm, network throughput predictions are requested for all connections to the available replica servers from the NWS before the transfer commences. The file is then divided into segments based on the expected throughputs



for each replica. Replica servers with higher throughput predictions are assigned to deliver larger portions of the data file.

In the NWS Dynamic algorithm, the desired file is divided into a fixed number of equal sized segments. The algorithm contacts the NWS to receive throughput forecasts for each replica server and assigns portions of a file segment based on the throughput predictions. When a replica completes a portion, the NWS is again contacted and additional portions are assigned based on the forecasts. The NWS Dynamic algorithm only schedules one segment of the entire file with each NWS prediction. If conditions change, then the next round of predictions should identify the changed conditions and re-distribute the workload accordingly.

In the authors' evaluations, they find that both of their NWS algorithms outperform a basic, brute-force, data co-allocation algorithm. They also find that their NWS Dynamic algorithm provides improved speedup over the Static algorithm.

Utilizing the NWS for predications provides additional overhead costs, which can vary depending on how frequently the service is used. The messages used by the service also produce additional traffic on the network. If the user's grid does not employ the NWS, then the user would be unable to utilize these techniques. Implementing and coordinating a NWS service on all servers in a grid is not a trivial task and would be outside the realm of the basic user's expertise and permissions.

Other mechanisms can be used to determine the status of connections between users and servers. Zhou et al. present a **probe-based data retrieval technique** (131), where a fixed sized pinging mechanism is used to probe network connections and determine network output. Based on the data returned by the probes, varying amounts of data are assigned to each replica. The authors find that their probe-based algorithm outperformed history-based techniques. They attribute this success with the fact that the probe gives an accurate representation of the current state of the network, unlike history-based techniques.

### 3.0.4 Dynamic Techniques - Equal Request Sizes

The following co-allocation retrieval techniques dynamically adapt to changing grid conditions by requesting small, equally sized, portions of a file from multiple replicas. Each technique uses different decision making algorithms on how to perform these requests and several of these algorithms are discussed in this section.

Vazhkudai finds that history-based techniques do not address dynamic network variations that can affect transfer rates between the replica servers and the client (110; 111). Servers that were previously determined to be fast or slow can behave differently than expected due to varying network traffic and system workloads. In order to address these issues, he develops a **conservative load balancing technique** that dynamically adapts to changing network and system conditions. The amount of data requested for a given server is decided dynamically instead of being based on previous history. The desired data file is divided into equal sized, disjoint blocks. Each available server is initially assigned one block to service in parallel. Once a server delivers the block, another block is assigned until the entire file is retrieved. Faster servers will transfer larger portions of the file.

Feng and Humphrey also develop a similar dynamic data co-allocation algorithm called, **NoObserve** (36). This algorithm differs from their other retrieval algorithms discussed in the previous section, since it adjusts to varying network conditions without utilizing the NWS. In the NoObserve algorithm, the source file is statistically divided into equal-sized segments. Initially, each replica is assigned one segment to service. When a replica finishes its segment, the replica is immediately assigned another segment until the entire file is retrieved. In the authors' evaluations, they find that the NoObserve technique provides a speedup over the baseline, brute-force technique. They also find that choosing the appropriate number of file segments is also important. The number of segments should not be too large, in order to minimize the overhead costs associated with transferring multiple small size file segments.

### 3.0.5 Dynamic Techniques - Varying Request Sizes

The techniques described in the previous section divide the desired data file into equal sized disjoint blocks. Other grid data retrieval techniques try to improve performance by varying the size of the blocks based on the performance of the replica servers. Faster servers are assigned larger blocks.

Vazhkudai develops an **aggressive load-balancing technique** (110; 111), which is a modified version his conservative load-balancing technique that was discussed in the previous section. Instead of requesting a single block from each replica, the amount of data requested from faster servers is progressively increased. The amount of data requested from slower servers is decreased or stopped completely. The transfer rate for each block request is compared to all other transfers. If the rate is higher than any other transfer, then the request size for that server is doubled to two blocks. If the rate is lower than other transfers, then the request size for that server is reduced to a single block. If the rate is significantly lower than all other transfers, then the replica no longer receives requests.

There are other dynamic data retrieval techniques that vary the amount of data requested from each server while still dividing the data file into blocks. The **recursively-adjusting co-allocation technique** (127; 128; 129) developed by Yang et al. is a combination of dynamic and predictive techniques, since it utilizes Network Weather Service forecasts. This technique works by continually adjusting the amount of data requested from each replica server to correspond to its real-time bandwidth during file transfers. Unlike the previous algorithm by Vazhkudai, the goal of this technique is to make the expected completion times for all servers the same. The recursively-adjusting algorithm continually monitors each server and adjusts the workloads to ensure that all servers deliver the last block at the same time. The goal of the algorithm is to eliminate the user from having to wait for a single server to deliver the last portion of the file.

The technique begins by dividing the desired data file into several sections. Each of these sections is then sub-divided into varying sized blocks that are individually assigned to all replicas. The number and size of the larger sections is variable and can be adjusted by the user. The size of each section is a percentage of the remaining file size to be retrieved. Each section size will therefore be progressively smaller than previous sections. The user can select the smallest section size that is used.

Initially, the algorithm assigns blocks from the first section to all available servers based on their bandwidths. The Network Weather Service is used to obtain the bandwidth forecast for each server. At this point, it is assumed that all servers will finish the section at the same time. Due to fluctuations in network conditions and server load, actual completion times may vary. When the fastest server completes its block of the current section, the next section of the file is divided into blocks using the NWS predictions and these blocks are assigned to the servers. The goal is for all servers to complete their outstanding work (first and second sections) at the same time. Slower servers will not be assigned additional blocks and faster servers will receive larger portions to service. This process repeats for all sections until the entire file has been requested.

Another dynamic data retrieval technique, which varies the amount of data requested from each server while still dividing the data file into blocks is the **MSDT algorithm** (121) developed by Wang et al. The MSDT algorithm is a combination of dynamic and predictive techniques, as it utilizes the past transfer histories for predictions. In theory, the MSDT algorithm is very similar to the recursively-adjusting co-allocation technique by Yang et. al. The MSDT algorithm just uses a different set of equations to predict the performance of a replica and to assign the workloads to each replica. The algorithm uses the overhead and bandwidth of previous segment transfers to predict the future performance of a replica. To begin, the source data file is divided into multiple segments of equal size. The MSDT algorithm autonomously

assigns a number of segments to each replica whenever the replica is idle. This algorithm assumes that the replicas will be solely dedicated to grid traffic and that the user has full knowledge of the workload present at the servers, which is not always the case in most grids. The amount of segments that are assigned varies depending on the transfer history for the particular replica.

### 3.0.6 Dynamic Techniques - Preemptive Measures

The dynamic techniques in the two previous sections retrieve portions of the data file from multiple replica servers. The amount of data retrieved may vary depending on the algorithm, however there is the possibility that a client will end up waiting for slower servers to deliver portions of the file. The previous techniques do not preempt transfers or re-distribute the workload to other servers when replicas become unresponsive. In this section, I examine several algorithms that utilize these preemptive measures.

The ReCon data retrieval service<sup>(131)</sup> designed by Zhou et al. offers a **Greedy retrieval algorithm** where the desired data file is divided into equal sized segments. Each replica is initially assigned one segment. As replicas complete their segments, they are assigned additional segments to service. A recursive scheduling mechanism handles any errors that occur. If the user does not receive a response from a server after a user-specified amount of time, the mechanism automatically reschedules the failed data request to another replica that is currently transferring data. Detailed information about re-submission process is not specified in the paper. Zhou et al. compare the Greedy algorithm with other algorithms developed for the ReCon. They find that their Greedy retrieval algorithm did not outperform other statistical based techniques (section 3.2) and their probe-based technique (section 3.3) provided faster retrieval.

Bhuvan et al. develop a different preemptive data co-allocation mechanism, the

### **Dynamic Co-allocation Scheme with Duplicate Assignments (DCDA) (24).**

This technique is used to cope with highly inconsistent network performance of replica servers. The authors develop this technique to enable efficient parallel download of replicated data from multiple servers without the use of past history or heuristics. In their algorithm, the desired data file is divided into disjoint blocks of equal size. Each server is initially assigned one block to service. When a server completes a request, it is assigned another outstanding block. The algorithm continues until all blocks have been assigned. If a server delivers a block and there are no blocks remaining that have not been initially assigned, the server will be given an outstanding block request that has not been completed. There will now be several servers working on the same request. When a server delivers a request, all other servers are notified to stop serving this request. In order to maintain a clear order of outstanding requests, the algorithm utilizes a circular queue to keep track of all requests.

In the evaluations of the DCDA technique, Bhuvan et al. assume that the overhead latency in assigning, delivering and killing of duplicate assignments is negligible. (In reality, this can be a complicated and costly procedure, depending on the infrastructure of the grid.) They compare their algorithm to the Vazhdukai's conservative load balancing technique (110) and find that the DCDA algorithm provides increased performance.

Chang et al. (29) develop an advanced preemptive technique, **Multiple Parallel Downloads with Bandwidth Considerations technique**, that considers both server throughput and client input bandwidth when assigning workloads to the replica servers. This paper is the first to discuss their technique in terms of multiple users. They realize that when everyone uses parallel downloads, they will compete for system resources that causes a degradation of system efficiency and unfairness for the users. They also determine that a server should not outdo its capacity by serving too many clients and a client should not download from too many servers with its limited

incoming bandwidth. Their scheme is divided into three stages: initial stage, steady stage and end stage.

In the initial stage, each replica server is assigned a priority value based on the round-trip-time between the client and server and based on the average wait time for the server. The block size is then determined using the cost of internal and external overheads as a factor. The steady stage begins by selecting two servers with the best priority values to download the desired file. As the file is being transferred, the download speed of the client is monitored. A client's download speed is limited by the speed of its network connection. If the client's download speed does not reach the maximum download bandwidth, an additional server is added to transfer the remaining file. If the client reaches the maximum download speed, then no more servers are utilized. The mechanism also monitors the throughput of each server and categorizes them based on their performance: ordinary servers are assigned one block at a time, fast servers are assigned two blocks at a time, and superior servers are assigned three blocks at a time. These categorizations are recreated before each block assignment. To ensure that all servers finish the last request as close to the same time as possible, the download efficiency of the last block is important. If the last block is to be transferred by a slow or disconnected server, it will increase the time for the entire data transfer. To ensure that this does not occur, a completed server will automatically be assigned an uncompleted block that was originally assigned to another server.

The authors discuss a multiple user environment and provide an example of how their technique would work with six users accessing a small number of files. Their experiments however, do not show the performance of their algorithm when many users are simultaneously utilizing their technique.

### 3.0.7 Peer-to-Peer Techniques

Peer-to-peer (P2P) and grid computing environments both address the problem of organizing large scale computing societies and have the same objective of coordinating large sets of distributed resources (39). Data retrieval techniques specifically designed for peer-to-peer environments have been adapted for use on the grid. Two of these techniques are GridTorrent (133) and the GridTorrent Framework (57). Both of these grid transfer mechanisms are based on the BitTorrent protocol.

The BitTorrent protocol (33) is a peer-to-peer protocol that enables users to retrieve data files from multiple sources while simultaneously uploading them to other clients, instead of obtaining them directly from a central server. BitTorrent is designed to work efficiently under flash crowd situations where a large number of users are concurrently downloading the same file. In this protocol, data files are segmented into pieces, which can be retrieved individually by clients from multiple sources. BitTorrent uses a distributed hash table to dynamically locate peers to participate in a file transfer. It limits the number of concurrent uploads for a user and gives priority to the peers with the best upload rates. The protocol also discourages free-riders, peers that download data without contributing to the system. It uses a tit-for-tat algorithm to ensure that all peers contribute to file downloading.

The GridTorrent transfer mechanism (133) is a modified BitTorrent implementation that is designed to interface with grid middleware components and protocols. GridTorrent can be used to receive data from GridFTP servers or other GridTorrent peers that are simultaneously requesting the same data. The mechanism utilizes the Replica Location Service (RLS) provided by the grid middleware to locate data sources. It extends the information stored in RLS records to include GridTorrent file sources, which allows any user to locate GridTorrent files. The developers also implement two new grid software components to facilitate GridTorrent data transfers: the PeerManager, which handles all communication with other GridTorrent peers and



the DiskManager, which handles all disk I/O for storing and receiving files.

The GridTorrent Framework (57) extends the BitTorrent protocol by adding a collaboration and content manager (CCM). The CCM allows users to publish and share their files with access control rights and allows users to search for available files. Both of these features are not present in the BitTorrent protocol. Kaplan et al. do provide details as to how their GridTorrent Framework interfaces with existing grid middleware and grid security protocols.

In a recent journal article, Al-Kiswany et al. (16) evaluate the effectiveness of peer-to-peer data dissemination techniques in large scientific collaborations, such as CERN's LHC experiment (5) and Fermi Lab's DZero experiment (6). They find that many of today's grids are over-provisioned and peer-to-peer solutions that adapt to dynamic and under-provisioned networks do not provide significant benefits and create unnecessary overhead expenses. In addition, datasets in scientific grid environments differ significantly from the data files typically transferred by peer-to-peer techniques, like BitTorrent. The popularity distributions for scientific data are more uniform than in peer-to-peer systems, which has a significant impact on the effectiveness of P2P techniques (16). It is not uncommon for a popular BitTorrent file to be requested by thousands of users or more, which exploits the benefits of the BitTorrent technique (21). In grid environments however, it is possible for a single data set to have an extremely large number of individual files, which are infrequently accessed concurrently. While observing Fermi Lab's DZero experiment between January 2003 and May 2005, Iamnitchi et al. (13) analyzed the data usage patterns of grid users. They found that 561 users processed more than 5 PB of data with 13 million file accesses to more than 1.3 million distinct data files. An individual file was requested by at most 45 different users during the entire analyzed time period. In the authors' evaluation, they also examine the feasibility of applying known peer-to-peer strategies, such as BitTorrent, using the real usage patterns that they observed. They find

that while the size of the data files being transferred may warrant the use of techniques like BitTorrent, the relatively small number of concurrent users of the same data files does not justify the overhead cost of the peer-to-peer technique.

### **3.1 Parallel transmission techniques used on the Internet**

Data retrieval techniques for the Internet are well established. Even though a majority of Internet data requests are small, various methods have been developed to facilitate the transfer of large data files on the Internet. I examine one group of mechanisms that uses parallel data retrieval from multiple servers. These mechanisms directly relate to many retrieval techniques developed for grid computing environments, which are discussed in the previous section.

Rodriguez and Biersack present mechanisms for parallel access to data on the Internet (101). They develop two different parallel-access schemes: history-based and dynamic. The goal for all of their schemes is to balance the load amongst all available servers by allocating a workload to each replica that is proportional to its service rate.

The authors state that parallel access has additional overhead in comparison to a single access. The additional overhead occurs when multiple connections are opened and extra traffic is generated to perform block requests. In order to minimize these overhead costs, these techniques should only be utilized for larger files.

Their history-based technique utilizes a database with information about previous rates from the different servers to the receiver in order to estimate future transfers. Using these estimates the algorithm assigns varying portions of the file to each replica with the goal that all servers will finish transferring the portions at the same time.

The authors evaluate their history-based technique using live web servers on the Internet distributed across the world. Due to the presence of other Internet traffic, the authors found that the performance of their technique varied at different times of the day. They found that network conditions rapidly change and estimating the transfer

rate to every server using past histories results in poor estimates. Their results show that during peak traffic times when transfer rates vary dramatically and historical information is not a good indicator of future performance, their history-based parallel access technique has higher download times than clients accessing a single server.

In response to the performance of their history-based technique, the authors develop a dynamic technique that adjusts to changing network conditions. Their dynamic technique divides the desired file into a fixed number of equal sized blocks. The client requests one block from every replica. When a server completes a request, another block is assigned. When there are a small number of blocks outstanding, idle servers are requested to deliver blocks that have already been assigned to another server, but have yet to be received. There will then be multiple servers working on the same requests. The authors state that the bandwidth wasted on overlapping these requests is smaller than the worst-case scenario of waiting for the slowest server to deliver the last block. To further enhance the performance of their technique, they utilize TCP-persistent connections between the client and every server to minimize the overhead of opening multiple TCP connections. They also propose pipelining the requests to each server in order to decrease interblock idle times. With pipelining, a new block request is sent to a server before the previous block request is completely received.

In the evaluations of their dynamic technique, the authors find that there is a significant speedup in comparison to a single server access. Since the dynamic technique is not relying on historical information and can adapt to changing network conditions, it has greater performance than requesting data from a single server even under peak traffic conditions. They also observe that the transfer time of a dynamic parallel access is very close to the optimum transfer time. Utilizing request pipelining, the authors demonstrate that their technique would be almost equal to the optimal transfer time.

## 3.2 Key Concepts of Parallel Transfer Techniques

After examining multiple parallel transfer techniques developed for grid computing environments, I can extract several key ideas and concepts about efficient grid data retrieval.

- Dynamic data retrieval techniques outperform static, predictive techniques. History-based techniques are not sensitive to dynamic and rapid changes in network conditions and server workloads. These types of techniques must constantly monitor recent transfers in order to adapt.
- Techniques that utilize the Network Weather Service can be beneficial under certain circumstances. A technique that constantly monitors the NWS forecasts and adapts its retrieval technique to changing conditions would be efficient. A downside to using the NWS is the overhead created by the probes and messages required to obtain the network forecasts. If a NWS service is already in place on the client's grid, then it can provide useful information. If a NWS service is not implemented on the client's grid however, then the task of implementing and coordinating the service is not a trivial task and beyond the scope of an average user.
- Dynamic techniques that divide the desired file into smaller blocks to be retrieved individually can dynamically react to changing conditions. In order for these methods to be efficient, they must carefully consider the following items.
  - Choosing the appropriate size and number of blocks for a data file is a complicated task and the efficiency of the entire download is dependent on the choice made. A large block size could place a significant portion of the workload at slower servers and small block sizes result in overhead costs outweighing the transfer times.

- Users should be cognizant of their network connection’s bandwidth and not attempt to retrieve data from more servers than their connection can handle. Using an excessive number of servers will not be beneficial to the user and will be detrimental to the servers as well.
- Re-issuing previously assigned requests creates duplicate work for the replica servers, however it can prevent the user from waiting for a single, slow server to deliver the last portion of the file. There are several important factors that must be considered when re-issuing requests.
  - \* Implementing a delay before re-issuing a request could prevent unnecessary additional work for servers. Choosing an appropriate delay value is another crucial decision.
  - \* Some algorithms notify replica servers when a request is completed, in order to minimize un-necessary work. These notifications produce additional overhead and add to the traffic on the network and servers. Developers should carefully examine whether the benefit of these messages outweigh their costs.

The data retrieval method utilized by a user can dramatically affect their performance. I have examined several different types of retrieval techniques from single server to multiple server utilizations. In single server techniques, the user selects a server based on an approximation of the best server to fit their needs at the current moment. The user’s performance is dependent on the performance of the server selected. If the server’s performance degrades, the user suffers. To help alleviate this situation and also to completely exploit a user’s bandwidth, multiple server techniques have been proposed. These techniques utilize multiple replica servers concurrently by allowing a user to download portions of the data file from several servers simultaneously.

The proposed techniques for data retrieval in grid environments are not adequate. There still exists a need for further study and development. Almost all of the multiple server techniques examined here are evaluated from a single user's perspective. Most of these techniques do not address situations where multiple users in different locations are simultaneously utilizing their techniques, nor do they discuss the effects that these additional users would have on overall grid performance. The Multiple Parallel Downloads with Bandwidth Considerations technique (29) by Chang et al. is the only paper which identifies that consideration should be given to the fact that server performance could degrade as the number of users increases. The authors however, neglect to perform an intensive evaluation of their technique for large, multi-user situations.

The overall performance effects of multiple server retrieval techniques are significant. Data co-allocation increases the workload on servers and networks in the system, especially as the number of users utilizing these strategies increases. Instead of a single user issuing a request to a single server, the user could be issuing tens or even hundreds of requests to various servers during the course of file retrieval. This increased workload has a negative effect on the servers receiving the requests and the networks transmitting the data. The impact is even more dramatic for other users in the system that are not using any co-allocation techniques, since co-allocation increases the workload at all of the servers, even though the number of users remains the same.

The studies of these techniques are superficial and only examine experimental situations under low demand. They neglect to examine their techniques under high-demand situations that would be present in real world grid environments. In addition, most of these studies evaluate their techniques in terms of network transfer time and network throughput. These performance values provide only a limited view of the impact of their techniques. They neglect to examine response times experienced

by users. Response time is an important performance value since it includes wait times, which are key indicators of queue lengths at resources in the grid. Without this information, it is difficult to ascertain the conditions of the resources in their experiments. In addition, they do not provide information about replica workloads or the number of users in the grid when their experiments were conducted. This information is important in order to understand and evaluate their results.

In the following chapter, I present my preliminary work related to grid computing and data retrieval in distributed systems.

## CHAPTER IV

### Preliminary work

In this chapter, I present my preliminary work that led me to the topic of my dissertation. The following sections summarize my research studies, which examine data replication in grid computing (4.1), multi-user co-allocation (4.2), and the performance impacts of parallel data retrieval (4.3).

#### 4.1 Replica Traffic Manager

My initial study in the area of grid computing involved managing users' file transfer requests to replicas in the system (114). Due to the distributed nature of the grid, users send their requests directly to replicas. There is no control over a request once it leaves the user. The focus of this study was controlling workload traffic at data grid replicas, by managing the flow of requests to each replica. I proposed the creation of a replica traffic manager that controls workload traffic sent to the individual replicas in the data grid. The traffic manager receives all user requests and manages the traffic for all replicas by maintaining a certain number of outstanding requests at each replica. When a particular replica is heavily loaded, all incoming requests for that replica would be held in a queue at the traffic manager and/or directed to another replica. Once the traffic decreases at the replicas, the queued requests would be immediately forwarded. By limiting the traffic to each replica, the traffic man-



ager has more control over the system than otherwise possible with individual users submitting requests directly to the replicas.

**Complications:** In my evaluations, I observed that my replica traffic manager has a beneficial effect on the performance of the data grid. In my simulations, I found that the traffic manager provided reliable and consistent response times for users' requests.

Since grids are distributed environments with replicas dispersed around the world, many of the sites in the grid are independently owned and managed. Due to the distributed nature of grids, a replica can be added or removed from the system at any time. In reality, implementing and managing the replica traffic manager service would not be trivial and might not even be possible in all grid environments. By centralizing the replica management service, the traffic manager could also become a bottleneck.

There are two major components of a data transfer between a replica and the user: the storage system component and the networking component. This study does not address the network portion of the data transfer, which is of significant importance. Regardless of which replicas are used to service the users' requests, the task of delivering the data to the user is not simple due to the size of the data sets being transferred.

## 4.2 Simulating multi-user parallel data transfers

As discussed in the previous chapter, there are several recent studies that suggest using parallel transfer (co-allocation) techniques can improve data transfer performance in replicated grid systems. These studies demonstrate that co-allocation techniques can improve network bandwidth and network transfer times by concurrently utilizing as many data grid replicas as possible. However, these prior studies evaluate their techniques from a single user's perspective and overlook evaluations

of system wide performance when multiple users are using co-allocation techniques. In this section, I summarize my paper (115) that provided multi-user evaluations of a co-allocation technique for replicated data in a controlled grid environment. I found that co-allocation works well under low-load conditions when there are only a few users using co-allocation. However, co-allocation performs poorly for medium and high-load conditions since the response time for co-allocating users grows rapidly as the number of grid users increases. The decreased performance for co-allocating users can be directly attributed to the increased workload that their greedy retrieval technique places on the replicas in the grid.

Overall, I found that the global use of co-allocating techniques for replicated data retrieval when done in a greedy or selfish manner is detrimental to user performance in a data grid. This study utilized a simulated environment that only contains grid data transfer workloads. It didn't take into account the workload of other non-grid users that would normally be present on the shared network connections and the Internet. In order to better understand the degree of impact that these types of large file transfers have on other users, I must conduct a study where I examine both large data transfers and normal Internet-user workloads. The following section details this study.

### **4.3 Impacting users with parallel transfers**

In this section, I summarize my study that examined the impact of parallel transfers on other users' workloads (120). It also evaluated the effects of placing retrieval restrictions on these parallel transfers. As previously discussed, the current trend of research on large file transfers is geared towards minimizing a user's service time by any means possible. The goal is to increase and maximize user throughput without regard for overall system performance or stability. Retrieve data as fast and hard as possible.

The users retrieving large data files from the grid/cloud are often on public networks, using a shared connection to the Internet. These public networks could be located in an academic campus or in a research institution, where there are potentially hundreds to thousands of users utilizing shared network resources. When users retrieve large files over these shared resources, everyone is affected. As the number of users retrieving data increases, the impact on the performance of the entire system multiplies. As the load grows, eventually there will be packet loss and failures. Transfer performance for all users will degrade as the demand rises. This is especially true when users utilize retrieval techniques that attempt to utilize as much bandwidth as possible.

The impact of big data transfers on other users as well as system resources has not been entirely examined. In order to fully understand these impacts, I evaluated big data transfers in a controlled testing environment to examine the effects of these transfers. In my experiments, I found that system and user performance suffer as the number of users retrieving large files increased. All users were affected by the increased workload of large file retrievals. The impact on other users that were sharing the public resources was significant. I found that a typical user could potentially see a 86% degradation in transfer performance when other users were concurrently retrieving large files.

Overall, I found that there is significant impact to local system performance when users retrieve large data files over shared, public networks. All resources in the system experience increased traffic and heavier workloads. The increased demand affected the performance of all users in the system. Normal users were unjustly penalized and observed decreased transfer times and longer service times. Restricting large file transfers allowed other users' workloads to have improved performance, however placing restrictions on these large transfers only prolonged their existence in shared system and is not an ideal solution. In order to truly understand the impact of big

data transfers, I needed to examine them on a live system with active user workloads. Only then could I garner insight into how to effectively support both normal user workloads and big data transfers. The following chapter details my evaluations of parallel transfers on a live, shared network.

## CHAPTER V

### Live evaluations of parallel transfers

In this chapter I present my study of parallel transfers on a live, actively used network (116). I utilize the campus network at the university to test parallel transfer techniques by retrieving data from servers distributed around the world. I compare the performance of recently proposed parallel techniques with a new dynamic technique that I developed. My technique is designed to minimize its impact on system resources while still providing fast download times for the user.

Users retrieving large scientific datasets are generally using public networks on academic campuses or in research institutions. Even though they possibly have private storage and computation resources, they must utilize a shared connection to the Internet. In a university environment, several thousand users might share this connection and a single user will be limited to only a portion of the bandwidth available to everyone. The conditions of the network can also vary greatly during different times of the day and different months of the academic year. There is no way to guarantee the network conditions at any given time. Due to these types of situations, parallel transmission techniques are available to users.

These advanced retrieval techniques allow a user to simultaneously use multiple data sources concurrently. The user is not reliant on one server connection for the entire transfer. A user could retrieve half of a file from one server and the remaining

portion from another server at the same time. The number of servers utilized in parallel depends on the algorithm for each technique.

## 5.1 Experiments and Observations

In my experiments I observe the process of a single user retrieving a large data file over a public network, using a shared Internet connection. I examine several different techniques that a user could potentially utilize to retrieve the data file. I evaluate their performance, as well as the difficulties that an average user faces when implementing and using these techniques.

Average users have limited capacity for data retrieval, which is governed by their network connection and their Internet service provider. A user may utilize a shared Internet connection, such as an academic campus network. The Internet connection for the entire network is fast, however all of the users on the network are sharing this resource. In my experimental setup, the user's computer is located on an academic campus network and uses a shared high-speed Internet connection.

Each end user (10,000+) shares the multiple high-speed Internet connections servicing the network. Network workload conditions vary throughout the day, as end users share the public resources. Figure 5.1 illustrates the variations in the user's transfer rate when retrieving a 1MB file from a remote server over the course of several weeks. Since the traffic on local and wide area networks can vary, as well as server workloads, I repeat my experiments several times over the course of three months. I present the average values for all data transfers.

I examine the performance of retrieving a 30GB data file over public networks, using a shared Internet connection. The data file being retrieved is replicated on thirty different servers located around the world. These servers are public servers not under my control and are concurrently servicing other users' requests. The user has the ability to retrieve the file from any of these servers.

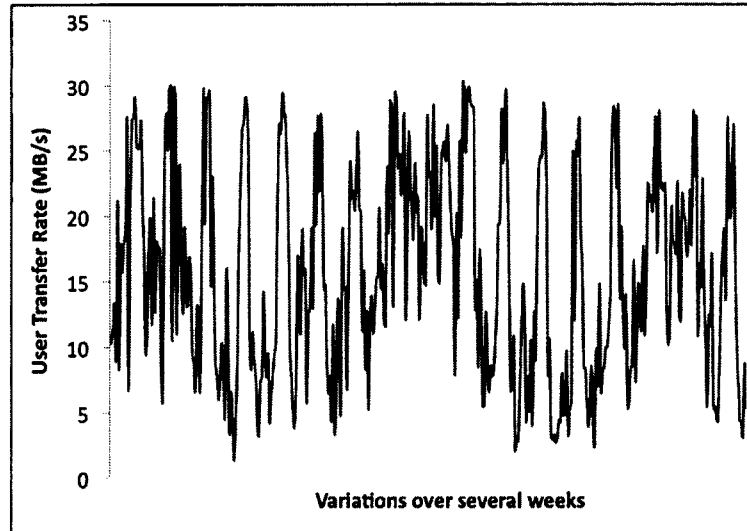


Figure 5.1: Variations in user transfer rates when retrieving a 1MB file from a remote server over the course of several weeks.

#### 5.1.1 Normal Data Retrieval

Normally, the user is faced with the decision of choosing a server from a listing of available servers to service their request. The user has no knowledge about the potential performance of any given server. In my experiments, I retrieve the data file from each server independently in order to observe the differences in service times that a user would experience. I begin my experiments by retrieving the desired 30GB data file from each one of the available 30 servers independently. I find that the data retrieval performance for each server varies greatly. Figure 5.2 illustrates the marked differences in the service times for each server. The fastest file transfer occurred in 11.7 minutes, while the slowest file transfer took over 19 hours. The median service time for all servers is 75.5 minutes.

During the transfers, the user's network utilization is monitored. I find that the user's retrieval capacity was not fully utilized during any of the transfers and was especially low for the transfers with the longest service times. This indicates that the bottleneck of the longest transfers lies with either the connection between the server and the user or with the server itself.

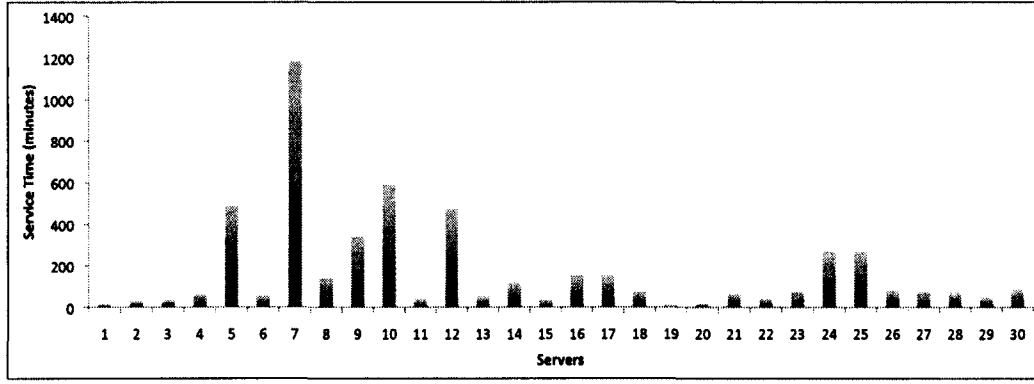


Figure 5.2: Normal Data Retrieval: Service times (minutes) for each server when retrieving the entire 30GB data file independently.

When retrieving data files, the replica selected can greatly impact a user's performance. The major difficulty for the average user is knowing how to select an appropriate replica. Choosing a lightly loaded server over a heavily loaded server can result in dramatically different completion times for a user. Finding the most efficient replica is a difficult and complicated task. As previously discussed, there are many studies (26; 82; 94; 95; 96; 113) that explore different mechanisms for efficient replica selection. All of these mechanisms require the user to implement and configure selection algorithms, which can be beyond the skill set of an average user.

### 5.1.2 Advanced Data Retrieval

Instead of relying on a single server for the entire data transfer, a user could potentially use multiple servers at the same time to transfer the desired data. Many recent studies explore advanced techniques for data retrieval known as distributed file retrieval (or data co-allocation), which allow a single user to simultaneously utilize multiple resources to service a request. Using data co-allocation, users can utilize many or all of the available replicas. The users would issue requests for portions of the data file from these replicas. The requests would then be serviced in parallel. The longest service time that any user would experience would be determined by the slowest replica to service any one of the partial data requests.



There are several different types of data co-allocation retrieval techniques. They can be grouped based on how they utilize the available replica servers. I examine the three most common groups of data co-allocation techniques: brute-force, performance-based, and dynamic. In the following sections, I examine the performance differences and user difficulties that I observe for these techniques when used in my experimental setup.

#### 5.1.2.1 Brute-force Technique

The basic, **brute-force**, data co-allocation technique (110) issues a request for equal sized portions of the file from all available replicas. Every replica that contains the file is utilized and each is responsible for servicing an equal amount of data. There is no consideration given to the performance of replica servers or network conditions. The workload at all servers is increased equally for each co-allocating user.

I evaluate the brute-force technique (BFT) by dividing my file request into 30 equal-sized portions and requesting one portion from each of the 30 replica servers. The requests are serviced concurrently and the data is retrieved from each server in parallel. Since the entire file request is not complete until all of the portions are retrieved, the performance of the request is dependent on the slowest file transfer. Similar to my normal data retrieval observations, I find that the performance of each of the transfers varies greatly. Figure 5.3 illustrates the differences in the service times for each of the individual file portion retrievals. As with normal data retrieval, server 7 provides the longest service time. The fastest file retrieval finishes in 2.5 minutes and the slowest file retrieval takes 76.8 minutes. Since the data retrieval is not complete until all portions are retrieved, the service time for the entire data file transfer using BFT is 76.8 minutes.

In comparison to my normal data retrieval experiments, the brute-force technique provides improvement over normal data retrieval for some of the servers. The average

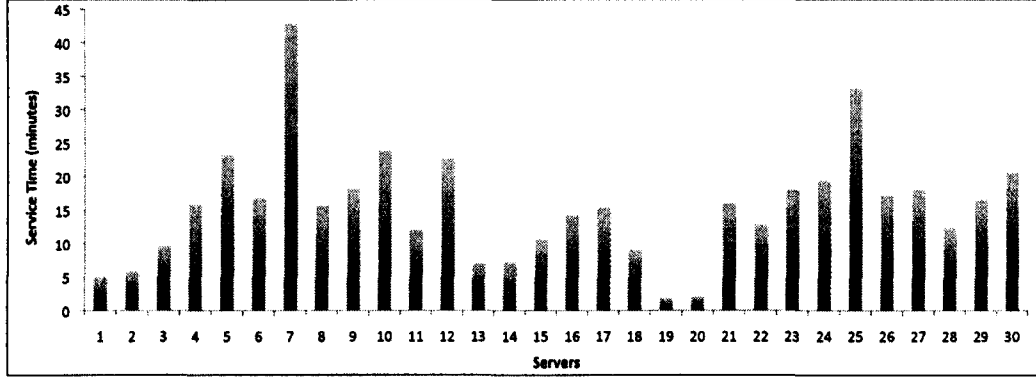


Figure 5.3: Advanced Data Retrieval - Brute Force Technique: Service times (minutes) for each server when retrieving equal 1GB portions of the 30GB data file.

service time for BFT is almost equal to the median service time for the single server technique. This indicates that the BFT provides improved performance in comparison to retrieving the file from a single server for 15 of the available 30 servers. Since the BFT technique utilizes all servers regardless of their retrieval capacity, the slower servers will always hinder the performance of the entire data transfer.

There are several difficulties that an average user would face when using the brute-force technique. Initiating and monitoring multiple transfers can be difficult. In my experiments, I utilized 30 concurrent transfers, which proved to be complicated to track. With multiple simultaneous transfers, the task of setting up and monitoring the individual transfers can be overly complex for the average user.

#### 5.1.2.2 Performance-based Technique

Performance based techniques utilize performance metrics when selecting replicas to utilize. There are two main groups of performance-based techniques: history-based and probe-based. Both of these groups attempt to exploit faster servers by assigning them greater portions of the workload. Depending on the user's choice, the number of servers utilized in parallel can vary from two to possibly all of the available servers.

In **history-based techniques** (110; 111), the retrieval algorithms address the

fact that each transfer between a replica and the client has varying transfer rates. These techniques adjust the amount of data retrieved from each replica by predicting the expected transfer rate for each replica. The algorithms create forecasts of future performance based on transfer history with network and disk load data. Historically faster servers are assigned to deliver larger portions of the file and slower servers are assigned smaller pieces.

In **probe-based techniques** (36; 131), the retrieval algorithms utilize network status information to create network throughput predictions. Some of these techniques utilize the Network Weather Service (124), which is a networking monitoring tool that utilizes sensors which gather data on the latency and bandwidth of end-to-end TCP/IP performance. Using these throughput forecasts for each replica server, the algorithms assign portions of data request to each available replica. Replicas predicted to have the best performance are assigned a larger portion of the request workload.

I evaluate a performance-based technique (PBT) by selecting servers using the round-trip time from a network ping and performance information from the transfers that I observed when examining the brute-force technique. I select servers with the lowest ping times and the shortest historical service times first. I vary the number of servers that are used concurrently from two to twenty. As the number of servers utilized increases, I use slower servers with larger round-trip times and longer historical service times. I compare the overall service times that I experience for the varying number of concurrently utilized servers in Figure 5.4. I find that as the number of servers increases, the overall service time also increases. When more servers are used, slower servers are required to service portions of the request. The request is not complete until the slow servers finish their portions and thus affect the overall service time. When only the two servers with the best metrics were utilized, the overall time to retrieve the file was the least.

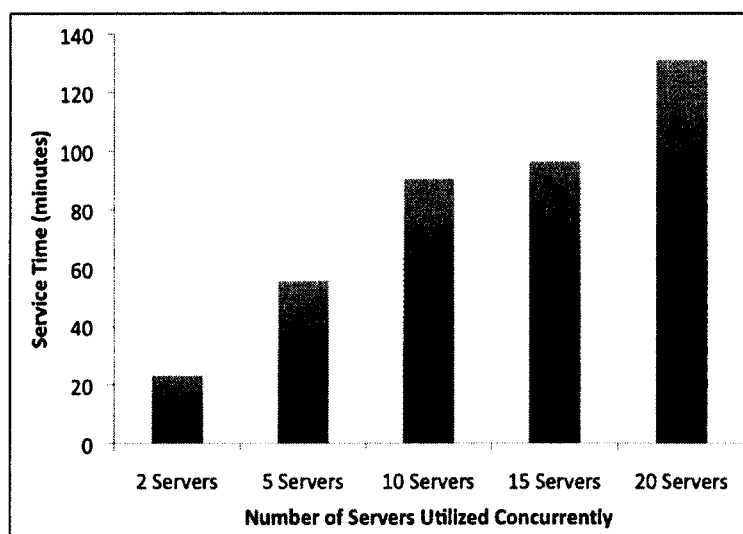


Figure 5.4: Advanced Data Retrieval - Performance-based Technique: Total service time (minutes) for the file transfer as the number of servers concurrently used increases.

I also observe that as more file transfers are added, the user's available retrieval capacity diminishes. Eventually, there are more file transfers than the user's connection can handle and the transfers will compete for the available retrieval capacity. This can negatively affect faster transfers. Figure 5.5 illustrates the effects of multiple parallel file retrievals on the transfer rate of the fastest connection observed. As the number of concurrent data retrievals increase, there is a decrease in the transfer rate for the fastest file transfer. There is a 77% decrease in the transfer rate when there are 29 other transfers competing for available retrieval capacity.

While probe-based techniques provide improved performance over brute-force techniques, they still attempt to utilize as many servers as necessary without regard for the user's limited retrieval capacity. In many cases these techniques create more transfers than the user's bandwidth can accommodate, which results in transfers competing for bandwidth. This situation diminishes the performance of the overall file transfer.

Users are faced with several difficulties when utilizing performance-based retrieval techniques. Since these techniques are more complex than the brute-force technique, their implementation could prove difficult for the average user. Another issue that a

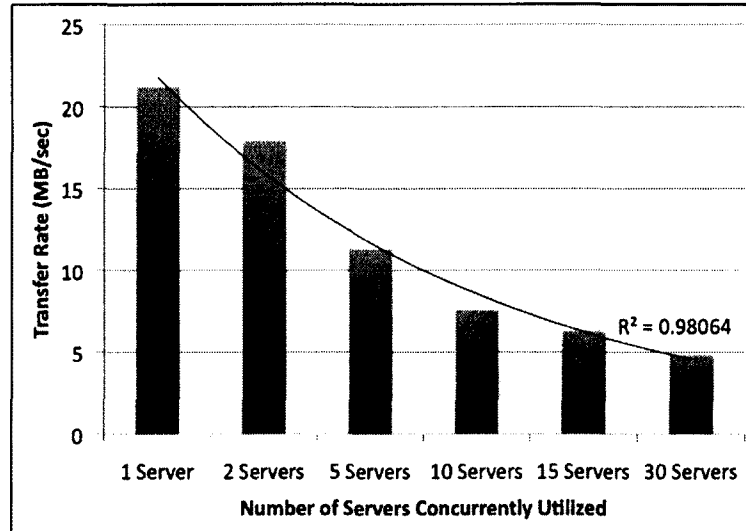


Figure 5.5: Transfer rates for the fastest server connection observed, as the number of servers concurrently used increases.

user faces is the problem of stale performance metrics. Network conditions and server workloads are constantly changing, which means that these metric values can quickly become inaccurate. Since these data transfers could potentially take multiple hours to several days, users will need to continuously update their performance metrics for all server connections.

### 5.1.2.3 Dynamic Techniques

Dynamic techniques (24; 29; 121; 127) attempt to automatically adapt to changing system conditions by requesting small, equally sized, portions of a file from multiple replicas. In many dynamic techniques, each replica is initially assigned one segment. As replicas complete their assigned segments, they are assigned additional portions of the data file to service. Each dynamic technique uses different decision making algorithms on how to schedule these requests, however faster servers will end up transferring larger portions of the file. Any failed or undelivered requests can be automatically rescheduled to other replica servers, potentially created duplicate work. Depending on the specific dynamic technique, the desired data file could be segmented so that a single server could receive tens to hundreds of requests for portions of one

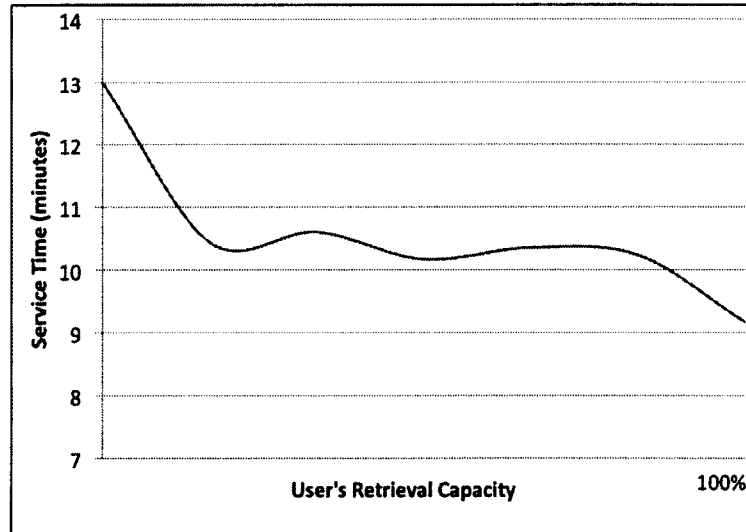


Figure 5.6: Incremental Distributed File Retrieval: Changes in service time (minutes) as the user's retrieval capacity approaches its maximum utilization.

file.

**My Dynamic Technique:** My technique attempts to fully utilize the user's retrieval capacity by dynamically and incrementally increasing the number of servers currently utilized until the user's maximum retrieval capacity is reached. This technique attempts to avoid situations where several transfers are fighting for available bandwidth. It allows requests for large amounts of sequential data from the same server, which enables the servers' storage systems to effectively utilize their caching and pre-fetching schemes.

This dynamic technique begins by selecting one server with the smallest round-trip time using a network ping. After the transfer has started, the user's available bandwidth is monitored. The technique then incrementally creates additional data transfers to other servers, as necessary until the user's retrieval capacity is fully utilized. Figure 5.6 illustrates the decrease in service time for the incremental technique as I approach full utilization of the user's retrieval capacity. The service time for this technique was 9.2 minutes, which was less than the fastest time observed using normal data retrieval.

In comparison to the other techniques, my dynamic technique produces the small-

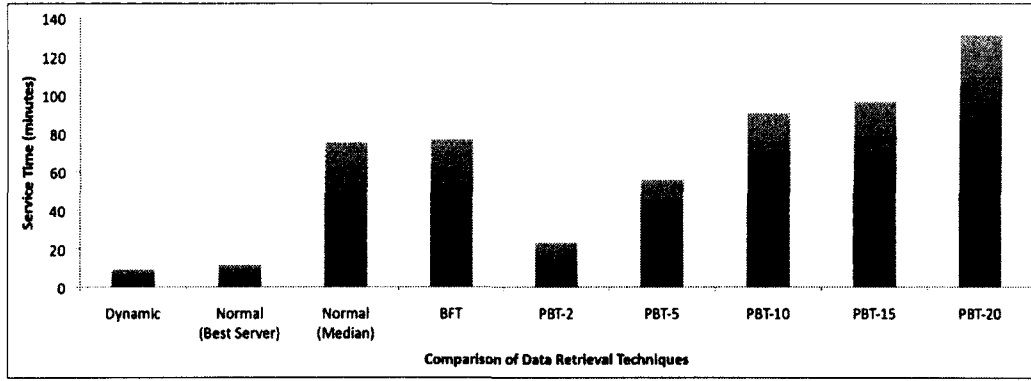


Figure 5.7: Comparison of service times (minutes) for all data retrieval techniques observed.

est service time for retrieving the 30GB data file. Figure 5.7 shows the difference in service times for all of the techniques that I observe. In addition to providing the smallest service time the user, the dynamic technique attempts to involve the smallest number of replica servers and attempts to fully utilize the user’s retrieval capacity.

The user difficulties associated with dynamic techniques are numerous. Many of these techniques are quite complicated and their algorithms are complex. Implementing them for automated use would require significant time for even an experienced programmer. The average user would find this task to be insurmountable. In addition, there are many aspects of these algorithms that are left for the user to decide and control. I detail some of these issues and challenges in the next section.

In summary, I find that advanced file retrieval provides improved performance in comparison to normal data retrieval. There are several advanced techniques available for the user to utilize. Choosing an appropriate and viable technique however is a difficult task. Implement, configuring and utilizing these techniques can be a challenge for even for the experienced user.

## 5.2 Issues and Challenges

Retrieving large data files (GB, TB, PB) is a complicated and time-consuming process. These long duration transfers could take tens of hours to several days and

a normal “one click and wait” method will not suffice. During the course of the transfer, servers may go off-line and network conditions may change that either hinder or stop the transfer completely. The user needs to know how to maintain the data transmission until completion.

Advanced retrieval techniques allow users to utilize multiple resources simultaneously. These advanced techniques provide improved performance for users, however they are quite complicated to implement and use. They require significant user involvement and require multiple user decisions that can dramatically affect the performance of the transfer. A user needs know-how in order to make these techniques function properly and efficiently.

Another configuration option that is frequently left for the user to determine is segment size. In some advanced techniques, the data file is divided into small portions called segments. The segment size is often left for the user to decide and the size chosen can affect the performance of the transfer. Determining the appropriate segment size is not a simple task. If the size is too small, a server may receive hundreds to thousands of requests for portions of a single file. This will result in longer disk service times at a server, as the number of users increases. A server’s storage system can best service requests if it has greater knowledge of a user’s workload. It can better schedule reading from the hard disks, as well as take advantage of pre-fetching and caching strategies.

A key issue that is not adequately addressed for retrieval techniques is failures. Since I am transferring extremely large data files over long periods of time, I will eventually encounter transfer failures. Many advanced techniques identify that failures can occur and provide mechanisms for issuing new requests, however specific details about the timings of these actions are not addressed and are left for the user to decide. The request re-issue delay is a common problem with these techniques. Determining the appropriate amount of time that the application should wait before



issuing a replacement request is a non-trivial task.

The most important outcome of this study is not the fact that my dynamic parallel transfer technique outperformed existing techniques, but the degree of impact that I had on the campus network and other users in the system. I address and discuss this impact in the following section.

### 5.3 Impacting other users

During my live experiments, I was contacted by the department network support team, as well as the university's telecommunication department. My experiments impacted the service of the subnet as well as the general Internet connections during peak usage times. During low usage intervals, other users' workloads were only minimally impacted due to the limited number of active users on the subnet. During high demand periods however, my experiments increased the load of the subnet link to near full capacity, which resulted in service problems for other users. The support teams received complaints from users that were experiencing problems with their network applications.

Working with university support teams and utilizing bandwidth-monitoring devices, I was able to obtain bandwidth utilization graphs for all of the shared Internet (WAN) connections for the campus. Figures 5.8a, 5.8b, and 5.8c show the bandwidth utilization for the Internet connections before, during and after my experiments. Before I began my evaluations, the university was on a mid-semester break. During this time, there was very little network load. Only automated and system traffic is present in the system at this time. Once users returned to campus, I initiated my experiments, so that my workload would be intermixed with normal everyday traffic.

Since my experimental traffic is mixed with all other users, I am unable to precisely isolate my traffic in the graphs. Figure 5.8a illustrates the bandwidth utilization for the Internet2 traffic. Since most student and staff traffic very rarely use the Internet2

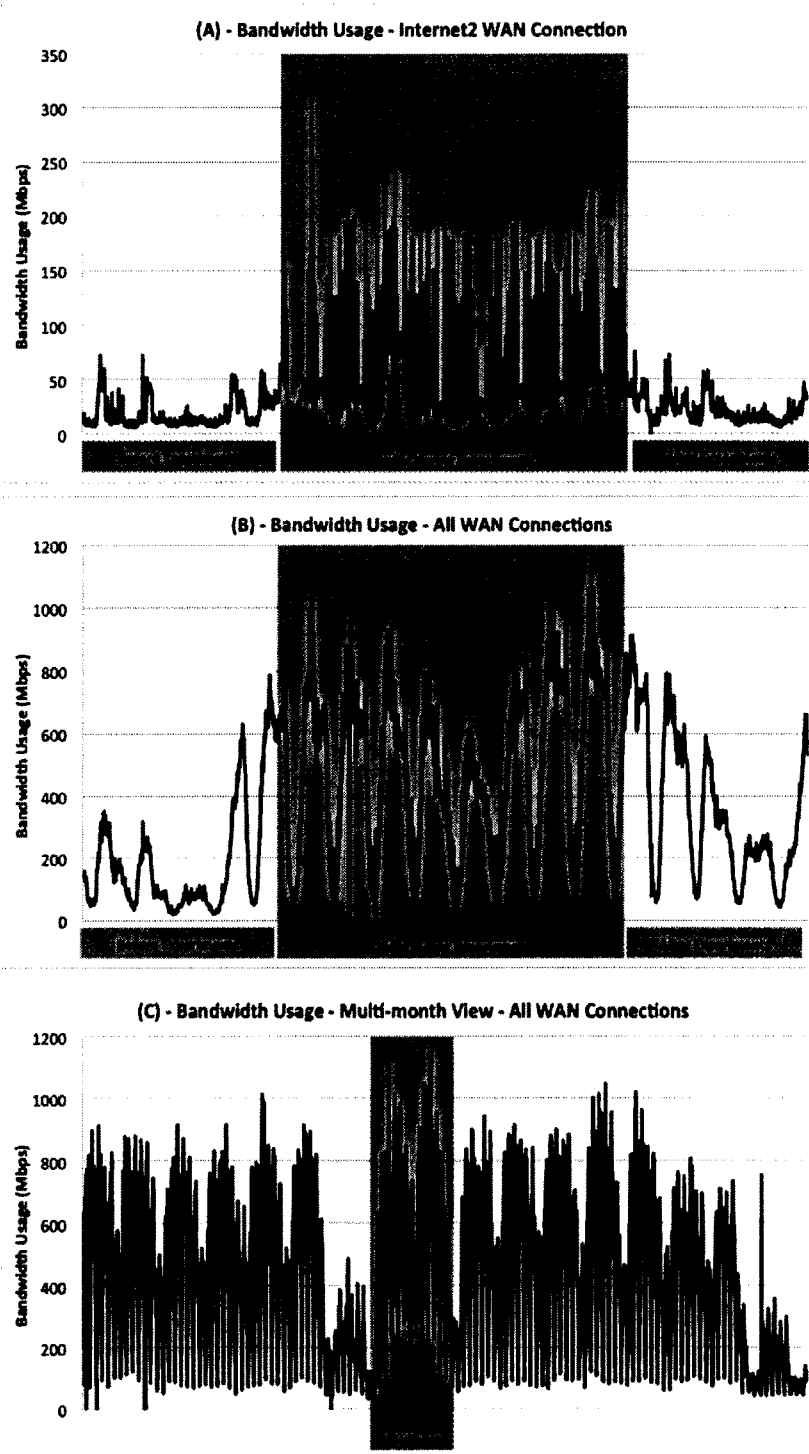


Figure 5.8: Bandwidth usage on wide area network connections before, during and after my evaluations. The shaded regions indicate the time period during my experiments. (a) - This graph shows bandwidth usage of the Internet2 connection for a two-week period. (b) - This graph shows total bandwidth usage of all WAN connections for a two-week period. (c) - This graph shows total bandwidth usage of all WAN connections for a four-month period.

link, it is easier to discern my experimental workload in the graph. Multiple sources in my experiments are located at universities on the Internet2 network and therefore my workload has a dramatic impact on the bandwidth utilization of the Internet2 link. As the graph indicates, the normal usage before and after my experiments is quite low in comparison. There is a significant increase in traffic on this link during my experiments.

The impact of my experiments on all WAN connections is slightly harder to see when it is intermixed with all other users' traffic. Figure 5.8b illustrates the total bandwidth utilization for all of the campus WAN connections. During my experiments, the total utilization reached its highest peaks during the two week time span. When the time range for this graph is increased to four months, as shown in Figure 5.8c, it is easier to notice the impact of my experiments. The bandwidth utilization again reached its highest peak during my experiments, as well as maintained a higher utilization for the entire experimental period. From all three of these graphs, it is clear that large file transfer workloads can impact the performance of the entire campus network, especially during high demand periods. It was during high demand periods when I impacted other users the most.

At the start of this study, my focus was on creating an efficient parallel transfer technique. At the end of the study, I realized that no matter the speed or efficiency of your transfer technique, it will only perform as well as the network/system you are on. Impacting other users with big data transfers is not responsible and could potentially cause you to have your Internet access restricted or even revoked.

In order to develop a new approach to big data transmissions, I needed to fully examine the campus network and its workloads. The following chapter details the findings of my campus network study.

## CHAPTER VI

# Examining the campus network and its user workloads

In order to understand how big data transmissions affect users on a shared network, I need to first analyze the architecture of these networks and their user workloads. In this chapter, I present my study of the UNH campus network (119). I first examine and detail the structure and setup of the network. I then analyze the network traffic to identify patterns in network usage and categorize the workloads of the campus users.

Campus networks are a microcosm of the Internet. The university campus is the workplace for researchers, faculty, staff, and students. Unlike commercial networks, the campus is also a home for the majority of the student body. The campus network must therefore support both academic and non-academic workloads in order to keep all users on campus content. The system must support a wide variety of application classes, such as: email, web browsing, streaming multimedia, gaming, video conferencing, voice over IP, cloud/grid workloads and file transfers. Each of these application classes has its own demands and requirements for bandwidth. In this chapter, I characterize bandwidth utilization rates, users' access patterns and data consumption amounts for these application classes on the UNH campus network.

Over the past few years there has been a major shift in Internet applications

used on campus networks. Users have progressed from low-bandwidth, best-effort applications to real-time and bandwidth intensive applications. One such application is streaming multimedia, which is capable of consuming extraordinary amounts of bandwidth (28; 22; 42; 53; 132; 51). Each year the bandwidth utilization rates are increasing for these types of applications. Since these applications can dynamically adjust their output quality based on bandwidth availability, they have unbounded demand for Internet resources. Users continually want better quality and high-definition viewing, which places extreme strain on system resources, especially on campus networks. A workload characterization is needed to determine the degree of impact these types of applications have on the campus network and how users are using these applications.

I realize that this network data represents only one possible network configuration used by academic institutions. Obtaining the following detailed data about bandwidth usage and user workloads required several rounds of authorization and working with network administrators to access live, mission critical hardware devices. Attempting to obtain similar in-depth data from other institutions and corporations proved impossible due to security concerns and confidentiality issues. I realize that some of the specifics from my analysis might only relate to the UNH network, but the trends that I observe are definitely present at universities throughout the country (78; 87; 102).

The rest of this chapter is organized as follows: first, I explain the configuration of the campus network in Section 6.1. I identify bandwidth usage information in Section 6.2. I then present the workload characterization for the Internet applications used on campus in Section 6.3. In Section 6.4, I examine system performance when users are given additional Internet bandwidth. Finally, I summarize my findings in Section 6.5.

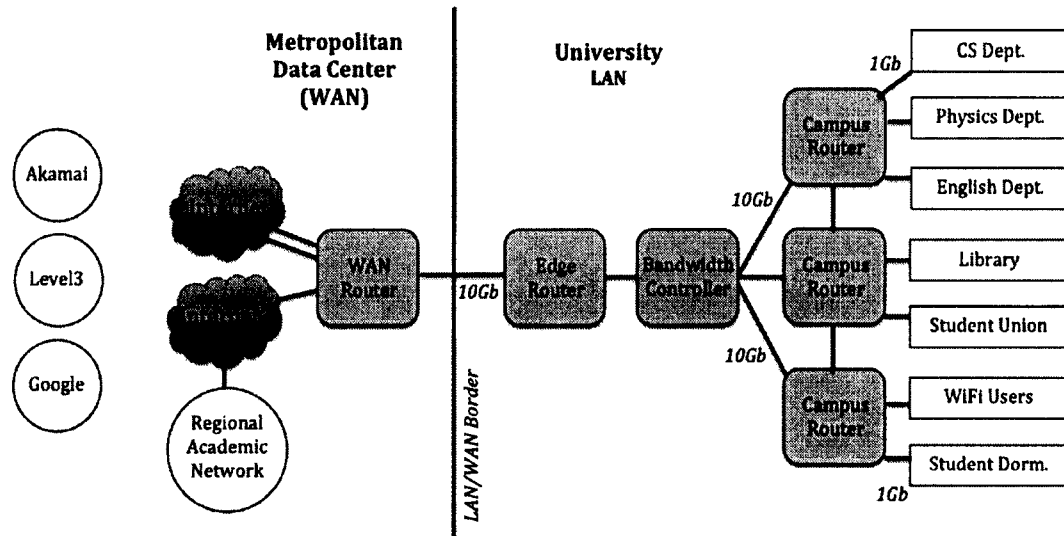


Figure 6.1: Network layout for the university network and its connection to the shared data center in a nearby metropolitan area.

## 6.1 Campus network configuration

In this section, I describe the general setup and configuration of the UNH campus network. This network, illustrated in Figure 6.1, is designed to support over 10,000 students with an average of 6000 concurrent connections. Users consist of students, faculty and staff. These users connect to the network through Ethernet or WiFi connections and are distributed across multiple subnets around campus that are connected to the campus core via 1 Gbps links. The core of the campus network consists of 10 Gbps connections.

At the edge of the campus network, all traffic destined for external locations passes through a bandwidth management device. This device monitors the workload for each IP address and controls the bandwidth usage for each user. After the traffic passes through the bandwidth manager, it continues through a private 10Gb fiber connection to a nearby metropolitan area, as shown in Figure 6.1. When it reaches the city, the traffic arrives at a shared data center, which contains access points to major telecom networks, regional universities and major corporations' services (such as Google, Akamai, Level3, etc.). The outgoing data are then routed to three different wide area

network connections. Two of these connections are to the public Internet and one of these connections is to Internet2, a non-profit network designed to support research and educational institutions (3). Traffic destined for the Internet is load balanced between the two general Internet WAN connections, which have a total bandwidth capacity of 1.5 Gb/s. The Internet2 connection has a variable bandwidth capacity, which allows on average 500 Mb/s. This results in a total bandwidth capacity of 2.0 Gb/s for the entire university network.

Incoming data destined for a user on the university's LAN comes into the three shared Internet connections. This data could be streaming multimedia from a nearby CDN server or a webserver at a regional university. All of this data crosses the LAN/WAN border and then passes through the private link to campus. All incoming data continues through the bandwidth manager before being routed to the correct subnet and finally to the end user.

Since the university supports over 10,000 users, the campus network has to ensure that each user has equal and fair access to the shared Internet connections. In order to accomplish this task, the university employs a bandwidth management device that is located at the edge or border of the LAN network. Each user device is limited to 8 Mb/s. As demand for bandwidth increases, the per device bandwidth allowance will be further restricted. The bandwidth manager is imperative in making sure that everyone has fair and equal access to the shared Internet connections. It however does not ensure that users will have sufficient bandwidth and capabilities to utilize their desired applications. Under high load conditions, a user might only receive a small fraction of available bandwidth. This amount might suffice for web browsing and email messaging, however streaming media and applications requiring low latency or quick response times will suffer.

In the following sections, I characterize the Internet workload for the campus network. I examine the total amount of traffic flowing into and out of the shared

Internet connections. I also examine the applications that are transferring the largest amounts of data over the campus network. In order to gain access to this information, I utilize network monitoring devices that are placed throughout the network. I gather live data from the network and perform off-line data analysis of all traffic flows. I also use the bandwidth manager to gather data regarding users' workloads.

## 6.2 Bandwidth usage

I begin my characterization of campus Internet workloads by examining the total bandwidth usage of the shared Internet connections for the campus network. I monitor and examine bandwidth consumption on the campus network for an entire academic year. Figure 6.2 illustrates the variations in the daily maximum bandwidth consumption for this 12 month period. I find that there is very high demand during academic semesters and reduced demand during breaks. Since students are the main consumers of bandwidth on campus, changes in consumption correlate to their leaving and returning to campus. There is however a constant level of usage throughout the year regardless of the month, as the university hosts multiple government run projects that continually transfer data. Internal services that connect to satellite and regional campus networks also conduct data transfers on regular schedules.

Figure 6.2 demonstrates that several times during the Fall 2010 and Spring 2011 semesters the maximum bandwidth usage rates reached the bandwidth limits of the shared Internet connections for the entire campus network. Multiple times throughout the semester users consumed their entire bandwidth allotments and were forced to utilize less than their maximum rate of 8 Mbps.

Bandwidth demand changes throughout the year, as illustrated by the peaks and valleys on the graph. In order to understand of these shifts in demand, I examine the bandwidth usage from a weekly perspective. Figure 6.3 shows the maximum, average and minimum bandwidth usage for a typical week during the Spring 2011 semester. I



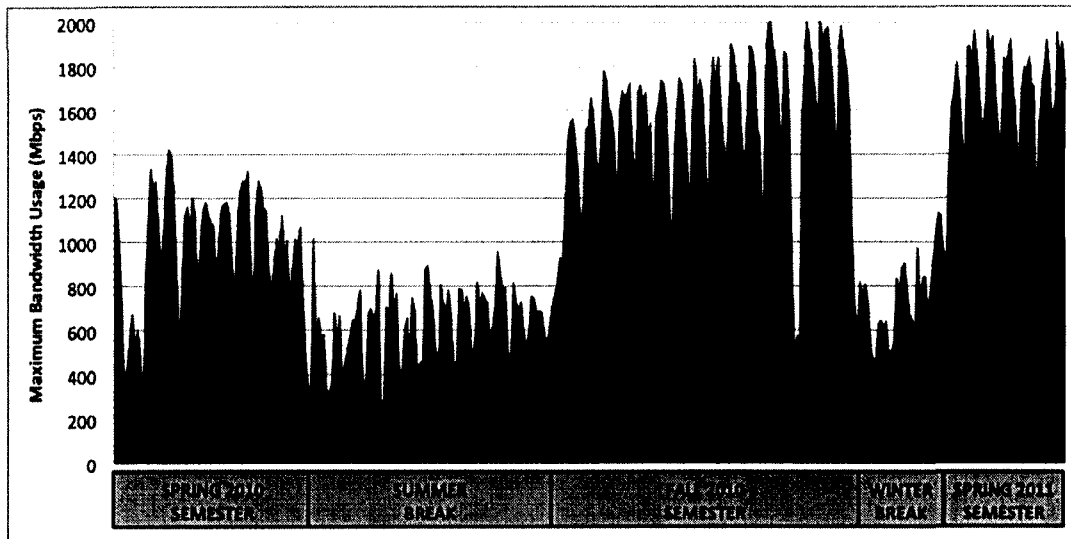


Figure 6.2: Changes in maximum bandwidth consumption over 12 months for all data passing through all of the university's shared Internet connections. Each semester user demand and bandwidth consumption increases.

find that network usage is the highest between Sunday evening and Friday afternoon. This correlates with classes starting and ending for a given a week. Between Friday night and Sunday afternoon, the network utilization is generally at its lowest. Even the maximum bandwidth rates during this period are much lower than during the rest of the week. I attribute this occurrence to the fact that many students and staff leave campus or reduce their network usage on the weekends.

As I observe that the network utilization changes from day to day, I also find that it changes from hour to hour. In Figure 6.4, I examine the maximum, average and minimum bandwidth usage for each hour in a typical weekday during the Spring 2011 semester. I find that peak usage occurs between noon and midnight. There is a slight dip around dinnertime and then usage increases until 1AM when demand starts to drop off. The lowest usage point occurs between 4 and 7 AM and then demand increases as faculty return to campus and students prepare for the start of classes. Throughout the 24-hour period, there is always some amount of bandwidth utilized as indicated by the minimum values on the graph. I observe very large differences between the minimum and maximum values, which indicates that users' workloads

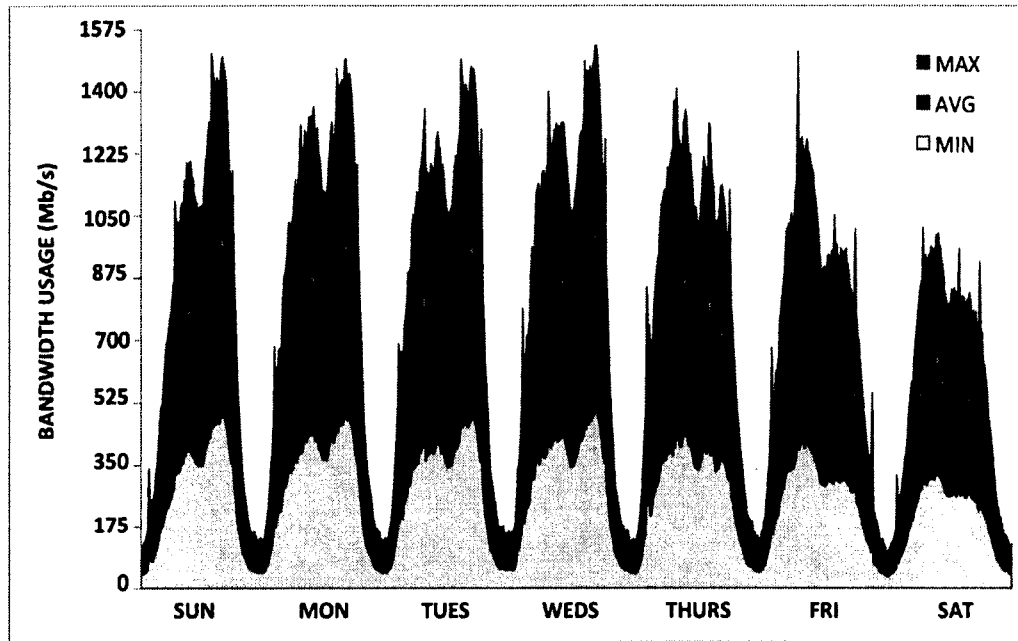


Figure 6.3: Changes in the minimum, average and maximum bandwidth usage (all receiving and transmitting traffic) for a typical week during the Spring 2011 semester.

are dynamically adapting to changing bandwidth availability.

Overall, I find that a significant amount of data is transferred between the campus network and the Internet daily. On an average day during an academic semester, about 7 TB of data is transferred through the shared Internet connections. 2.5 TB of outgoing data is sent to the Internet and 5.5 TB of data is transferred into the campus network. Figure 6.5 demonstrates the total amount of data transferred each day using the campus network's shared Internet connections. The maximum amount of data ever transferred in a single day is roughly 10 TB. As in Figure 6.2, I also observe usage patterns that correlate to the academic calendar. More data is transferred during the Fall and Spring semesters than any other time. As previously discussed, there is a constant workload for the shared Internet connections and they are never completely idle. The minimum amount of data transferred on any day in the year is 870 GB, which occurred on Christmas day.

**Bandwidth Summary:** Overall, I find that largest amount of data transferred

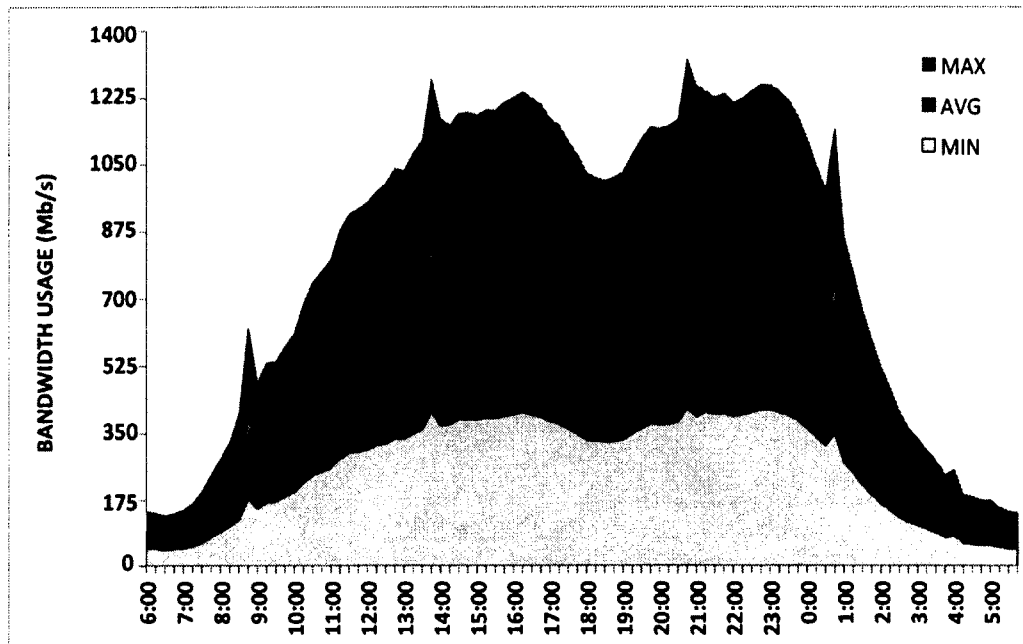


Figure 6.4: Changes in the minimum, average and maximum bandwidth usage (all receiving and transmitting traffic) for a typical day during the Spring 2011 semester.

between the campus network and the Internet occurs during academic semester in the Fall and Spring. There is a continual amount of traffic regardless to the time of year, which is created by special projects and internal services on campus. The peak usage time for the campus network is between noon and midnight from Sunday to Friday. I see decreased usage during the early morning hours (4AM to 10AM) and on the weekends. On a typical day the campus network is transferring roughly 7 TB of data to and from the Internet.

### 6.3 Internet application workloads

In the previous section, I characterized the amount of data being transferred to and from the Internet on the campus network. The next component of my characterization is to identify the applications that are transferring these large amounts of data.

Working with network management devices on campus, I was able to obtain usage

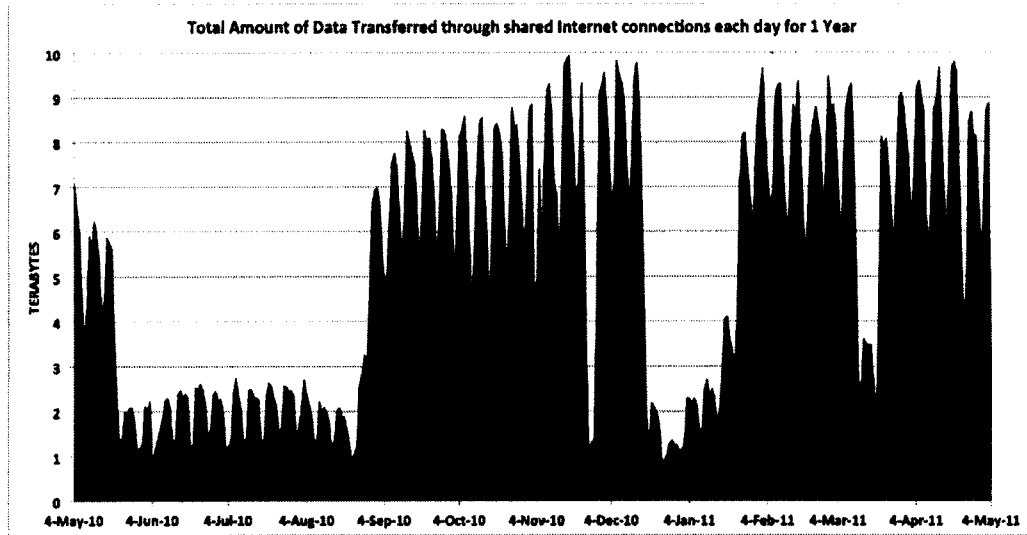


Figure 6.5: Total amount of data transferred by the campus network each day.

profiles for users on campus. I monitored user traffic for a period of 35 days during the Spring 2011 semester. I examined the traffic workload to identify the top applications consuming Internet bandwidth during this time period. Figure 6.6A illustrates the applications that consume the most amount of bandwidth on a typical day for all users. I found that the applications utilizing the largest amount of Internet bandwidth are streaming multimedia applications, such as Netflix, HTTP Streaming and YouTube. On a typical day, these three application classes consume more than triple the bandwidth of general web browsing. This is the case on many campus networks, as well as the entire Internet (78; 87; 102). Netflix currently consumes the most amount of bandwidth for the entire Internet (130; 107). I also found that Skype and file transfers register in the top ten application classes. Popular applications such as Facebook and iTunes rank in the top 15 user applications.

I continued my workload characterization by examining application usage by user type. I began by comparing the usage patterns for students and faculty staff. In Figure 6.6B, I identified the top bandwidth consuming applications for faculty and staff users. I found that their workload is dominated by web browsing and file transfers. The applications with the next highest levels of bandwidth consumption are

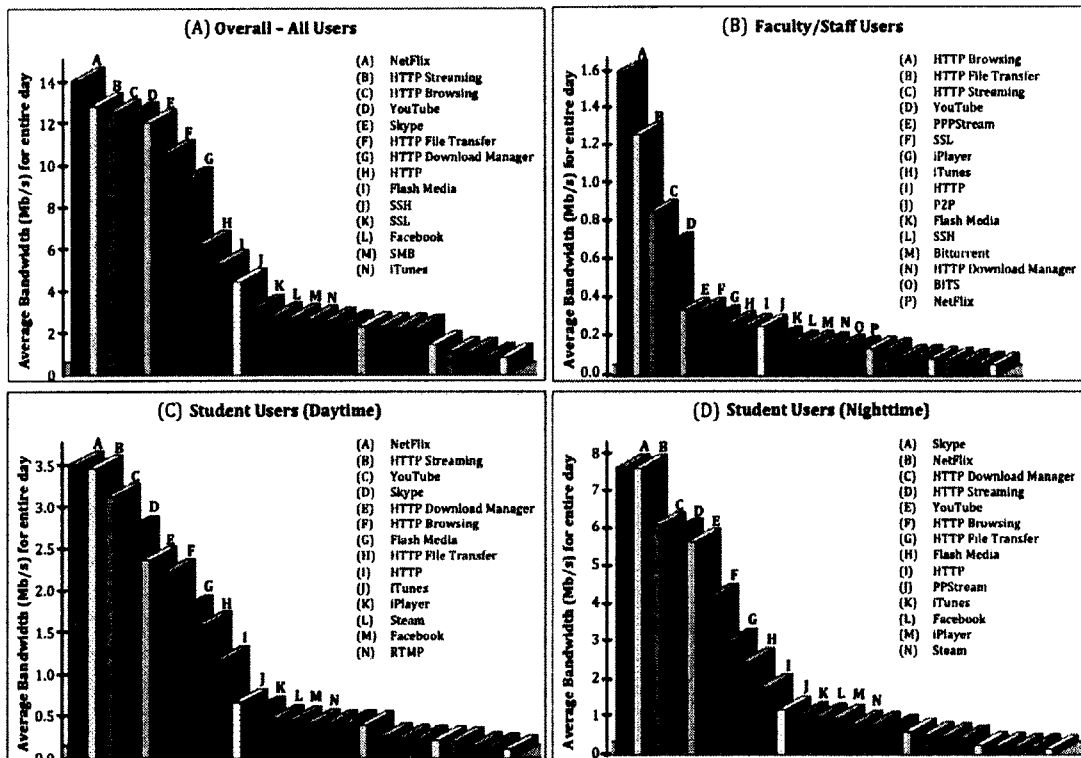


Figure 6.6: Most active protocols utilized on an average day. Protocol usage by types of users are shown: A) all users, B) faculty/staff users, C) student users during the daytime and D) student users during the nighttime.

streaming video and YouTube. Netflix is very low on the list of applications for the staff users. The bandwidth used by web browsing for the faculty is double that of any streaming application for their user group, very unlike the student users.

Figures 6.6C and 6.6D illustrate the top applications for the student users on campus. I separated the applications by daytime and nighttime usage. During the day, the applications utilized by the students are mainly streaming multimedia (Netflix, YouTube, HTTP Streaming). At nighttime, the same streaming applications are still high in the list of applications consuming the most Internet bandwidth, however the bandwidth usage for these applications increases in the evening time. The major difference between daytime and nighttime periods is that Skype utilization increased dramatically. Skype is the top application for bandwidth consumption during nighttime hours.

My user workload characterization also examined the changes in application usage based on the time of day. I have already compared student usage during the day to nighttime. I continued my characterization by looking at all users for specific hourly periods over the course of 24 hours. Figure 6.7 illustrates the changes in bandwidth utilization of the top application classes over the course of a typical day. The top bandwidth consuming application, Netflix, is used to the greatest extent between 6PM and midnight. Netflix utilization is double during this time period in comparison to other parts of the day. Skype also has a significant increase in utilization during the evening time. Skype bandwidth consumption increased by 300% at night. Web browsing, YouTube viewing and HTTP streaming applications have the highest usage levels between noon and midnight. All applications see decreased usage between 6AM and noon. Skype and Netflix have the most noticeable decreases when compared to their peak periods. Web browsing is the only application to have usage levels during the 6AM to noon period that are comparable to normal daytime rates. The SSH application class has a fairly consistent level of usage regardless of the time of day. Many internal services (data backups and replicated data sets) utilize SSH for automatic file transfers throughout the day. The average daytime (6AM-6PM) rate is almost equal to the average nighttime rate (6PM-6AM) for the SSH application class.

In addition to characterizing the bandwidth usage rates for the application classes that make up the Internet workload on campus, I also identified the total amount of data being utilized by each application class. In Figure 6.8, I display the applications that received and transmitted the greatest amount of data between October 2010 and May 2011. Since this time period includes Winter break, the data essentially display usage information for six months. I identified the top five applications for both sending and receiving. I found that users utilizing the Netflix application were able to receive over 25,000 GB of data during the six month period. Both HTTP streaming and

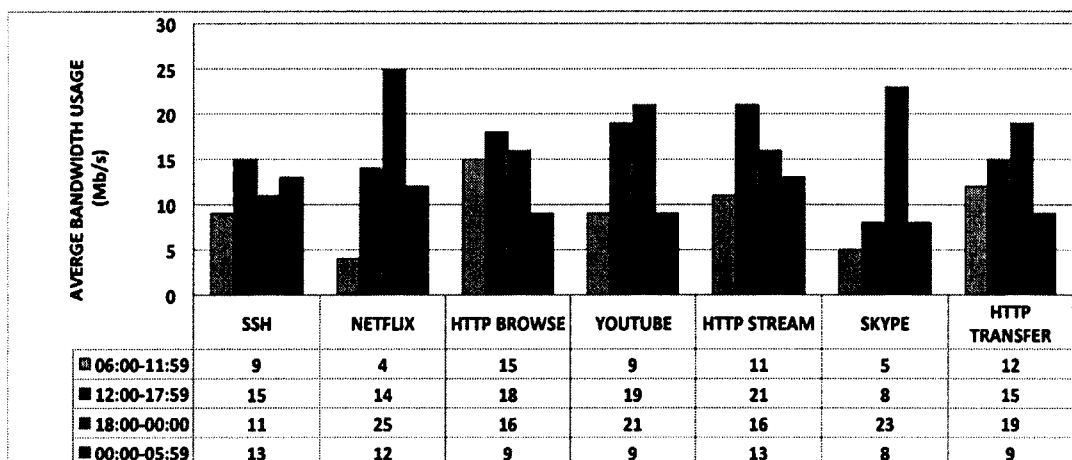


Figure 6.7: Changes in protocol usage for the most actively used protocols on campus throughout a typical day.

YouTube received over 44,000 GB combined. Web browsing and HTTP file transfers each consumed roughly 15,000 GB of data individually.

I also examined the applications sending the most amount of data from campus to the Internet. The amount of data leaving the campus network for the Internet is considerably lower than the amount of data being received. Skype sent the largest amount of data during the six month time period, almost 10,000 GB. Both the sending and receiving amounts for Skype were almost identical. The next two applications that sent the largest quantities of data out of the UNH network were secure communications (IPSEC-ESP and SSH). Each of these application classes transferred over 7000 GB of data out of the campus network. File transfers and web browsing also sent about 6000 GB. Web browsing had a bandwidth usage ratio of 2:1. The amount of data being received by web browsing users was double that of the data being sent by the same users. A full table of the data amounts by application is displayed in Figure 6.9.

**Application Summary:** Overall, I found that the real-time, bandwidth-intensive applications dominate the Internet workload on campus. Users are utilizing interactive applications that are sensitive to changes in latency and network congestion.

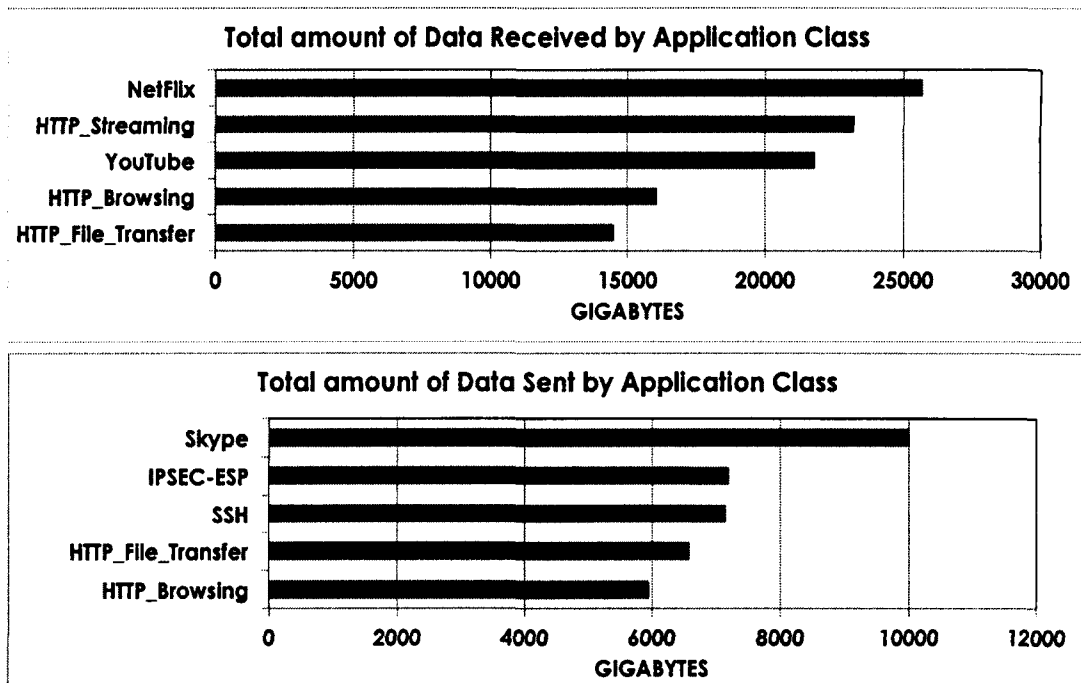


Figure 6.8: Total amount of data transferred by each application class between October 2010 and May 2011 for all users on campus.

Netflix consumes the maximum bandwidth and receives the largest amount of data in comparison to all other applications on campus. Skype transmits the largest amount of data to the Internet. Web browsing and SSH communications have fairly stable usage patterns in comparison to other applications.

## 6.4 Increasing bandwidth

In the previous two sections, I characterized bandwidth consumption and user workloads. I continued my characterization by examining changes in user workloads when bandwidth is increased for users on campus. It is common to think that giving users additional bandwidth will solve any performance problems on campus networks, however I found that any extra bandwidth is quickly consumed. I discovered this situation in multiple instances.

During the nighttime hours, students are given a portion of the faculty's band-



Protocol	Total Bandwidth (GB)	% Total Bandwidth	In Bandwidth (GB)	Out Bandwidth (GB)
NetFlix	26221.97	5.7	25640.919	581.052
HTTP_Streaming	24088.721	5.2	23172.675	916.045
YouTube	22511.961	4.9	21757.611	754.35
HTTP_Browsing	21971.461	4.8	16035.051	5936.41
HTTP_File_Transfer	21036.135	4.6	14472.353	6563.782
HTTP_DownloadManager	13678.501	3	13108.437	570.064
Skype	19041.435	4.1	9052.676	9988.759
Flash Media	8758.3	1.9	8524.36	233.94
HTTP	9052.033	2	7681.611	1370.422
SSH	12884.821	2.8	5753.53	7131.291
iPlayer	4914.076	1.1	4642.528	271.548
iTunes	4890.349	1.1	4579.231	311.118
Facebook	4502.95	1	3866.792	636.158
SSL	7287.619	1.6	3552.22	3735.399
FTP-DATA	4662.748	1	3434.913	1227.834
PPStream	7643.39	1.7	2947.715	4695.676
Steam	2604.319	0.6	2541.917	62.402
RTMP	2640.165	0.6	2375.654	264.51
Other P2P	4473.183	1	2303.007	2170.176
YouTube-HD	1793.975	0.4	1754.968	39.007
BITS	1679.611	0.4	1640.144	39.466
MegaUpload	1439.513	0.3	1298.464	141.05
HTTPS	1170.22	0.3	990.882	179.338
BitTorrent Enc	2299.763	0.5	990.672	1309.091
Microsoft Live	1758.971	0.4	979.049	779.923
STUN	1801.377	0.4	865.712	935.665
IPSEC-ESP	8019.213	1.7	836.415	7182.799
HTTP_Audio	968.473	0.2	834.739	133.734
BitTorrent	977.689	0.2	496.192	481.497
SMB	3814.965	0.8	170.655	3644.309
All Others	211854.784	45.7	176674.118	35180.667
<b>TOTAL</b>	<b>460442.691</b>	<b>100</b>	<b>362975.21</b>	<b>97467.482</b>

Figure 6.9: This table lists the total amount of data transferred by application class between October 2010 and May 2011 for all users on campus.

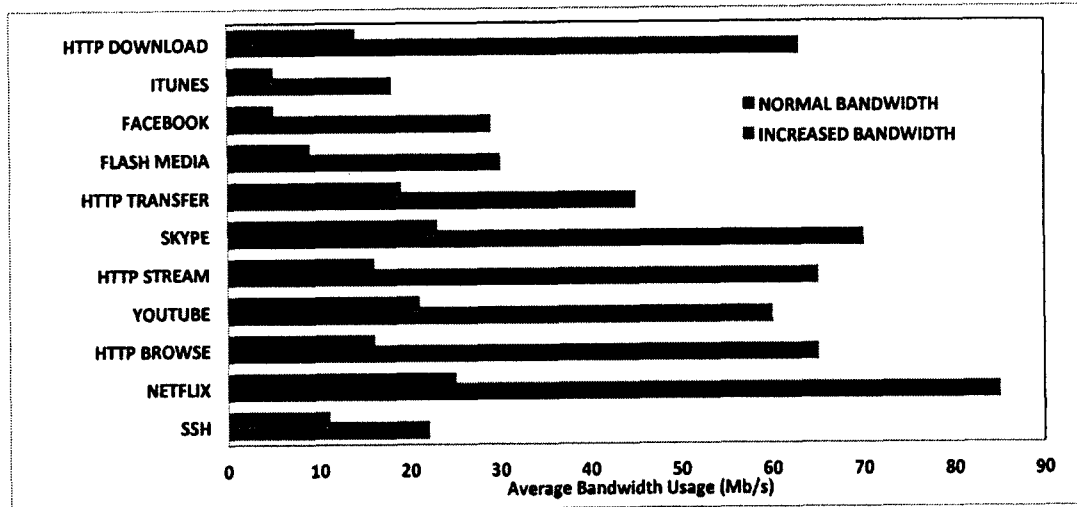


Figure 6.10: Changes in bandwidth usage for top applications when the students' bandwidth is increased.

width since there is decreased demand from the faculty/staff users during this time. When comparing the scale of bandwidth usage between Figure 6.6C and Figure 6.6D, I observed a significant increase in average bandwidth usage for each of the application classes. NetFlix uses an average of 3.5 Mbps during the daytime hours and 7.5 Mbps during the nighttime hours for a typical day. I found a similar increase for all other applications.

Figure 6.10 illustrates another example of increased bandwidth being quickly consumed by users. In this graph, I compare the average bandwidth usage of the most popular applications when the students' bandwidth is temporarily increased. As the graph shows, there is a significant increase in the bandwidth utilization for all applications when more bandwidth is given to students. Streaming multimedia increases by more than 200% for all users. Skype usage triples and Netflix quadruples in usage.

I also found that each semester overall user demand and bandwidth consumption increases. In Figure 6.2, I observed a significant difference in bandwidth utilization from the Spring 2010 semester to the Fall 2010 semester. The median receive rate increased by 41% and the median transmit rate increased by 132%. During the summer, network configurations were modified and users were given increased bandwidth

allotments. It is clear that this increased bandwidth was quickly consumed and utilized by the users at the start of the next semester. Several times during the Fall 2010 semester, campus data rates reached the maximum bandwidth available for all WAN connections. Comparing the Fall 2010 usage to the current Spring 2011 semester usage to date, I found that the median usage rates have already increased by 8% for receive and 9% for transmit. Even with user allotments remaining constant, there is increased demand for resources from users. In the past few years, there has been an explosion of devices registered on the network. This is especially true for WiFi connected devices. It is not uncommon for users to have multiple computers and devices concurrently connected and transferring data such laptops, mobile phones, iPods and iPads. During class, a professor and students could all be using computers to view online videos or demonstrations while concurrently utilizing mobile devices for messaging and personal multimedia.

In February 2011, Dartmouth College temporarily increased available bandwidth in order to improve their users' Internet experience(78). The college was experiencing increased demand for network resources, which resulted in poor performance for users throughout campus. The network administrators doubled the bandwidth for the college from 200 Mb/s to 400 MB/s for a two week trial period. The newly available bandwidth was quickly consumed by users and there were still performance problems. One class attempted to watch a 15 minute streaming video from a remote server and the entire video took over 45 minutes to watch, even with the increased bandwidth connection. Higher capacity connections are expected to be added over the course of the next two years, however demand is also expected to increase just as quickly.

Another example of increasing bandwidth to deal with high demand issues can be found at Ohio University (102). On their academic network, users were experiencing high levels of congestion, so much so that administrators actually fully restricted portions of their network during finals week. They identified that students utilize

over 70% of the campus bandwidth and Netflix was the “largest single consumer of Internet capacity”. The top three bandwidth consuming categories on their networks are: streaming media (60%), web browsing (25%) and file sharing (7%). In order to deal with high demand, the university implemented per user bandwidth limits and increased the total bandwidth capacity for the entire network by 10%. Even with these measures in place, administrators have noted that demand continues to exceed available capacity and they are repeatedly utilizing the maximum available bandwidth. University officials agree that there needs to be work done to “address the challenge of rising demand for Internet capacity” (87).

**Increasing bandwidth summary:** When bandwidth rates are increased for users, I found that any newly available bandwidth is quickly and easily consumed. Applications like Netflix dynamically adapt to changing bandwidth conditions. When more bandwidth is made available, the applications attempt to transfer larger amounts of data for higher quality output. Netflix will shift from standard to high-definition viewing if the appropriate amount of bandwidth is available. A standard definition movie requires about 1 GB of data for an hour of video (2.3 Mbps), whereas an HD movie requires almost 2.5 GB of data in an hour (5.7 Mbps) (48; 52). Other applications will also increase their transfer rates when given additional bandwidth (93; 106; 34; 104; 67; 103; 14).

## 6.5 Summary

The UNH campus network supports over 10,000 users and allows each user’s device to utilize up to 8 Mbps. During peak periods, the bandwidth limit per device decreases automatically based on demand. The entire campus network currently shares multiple connections to both the Internet and Internet2. Given this configuration, I examined the bandwidth and application usage for all users. The following points represent the

main findings of my workload characterization.

- Internet demand varies throughout the academic year, however each semester more and more bandwidth is consumed by users as demand grows.
- User demand and bandwidth usage is greatest between Sunday evening and Friday afternoon. On average, bandwidth usage reaches a high-load condition between noon and midnight each day.
- On a typical academic day, the campus network transfers 7 TB of data. The network has frequently transferred up to 10 TB during peak periods. The minimum amount of data transferred on a given day is 0.8 TB.
- The applications consuming the most bandwidth on an average day are streaming multimedia (Netflix, YouTube), web browsing and Skype.
- The top applications for student users are Netflix, streaming web videos and Skype. Faculty and staff users' workloads are dominated by web browsing and file transfers.
- During the daytime hours (6AM-6PM), Internet traffic is mostly web browsing, file transfers, SSH and streaming multimedia. At nighttime, Skype, Netflix, YouTube and other streaming multimedia take over as the applications demanding the most bandwidth.
- During a six month period, the top applications transferred tens of thousands of gigabytes of data. Netflix (25,000 GB), HTTP streaming (23,000 GB), YouTube (21,000 GB) and web browsing (16,000 GB) had the largest amounts of received data. Skype sends the most amount of data on a given day (10,000 GB) and receives roughly the same amount of data.

- When bandwidth is increased, users quickly utilize any new capacity made available to them and the data transfer rates for the top bandwidth consuming applications greatly increase.

At the time of this study, users were restricted to a combined bandwidth limit of 1.2 Gbps. Each year additional bandwidth is acquired by the university and the users' limits are increased. If the students' usage patterns remain the same and their bandwidth partition is increased to 2 Gbps, the campus network is estimated to transfer over 12 TB of data daily. Given that applications like Netflix dynamically adjust to available bandwidth and attempt to utilize as much as possible to achieve high definition viewing, I expect that the daily data consumption amount will be even higher.

Given the results of this study, I must next ascertain whether or not the campus network can accommodate big data transfers. With a better understanding of the campus network and its workload, I need to determine if it is feasible for the system to support big data transfers for all users on campus. The following chapter presents my feasibility study.

## CHAPTER VII

# Feasibility of big data transfers on the campus network

In the previous chapter I identified the architecture and workloads of the campus network. I found that the network is heavily utilized by thousands of users and its workload is dominated by congestion sensitive applications. Given the results of the campus network study, I must determine whether or not it is feasible for the network to support multiple users transferring big data. This chapter details the finds from my study (117).

Currently, only a small percentage of academic users, mainly researchers in disciplines like physics and biochemistry, need access to large data sets stored elsewhere. Since electronic transmission of large data sets is difficult, these researchers often transfer their data via hard disks transported by snail mail (1). In rare cases, fast links can be manually and temporarily set up between two locations for transfers of large data sets by working with network administrators. As large scientific and commercial data sets become available in a growing number of disciplines, a greater number of academic users will require access to these data files. Since these data sets are often multiple petabytes in size, researchers will often require subsets of the data with file sizes in the range of hundreds of gigabytes to several terabytes. Users routinely require and prefer local access to these files for processing and other tasks.

Even the output of remote computations in both grid and cloud environments can be in the same magnitude of file size. The movement of large private files between the cloud/grid and its clients is a commonplace occurrence. Therefore, in addition to specific research groups, individual users on campuses will require access to large files. Current trends in computing and the increasing sizes of files predicate the need for efficient techniques to transfer large files to and from campuses.

Big data transfers impose a much higher bandwidth burden than any other application. Users want to be able to retrieve/transmit large files quickly with a click of a mouse, without having to worry about errors and retransmissions. In order to move files quickly, the campus user must have fast links. For example, to transmit a 1 terabyte file over a 1 Gb link would take at least 2.3 hours, and over a 5 Mb link would take at least 19.4 days. However, increasing the bandwidth for large file transfer users would limit the available bandwidth for other users. Satisfying the performance requirements of large file transfers is important, but it should not inhibit the performance of other applications.

There is considerable research interest in techniques for large file transmission (18; 17; 49; 69; 86). The majority of existing research focuses on new network hardware and new network protocols for large file transfers (56; 61; 68; 77; 99). Purchasing new hardware for a particular application may not be cost effective. Before investing in new hardware for large file transfers, it is prudent to investigate whether the existing campus computer and network infrastructure can be parlayed to support large file transfers. An efficient solution must not only ensure that the performance requirements of users transmitting large files are satisfied but also that the addition of large file transfers does not impede other users and applications. The problem of adding a high-load application such as large file transfers to a shared network environment is not just a network issue. It is a systems issue and requires an understanding of the milieu in which these transfers occur. The users transmitting large files share the



campus network with myriad users running a variety of applications with different performance requirements. In order to satisfy the performance requirements of all network users/applications, it is necessary to understand the issues and challenges of incorporating large file transfers into the existing campus design.

## **7.1 Campus Network**

In order to accommodate large file transfers, system resources need to be able to handle the increased burden created by these workloads. After examining the infrastructure and configuration of our campus network, which has a similar structure to other campus networks, we find that it is capable of supporting large file movements without significant modifications to the infrastructure. The core of the campus network and the link to the WAN connections is 10 Gb, which would theoretically allow the transfer of a terabyte file in under 15 minutes. The links to the end user on campus could support a maximum of 1 Gb/s, which provides a theoretical time of 2.3 hours. If all of the connections in the data path are able to support these rates, then the storage systems will be the limiting factor of the transfer rate. The bandwidth controller that manages the interface between the LAN and WAN has a bird's eye view of all traffic passing through the border. Since it has complete knowledge of the workload present in the system, it could be utilized to schedule big data transfers and to allow these tasks increased bandwidth in order to complete quickly.

## **7.2 Traffic on the Campus Network**

I find that users' bandwidth demands are unbounded and users will utilize any bandwidth that is provided to them, especially during peak periods. The composition of user traffic is dominated by real-time applications, such as streaming multimedia, web browsing and VoIP, which are highly sensitive to changes in network latency and

congestion. I also find that there are varying levels of demand during different times of the day and on different days of the week. High demand is present during the week when students are actively connected to network, specifically between noon and midnight. When the campus network is under high load, big data transfers should not be placed in the system, as they will negatively impact other applications and will take longer than necessary. Between midnight and noon and on the weekends, the number of connected users is significantly lower and so is bandwidth demand. It is during these low usage periods that big data transfers should take place.

### **7.3 Impact of Big Data Transmissions**

From my experiments in the previous chapter, I find that it is possible to retrieve large data files over the campus network. I identify that these workloads impact system performance and cause congestion during peak periods. There is no benefit to any user by running these transfers during high load times. Campus users will experience delays and jitter in their time critical applications and the large file transfers will see decreased transfer rates and longer durations. If the transfers are restricted to only operate during low utilization periods however, then the performance impact on user workloads will be minimal and the large file transfers will find faster transfer rates and shorter service times.

### **7.4 Potential and Limitations**

During my system level feasibility study, I identify two key challenges that must be addressed before big data transmission can become commonplace on the campus network.

1. As Internet applications evolve and new services become available, the demand for bandwidth is expected to outpace the available bandwidth on several cam-

puses (78; 87). In order to ensure fairness for all users, the bandwidth allotted to individual users is limited. Big data transfers have the highest bandwidth requirements in comparison to other Internet applications utilized by users. With these restricted bandwidth allotments, big data transmissions would take several weeks to complete. On the other hand, allowing unrestricted bandwidth to large file transfers would greatly reduce the bandwidth available for other applications.

2. The majority of campus users are running real-time applications. Any loss of bandwidth or congestion can result in jitter and slowdowns for these applications. To a user staring at the “screen,” even a small delay can appear endless and frustrating. These services can therefore not be impacted by big data transmissions.

I conclude that big data transmissions should not be allowed free rein on campuses, but should be restricted to operate only during low demand periods. My feasibility study also identifies the advantages provided by the campus infrastructure with regard to incorporating big data transfers:

- The bandwidth controller placed at the border between the campus LAN and WAN manages all traffic moving in and out of campus. The controller has a complete view of the campus traffic conditions. Moreover, the controller manages the bandwidth given to each user at all times. Therefore, the bandwidth controller has the knowledge and the authority to control the bandwidth given to big data transmissions.
- My study show that while campus users place heavy load on the network, the load is not consistent during all times of the day. There are periods during each day when there is very low usage of the network. During these times, the network can be specifically employed for terabyte transfers.

- File transmissions are not time critical applications. Users do not want to deal with errors, timeouts and retransmissions, they just want to upload/download files with minimum problems.

I identify that it is feasible to support big data transmissions on the campus network by utilizing idle bandwidth available during low demand periods. Taking advantage of this free bandwidth for big data transmissions needs to be examined further. The following chapter presents our model for big data transmissions, which utilizes off-peak periods to transfer data.

## CHAPTER VIII

### Nice model for big data transfers

In this chapter, I present our “nice” algorithm for handling big data transmissions (109; 118). As discussed in the previous chapter, it is feasible to support these types of large file transfers on the campus network. In order to accommodate this additional workload, these transfers must occur during low demand periods and be allowed access to full bandwidth availability.

A new, nice algorithm for Big Data transfers, which is based on a store-and-forward model instead of an end-to-end approach, is presented. This nice algorithm ensures that Big Data transfers only occur during low demand periods when there is idle bandwidth that can be repurposed for these large transfers. Under this algorithm, Big Data are transmitted when the Internet traffic at the senders LAN is low. If the Internet traffic at the receivers LAN is high at this time, then the data are stored at a staging server and later transmitted to the receiver. Similar to the nice command in Linux, a transfer tool based on the nice algorithm, gives itself low priority and is nice to other applications using the Internet.

The overall goal is to develop an application that can transmit big data via the Internet for all users on campus. In order to develop the application, we first abstract the essential features of the hardware/software platform over which big transmissions execute.

## 8.1 Platform

**Hardware:** The tool transmits files between two campuses, so the hardware of significance is the transmission media, with the assumption that the sender/receiver computers are fast enough to handle the upload/download. From a modeling perspective, the hardware platform can be divided into 3 network zones: the LANs at the two campuses and the Internet connecting these LANs. The data transmission rate of a network depends on the maximum amount of data that can be transmitted per second. This rate is determined by the smallest bandwidth link in the route. The rest of the network hardware, such as routers and switches, can be abstracted out of the model. Therefore, each network zone can be represented by a single link, namely, the smallest link in the zone.

The smallest Internet link is usually greater than the smallest campus link. Moreover, there could be several alternate Internet paths from the sender's campus LAN to the receiver's campus LAN. Since the Internet has more bandwidth than the campus links, the transmission rate is determined by the campus links. Consequently, the Internet zone can be abstracted out of the hardware model, without impacting the performance of the transfer. The hardware platform reduces to 2 links, the smallest sender link and the smallest receiver link. Define two variables, `SendBW` and `RecBW`, to represent the smallest sender link bandwidth and the smallest receiver link bandwidth, respectively.

**Workload:** Big data transmissions have to share the network with other time critical applications. From a modeling perspective, the impact of this other workload on the network can be incorporated by reducing the amount of bandwidth available to big transmissions. Therefore, even if `SendBW` is the smallest physical bandwidth, the bandwidth available to big transmissions may be smaller. From analyzing the campus network traffic, there are periods during the day when the network is heavily used,

Time	Utilization (%)	SendBW/RecvBW
12PM-12AM	95%	15 Mbps
12AM-1AM	50%	500 Mbps
1AM-2AM	25%	750 Mbps
2AM-10AM	< 10%	1 Gbps
10AM-11AM	25%	750 Mbps
11AM-12PM	50%	500 Mbps

Figure 8.1: Network utilization and bandwidth availability for each hour of a typical day.

while there are other periods when the network is largely idle. Therefore, the amount of bandwidth available for big transmissions varies depending on the time of the day.

Figure 6.3 shows that the traffic pattern is largely stable from day-to-day and varies hourly. For simplicity, the model assumes that the weekend traffic pattern is similar to the weekday traffic pattern. In order to incorporate the impact of work-load traffic, the notations SendBW and RecBW have to be modified to represent the maximum available bandwidth (Mb) during each hour of the day:

Arrays SendBW[i], RecBW[i] where  $i$  represents the hour of the day  $0 \leq i \leq 23$  where hour 0 is 12 AM, hour 1 is 1 AM,..., hour 23 is 11 PM; SendBW[i], RecBW[i] represent the smallest available bandwidth (in Mb) at the sender's/receiver's campus during hour  $i$ .

Figure 8.1 shows bandwidth availability SendBW[i], RecBW[i] used in our modeling. At 4pm, the maximum bandwidth is 15 Mb/s so SendBW[16] = 15. During non peak hour, say at 2am, the available bandwidth is 1 Gb/s, so SendBW[2] = 1024.

The sender and receiver campuses may be on different time zones. The variable TimeDiff represents the number of hours by which the receiver campus is ahead or behind the sender's campus.

TimeDiff =  $j$  where  $j \in \{\dots, -3, -2, -1, 0, 1, 2, \dots\}$

For example, if the sender is situated in California and the receiver is situated in

Japan, then TimeDiff= 15.

**Tool:** The user of a file transmission tool either wants to transmit a file or receive a file. The tool inputs of relevance are the file's size and the time that the user initiates the transmission. Let the file size be represented in MB.

FileSize =x MB where  $x \in \{1, 2, \dots\}$

Let InitiateTime represent the hour at which the user initiates the transmission.

InitiateTime = j where  $j \in \{0, 1, 2, \dots, 22, 23\}$ : the hour (time) at which the user initiates the transfer.

### 8.1.1 Performance metrics

The following performance metrics are of interest:

#### 1. Transmission Time

sendTT, recTT  $\in \{1, 2, \dots\}$ : number of hours that sender and receiver's LANs, respectively, are busy transmitting the file.

TT  $\in \{1, 2, \dots\}$ : maximum number of hours that the network is utilized during the transmission of the file's data.

Suppose a file is transmitted from sender to an intermediate server in 2 hours; the file is then transmitted to the receiver in 1 hour. In this case, sendTT= 2, recTT= 1, TT= 2, the maximum of the two transmission times.

#### 2. Wait Time

sendWT  $\in \{0, 1, 2, \dots\}$ : number of idle (no transmission) hours between InitiateTime and completion of transmission from the sender's LAN.

recWT  $\in \{0, 1, 2, \dots\}$ : number of idle (no transmission) hours at the receiver between InitiateTime and completion of transmission (arrival of file) at the receiver's computer.



### 3. Response Time

$RT \in \{1, 2, \dots\}$ : number of hours between `InitiateTime` and the completion of the transmission.

Note that the time unit is an hour - transmissions start on the hour and end on the hour; if a transmission completes in less than an hour, the transmission time is rounded up to an hour.

## 8.2 Nice model

In order to avail of maximum installed bandwidth without impacting other users, the key is to open multiple transmission streams during low traffic. If the sender and receiver are in the same time zone then a direct transmission from sender to receiver is feasible. If the sender and receiver are in different time zones, then the low traffic periods at the two end points do not coincide. In this instance, the file is transmitted from the sender to one or more staging server(s), placed in the Internet zone. Depending on the Internet configuration between the sender and receiver, the file may be transmitted to a single staging server via multiple streams or the file may be divided and parts of the file are transmitted concurrently to multiple staging servers. When the receiver's LAN traffic is low, the file can be transmitted from the staging server(s) to the receiver. A file transmission tool such as GridFTP could be used for the transmission from the sender to the staging server(s) and from the staging server(s) to the receiver. Figure 8.2 represents the nice model. A transmission tool based on the nice model is parallel, store-and-forward.

It is assumed that the sender and receiver LAN traffic pattern is similar. This is a reasonable assumption since the traffic pattern is modeled after human behavior. Using Figure 8.1 as the basis, the high traffic period starts at 9:00AM and ends at midnight 12:00AM. `HighTrafficHours` = { 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,

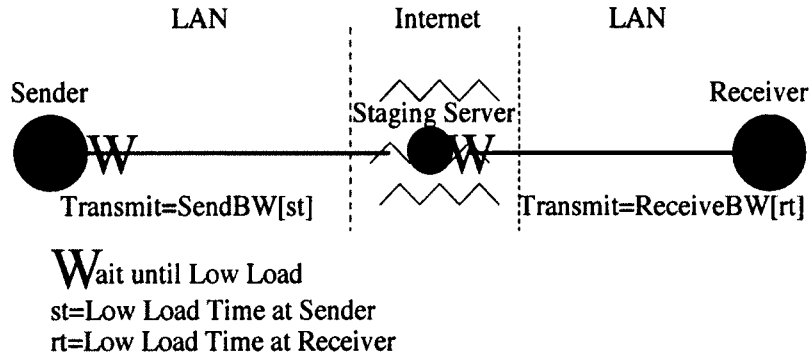


Figure 8.2: Nice model

22, 23 }

Note that while the traffic pattern is similar, the installed bandwidth at the sender and receiver campuses may be different. For example, during low traffic hours 0-9, the bandwidth at the sender's LAN may be 5Gb/s while the bandwidth at the receiver's LAN is 1Gb/s.

The user initiates the transmission; if it is high traffic at the sender's LAN, then the transmission does not begin until low traffic. At low traffic, the file transmission starts from the sender to the staging server at rate  $\text{SendBW}$ . If it is currently high traffic at the receiver, the transmitted portions of the file remain at the staging server. When low traffic period starts at the receiver's campus, then transmission proceeds to the receiver.

In order to focus on the essential aspects of nice's design, we do not explicitly code parallelism, but we capture the impact of parallelism by using the maximum available bandwidth. If the sender and receiver are in the same time zone, the staging server can be used to absorb the bandwidth differential between sender and receiver. For example, if the sender has a bandwidth of 5 Gb/s and the receiver has a bandwidth of 1 Gb/s, the sender can transmit at the higher rate using the staging server as a buffer.

The nice algorithm computes performance metrics for a transmission tool based

on the nice model. To keep code simple, we have set  $\text{SendBW}[i] = \text{RecBW}[i] = 0$  when  $i \in \text{HighTrafficHours}$ . This initialization does not impact performance metrics since a transmission tool based on the nice model does not transmit during high traffic hours. However, if the remaining portion of a file is small enough (equivalent to standard files), the code can be modified to allow transmission of this remaining portion of the file during the high traffic time. The variables  $\text{SendRemain}$ ,  $\text{StageFile}$ ,  $\text{RecFile}$  represent the file sizes at the sender, staging server, and receiver. The variables  $\text{SendTransmit}$  and  $\text{RecTransmit}$  represent the amount of data transmitted at the sender and receiver during the current hour. In line 4, the function *TimeZone* computes the time at the receiver given the time at the sender and the difference in time zones between sender and receiver.

### 8.3 Parallel model

Current large file transmission tools such as GridFTP and BitTorrent are designed on the parallel model. Both protocols allow the downloading of parts of the file from different locations - one receiver, several senders. We are interested in how these protocols transmit big data between two locations - one receiver, one sender. Since bandwidth is critical to transfer time, multiple concurrent TCP streams are opened between the two locations, and parts of the file are transmitted concurrently. Opening multiple streams potentially distributes Internet load if a different network route is used for each stream. However, the multiple streams converge at both end point LANs, thereby straining the capacity of congested campus LANs.

Figure 8.3 depicts the parallel model for file transmissions. The transmission model is end-to-end with streams of transmission from sender to receiver. Thus, the rate of transmission is determined by the minimum bandwidth along the path. The parallel algorithm computes performance metrics for the parallel transmission tool. The transmission begins at  $\text{InitiateTime}$  so  $\text{sendWT}$  and  $\text{recWT}$  are 0.

---

**Algorithm 1: NICE ALGORITHM**

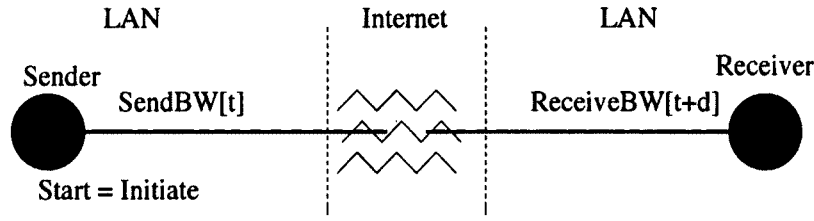
---

```
Initialize sendTT, recTT, StageFile, RecFile, RT, recWT to 0;
FileNotTransferred= TRUE;
SendRemain= FileSize;
i = InitiateTime;
j = TimeZone(InitiateTime, TimeDiff);
while FileNotTransferred do
    SendTransmit= SendBW[i] * 60 * 60;
    if SendRemain > 0 && SendTransmit > 0 then
        sendTT++;
        if SendTransmit > SendRemain then
            SendTransmit= SendRemain;
        SendRemain= SendRemain- SendTransmit;
        StageFile= StageFile+ SendTransmit;
    RecTransmit= RecBW[j] * 60 * 60;
    if StageFile > 0 && RecTransmit > 0 then
        recTT++;
        if RecTransmit > StageFile then
            RecTransmit= StageFile;
        StageFile= StageFile- RecTransmit;
        RecFile= RecFile+ RecTransmit;
    if (SendRemain > 0 && SendTransmit == 0) || (RecTransmit == 0) then
        recWT++;
    RT++;
    if RecFile >= FileSize then
        FileNotTransferred= FALSE;
    i = (i+1) MOD 24;
    j = (j+1) MOD 24;
TT= MAXIMUM(sendTT, recTT);
print RT, TT;
```

---

## 8.4 Related work: Data transfers over the Internet

Data transfers between users on shared networks, like campus networks, will utilize a data path similar to the following. The transfer initiates on a node on the campus network, which is generally a well-managed and heavily utilized system depending on the size of the university. It can be considered its own autonomous system (AS), which is a network that is administrated independently (90). The campus network connects to the Internet via one or more ISPs that provide dedicated bandwidth availability.



$d = \text{Time Difference}$   
 $\text{TransmitBW} = \text{minimum}\{\text{SendBW}[t], \text{ReceiverBW}[t+d]\}$

Figure 8.3: Parallel model

---

**Algorithm 2: PARALLEL ALGORITHM**

---

```

RemainingFile = FileSize;
CompleteTime = 0;
i = InitiateTime;
while RemainingFile > 0 do
    j = TimeZone(i, TimeDiff);
    CurrentBW = MIN(SendBW[i], RecBW[j]);
    TransmittedFile = CurrentBW * 60 * 60;
    RemainingFile = RemainingFile - TransmittedFile;
    CompleteTime = CompleteTime + 1;
    i = (i + 1) MOD 24;
RT = CompleteTime;
TT = CompleteTime;

```

---

Many large universities have several Internet connections and can be considered to be multi-homed, in that it has the access to the Internet via different service providers and different physical links (35; 122). After leaving the campus network, data pass through the server providers' AS and reach a peering point where their networks connect with other service providers and telecommunication companies' ASes. The data transfer operates as transit traffic on backbone connections towards the destination. This middle portion of the data path can be referred to as the transit networks. These high-speed, high-capacity links are engineered and managed for high efficiency and availability. Eventually the data will reach the service provider for the receiver's campus network. The data are then forwarded through recipient's network to the final destination. The transit network portion of the data path is the most abstract

from the end user. The specifics of the routing, such as next-best-hop heuristics (92), and bandwidth availability are hidden from users. Network providers also use traffic engineering to determine efficient routing and to satisfy economical objectives (126). It is therefore difficult to identify the ability of these networks to move large amounts of data without access to proprietary network information.

The ability to transfer large amounts of delay tolerant data through transit networks on the Internet is examined in (31; 63; 64; 75). These studies have unfettered access to large ISP/transit networks' configurations and actual workloads. They find that there is ample bandwidth available in the major transit networks of the Internet to move large amounts of data without incurring additional cost or overhead for telecommunication companies. They find that the Internet is not the bottleneck in large data transfers between end users, since there is enough capacity to move massive data sets during off-peak hours. Internet transit networks exhibit a diurnal pattern where load peaks between noon and midnight and then shows a dramatic decrease until the following afternoon (60; 105). The studies also find that under varying pricing schemes used by service providers (32; 43; 47; 89; 108) they were able to transfer data at no cost or at a minimal expense, especially in comparison to physically shipping data at regular intervals. For some users with very limited bandwidth, utilizing postal mail or courier services still remain the best option (123).

In order to gain access to larger amounts of bandwidth for large data transfers, a consumer could purchase dedicated network connections. Backbone optical networks can be provisioned for a customer's private connection, however the process can take several weeks and be very costly (74). For most users dedicated communication lines are not a viable option, so users must make the best of their existing connections. There are several technologies that allow users to customize their data paths to maximize throughput. Overlay networks are one such technology, where users have the ability to specify their own route through the Internet and utilize faster/less con-

gested links (15; 20; 83; 99). Using smart algorithms specific or dynamic pathways through multi-hop networks can be devised (55). A user connected to a multi-homed network has the ability to select from a set of network pathways and utilize them to varying degrees in order to gain access to larger bandwidth links (43).

An emerging network technology, OpenFlow, is a perfect fit for transferring large amounts of data through local networks and transit ISPs. OpenFlow is based on software-defined networking where the individual network components are programmable entities that a high-level management application can control in order to optimize traffic flows to take the shortest path and to optimize the network to maximize link utilization (70). Typical wide area networks may only have a 30 % utilization on average, since administrators must save bandwidth for bursting periods. Using OpenFlow it is possible to repurpose this idle bandwidth for bulk data transfers and thereby increasing the overall utilization of network connections. Since the OpenFlow management software controls all aspects of the network, it is able to successfully operate at 90-95 % utilization, which is something that large companies like Google is already doing today (50). The OpenFlow technology removes the burden of routing calculations from the individual routers in the network and eliminates duplicated work by having a centralized “route compiler in the sky”, RCITS (90). Using this methodology, OpenFlow enables administrators to program the network for different optimizations on a per-flow basis, which means that latency-sensitive traffic can take the fastest path and bulk flows can take the cheapest (70). As the technology is further developed, it may become the ideal candidate for moving large amounts of data through transit ISPs.

## 8.5 Summary

Current big data transmission tools are based on the parallel model where the goal is to access a major share of available bandwidth. The parallel model is short

sighted in that the design does not look beyond the tool's requirements to the impact on other users of the Internet. We have developed a nice model that utilizes available bandwidth without impacting other Internet applications. The nice model goes beyond the parallel model in that it incorporates both the requirements of big data transfers and the architecture and public accessibility of the Internet.

In the next chapter, I experimentally evaluate the nice and parallel models. I show that the nice model is superior to the parallel model for big data transfers across the globe.



## CHAPTER IX

### Evaluating the nice model for big data transfers

In this chapter, I evaluate the performances of the nice and parallel models using the OPNET network simulator and a simulator that I developed. The OPNET simulator is a well-known, commercial simulator capable of simulating a wide variety of network components and workloads (73; 88). Figure 9.1 represents the setup for our experiments. Due to the constraints of the simulator, which only allows file transfers up to 50000000 bytes, the bandwidth of the shared Internet link is set to a maximum of 1.5 Mb/s. The results of the simulations are scaled appropriately so that the transmission rate is 1 Gb/s and the file size is 1 TB. As presented in the previous chapter, the time unit used in my evaluations is an hour - transmissions start on the hour and end on the hour; if a transmission completes in less than an hour, the transmission time is rounded up to an hour.

Three client/server machine pairs are setup to emulate the workload of the most popular traffic classes found on the campus network: streaming video, web browsing and VoIP. In the simulations, the workloads of the three popular traffic classes are varied to represent the background utilization of the shared Internet connection at various times of the day. The fourth server is used to simulate big data transfers. Another server handles staging for big transmissions. The following sections present the main results of my experimental evaluation.

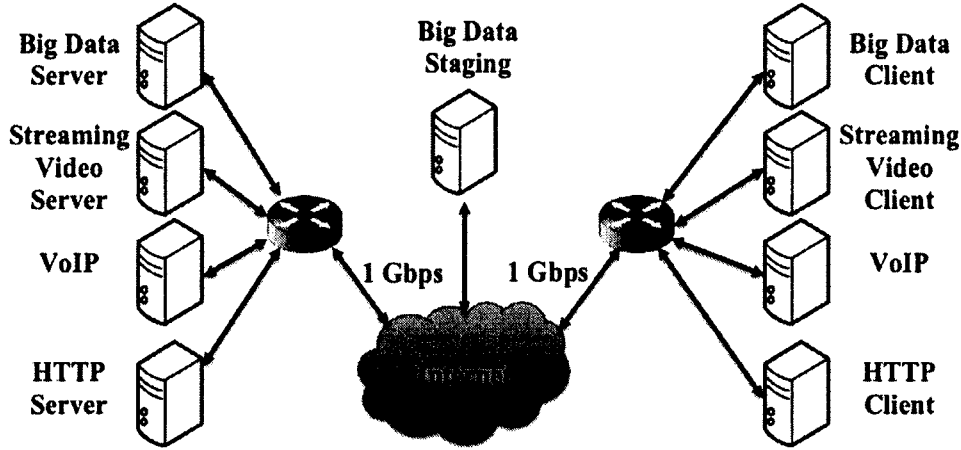


Figure 9.1: Simulator configuration map: The left side of the map represents that sending campus network and its client/server machines. The right side of the map represents the receiving campus network and its client/server machines. The staging server in the middle of the map is utilized when the sender/receiver networks have non-synchronous low demand periods.

## 9.1 Transmission time (TT as TimeDiff and InitiateTime varies):

Figure 9.2 plots the transmission time (TT) and response time (RT), as the request submission time (InitiateTime) varies during a 24 hour period starting at 8AM. In the top graph, the sender and receiver are in the same time zone (TimeDiff = 0). In the bottom graph, there is a 12 hour time zone difference (TimeDiff = 12). The transmission time (TT) of the nice model is invariant of the request submission time (InitiateTime) and the time zone difference (TimeDiff). For the parallel model, the response and transmission times (TT) are sensitive to both InitiateTime and TimeDiff. The response time (RT) of the nice model varies with the request submission time (InitiateTime) due to the necessary waiting times for the low demand period to commence.

Figures 9.3, 9.4, 9.5, 9.6 plot the transmission time (TT) in hours, as the request submission time (InitiateTime) varies. Each graph displays a different time zone variance (TimeDiff). The graphs shows that the transmission times for parallel and nice are closest in performance when the start time (InitiateTime) is in a low traffic period

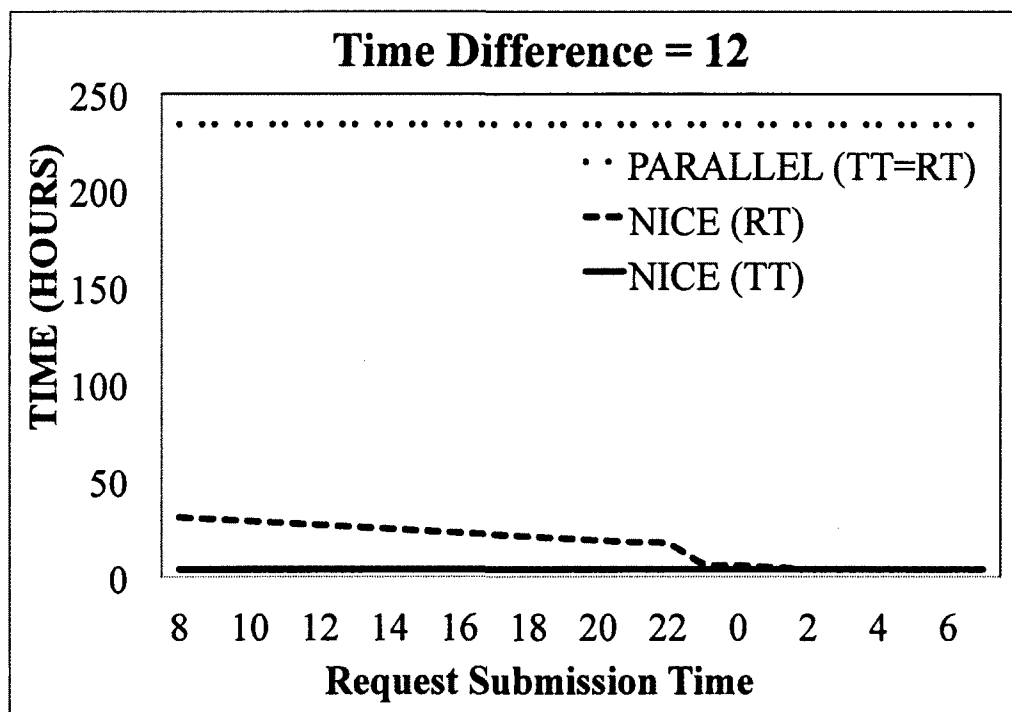
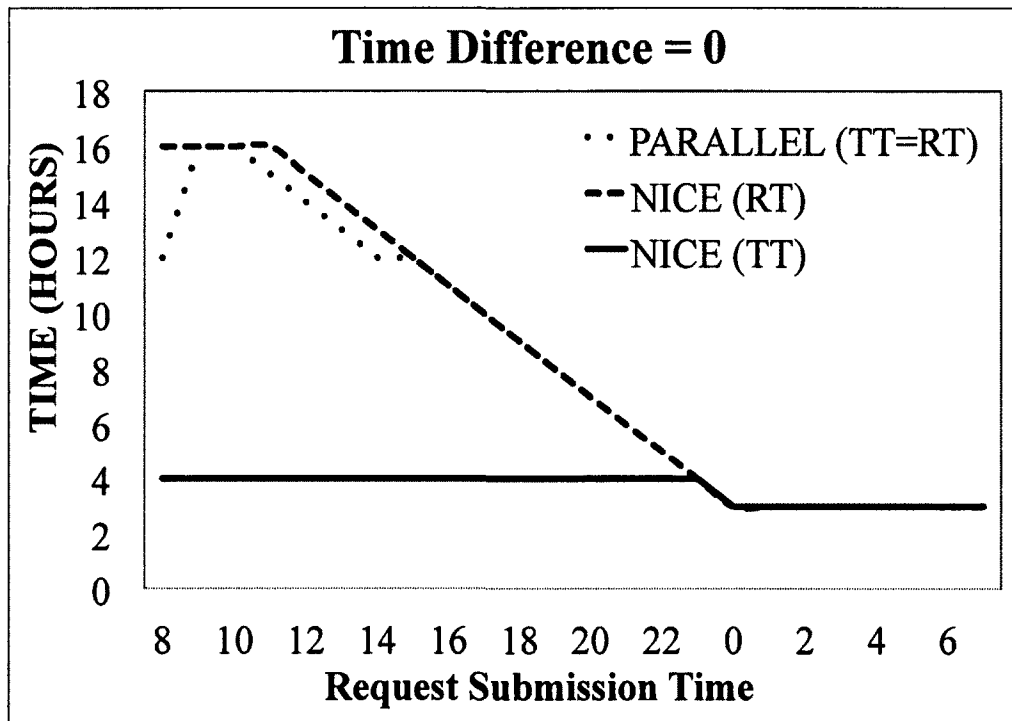


Figure 9.2: Transmitting a 1 TB data set when TimeDiff = 0 (top graph) and TimeDiff = 12 (bottom graph).

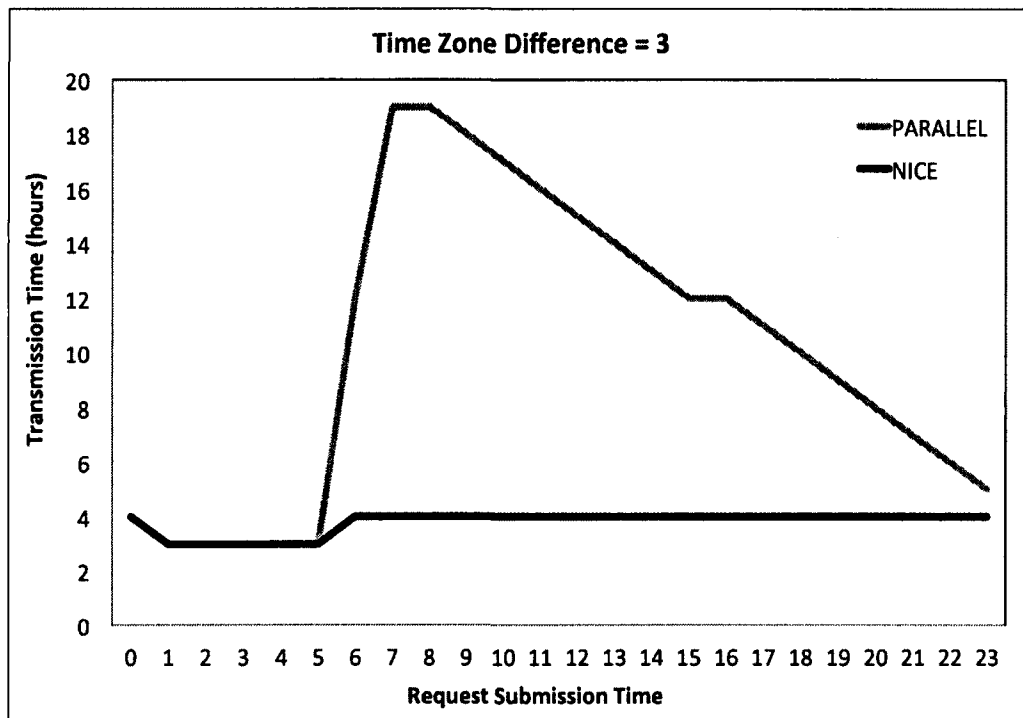
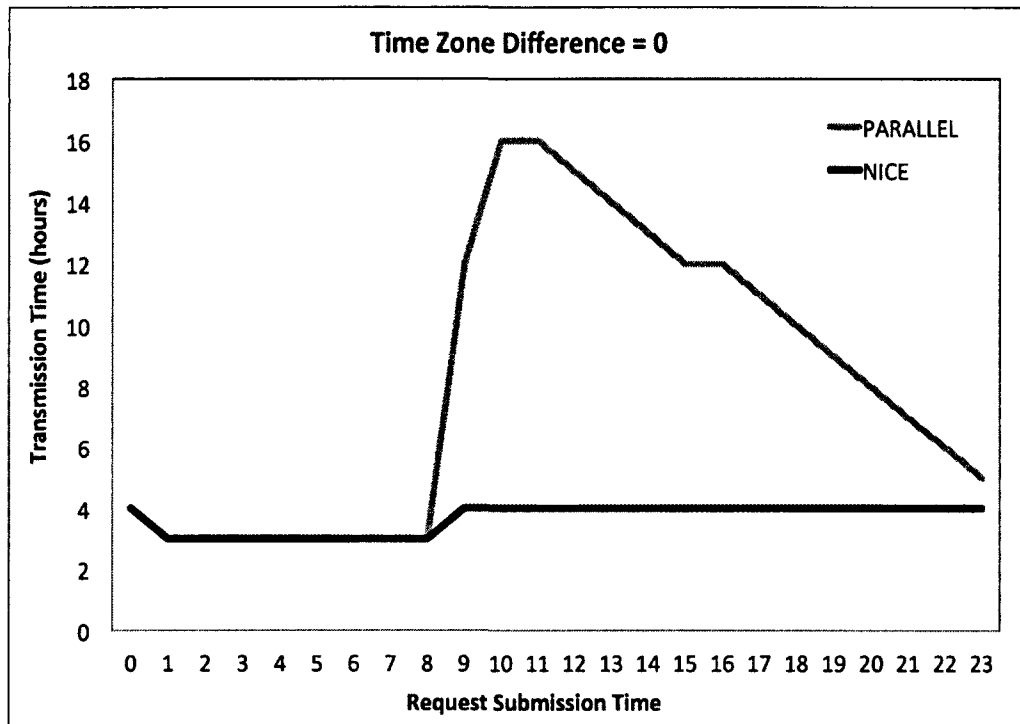


Figure 9.3: Transmitting a 1 TB data set when TimeDiff = 0 (top graph) and TimeDiff = 3 (bottom graph).

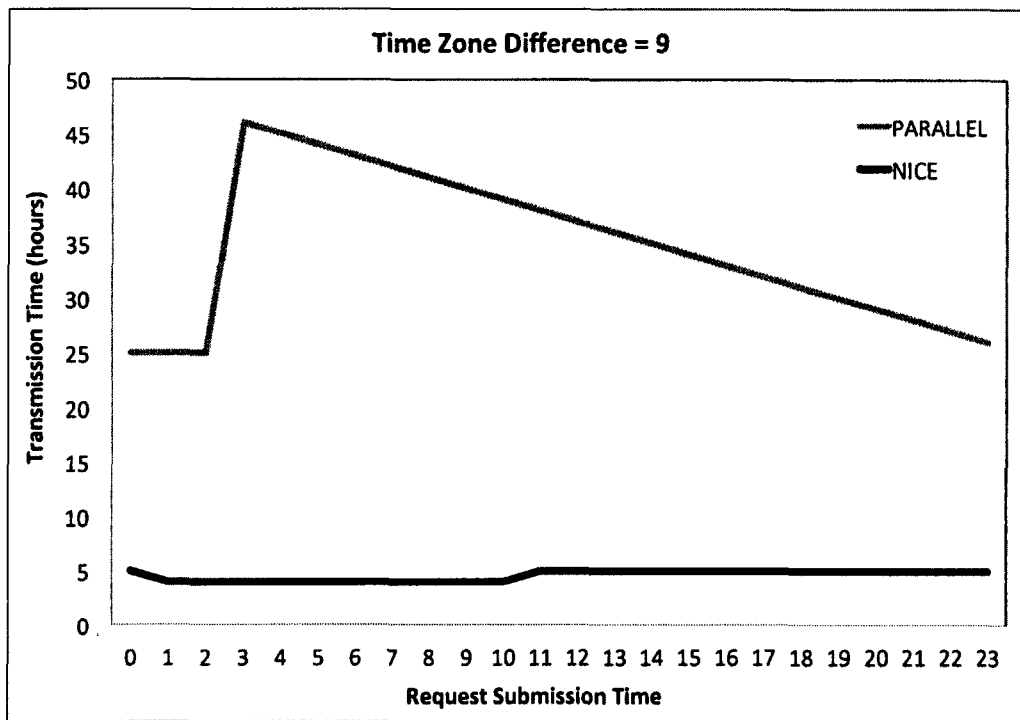
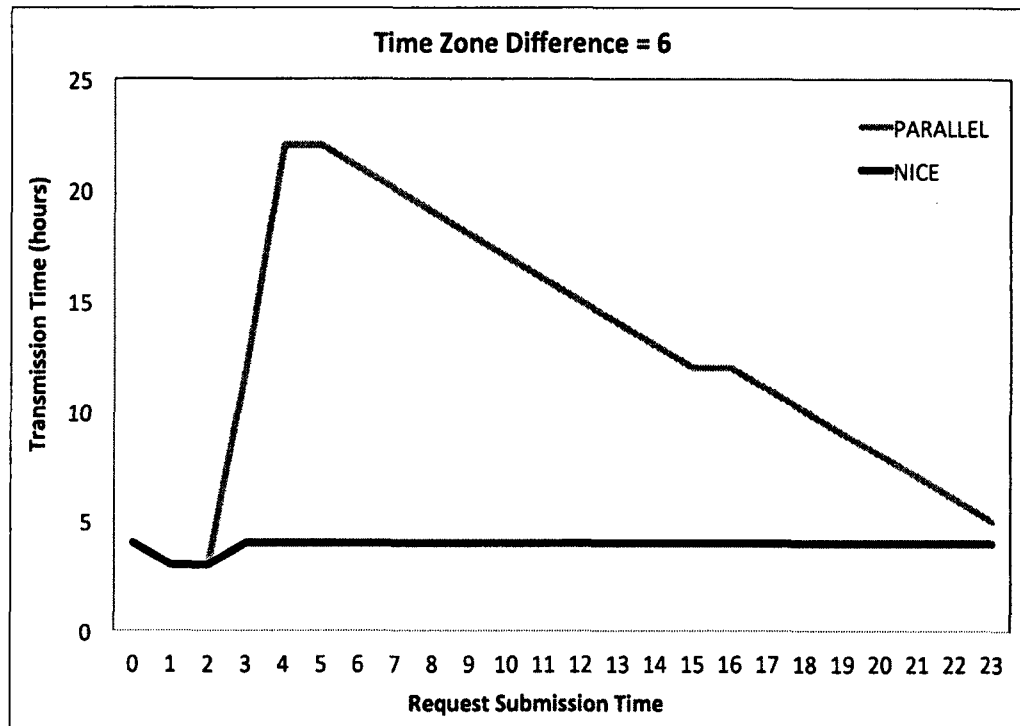


Figure 9.4: Transmitting a 1 TB data set when TimeDiff = 6 (top graph) and TimeDiff = 9 (bottom graph).

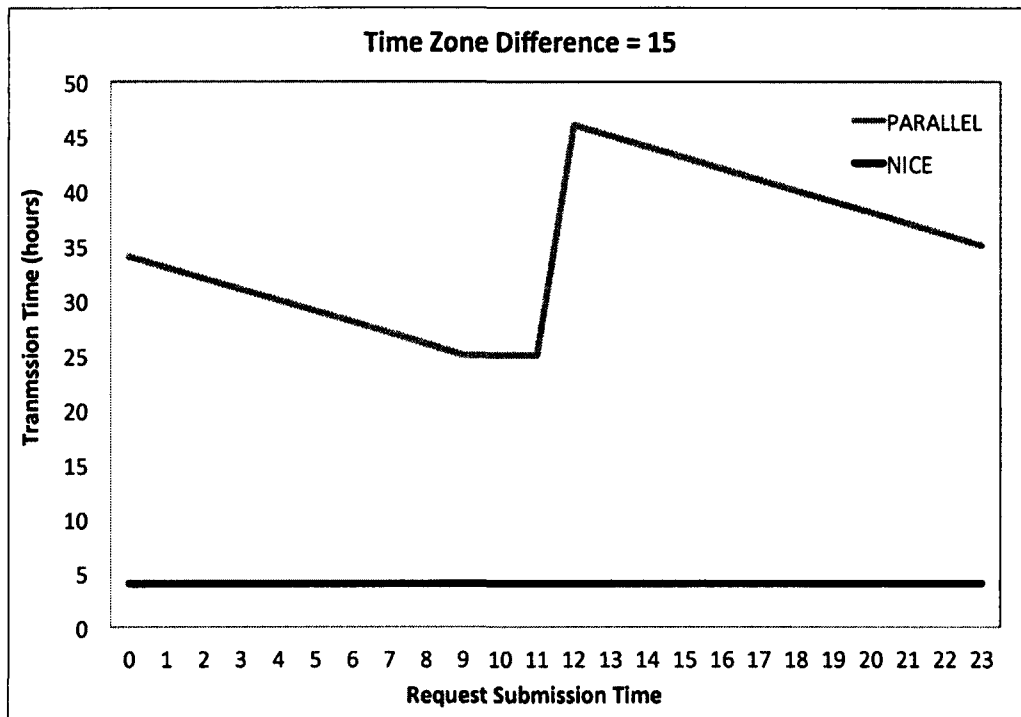
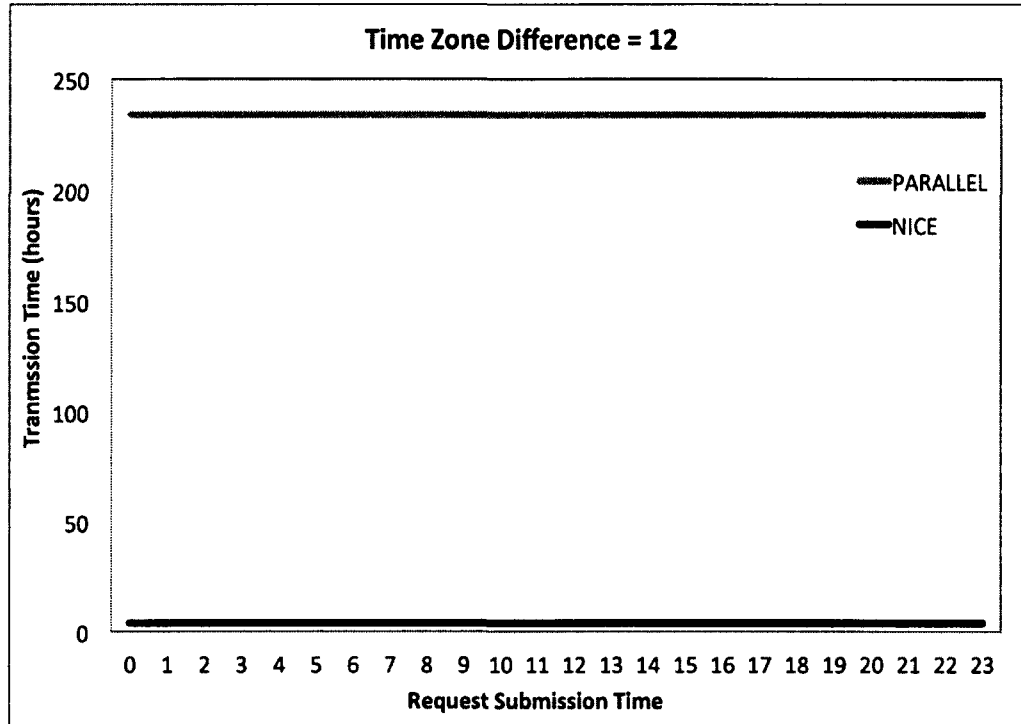


Figure 9.5: Transmitting a 1 TB data set when TimeDiff = 12 (top graph) and TimeDiff = 15 (bottom graph).

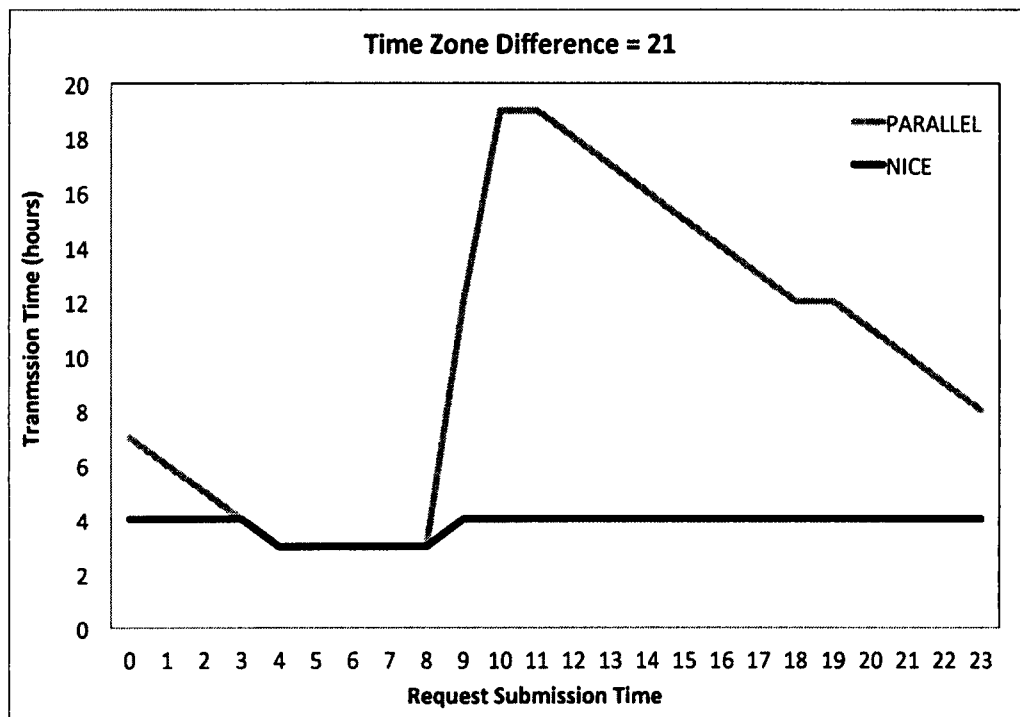
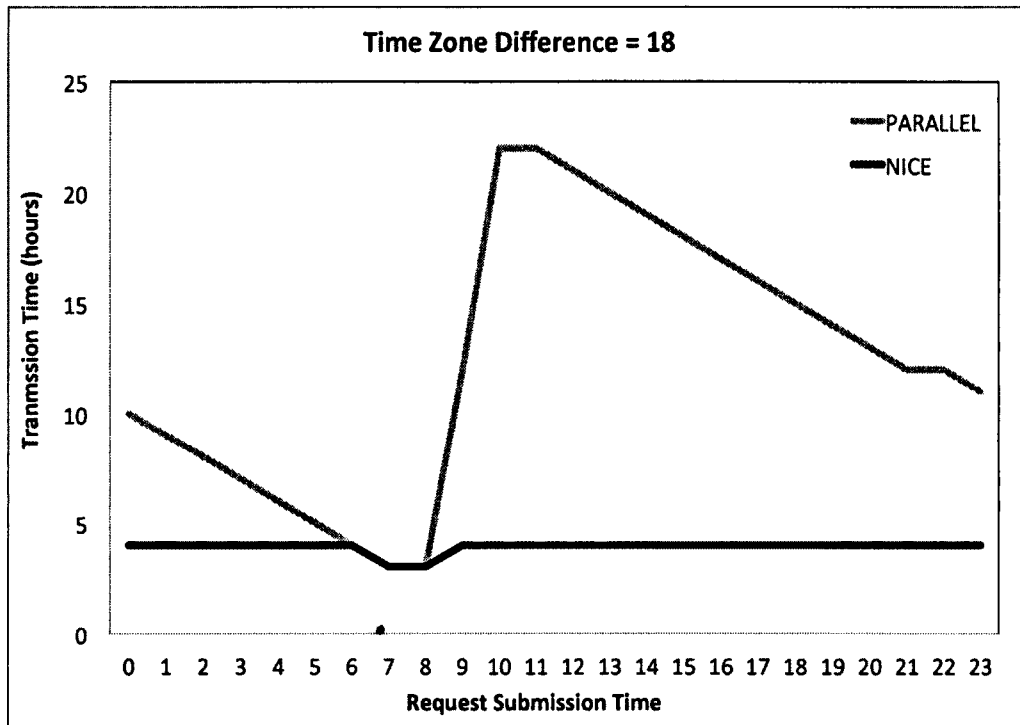


Figure 9.6: Transmitting a 1 TB data set when TimeDiff = 18 (top graph) and TimeDiff = 21 (bottom graph).

and the time zone difference (TimeDiff) is small. Nice significantly outperforms parallel when the time zone difference is 12 hours, which indicates that the sender and receiver have completely opposite low and high demand periods.

A comparison of the transmission times between the parallel and nice models for varying time zone differences is shown in the next group of graphs. Figures 9.7 and 9.8 illustrate the variations in transmission times for the parallel model under all time zone differences. Figure 9.9 shows the variations in transmission times for the nice model under all time zone differences. The parallel model experiences vast fluctuations in transmission time for all time zone differences, whereas the nice model's transmission times only varies by one or two hours. The nice model significantly reduces the transmission time for transferring the 1 GB data set when the sender and receiver have the greatest variations in low demand periods.

The percentage improvement in transmission time when the nice model is used instead of the parallel model is shown in Figure 9.10. As the time zone difference (TimeDiff) increases, the performance of the parallel model degrades, while the performance of nice remains unchanged. There is a near 100% improvement in transmission time (TT) of nice when the time zone difference is 12 (TimeDiff = 12). Figure 9.11 reconfirms the percentage improvement by graphing the reduction in transmission times afforded by the nice model in comparison to the parallel. Again, the nice model provides the greatest reduction in transfer time (over 240 hours) when the time zone difference is 12.

The improvement offered by the nice model can also be illustrated in figures 9.12, 9.13, 9.14, and 9.15. These graphs plot transmission time (TT) as time zone difference (TimeDiff) changes for various request submission times. When TimeDiff = 12, the sender and receiver are in orthogonal time zones (*i.e.*, there is no overlap of low traffic times at the sender and receiver), and the parallel model always transmits over small bandwidth.



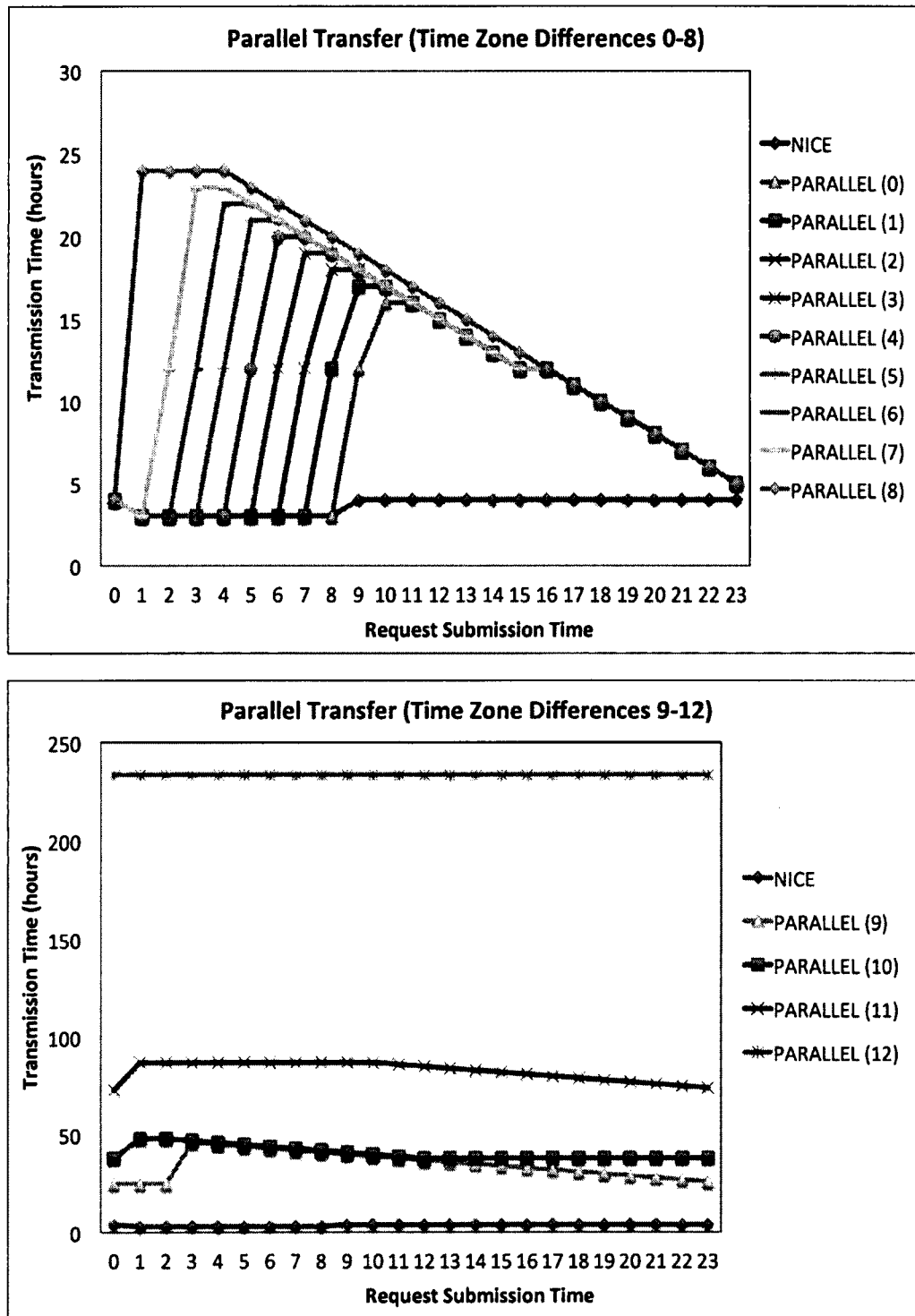


Figure 9.7: Transmission time comparison for transmitting a 1 TB data set using the parallel model when the time zone difference is between 0-8 hours (top graph) and 9-12 hours (bottom graph).

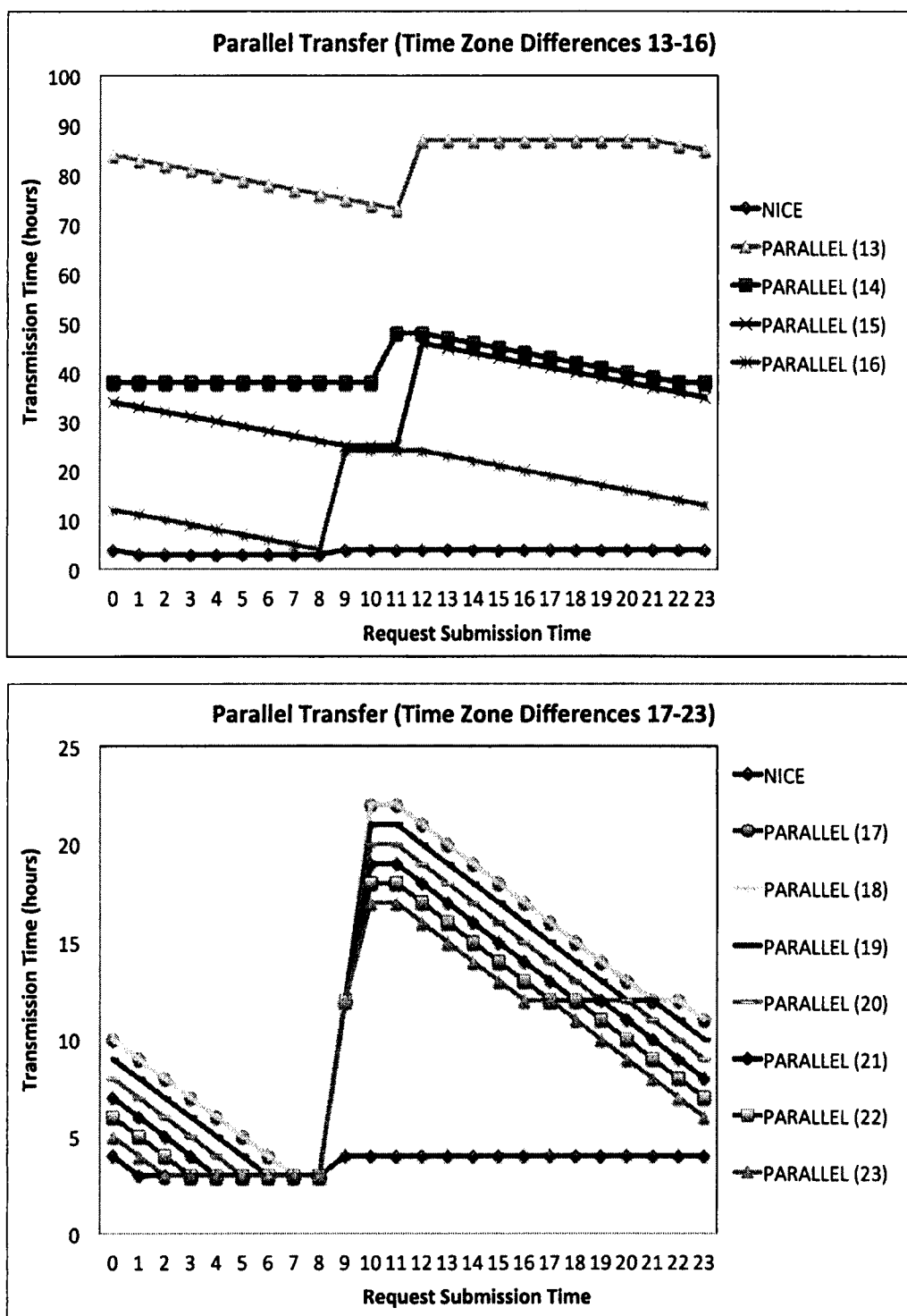


Figure 9.8: Transmission time comparison for transmitting a 1 TB data set using the parallel model when the time zone difference is between 13-16 hours (top graph) and 17-23 hours (bottom graph).

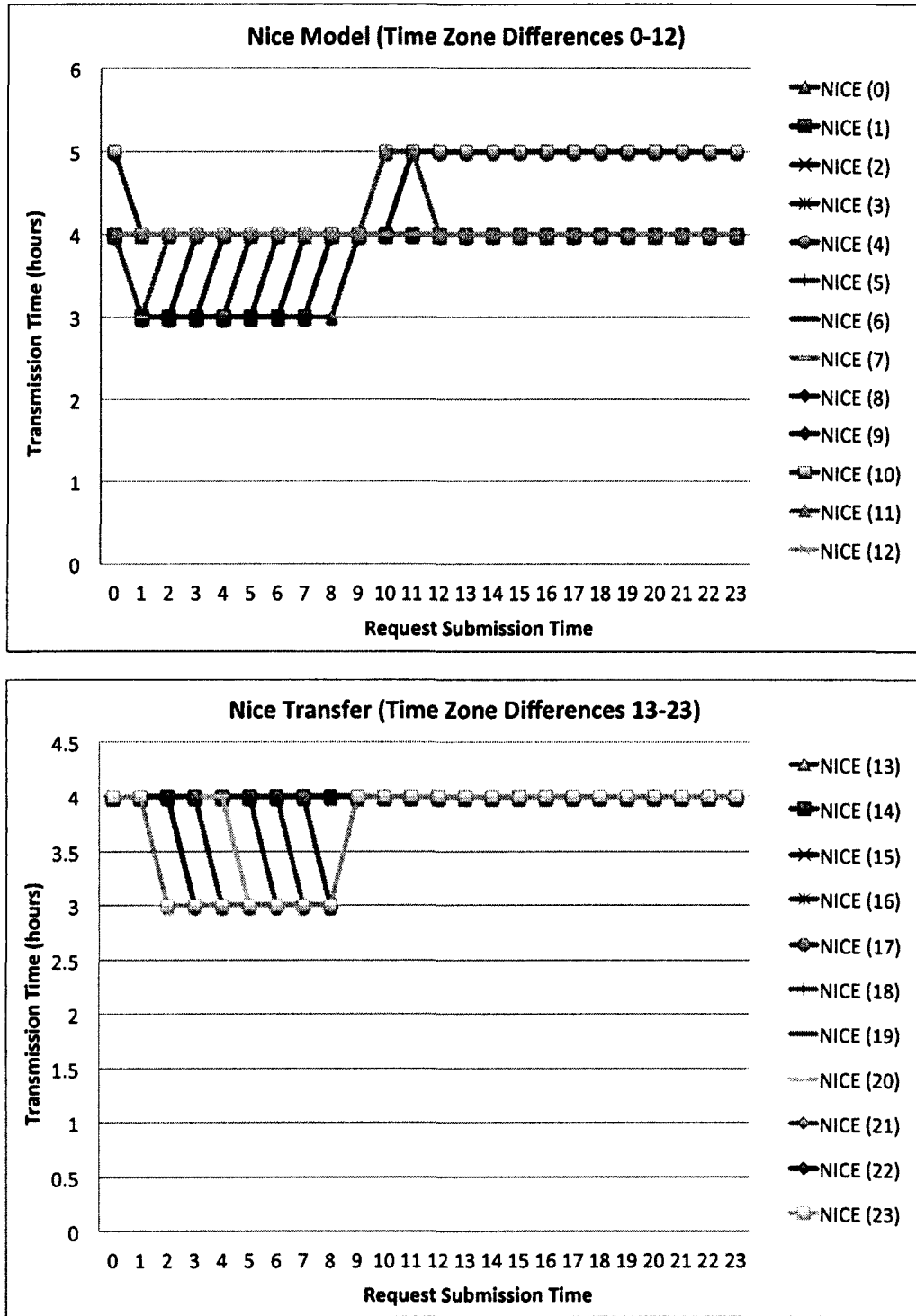


Figure 9.9: Transmission time comparison for transmitting a 1 TB data set using the nice model when the time zone difference is between 0-12 hours (top graph) and 13-23 hours (bottom graph).

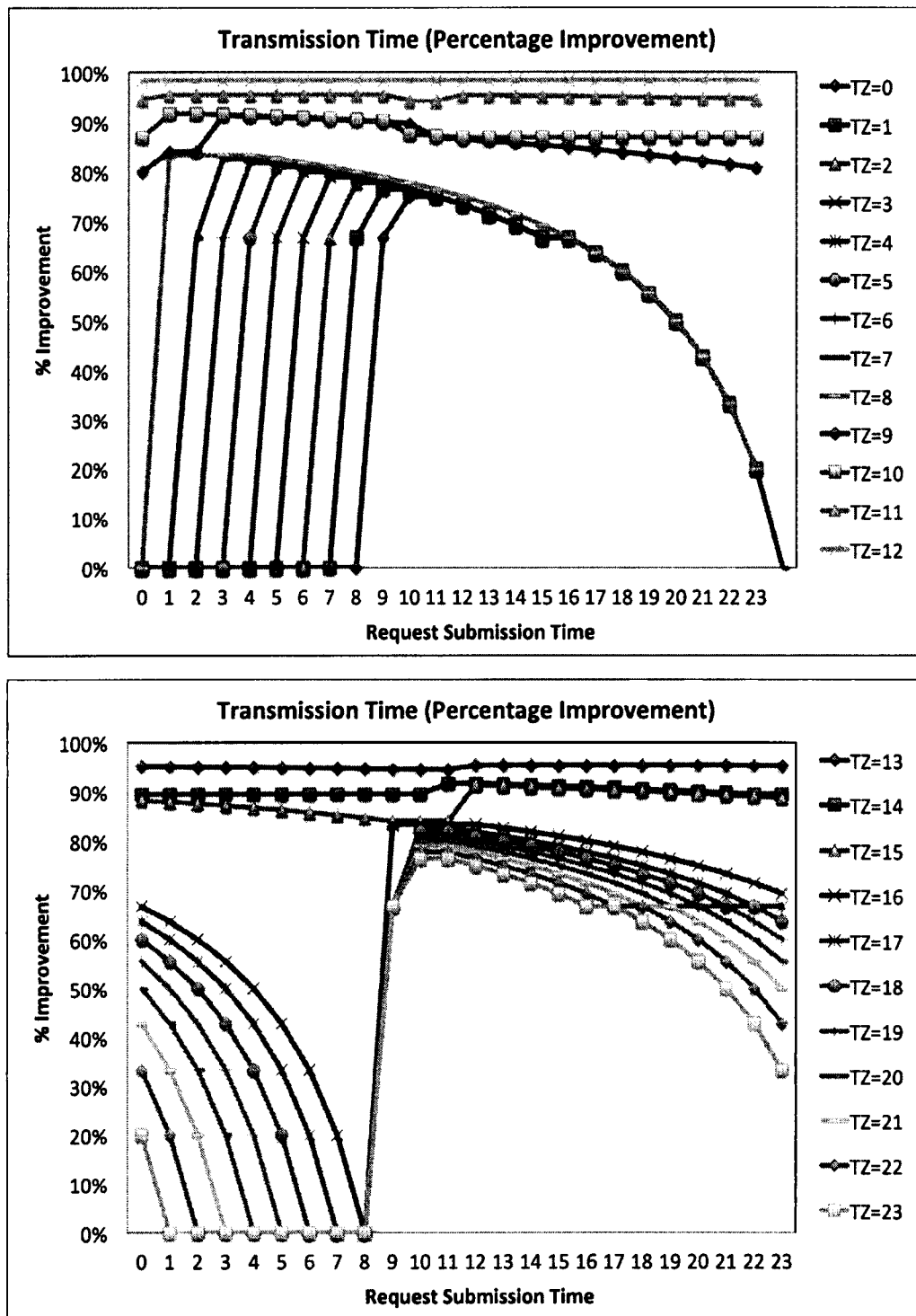


Figure 9.10: Percentage improvement (max 100%) in transmission time when the nice model is used instead of the parallel model for time zone differences 0-12 (top graph) and 13-23 (bottom graph).

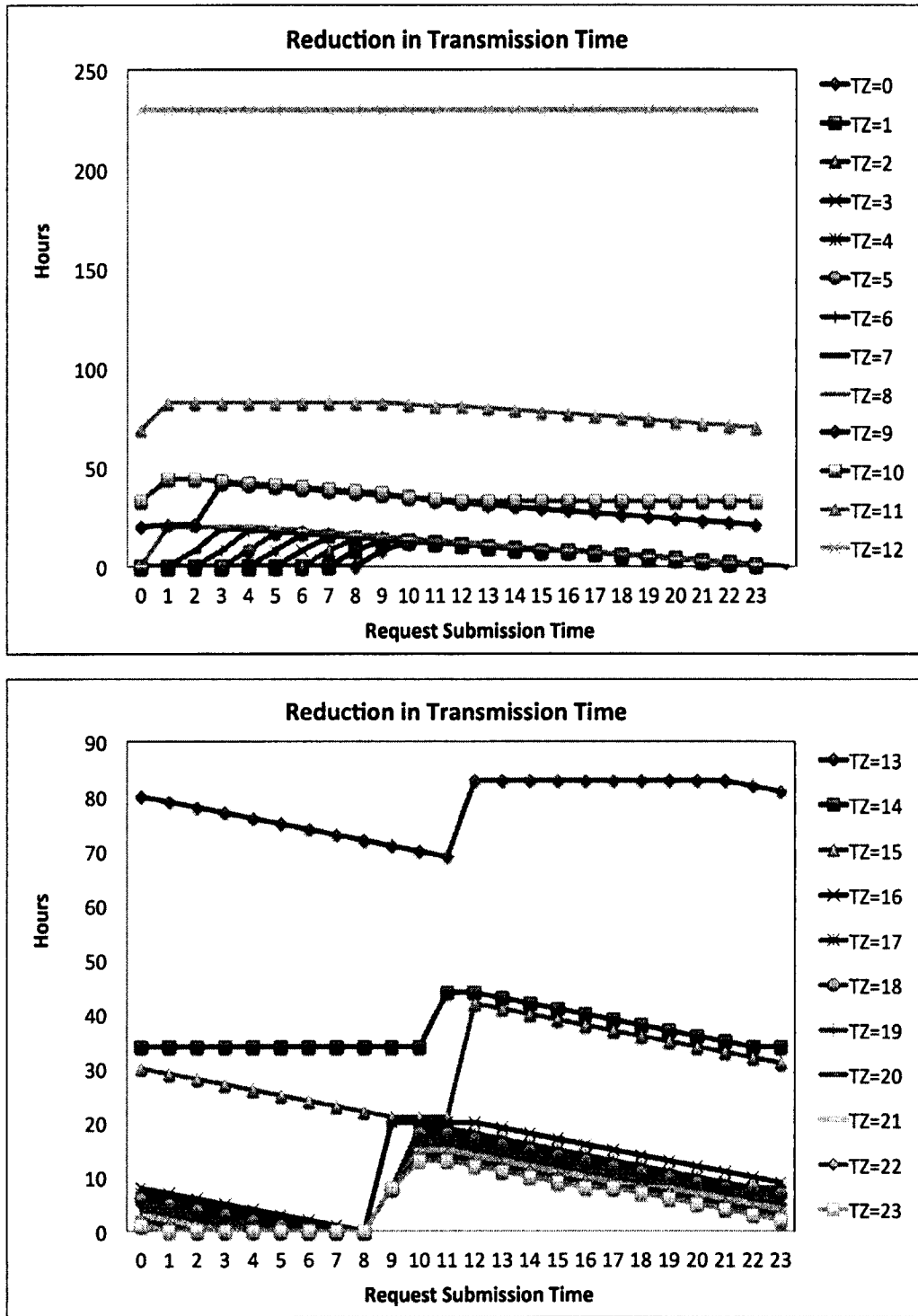


Figure 9.11: Reduction in transmission time when the nice model is used instead of the parallel model for time zone differences 0-12 (top graph) and 13-23 (bottom graph).

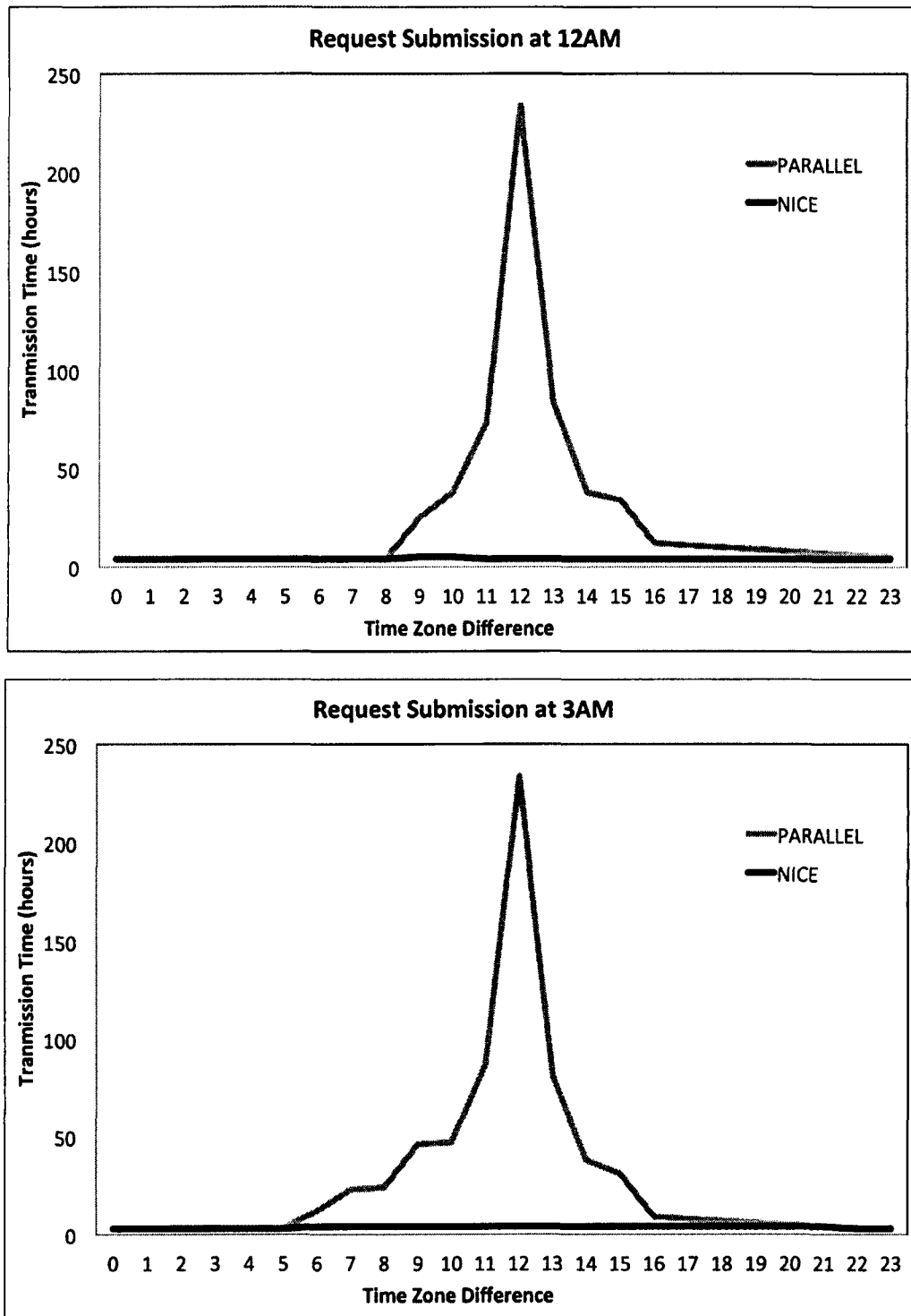


Figure 9.12: Comparison of transmission times for the 1 TB data set for both nice and parallel models under all time zone differences when the request submission time is 12AM (top graph) and 3AM (bottom graph).

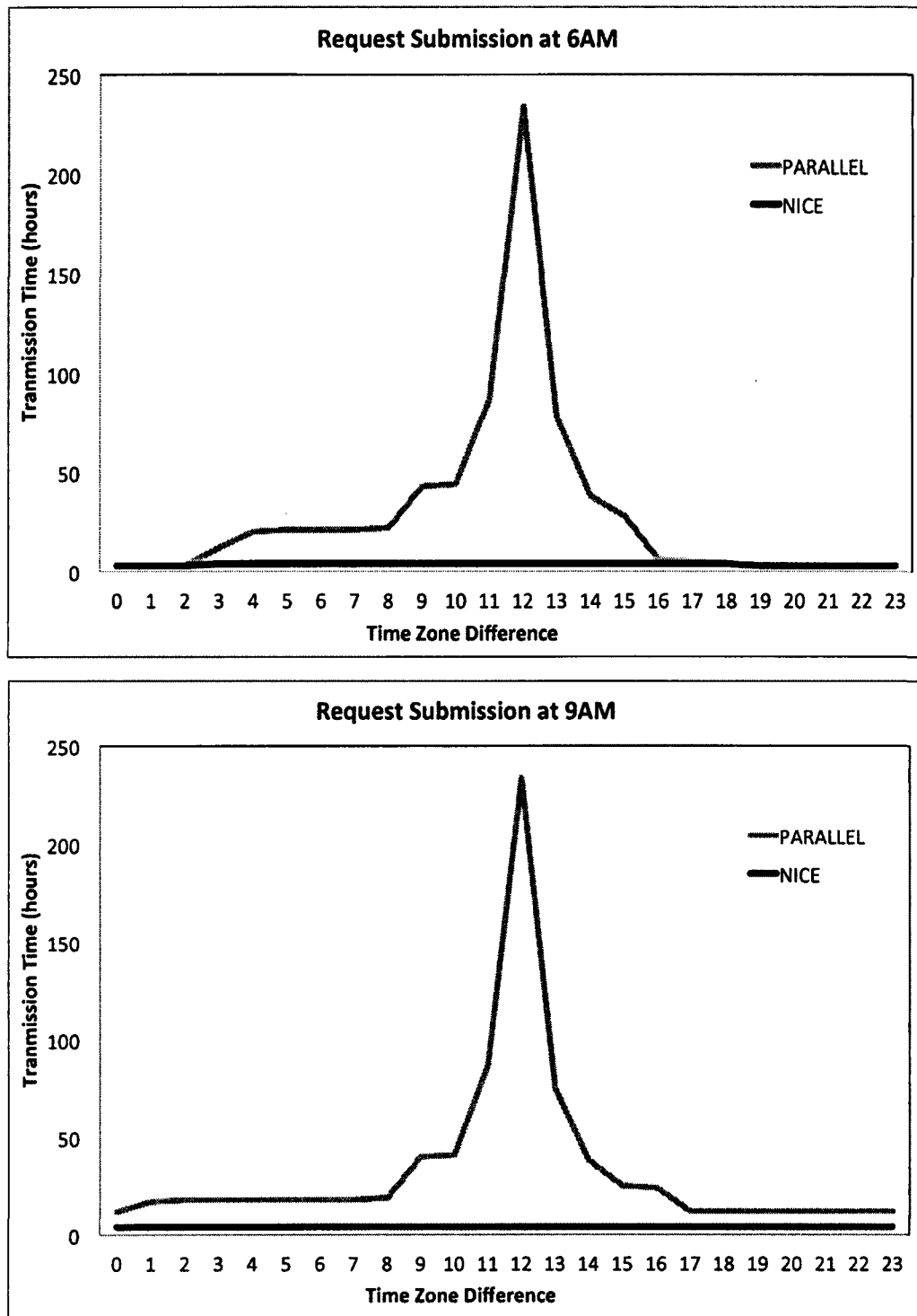


Figure 9.13: Comparison of transmission times for the 1 TB data set for both nice and parallel models under all time zone differences when the request submission time is 6AM (top graph) and 9AM (bottom graph).

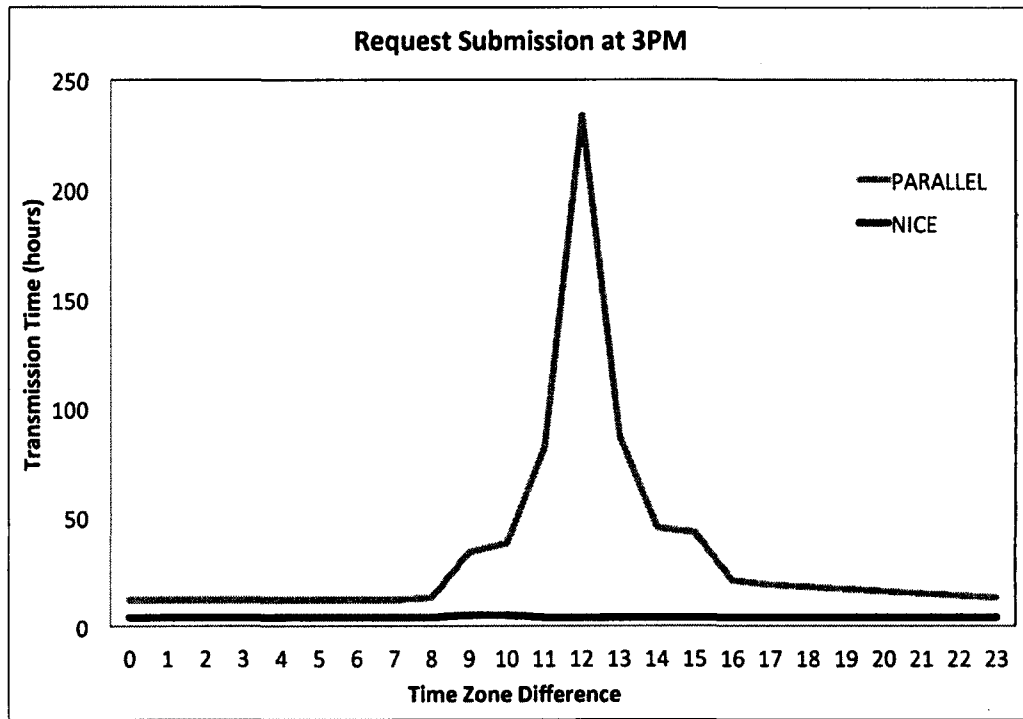
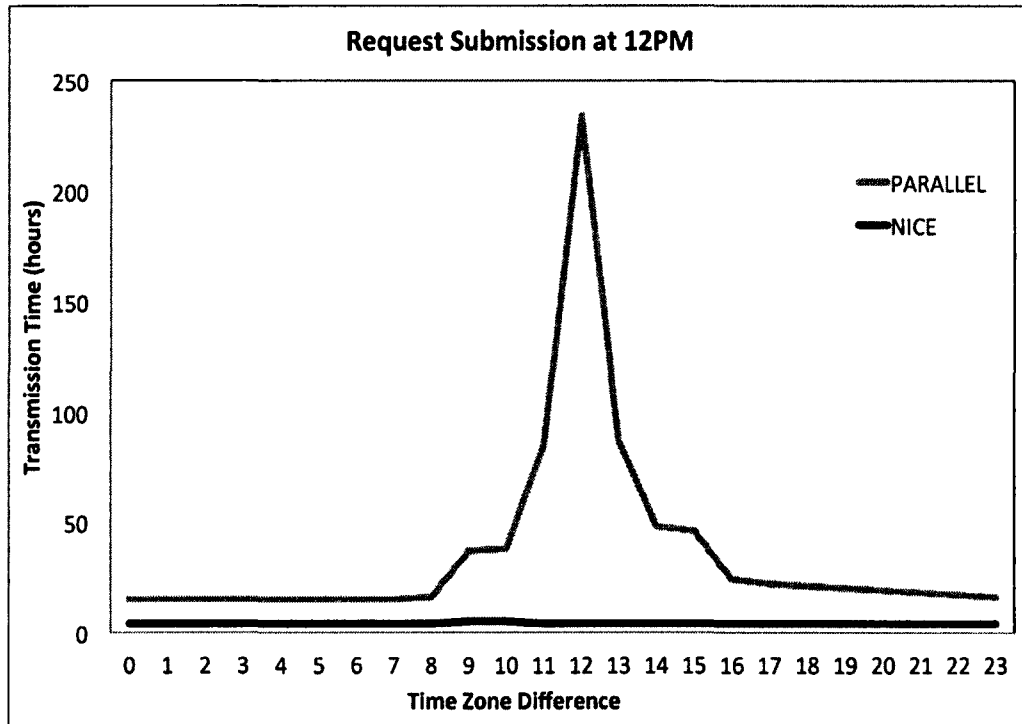


Figure 9.14: Comparison of transmission times for the 1 TB data set for both nice and parallel models under all time zone differences when the request submission time is 12PM (top graph) and 3PM (bottom graph).



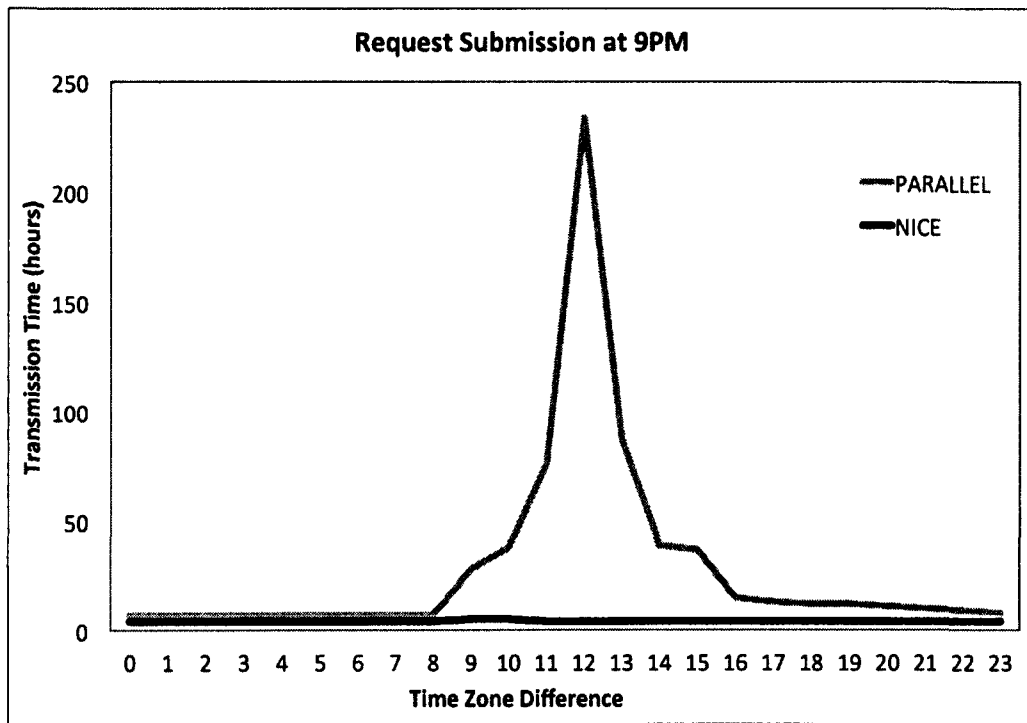
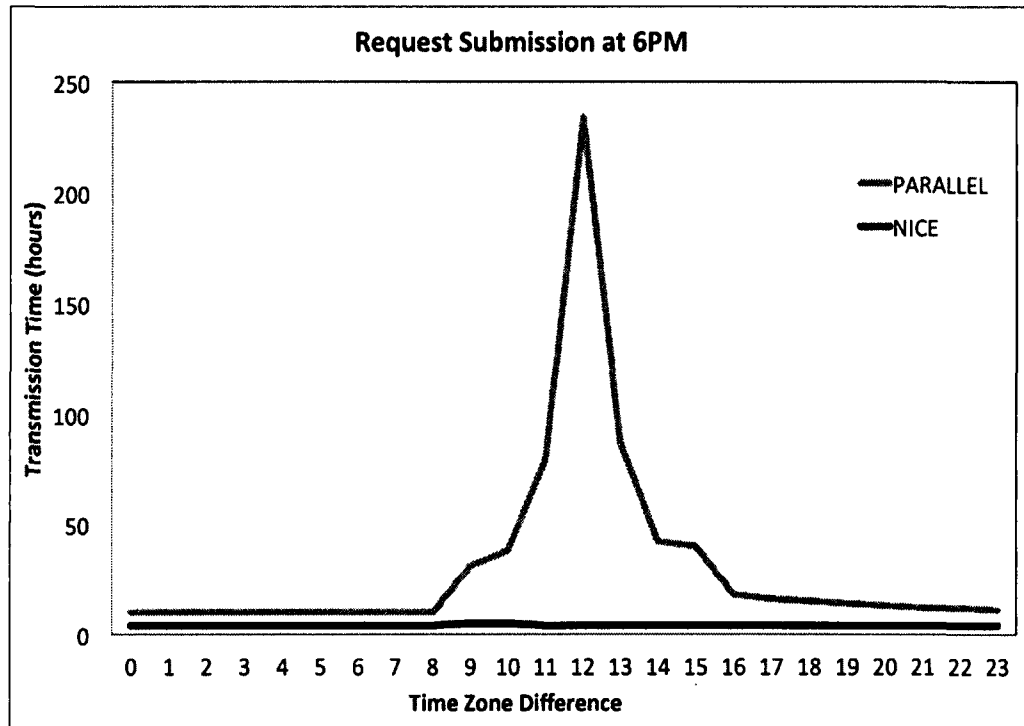


Figure 9.15: Comparison of transmission times for the 1 TB data set for both nice and parallel models under all time zone differences when the request submission time is 6PM (top graph) and 9PM (bottom graph).

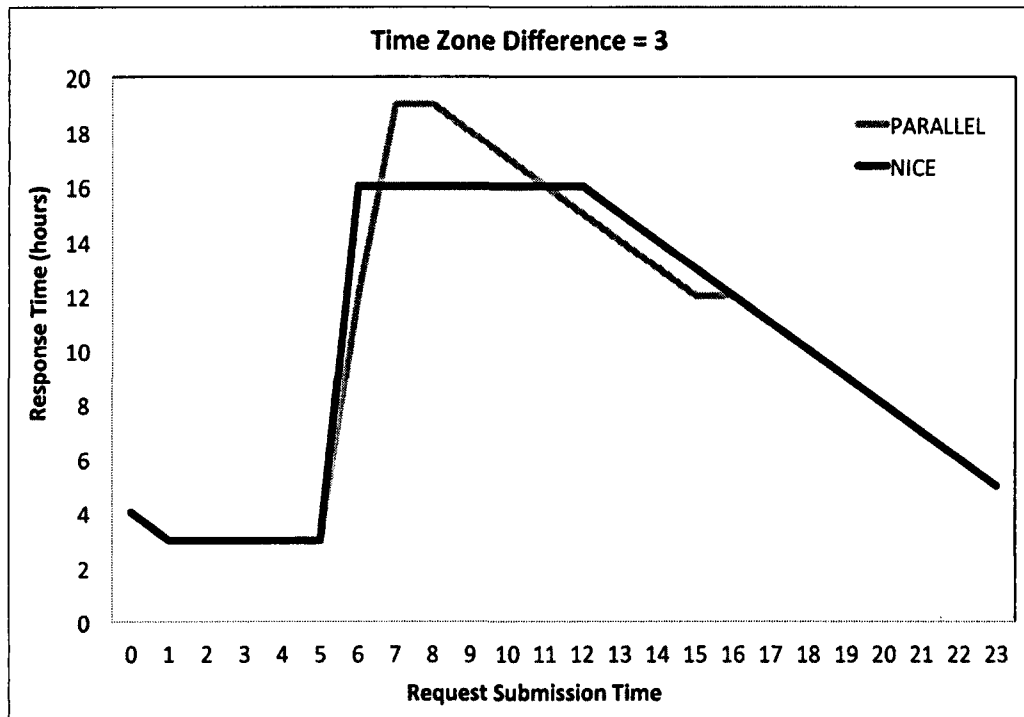
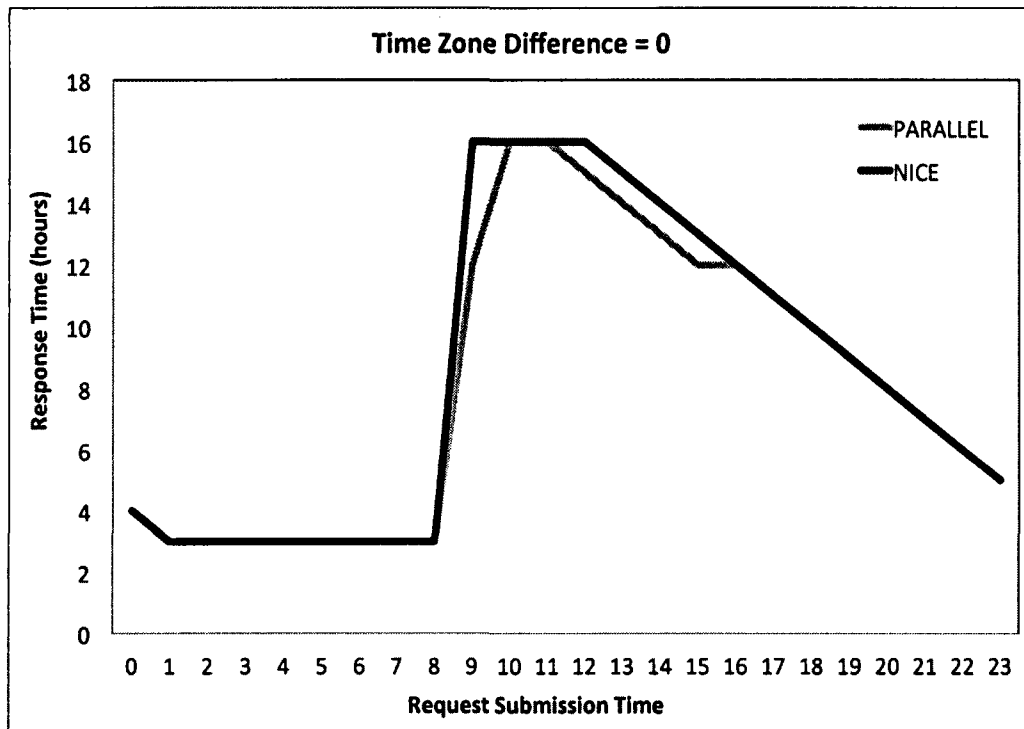


Figure 9.16: Comparison of response times for the 1 TB data set for both nice and parallel models when the time zone differences are 0 (top graph) and 3 (bottom graph).

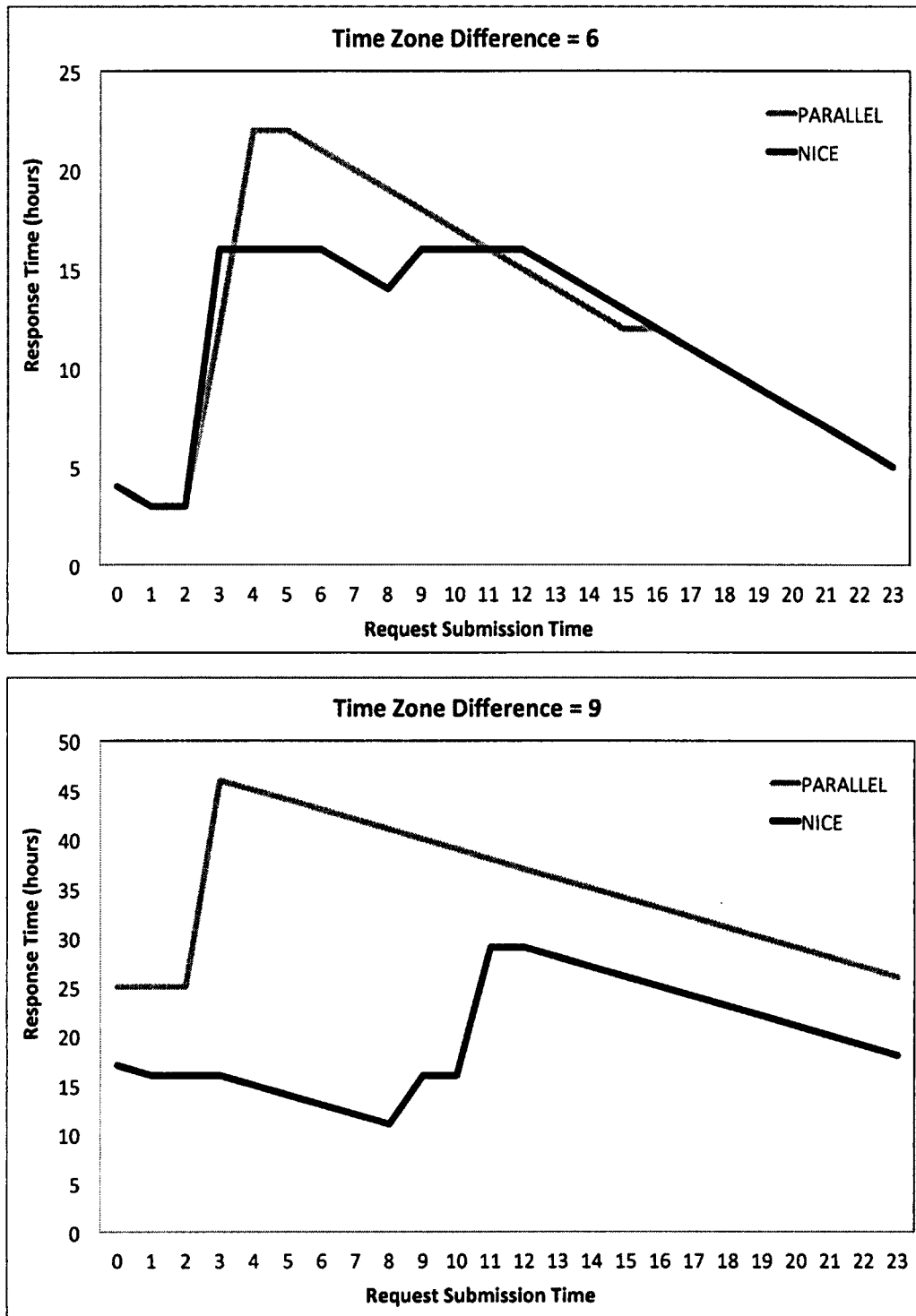


Figure 9.17: Comparison of response times for the 1 TB data set for both nice and parallel models when the time zone differences are 6 (top graph) and 9 (bottom graph).

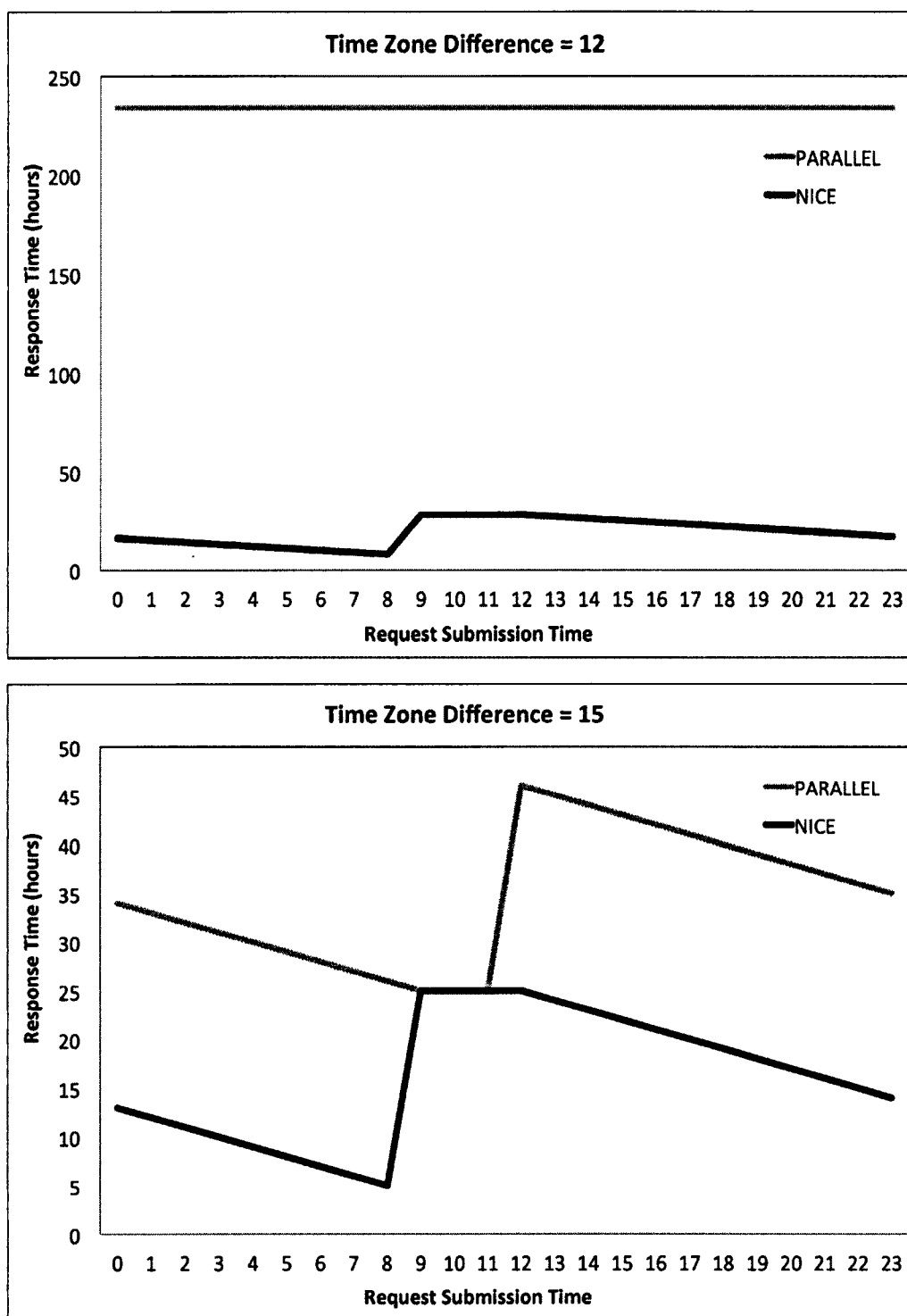


Figure 9.18: Comparison of response times for the 1 TB data set for both nice and parallel models when the time zone differences are 12 (top graph) and 15 (bottom graph).

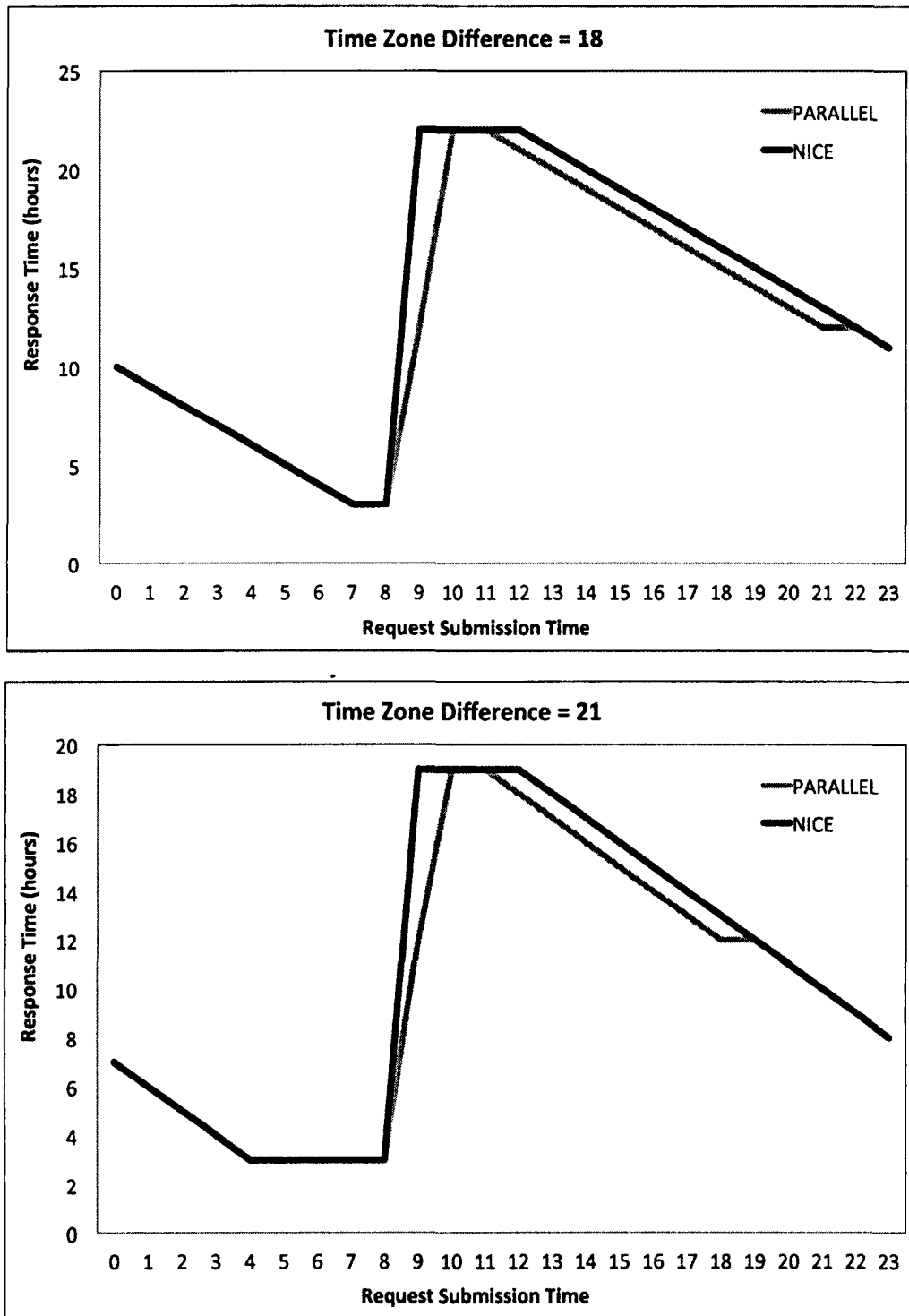


Figure 9.19: Comparison of response times for the 1 TB data set for both nice and parallel models when the time zone differences are 18 (top graph) and 21 (bottom graph).

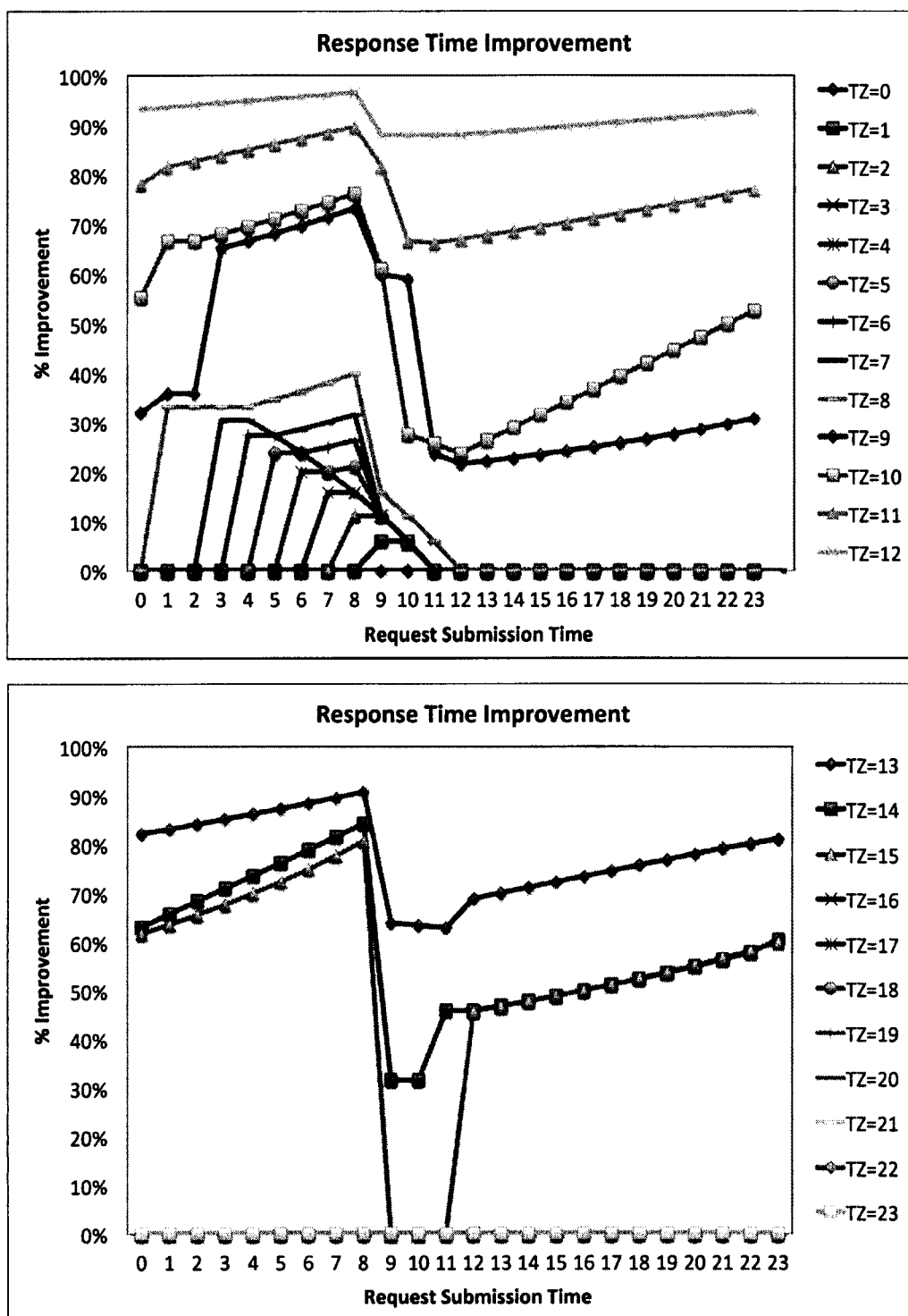


Figure 9.20: Percentage improvement (max 100%) in response time when the nice model is used instead of the parallel model for time zone differences 0-12 (top graph) and 13-23 (bottom graph).

## 9.2 Response time (RT as TimeDiff and InitiateTime varies):

Figures 9.16, 9.17, 9.18, and 9.19 plot the response times (RT) for both the nice and parallel models when transferring the 1 GB data set with varying time zone differences. Note that for the parallel model, the response time is the same as the transmission time ( $RT = TT$ ). Both models perform better when  $TimeDiff = 0$ . As the time difference increases, the wait time in the nice model increases and in the parallel model, the time zones don't synchronize resulting in low bandwidth transmission. The percentage improvement in response time is shown in figure 9.20. The greatest improvement when using the nice model over the parallel model occurs at time difference 12.

## 9.3 Bandwidth differential between sender and receiver:

In the previous sections, both the sender and receiver had equal bandwidth capacities. The maximum available bandwidth for both was 1 Gbps. In this section, I set the receiver to have 4 times the available bandwidth of the sender. Both the sender and receiver still follow the bandwidth availability percentages provided in Figure 8.1. In the evaluation of this setup, I focus on the receiver's transmission time. Since the nice model uses a store-and-forward approach instead of parallel models' end-to-end technique, the receiver has the ability to receive the data faster than the sender transmits the data to the staging server.

Figures 9.21, 9.22, 9.23, and 9.24 plot the transmission time at the receiver's LAN (recTT) when the receiver has 4 times as much bandwidth as sender. For the parallel model, the faster transmission rate at receiver has no impact on performance. For the nice model, when there is no time difference ( $TimeDiff = 0$ ), the faster transmission rate of receiver has no impact. Comparing the top graphs in Figure 9.3 and Figure 9.21 shows the same transmission times for the nice model.

When there is a time difference ( $\text{TimeDiff} > 0$ ) however, the receiver's transmission time ( $\text{recTT}$ ) is faster under the nice model. This improvement is shown when  $\text{TimeDiff} = 9$  in Figure 9.4 and Figure 9.22 are compared. A comparison of Figures 9.5 and 9.23 also shows the decreased receiver's transmission times when  $\text{TimeDiff} = 15$ .

## 9.4 Summary

These evaluations show that the performance of the parallel model is dependent on the time difference ( $\text{TimeDiff}$ ) and the request submission time ( $\text{InitiateTime}$ ), while the performance of the nice model is not. The nice model performs better than parallel. In the next chapter, I explain these experiments using theoretical evaluation.

On a side note, I also collect data regarding the negative impact of big transmissions on other applications during high traffic periods. I observe significant increases in end-to-end packet delays: the streaming video client experiences a 52% increase in packet delays on average and the VoIP client has an even higher increase of 67%.



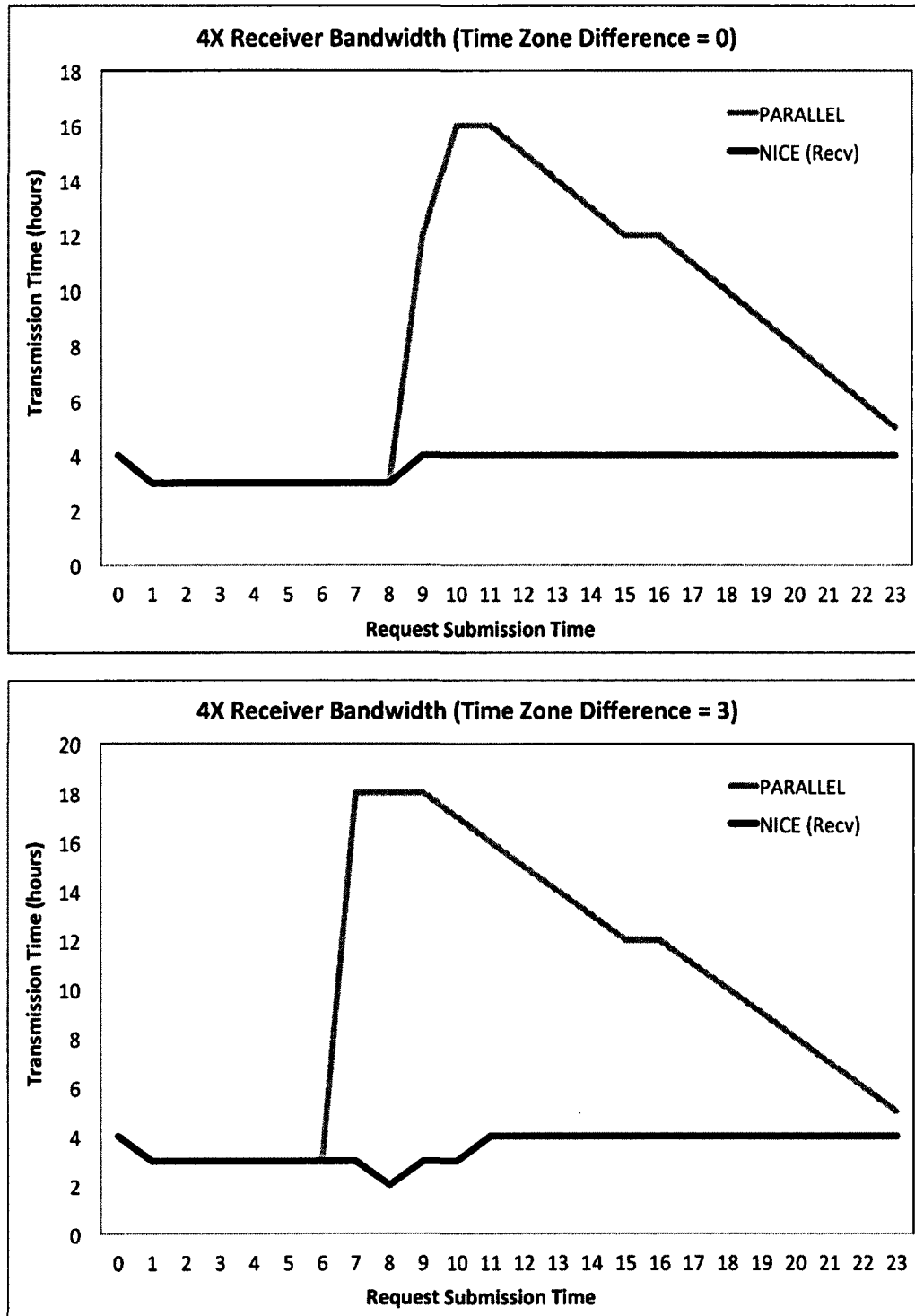


Figure 9.21: Transmitting a 1 TB data set when TimeDiff = 0 (top graph) and TimeDiff = 3 (bottom graph) where the receiver has 4 times the available bandwidth than the sender.

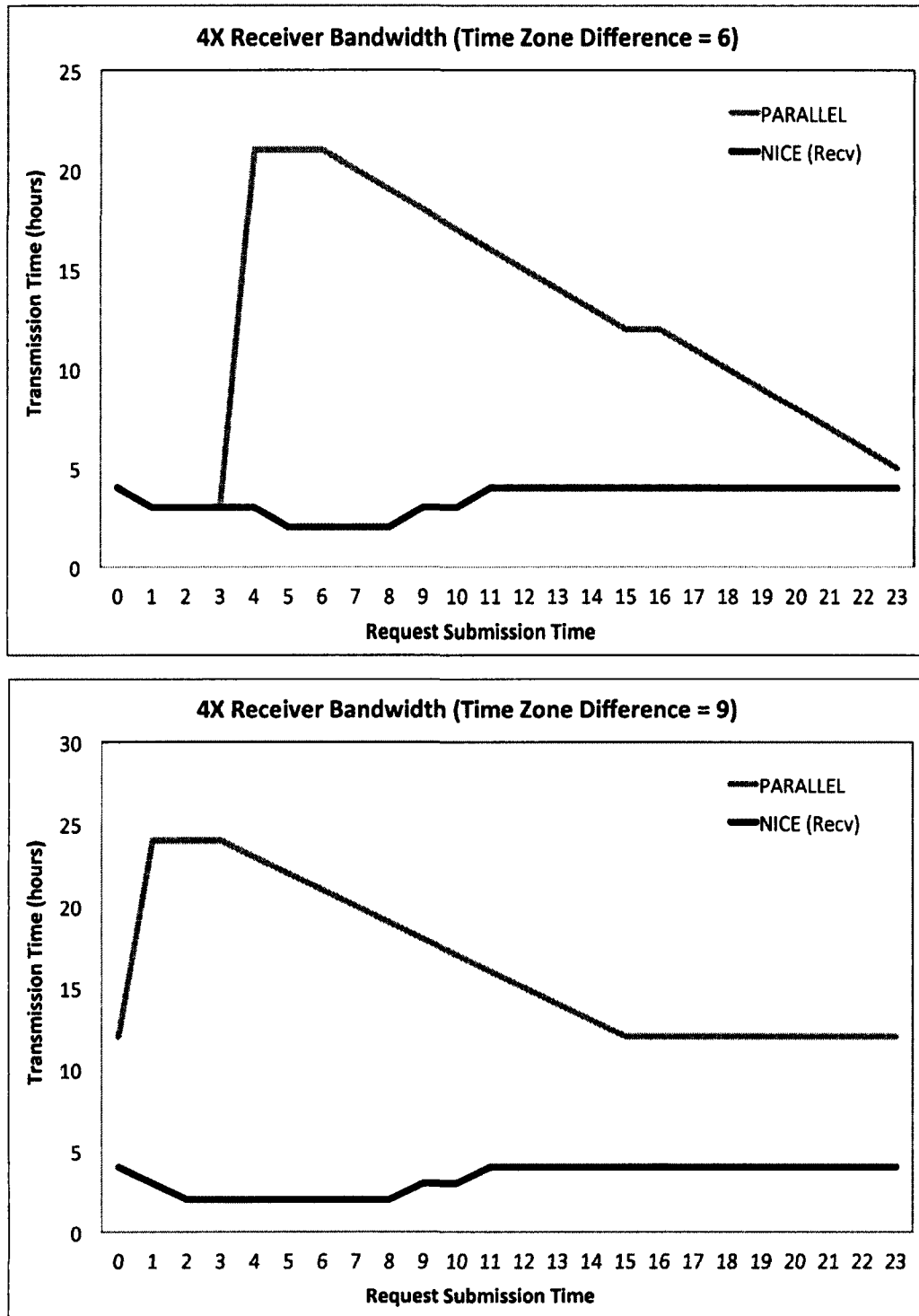


Figure 9.22: Transmitting a 1 TB data set when TimeDiff = 6 (top graph) and TimeDiff = 9 (bottom graph) where the receiver has 4 times the available bandwidth than the sender.

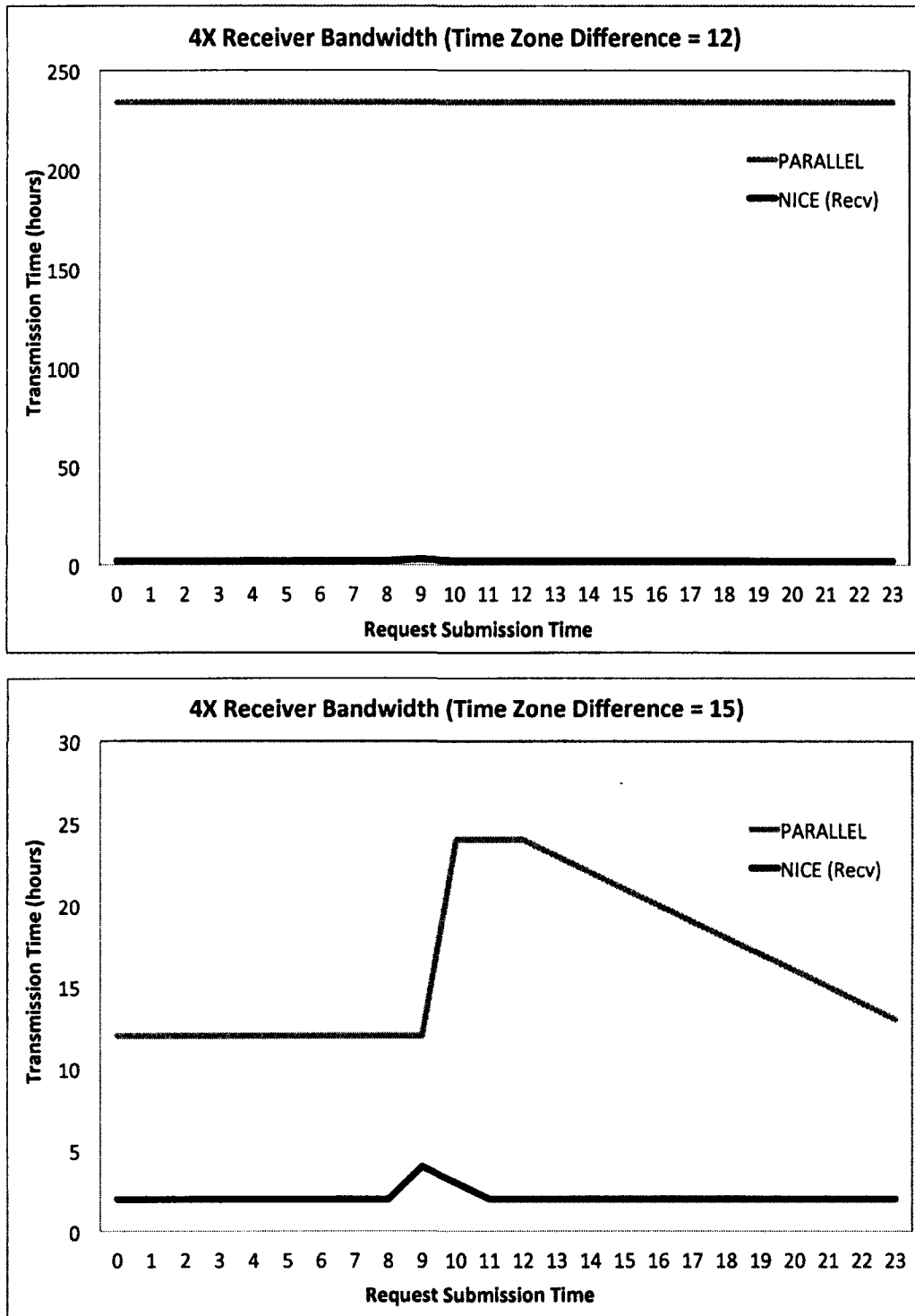


Figure 9.23: Transmitting a 1 TB data set when TimeDiff = 12 (top graph) and TimeDiff = 15 (bottom graph) where the receiver has 4 times the available bandwidth than the sender.

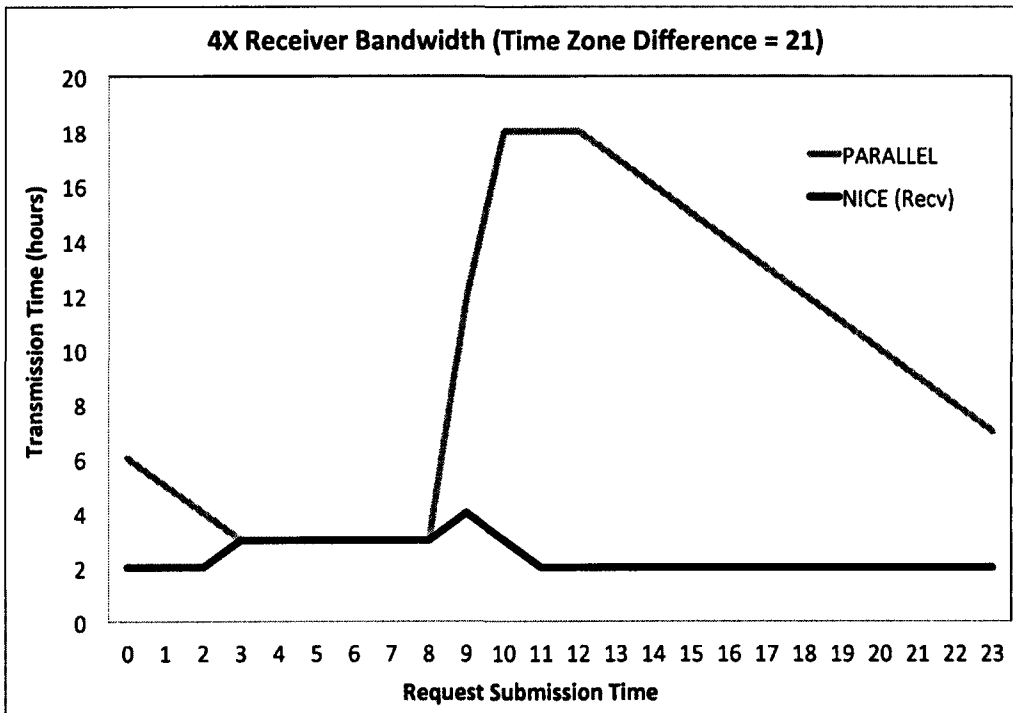
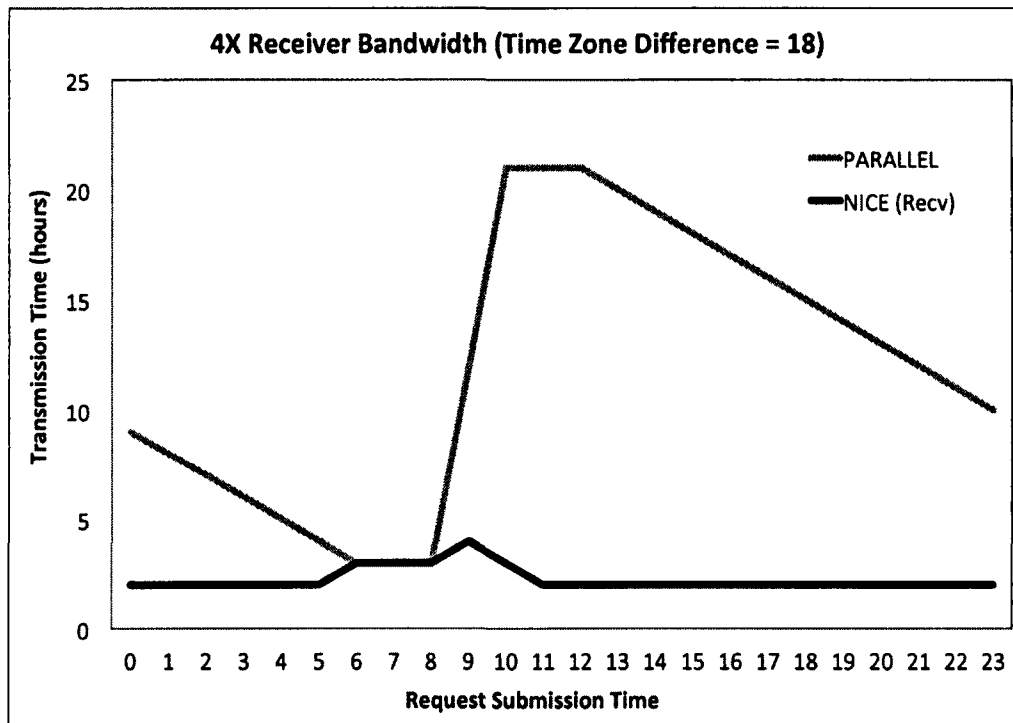


Figure 9.24: Transmitting a 1 TB data set when TimeDiff = 18 (top graph) and TimeDiff = 21 (bottom graph) where the receiver has 4 times the available bandwidth than the sender.

## CHAPTER X

### Analysis of Evaluations

The experiments in the previous chapter clearly show that the nice model far outperforms the parallel model. From experimental analysis alone, it is hard to quantify the impact of the request submission time (`InitiateTime`), the time zone difference (`TimeDiff`), and the bandwidth differential on performance. The goal of this theoretical analysis is to better understand the experiment results.

Before starting the theoretical analysis, we define a few more variables. All the variables are either defined here or in Chapter VIII. The parameters `SendBW[i]`, `RecBW[i]` capture the impact of traffic intensity on the transmission tool. The analysis can be simplified, without changing the essential performance characteristics, by assuming that `SendBW` and `RecBW` do not vary by the hour. Instead, the bandwidth available for big data transmissions is a fixed low value during the high traffic period and a fixed high value during the low traffic period. Let `SendLowBW` and `SendHighBW` represent the low and high transmission rate per hour, respectively, at the sender's LAN; let `RecLowBW` and `RecHighBW` represent the low and high transmission rate per hour, respectively, at the receiver's LAN. Note that the transmission rate is given in units of per hour, not per second; If low transmission rate is 10 Mb/s, then  $\text{SendLowBW} = 10 * 60 * 60 = 36000 \text{ Mb/h}$ . The only reason for using hour as the unit is to improve the readability of the analysis by not having to constantly multiply

by  $60 * 60$ .

At any hour, the sender and receiver LANs are in one of the following four states: 1) SendLowBW, RecLowBW; 2) SendLowBW, RecHighBW; 3) SendHighBW, RecLowBW; and 4) SendHighBW, RecHighBW. Since end-to-end transmission rate is dependent on the smallest bandwidth, the bandwidth rate at any hour would be one of:

$$\text{Low} = \text{MIN}\{\text{SendLowBW}, \text{RecLowBW}\};$$

$$\text{SendLow} = \text{MIN}\{\text{SendLowBW}, \text{RecHighBW}\};$$

$$\text{RecLow} = \text{MIN}\{\text{SendHighBW}, \text{RecLowBW}\};$$

$$\text{High} = \text{MIN}\{\text{SendHighBW}, \text{RecHighBW}\};$$

Let #HighBWHrs and #LowBWHrs represent the number of hours in a day when traffic is off-peak and peak, respectively. Recall that, on our campus LAN, the off-peak traffic period is from hours 0 to 9, while the peak traffic period is from hours 10 to 23. Therefore, #HighBWHrs = 10 and #LowBWHrs = 14.

### 10.0.1 Nice

Maximum FileSize transmitted in 24 hours:

Let 24hrFileSize represent the maximum FileSize that can be transmitted during 24 hours.

$$24\text{hrFileSize} = \text{High} \times \# \text{HighBWHrs}$$

**Result 1.** *For nice, the maximum data that can be transmitted during 24 hours is determined only by High; 24hrFileSize is independent of TimeDiff and InitiateTime.*

Response time RT:

The RT is computed in terms of receiver wait time and receiver transmission time.

$$\text{RT} = \text{recWT} + \text{recTT}$$

recWT is the time from InitiateTime until the start of transmission to the receiver. The receiver's transmission starts at the first high bandwidth hour that is greater than or equal to the sender's first transmission hour. Since the staging server transmits only when the receiver starts off-peak period, recWT is a function of InitiateTime and TimeDiff.

**Result 2.** *sendWT is dependent on InitiateTime. recWT is dependent on both InitiateTime and TimeDiff. Consequently, RT is dependent on both parameters.*

Transmission times sendTT, recTT, TT:

$$TT = \text{MAX}(\text{sendTT}, \text{recTT}) = \lceil \frac{\text{FileSize}}{\text{High}} \rceil$$

The sender's transmission time depends only on the bandwidth at the sender due to the buffering available at the staging servers.

$$\text{sendTT} = \lceil \frac{\text{FileSize}}{\text{SendHighBW}} \rceil$$

Calculating recTT is tricky; depending on TimeDiff, the transmission from sender to receiver may be completely concurrent, partially concurrent, or serial. The computation of recTT also depends on whether the sender or the receiver is faster.

If  $\text{SendHighBW} \geq \text{RecHighBW}$ , then

$$\text{recTT} = \lceil \frac{\text{FileSize}}{\text{RecHighBW}} \rceil.$$

If  $\text{SendHighBW} < \text{RecHighBW}$  and  $\text{TimeDiff} = 0$ , then

$$\text{recTT} = \lceil \frac{\text{FileSize}}{\text{SendHighBW}} \rceil.$$

If  $\text{SendHighBW} < \text{RecHighBW}$  and  $|\text{TimeDiff}| > 0$ , then the value of recTT depends on how much of the file, StageFile, has been transmitted to the staging server before the receiver's transmission time starts. While the receiver catches up with the sender, the file is transmitted at the receiver's faster rate and then afterward, any remaining portion is transmitted at the sender's slower rate.

$$\text{recTT} = \lceil \frac{(\text{StageFile}+)}{\text{RecHighBW}} + \frac{\text{FileSize}-(\text{StageFile}+)}{\text{SendHighBW}} \rceil.$$

The + in StageFile+ represents the additional transmission from the sender while the receiver is trying to catch up. As TimeDiff increases, recTT becomes more dependent on RecHighBW (and less dependent on High = MIN(SendHighBW, RecHighBW)).

**Result 3.** *sendTT only depends on SendHighBW.*

*As TimeDiff increases, recTT becomes less dependent on High and more dependent on RecHighBW.*

Result 3 explains the bandwidth differential graphs in Figures 9.21, 9.22, 9.23, and 9.24. The presence of the staging servers ensures that the bandwidth differential between sender and receiver is hidden.

**Result 4.** *The TT of the nice model only depends on High; TT is insensitive to TimeDiff and InitiateTime.*

## 10.0.2 Parallel

### Maximum FileSize transmitted in 24 hours:

The total data transmitted depends on TimeDiff between sender and receiver.

1) TimeDiff= 0:

$$24\text{hrFileSize} = (\text{High} \times \#\text{HighBWHrs}) + (\text{Low} \times \#\text{LowBWHrs})$$

2) |TimeDiff| = d where  $0 < d \leq \#\text{HighBWHrs}$ :

$$\begin{aligned} 24\text{hrFileSize} = & (\text{RecLow} \times d) + (\text{High} \times (\#\text{HighBWHrs} - d)) + (\text{SendLow} \times d) \\ & + (\text{Low} \times (\#\text{LowBWHrs} - d)) \end{aligned}$$

3) |TimeDiff| = d where  $\#\text{HighBWHrs} < d \leq \#\text{LowBWHrs}$

$$\begin{aligned} 24\text{hrFileSize} = & (\text{RecLow} \times \#\text{HighBWHrs}) + (\text{Low} \times (d - \#\text{HighBWHrs})) \\ & + (\text{SendLow} \times \#\text{HighBWHrs}) + (\text{Low} \times (\#\text{LowBWHrs} - d)) \end{aligned}$$

From 3), it follows that **when transmitting between LANs in India and the US, or between LANs in Japan and the US, the entire parallel transmission is**



carried out in low bandwidth. Equations 1 to 3 (above) explain the performance of parallel in our experiments. Note that  $23 \geq d > \#LowBWHrs$  is not evaluated since it reduces to one of the above cases.

**Result 5.** *Parallel: When TimeDiff=0, 24hrFileSize is maximum. As TimeDiff increases, 24hrFileSize decreases. At TimeDiff  $\geq \#HighBWHrs$ , the data are entirely transmitted at low bandwidth, so 24hrFileSize is smallest.*

When TimeDiff = 0, parallel transmits more data than nice. However, depending on the difference in bandwidth during high traffic and low traffic times, the percentage improvement is insignificant. For example, for a 10Mb low bandwidth and a 1 Gb high bandwidth, parallel transmits only 1.6% more data than nice during the high traffic hours.

For the parallel model:  $RT = TT = sendTT = recTT$ .

**Result 6.** *The transmission times of the parallel model are sensitive to parameters InitiateTime, TimeDiff, bandwidth availability, and transmission rate differential between sender and receiver.*

### 10.0.3 Summary

**Theorem 1.** *For a given FileSize, the TT of the nice model is faster than that of the parallel model.*

*Parallel is best in comparison to nice when TimeDiff is close to 0 and the sender and receiver have similar transmission rates.*

*Parallel is worst in comparison to nice when there is no overlap of low traffic times between sender and receiver.*

Thus, the nice model is better suited to big transmissions over large distances that span time zones and varying network capabilities. In essence, big data transmission via public Internet is a “first mile, last mile” problem. The sender has to wait for

ample bandwidth at its LAN before transmitting; if the receiver LAN is not in synch with the sender, then the big file has to wait at one or more intermediate server(s) until ample bandwidth is available at the receiver - parallel, store-and-forward. Again, the name, nice, is a play on words linking our transmission model to the nice program in Unix. The nice transmission model waits for off-peak hours when bandwidth is available, so it is nice to other Internet users.

## CHAPTER XI

### Conclusions and Future Work

My research journey from grid computing to the issues and challenges of big data transfers was not straightforward. It was through my many discoveries along the way that led me to this point. From the outset it was not clear that this was even a problem that needed to be examined. While attempting to transfer big data sets, I experienced the difficulties associated with transferring large amounts of data through shared networks firsthand. It was clear that this a problem that needs to be investigated.

Big data transfers via the Internet are not a commonplace task for most users today. Currently, there are no tools to facilitate these kinds of transmissions. The task of transferring massive amounts of data across the country or even the globe is a challenging and daunting undertaking for any user. As the popularity of distributed storage propagates and the amount of scientific data continues to surge, the demand for big data transfers will grow at a tremendous rate. The existing tools for moving large amounts of data are based on the parallel model, which is designed to grab as much bandwidth as possible by opening concurrent data streams. This greedy approach may be good for a single user's transfer, however it is not scalable for multiple users on a shared network. The entire system suffers when users attempt to grab bandwidth.

My solution to this problem is the nice model for big data transfers. Under this model, these transfers are relegated to low demand periods when there is ample, idle bandwidth available. This bandwidth can then be repurposed for big data transmissions without impacting other users in the system. Since the nice model uses a store-and-forward approach by utilizing staging servers, the model is able to accommodate differences in time zones and variations in bandwidth. From my evaluations and theoretical analysis, I have shown that the nice model significantly outperforms the existing greedy, parallel model. It is clear that nice is better than greedy when it comes to big data transmissions.

### **11.1 Future Work: CargoExchange application**

In order for multiple users to successfully utilize the nice model for big data transmissions in a shared system, like the campus network, there should be a system-level service that supports these users' workloads. After speaking with researchers in various departments around campus, I received an overwhelming response from these users that they only want guaranteed delivery and ease of use when it comes to this type of transfer. Users do not want to be burdened with error recovery, re-transmissions, security and bandwidth issues. Administrators want to ensure that big data transfers do not impact other applications/users on the network. A system-level service, called CargoExchange, could provide users with this simplicity and administrators with quality of service guarantees.

The CargoExchange service would handle big data transmission for all users on the shared network. The service is called CargoExchange since it bears similarities to companies like UPS and FedEx that are specifically designed to transport large amounts of goods or cargo. The CargoExchange service would be tasked solely with transporting users' big data. This service at the sender, receiver and staging sites would be able to communicate and facilitate all facets of the transfers. In order for

this service to function properly, there are several aspects that would need to be addressed.

**Client Interface:** The client interface would be web based. The client would enter the specifications for the data transfer. Clients have no control over how the data are transmitted, but they can log in and check on the status of their transfers. There may be other features such as payment options and delivery options (fast, regular, etc.). Once a transmission is completed, a client will receive an email notification.

**File System:** The servers utilized by the CargoExchange service are required to store big data sets and retrieve all or parts of files at any time. Since standard file systems are primarily designed for smaller files, the CargoExchange servers should have file systems specifically designed for storage and retrieval of big data (2).

**Tracking System:** The CargoExchange servers must keep track of the users' files while they are being transmitted and once they arrive at their location. The algorithms for keeping track of file movement will need to be explored and potentially developed.

**Security:** The files transmitted via the CargoExchange service would be private. Before transmission, files must be encrypted. Recent advances in encryption (23; 44; 72) can be utilized to ensure that scalable, fast, and reliable security is available to users.

**Compression:** The service could utilize compression techniques in order to reduce file sizes. These techniques must be able to compress large data files at a fairly fast rate. Recent work (59; 71; 79; 85) in this area has resulted in improved performance, however new compression techniques may need to be developed for big data.

**Routing Algorithms:** This is a critical aspect of the CargoExchange service. The routing algorithms refer to routing at a high level, not at the network level. Using the analogy of the roadway system for example, in order to travel from Durham, NH to

Montreal, Canada, one could choose to go via Maine or Vermont. The CargoExchange would determine the selection of Maine or Vermont (or both in the case of parallel transmission). The CargoExchange decides when and how to route users' data, which may include multiple paths.

**Traffic Monitoring:** The CargoExchange service must have knowledge of the traffic present on network links in order to properly schedule data transmissions. The servers will work directly with traffic monitoring and bandwidth management devices, which have an accurate view of the traffic on their networks. CargoExchange servers will use and share this information with other servers in order to facilitate big data transmissions and to utilize available bandwidth.

**Network Protocols:** New network protocols may be required for big data transmissions (58; 76; 91). Advances in network hardware technologies could also be utilized by the CargoExchange service. Since the nitty-gritty of the transfers is removed from the users, the service could implement and utilize any new software/hardware advances that might improve transfer performance or reliability.

## **BIBLIOGRAPHY**

## BIBLIOGRAPHY

- [1] Amazon web services (aws). <http://aws.amazon.com/importexport/>.
- [2] Hadoop distributed file system (hdfs). <http://hadoop.apache.org/>.
- [3] Internet2. <http://www.internet2.org>.
- [4] The teragrid project. <http://www.teragrid.org>.
- [5] Worldwide lhc computing grid (wlcg). <http://lcg.web.cern.ch/LCG/>.
- [6] Dzero experiment, <http://www-d0.fnal.gov>, 2009.
- [7] Enabling grids for escience (egee), <http://www.eu-egee.org/>, 2009.
- [8] The globus alliance, <http://www.globus.org/>, 2009.
- [9] Naregi: National research grid initiative, <http://www.naregi.org>, 2009.
- [10] Ogsa-dai, <http://www.ogsadai.org.uk>, 2009.
- [11] Open science grid (osg), <http://www.opensciencegrid.org>, 2009.
- [12] Uk national grid service, <http://www.ngs.ac.uk>, 2009.
- [13] AAMNITCHI, A., DORAIMANI, S., AND GARZOGGIO, G. Filecules in high-energy physics: Characteristics and impact on resource management. *High Performance Distributed Computing* (2006), 69–80.
- [14] ADHIKARI, V. K., AND ET. AL. Youtube traffic dynamics and its interplay with a tier-1 isp: an isp perspective. In *Internet Measurement (IMC)* (2010).
- [15] AGGARWAL, V., FELDMANN, A., AND SCHEIDELER, C. Can isps and p2p users cooperate for improved performance? *SIGCOMM Comput. Commun. Rev.* 37, 3 (July 2007), 29–40.
- [16] AL-KISWANY, S., RIPEANU, M., IAMNITCHI, A., AND VAZHKUDAI, S. Beyond music sharing: An evaluation of peer-to-peer data dissemination techniques in large scientific collaborations. *Journal of Grid Computing* (March 2009).



- [17] ALLCOCK, W., BESTER, J., BRESNAHAN, J., CHERVENAK, A. L., FOSTER, I., KESSELMAN, C., MEDER, S., NEFEDOVA, V., QUESNEL, D., AND TUECKE, S. Data management and transfer in high performance computational grid environments. *Parallel Computing Journal* 28, 5 (May 2002), 749–771.
- [18] ALLCOCK, W., BESTER, J., BRESNAHAN, J., CHERVENAK, A. L., FOSTER, I., KESSELMAN, C., MEDER, S., NEFEDOVA, V., AND STEVEN, D. Q. Secure, efficient data transport and replica management for high-performance data-intensive computing. In *IEEE Mass Storage Conference* (2001).
- [19] ALLCOCK, W., BRESNAHAN, J., KETTIMUTHU, R., LINK, M., DUMITRESCU, C., RAICU, I., AND FOSTER, I. The globus striped gridftp framework and server. In *Supercomputing* (2005).
- [20] ANDERSEN, D., BALAKRISHNAN, H., KAASHOEK, F., AND MORRIS, R. Resilient overlay networks. *SIGOPS Oper. Syst. Rev.* 35 (October 2001), 131–145.
- [21] BELLISSIMO, A., LEVINE, B. N., AND SHENOY, P. Exploring the use of bittorrent as the basis for a large trace repository. Tech. rep., University of Massachusetts at Amherst, 2004.
- [22] BEN MOSHE, B., DVIR, A., AND SOLOMON, A. Analysis and optimization of live streaming for over the top video. In *IEE CCNC* (2011).
- [23] BETHENCOURT, J., SAHAI, A., AND WATERS, B. Ciphertext-policy attribute-based encryption. SP '07, IEEE, pp. 321–334.
- [24] BHUVANESWARAN, R. S., AND ET AL. Redundant parallel data transfer schemes for the grid environment. In *ACSW Frontiers* (2006).
- [25] BRESNAHAN, J., LINK, M., KHANNA, G., IMANI, Z., KETTIMUTHU, R., AND FOSTER, I. Globus gridftp: What's new in 2007. In *GridNets* (October 2007).
- [26] CAMERON, D., AND ET. AL. Replica management in the european datagrid project. *Journal of Grid Computing* 2, 4 (2004), 341–351.
- [27] CERN. Lhc physics data taking gets underway at new record collision energy of 8tev. <http://press.web.cern.ch> (2012).
- [28] CHA, M., AND ET. AL. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *Internet Measurement Conference* (2007).
- [29] CHANG, R.-S., GUO, M.-H., AND LIN, H.-C. A multiple parallel download scheme with server throughput and client bandwidth considerations for data grids. *Future Generation Computer Systems* 24, 8 (2008), 798–805.

- [30] CHERVENAK, A., FOSTER, I., KESSELMAN, C., SALISBURY, C., AND TUECKE, S. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications* 23 (2001), 187–200.
- [31] CHHABRA, P., ERRAMILI, V., LAOUTARIS, N., SUNDARAM, R., AND RODRIGUEZ, P. Algorithms for constrained bulk-transfer of delay-tolerant data. In *Communications (ICC), 2010 IEEE International Conference on* (may 2010), pp. 1 –5.
- [32] CHHABRA, P., LAOUTARIS, N., RODRIGUEZ, P., AND SUNDARAM, R. Home is where the (fast) internet is: flat-rate compatible incentives for reducing peak load. In *Proceedings of the 2010 ACM SIGCOMM workshop on Home networks* (New York, NY, USA, 2010), HomeNets '10, ACM, pp. 13–18.
- [33] COHEN, B. Bittorrent, <http://www.bittorrent.com>.
- [34] DE CICCIO, L., MASCOLO, S., AND PALMISANO, V. Skype video responsiveness to bandwidth variations. In *NOSSDAV* (2008).
- [35] DHAMDHERE, A., AND DOVROLIS, C. Isp and egress path selection for multi-homed networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings* (april 2006), pp. 1 –12.
- [36] FENG, J., AND HUMPHREY, M. Eliminating replica selection - using multiple replicas to accelerate data transfer on grids. In *ICPADS* (2004), p. 359.
- [37] FOSTER, I. What is the grid? a three point checklist. *GRIDToday* (July 2002).
- [38] FOSTER, I. Globus toolkit version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing* (2006), pp. 2–13.
- [39] FOSTER, I., AND IAMNITCHI, A. On death, taxes, and the convergence of peer-to-peer and grid computing. *Peer-to-Peer Systems II* (2003), 118–128.
- [40] FOSTER, I., AND KESSELMAN, C. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.
- [41] FOSTER, I., KESSELMAN, C., AND TUECKE, S. The anatomy of the Grid: Enabling scalable virtual organizations. *The International Journal of High Performance Computing Applications* 15, 3 (Fall 2001), 200–222.
- [42] GILL, P., ARLITT, M., LI, Z., AND MAHANTI, A. Youtube traffic characterization: a view from the edge. In *Internet Measurement Conference (IMC'07)* (2007).

- [43] GOLDENBERG, D. K., QIU, L., XIE, H., YANG, Y. R., AND ZHANG, Y. Optimizing cost and performance for multihoming. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2004), SIGCOMM '04, ACM, pp. 79–92.
- [44] GOYAL, V., PANDEY, O., SAHAI, A., AND WATERS, B. Attribute-based encryption for fine-grained access control of encrypted data. CCS '06, ACM.
- [45] GRANT, A., ANTONIOLETTI, M., HUME, A. C., KRAUSE, A., DOBRZELECKI, B., JACKSON, M. J., PARSONS, M., ATKINSON, M. P., AND THEOCHAROPOULOS, E. Ogsa-dai: Middleware for data integration: Selected applications. *eScience, 2008. eScience '08. IEEE Fourth International Conference on* (Dec. 2008), 343–343.
- [46] GRIM, K. Tier-3 computing centers expand options for physicists. *International Science Grid This Week (iSGTW)* (January 2009).
- [47] GYARMATI, L., SIRIVIANOS, M., AND LAOUTARIS, N. Sharing the cost of backbone networks: Simplicity vs. precision. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on* (march 2012), pp. 171–176.
- [48] HAKKEN, N. Netflix and akamai reports show sustained broadband speeds falter in u.s.
- [49] HEY, A. J. G., AND TREFETHEN, A. E. The data deluge: An e-science perspective, 2003.
- [50] HOELZLE, U. Keynote speech at the open networking summit. <http://www.youtube.com/watch?v=VLHJUfgxEO4> (2012).
- [51] HUANG, N.-F., CHANG, H.-Y., LIN, Y.-W., AND HSU, K.-S. A novel bandwidth management scheme for video streaming service on public-shared network. In *IEEE ICC* (2008).
- [52] HUNT, N. Netflix lowers data usage by 2/3 for members in canada. Tech. Rep. <http://blog.netflix.com/2011/03/netflix-lowers-data-usage-by-23-for.html>, 2011.
- [53] HUR, N., AND ET. AL. 3dtv broadcasting and distribution systems. *IEEE Transactions on Broadcasting* (2011).
- [54] J. KNOBLOCH, L. R. Lhc computing grid - technical design report. Tech. Rep. LCG-TDR-001, CERN, June 2005.
- [55] JAIN, S., FALL, K., AND PATRA, R. Routing in a delay tolerant network. *SIGCOMM Comput. Commun. Rev.* 34, 4 (Aug. 2004), 145–158.

- [56] JONES, E. P., LI, L., SCHMIDTKE, J. K., AND WARD, P. A. Practical routing in delay-tolerant networks. *IEEE Transactions on Mobile Computing* 6 (2007), 943–959.
- [57] KAPLAN, A., FOX, G. C., AND VON LASZEWSKI, G. Gridtorrent framework: A high-performance data transfer and data sharing framework for scientific computing. In *Grid Computing Environments (GCE) workshop* (2007).
- [58] KATABI, D., HANDLEY, M., AND ROHRS, C. Congestion control for high bandwidth-delay product networks. *SIGCOMM Comput. Commun. Rev.* 32 (August 2002), 89–102.
- [59] KOTHIYAL, R., TARASOV, V., SEHGAL, P., AND ZADOK, E. Energy and performance evaluation of lossless file data compression on server systems. In *SYSTOR 2009*, ACM.
- [60] LAKHINA, A., PAPAGIANNAKI, K., CROVELLA, M., DIOT, C., KOLACZYK, E. D., AND TAFT, N. Structural analysis of network traffic flows. In *Proceedings of the joint international conference on Measurement and modeling of computer systems* (New York, NY, USA, 2004), SIGMETRICS '04/Performance '04, ACM, pp. 61–72.
- [61] LAM, C., LIU, H., KOLEY, B., ZHAO, X., KAMALOV, V., AND GILL, V. Fiber optic communication technologies: What's needed for datacenter network operations. *Communications Magazine, IEEE* 48, 7 (2010), 32–39.
- [62] LAMEHAMEDI, H., SZYMANSKI, B., SHENTU, Z., AND DEELMAN, E. Data replication strategies in grid environments. In *Proc. Of the Fifth International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'02)* (2002).
- [63] LAOUTARIS, N., SIRIVIANOS, M., YANG, X., AND RODRIGUEZ, P. Inter-datacenter bulk transfers with netstitcher. In *Proceedings of the ACM SIGCOMM 2011 conference* (New York, NY, USA, 2011), SIGCOMM '11, ACM, pp. 74–85.
- [64] LAOUTARIS, N., SMARAGDAKIS, G., RODRIGUEZ, P., AND SUNDARAM, R. Delay tolerant bulk data transfers on the internet. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems* (New York, NY, USA, 2009), SIGMETRICS '09, ACM, pp. 229–238.
- [65] LAURE, E., FISHER, S. M., FROHNER, A., GRANDI, C., KUNSZT, P. Z., KRENEK, A., MULMO, O., PACINI, F., PRELZ, F., WHITE, J., BARROSO, M., BUNCIC, P., HEMMER, F., DI MEGLIO, A., AND EDLUND, A. Programming the grid with glite. *Computational Methods in Science and Technology* 12, 1 (2006), 33–45.

- [66] LAURE, E., HEMMER, F., PRELZ, F., BECO, S., FISHER, S., LIVNY, M., GUY, L., BARROSO, M., BUNCIC, P., KUNSZT, P. Z., DI MEGLIO, A., AIMAR, A., EDLUND, A., GROEP, D., PACINI, F., SGARAVATTO, M., AND MULMO, O. Middleware for the next generation grid infrastructure. *Computing in High Energy Physics and Nuclear Physics* (October 2004), 826.
- [67] LEVIN, D., AND ET. AL. Bittorrent is an auction: analyzing and improving bittorrent's incentives. *SIGCOMM Comput. Commun. Rev.* (2008).
- [68] LI, J., QIAO, C., XU, J., AND XU, D. Maximizing throughput for optical burst switching networks. *IEEE/ACM Trans. Netw.* 15 (October 2007), 1163–1176.
- [69] LI, M., AND BAKER, M. *The Grid - Core Technologies*. John Wiley and Sons, Inc., 2005.
- [70] LIMONCELLI, T. A. Openflow: A radical new idea in networking. *Queue* 10, 6 (June 2012), 40:40–40:46.
- [71] LIN, M.-B., LEE, J.-F., AND JAN, G. E. A lossless data compression and decompression algorithm and its hardware architecture. *IEEE Trans. Very Large Scale Integr. Syst.* 14 (2006), 925–936.
- [72] LIU, J. K., AU, M. H., AND SUSILO, W. Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model: extended abstract. ASIACCS '07, ACM.
- [73] LUCIO, G. F., PAREDES-FARRERA, M., JAMMEH, E., FLEURY, M., AND REED, M. J. Opnet modeler and ns-2. In *ICOSMO* (2003), pp. 700–707.
- [74] MAHIMKAR, A., CHIU, A., DOVERSPIKE, R., FEUER, M. D., MAGILL, P., MAVROGIORGIS, E., PASTOR, J., WOODWARD, S. L., AND YATES, J. Bandwidth on demand for inter-data center communication. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks* (New York, NY, USA, 2011), HotNets-X, ACM, pp. 24:1–24:6.
- [75] MARCON, M., SANTOS, N., GUMMADI, K. P., LAOUTARIS, N., RODRIGUEZ, P., AND VAHDAT, A. Netex: efficient and cost-effective internet bulk content delivery. In *Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems* (New York, NY, USA, 2010), ANCS '10, ACM, pp. 30:1–30:2.
- [76] MASCOLO, S., AND VACIRCA, F. Congestion control and sizing router buffers in the internet. In *CDC-ECC '05*.
- [77] MCGARRY, M., REISSLEIN, M., AND MAIER, M. Ethernet passive optical network architectures and dynamic bandwidth allocation algorithms. *Communications Surveys Tutorials, IEEE* 10, 3 (2008), 46 –60.

- [78] MCNIERNEY, M. College increases web speed for trial period. *The Dartmouth* (2011).
- [79] MILWARD, M., NÚÑEZ, J. L., AND MULVANEY, D. Design and implementation of a lossless parallel high-speed data compression system. *IEEE Trans. Parallel Distrib. Syst.* 15 (June 2004), 481–490.
- [80] MINOLI, D. *A Networking Approach to Grid Computing*. John Wiley and Sons, Inc., 2005.
- [81] MITZENMACHER, M. How useful is old information? *Parallel and Distributed Systems, IEEE Transactions on* 11, 1 (Jan 2000), 6–20.
- [82] MITZENMACHER, M. The power of two choices in randomized load balancing. *Parallel and Distributed Systems, IEEE Transactions on* 12, 10 (Oct 2001), 1094–1104.
- [83] NAKAO, A., PETERSON, L., AND BAVIER, A. A routing underlay for overlay networks. *SIGCOMM '03* (2003).
- [84] NICHOLSON, C., AND ET. AL. Dynamic data replication in lcg 2008. *Concurrency and Computation: Practice and Experience* 20, 11 (2008), 1259–1271.
- [85] NÚÑEZ, J. L., AND JONES, S. Gbit/s lossless data compression hardware. *IEEE Trans. Very Large Scale Integr. Syst.* 11 (June 2003), 499–510.
- [86] NYGREN, E., SITARAMAN, R. K., AND SUN, J. The akamai network: a platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.* 44 (August 2010), 2–19.
- [87] O'MALLEY, S. Flood watch: Ohio's internet connection overflows. <http://www.ohio.edu/oit/news/ohio-internet-connection-overflows.cfm> (March 2011).
- [88] OPNET. <http://www.opnet.com>.
- [89] OTTO, J., STANOJEVIC, R., AND LAOUTARIS, N. Temporal rate limiting: Cloud elasticity at a flat fee. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on* (march 2012), pp. 151–156.
- [90] PETERSON, L., AND DAVIE, B. *Computer Networks: A Systems Approach*. Morgan Kaufmann Publishers Inc., 2012.
- [91] PLANK, J., BASSI, A., MOORE, T., SWANY, M., AND WOLSKI, R. Managing data storage in the network. *Internet Computing, IEEE* 5, 5 (2001), 50–58.
- [92] PUJOL, J., TOLEDO, A., AND RODRIGUEZ, P. Fair routing in delay tolerant networks. In *INFOCOM 2009, IEEE* (april 2009), pp. 837–845.

- [93] QI, J., ZHANG, H., JI, Z., AND YUN, L. Analyzing bittorrent traffic across large network. In *Cyberworlds* (2008), pp. 759–764.
- [94] RABINOVICH, M., AND SPATSCHECK, O. *Web Caching and Replication*. Addison-Wesley, 2002.
- [95] RAHMAN, R. M., ALHAJJ, R., AND BARKER, K. Replica selection strategies in data grid. *Journal of Parallel and Distributed Computing* (2008).
- [96] RAHMAN, R. M., BARKER, K., AND ALHAJJ, R. Replica selection in grid environment: a data-mining approach. In *SAC '05* (New York, NY, USA), pp. 695–700.
- [97] RAJASEKAR, A., WAN, M., AND MOORE, R. Mysrb and srb - components of a data grid. *High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on* (2002), 301–310.
- [98] RAJASEKAR, A., WAN, M., MOORE, R., SCHROEDER, W., KREMENEK, G., JAGATHEESAN, A., COWART, C., ZHU, B., CHEN, S.-Y., AND OLSCHANOWSKY, R. Storage resource broker-managing distributed data in a grid. In *Computer Society of India Journal, Special Issue on SAN* (October 2003), vol. 33, pp. 42–54.
- [99] RAMAKRISHNAN, L., GUOK, C., JACKSON, K., KISSEL, E., SWANY, D. M., AND AGARWAL, D. On-demand overlay networks for large scientific data transfers. In *CCGrid* (2010).
- [100] RANGANATHAN, K., AND FOSTER, I. T. Identifying dynamic replication strategies for a high-performance data grid. In *GRID* (2001), pp. 75–86.
- [101] RODRIGUEZ, P., AND BIRSACK, E. W. Dynamic parallel access to replicated content in the internet. *IEEE/ACM Trans. Netw.* 10, 4 (2002), 455–465.
- [102] ROETTIGERS, J. Ohio university blocks netflix, backpedals. <http://gigaom.com/video/ohio-university-blocks-netflix> (March 2011).
- [103] ROSSI, D., MELLIA, M., AND MEO, M. Evidences behind skype outage. In *IEEE ICC* (Piscataway, NJ, USA, 2009).
- [104] ROSSI, D., MELLIA, M., AND MEO, M. Understanding skype signaling. *Comput. Netw.* 53 (February 2009), 130–140.
- [105] ROUGHAN, M., GREENBERG, A., KALMANEK, C., RUMSEWICZ, M., YATES, J., AND ZHANG, Y. Experience in measuring backbone traffic variability: models, metrics, measurements and meaning. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement* (New York, NY, USA, 2002), IMW '02, ACM, pp. 91–92.

- [106] SEN, S., AND WANG, J. Analyzing peer-to-peer traffic across large networks. *Networking, IEEE/ACM Transactions on* 12, 2 (april 2004), 219 – 232.
- [107] SINGEL, R. Most content online is now paid for, thanks to netflix. *Wired* (May 2011).
- [108] STANOJEVIC, R., LAOUTARIS, N., AND RODRIGUEZ, P. On economic heavy hitters: shapley value analysis of 95th-percentile pricing. In *Proceedings of the 10th annual conference on Internet measurement* (New York, NY, USA, 2010), IMC '10, ACM, pp. 75–80.
- [109] VARKI, E., AND VILLA, A. H. Impact of round world on etransfer of big data. In *Submitted to 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (October 2012).
- [110] VAZHKUDAI, S. Enabling the co-allocation of grid data transfers. In *GRID* (2003), p. 44.
- [111] VAZHKUDAI, S. Distributed downloads of bulk, replicated grid data. In *Journal of Grid Computing* (March 2004), vol. 2, pp. 31–42.
- [112] VAZHKUDAI, S., AND SCHOPF, J. M. Predicting sporadic grid data transfers. In *HPDC '02: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing* (Washington, DC, USA, 2002), IEEE Computer Society, p. 188.
- [113] VAZHKUDAI, S., TUECKE, S., AND FOSTER, I. Replica selection in the globus data grid. In *CCGRID* (May 2001).
- [114] VILLA, A. H., AND VARKI, E. Replica traffic manager for data grids. In *20th International Conference on Parallel and Distributed Computing Systems* (September 2007).
- [115] VILLA, A. H., AND VARKI, E. Co-allocation in data grids: A global, multi-user perspective. *Advances in Grid and Pervasive Computing* (2008), 152–165.
- [116] VILLA, A. H., AND VARKI, E. It takes know-how to retrieve large files over public networks. In *1st International Conference on Advanced Computing and Communications (ACC)* (2010).
- [117] VILLA, A. H., AND VARKI, E. The feasibility of moving terabyte files between campus and cloud. In *23rd IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS)* (December 2011).
- [118] VILLA, A. H., AND VARKI, E. Automating large file transfers. In *13th International Conference on Internet Computing (ICOMP'12)* (July 2012).
- [119] VILLA, A. H., AND VARKI, E. Characterization of a campus internet workload. In *27th International Conference on Computers and their Applications (CATA)* (March 2012).



- [120] VILLA, A. H., AND VARKI, E. Tortoise vs. hare: a case for slow and steady retrieval of large files. In *2nd International Conference on Advanced Computing and Communications (ACC 2012)* (June 2012).
- [121] WANG, C.-M., HSU, C.-C., CHEN, H.-M., AND WU, J.-J. Efficient multi-source data transfer in data grids. In *CCGRID* (2006), pp. 421–424.
- [122] WANG, H., XIE, H., QIU, L., SILBERSCHATZ, A., AND YANG, Y. Optimal isp subscription for internet multihoming: algorithm design and implication analysis. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE* (march 2005), vol. 4, pp. 2360 – 2371 vol. 4.
- [123] WANG, R. Y., SOBTI, S., GARG, N., ZISKIND, E., LAI, J., AND KRISHNAMURTHY, A. Turning the postal system into a generic digital communication mechanism. *SIGCOMM Comput. Commun. Rev.* 34, 4 (Aug. 2004), 159–166.
- [124] WOLSKI, R., SPRING, N. T., AND HAYES, J. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Gener. Comput. Syst.* 15, 5-6 (1999), 757–768.
- [125] XIAO QIN, H. J. *Data Grids: Supporting Data-Intensive Applications in Wide-Area Networks*. 2006, pp. 481–494.
- [126] XIE, H., YANG, Y. R., KRISHNAMURTHY, A., LIU, Y. G., AND SILBERSCHATZ, A. P4p: provider portal for applications. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication* (New York, NY, USA, 2008), SIGCOMM '08, ACM, pp. 351–362.
- [127] YANG, C.-T., CHI, Y.-C., AND FU, C.-P. Redundant parallel file transfer with anticipative adjustment mechanism in data grids. In *Journal of Information Technology and Applications* (March 2007), vol. Vol. 1, pp. 305–313.
- [128] YANG, C.-T., YANG, I.-H., CHEN, C.-H., AND WANG, S.-Y. Implementation of a dynamic adjustment mechanism with efficient replica selection in data grid environments. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing* (2006), pp. 797–804.
- [129] YANG, C.-T., YANG, I.-H., LI, K.-C., AND WANG, S.-Y. Improvements on dynamic adjustment mechanism in co-allocation data grid environments. *The Journal of Supercomputing* 40, 3 (2007), 269–280.
- [130] YARROW, J. Netflix is eating up more of north america’s bandwidth than any other company. *Business Insider* (May 2011).
- [131] ZHOU, X., KIM, E., KIM, J. W., AND YEOM, H. Y. Recon: A fast and reliable replica retrieval service for the data grid. In *CCGRID* (2006), pp. 446–453.

- [132] ZINK, M., SUH, K., GU, Y., AND KUROSE, J. Characteristics of youtube network traffic at a campus network - measurements, models, and implications. *Comput. Netw.* 53 (March 2009), 501–514.
- [133] ZISSIMOS, A., DOKA, K., CHAZAPIS, A., AND KOZIRIS, N. Gridtorrent: Optimizing data transfers in the grid with collaborative sharing. In *Proceedings of the Panhellenic Conference on Informatics (PCI)* (2007).