MODELING AND ANALYSIS OF IP STORAGE PROTOCOLS FOR TIME CRITICAL APPLICATIONS

A Thesis by

Chandu Nuthakki

Bachelor of Engineering, CBIT, O.U, 2003

Submitted to Department of Electrical and Computer Engineering
and faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Master of Science

July 2007

MODELING AND ANALYSIS OF IP STORAGE PROTOCOLS FOR TIME CRITICAL APPLICATIONS

I have examined the final copy of this thesis for form and content, and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Electrical and Computer Engineering.

_____
Ravi Pendse, Committee Chair

We have read this thesis and recommend its acceptance:

_____
Kamesh Namuduri, Committee Member

_____
Krishna K. Krishnan, Committee Member

DEDICATION

To my parents, my sister and my friends who have supported me throughout my life.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Ravi Pendse, for giving me an opportunity to do my Master's thesis under him in the field of storage networking. I would also like to thank my committee for taking the time to work with me in this endeavor.

I am thankful to Mr. Amarnath Jasti and all my friends at the Advanced Networking Research Center (ANRC) at Wichita State University for their help during this time. Finally, I am forever indebted to my parents and family for supporting and encouraging me throughout my life to achieve higher goals.

ABSTRACT

In the current storage era, Fibre Channel (FC) protocol is used for high performance and reliability, providing different levels of service with 1Gbps to 4Gbps physical interfaces. FC transports Small Computer System Interface (SCSI) data, between storage devices. FC spans from 500 meters to 10 Kilometers [27, 28] using multimode, single mode fibers respectively, limiting the storage to a site or between two sites. In order to overcome this distance limitation and implementation costs, Fibre Channel over Internet Protocol (FCIP) and Internet Fiber Channel Protocol (iFCP) were introduced [30, 31]. FCIP tunnels the fibre channel frames over Internet between two fiber switches, with in the same network. iFCP is an IP Based gateway to gateway protocol, which interconnects different FC-SANs. Again, all these protocols are involved with fiber channels which increase the cost. iSCSI is a new IETF standardized protocol [26], which transports SCSI data over Internet. Advancements in Ethernet technologies from 1Gbps to 10 Gbps make iSCSI deployment to yield better results in terms of performance and can be very cost effective compared to FCIP and iFCP.

Some of the widely used storage applications in the industry are archiving and mirroring, these applications are used for backup/recovery process in IT industry. Archiving is a process in which data is written to portable media such as optical disks or magnetic tapes. Mirroring is a process of data replicated on the remote disk. Mirroring can be done in two ways, synchronous and asynchronous. In synchronous mirroring, when ever there is an update with data, it is written to both local and remote disk at the same time. In asynchronous mirroring, data is updated periodically irrespective of the actual update. If iSCSI is used for remote mirroring, end users need to ensure the performance of iSCSI should meet the requirements of the application. Most of the studies proved deficient in considering some or the other aspects. In this research work,

the author presents the modeling and analysis of iSCSI between two SAN Islands considering iSCSI level errors, which will enable the IT industry to use this model for their analyze before they actually deploy. Throughout the analysis, the author employs asynchronous mirroring between the SAN Islands. iSCSI level errors need to be considered when the SCSI data is on the Internet, which will seriously effect the performance of the application in real time. The prototype was analyzed using TCP/IP and UDP traffic with both dedicated links and Internet links. When iSCSI is used to interconnect different SAN islands, one should ensure its performance to meet the application requirements. The bandwidth management for time critical applications like synchronous and asynchronous mirroring is essential while analyzing performance on IP networks, for this the key operations of iSCSI like iSCSI read, iSCSI write were modeled. Also, the throughput under realistic traffic conditions varying different parameters like network types, round trip time, bandwidth and distance was modeled. Security is another important attribute which should be taken in to account while IP networks are involved. It is essential to consider security of data while block level data is being transferred between two storage islands using iSCSI. The errors processed by this protocol on the IP network were considered and suitable iSCSI error recovery procedures were analyzed.

TABLE OF CONTENTS

Chapter Page

TABLE OF CONTENTS (Cont.)

Chapter Page

TABLE OF CONTENTS (Cont.)

Chapter Page

LIST OF FIGURES

Figure Page

LIST OF TABLES

Table Page

LIST OF ACRONYMS

CDB	Command Descriptor Block
CHAP	Challenge Handshake Authentication Protocol
CID	Connection Identifier
CPU	Central Processing Unit
CRC	Cyclical Redundancy Check
CSG	Current Stage
EUI	Enterprise Unique Identifier
ExpCmdSN	Expected Command Sequence Number
ExpDataSN	Expected Data Sequence Number
ExpStatusSN	Expected Status Sequence Number
DAS	Direct Attached Storage
DataSN	Data Sequence Number
DNS	Domain Name Service
FC	Fedora Core
GUI	Graphical User Interface
HBA	Host Bus Adapter
IET	iSCSI Enterprise Target
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
I/O	Input/Output
IOPS	Input/Output Per Second
IP	Internet Protocol

LIST OF ACRONYMS (Cont.)

IPSEC	Internet Protocol Security
IQN	iSCSI qualified name
ISID	Initiator Session Identification
IT	Information Technology
iSCSI	Internet Small Computer System Interface
iqn	iSCSI Qualified Name
LAN	Local Area Network
LBA	Logical Block Address
LONP	login operational negotiation phase
MaxCmdSN	Maximum Command Sequence Number
MSB	Most Significant Bit
NAS	Network Attached Storage
NOP	No-Operation
NSG	Next stage
PDU	Protocol Data Units
R2T	Ready to Transmit
R2TSN	Request to Send Sequence Number
SAN	Storage Area Network
SCSI	Small Computer System Interface
SAP	Security Authentication Processes
SNACK	Selective Negative Acknowledgement
SPKM	Simple Public Key Mechanism

LIST OF ACRONYMS (Cont.)

SRP	Secure Remote Password
StatSN	Status Sequence Number
TCP/IP	Transmission Control Protocol/Internet Protocol
TPGT	Target Portal Group Tag
TSIH	Target Session Identification handle
ULP	Upper Layer Protocol

CHAPTER 1

INTRODUCTION

1.1 Introduction to Storage Technologies

The concept of storage can be broadly classified into two categories [19, 20, 21, 22], direct attached storage and network attached storage. Direct Attached Storage is a storage which is connected to a workstation and is independent of any other device connected to the network. The other category is storage over the IP network which includes Network Attached Storage (NAS) and Storage Area Networks (SAN). The main differences between NAS and SAN are (i) NAS has all the devices including the storage in the same network (LAN), whereas SAN has a segregated network for storage in which all the storage devices communicate effectively without congesting traffic on the LAN. (ii) NAS interacts with file level I/O (Input/Output) and SAN does with Block level I/O's over network. Though SAN gives the best performance over NAS [16], it is geographical limited. In order to overcome this limitation and span for longer distances, they were incorporated with IP networks. Over the past years, various technologies were developed to incorporate storage over the IP networks. Some of these technologies include Fibre Channel Over IP (FCIP), Internet Fibre Channel Protocol (iFCP) and Internet Small Computer System Interface (iSCSI) [15]. Each of the technology is explained in detail in the following discussion.

1.1.1 Direct Attached Storage (DAS)

DAS is most widely used data storage technology from home PC's to the enterprise servers at office [19, 22]. In this technology, the storage is directly connected to the host computer using one of these technologies/buses like Advanced Technology Attachment (ATA),

Serial Advanced Technology Attachment (SATA), SCSI bus or Fibre Channels (FC). The following figure1.1 illustrates the DAS in brief.

1.1.1.1 Advantages

1. Ease of deployment, easy to understand and low establishment cost.
2. Can be deployed anywhere from home to small office, where high performance, capacity and high-availability along with sharing are not key requirements.

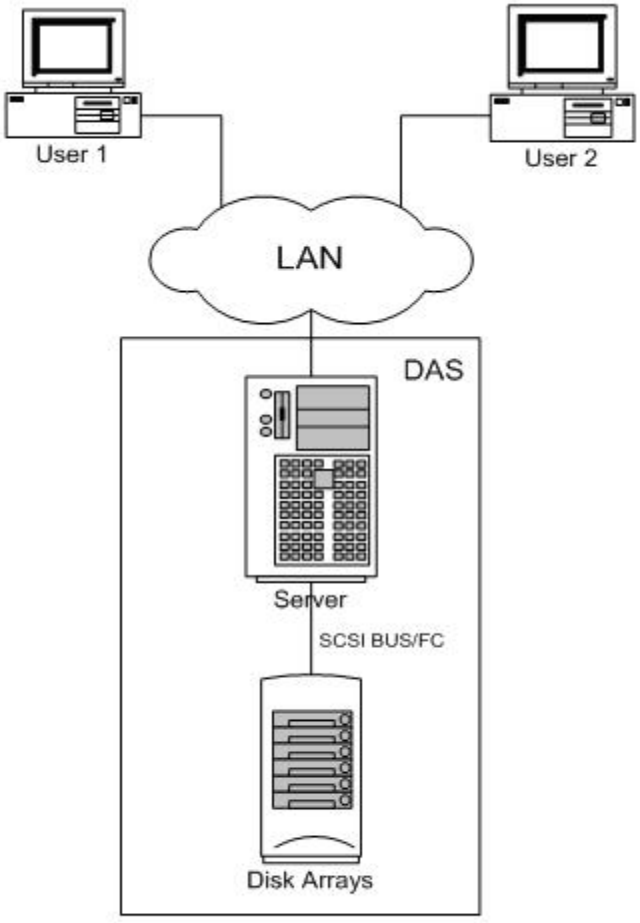

Figure 1.1 Direct Attached Storage

1.1.1.2 Disadvantages

1. *Scalability:* As industry grows, storage capacity needs to be increased significantly to meet the requirements. In DAS scalability is limited to the storage capacity of a server i.e. a server with SCSI bus can hold a maximum of fifteen hard drives.
2. *Reliability:* Down time of storage is very important in industry, which causes loss of business. In DAS the down time may be due to failure of a storage drive or additional storage drives need to be added, which causes unavailability of data.
3. *Performance:* The performance of the DAS is purely dependent on the host system processing speed.
4. *Efficiency:* High level of efficiency can't be achieved as the resources are limited to that host computer, to which the storage is connected. This storage cannot be shared with other computers.
5. *Maintenance:* Maintenance of DAS is a tedious process, as backup/recovery process of data should be done with every host computer, which takes lot of time and human intervention.

1.1.2 Network attached storage

NAS was introduced to overcome the disadvantages of DAS. In NAS the storage is centralized and connected to the computer network (LAN), so that every one on the network can access the storage through Network File System (NFS) and Common Internet File System protocol (CIFS). NAS performs file level I/O's on the network between storage and host computer to share data. As storage should perform file level I/O on the network to share the data, it should be running a customized operating system, which implements the functionality of NFS

and CIFS. On the other hand, host machine should support the file-level I/O's redirection on to the network. The following figure1.2 illustrates the NAS in brief.

1.1.2.1 Advantages

1. Easy to implement and use.
2. Supports heterogeneous platforms. i.e. storage can be accessed from the hosts running different operating systems like Linux and Windows.
3. Can use the present existing infrastructure to deploy and no additional cost involved in establishing infrastructure Ex: Ethernet switches etc.
4. Additional storage can be added to the present storage without bringing the storage down.

1.1.2.2 Disadvantages

1. While running data backup and mirroring applications, total network resources are utilized by them, which make other applications and users to starve for the network resources.
2. NFS and CIFS will add additional overhead on the network to transmit the data between storage and host computer.

1.1.3 Storage Area Network

SAN is a dedicated network for connecting storage and servers with either Fibre Fabric or Ethernet. In fibre fabric, the storage and servers are connected through fibre fabric and fibre switches while in Ethernet they are connected through Ethernet and high speed LAN switches, in NAS performs file-level I/O's on the network, where as SAN performs Block level I/O's on the network. The following figure1.3 illustrates the SAN in brief.

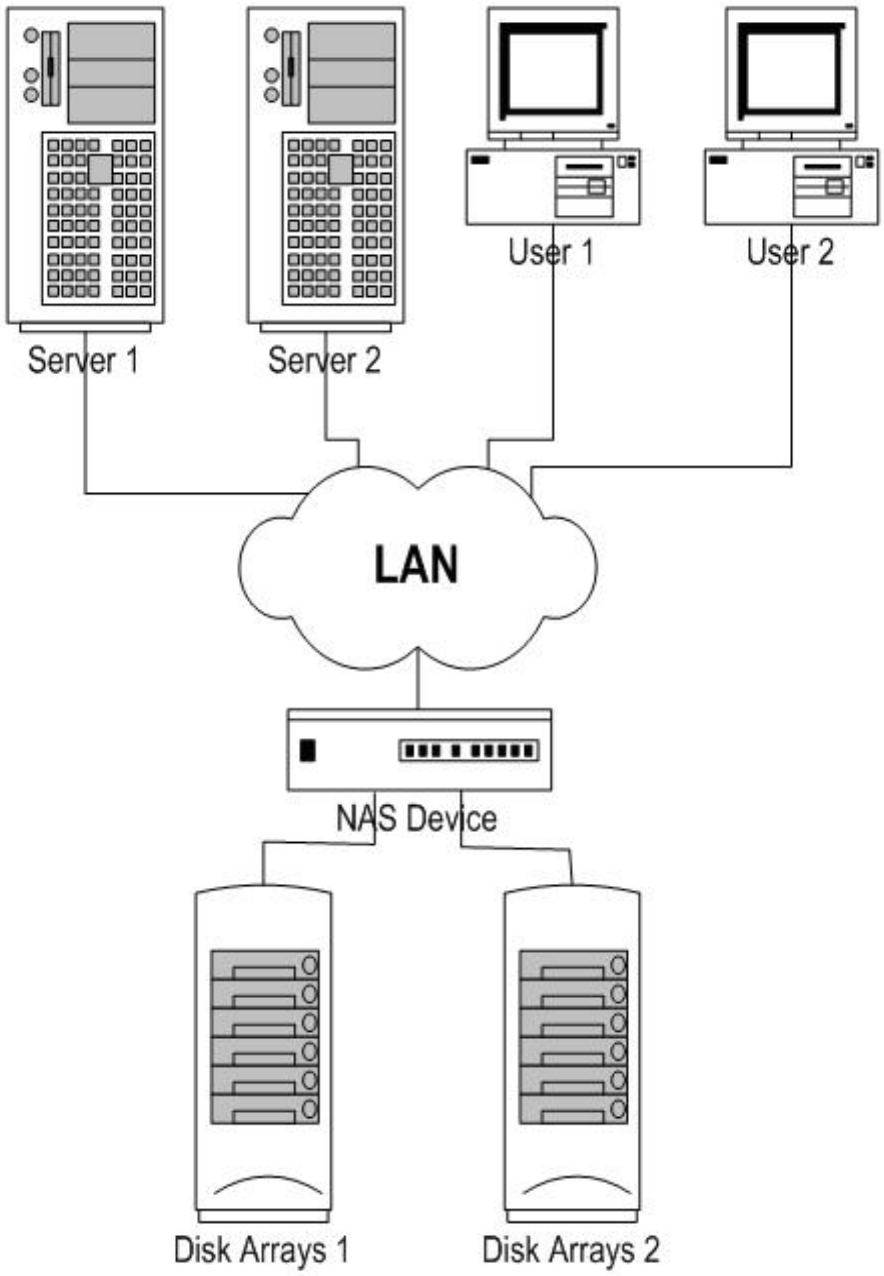

Figure1.2 Network Attached Storage

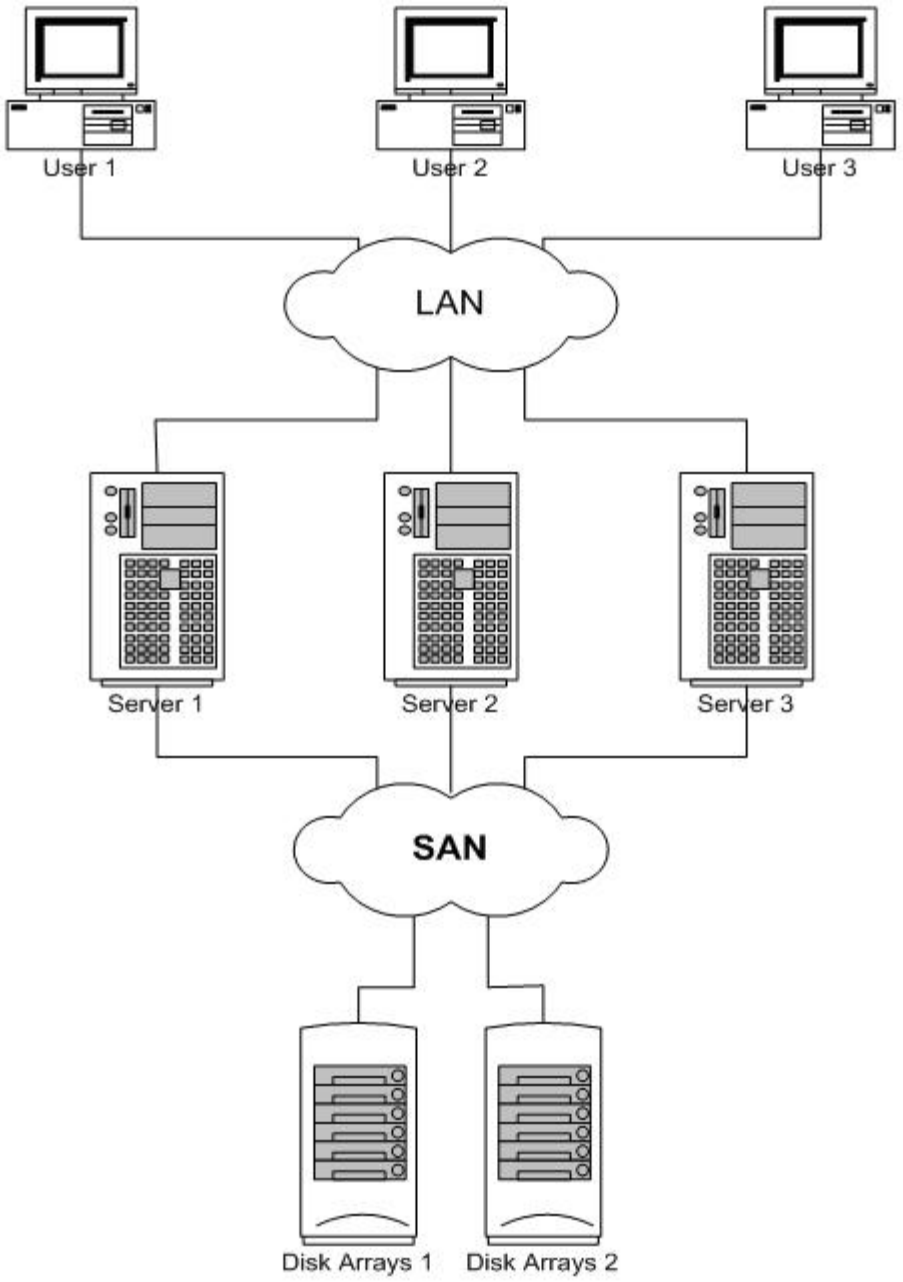

Figure1.3 Storage Area Network

1.1.3.1 Advantages

1. Storage traffic is totally isolated from LAN traffic, which completely frees the LAN and provides more bandwidth to users on the LAN [19].

2. Provides high scalability, performance and security [22].
3. Block level I/O will eliminate additional overhead on the network and improves the performance of the system [19].

1.1.3.2 Disadvantages

1. Establishing infrastructure for SAN with fibre fabric is highly expensive.
2. Maintenance of fibre channels needs skilled people.

1.2 Storage over IP

In NAS and SAN, the storage is limited to a building or campus. What if we need to connect the storage which is located totally in a different geographical location? IP storage solves this problem by connecting the remote storage to the local servers over IP. Storage over IP will make use of present existing IP infrastructure to interconnect storage and servers, which will minimize the deployment cost. No additional skilled people are required to maintain the network, which reduces the maintenance cost. IP storage can be defined using different technologies like Fibre Channel over Internet Protocol (FCIP), Internet Fibre Channel Protocol (iFCP) and Small Computer System Interface over Internet Protocol (iSCSI).

1.2.1 FCIP

FCIP encapsulates fibre channel frames over TCP/IP. FCIP devices are used to interconnect SAN islands over IP. FCIP tunnel runs between two FCIP devices over TCP/IP. Encapsulation and decapsulation of FCIP packets are carried out at the end FCIP devices. In this way different SAN islands are interconnected virtually via FCIP tunnels over IP irrespective of their geographical locations.

1.2.2 iFCP

iFCP is a gateway to gateway protocol for providing the fibre channel fabric services to end fibre channel devices over TCP/IP network. End devices like host computer and storage are connected to the iFCP gateways and operate as if they are connected by a virtual fibre fabric. Ethernet IP networks are used to interconnect iFCP gateways.

1.2.3 iSCSI

iSCSI transports SCSI commands over TCP/IP between iSCSI initiator and iSCSI target. iSCSI simply encapsulates the SCSI commands over TCP/IP. The concept of initiator and target forms an analogy of a client and server architecture in which initiator can be considered equivalent to the client and the target to a server. As all hosts and storage support Ethernet/IP and iSCSI stack, they can be directly connected to the network to exchange and share data. iSCSI uses the present existent IP infrastructure like LAN switch etc, which reduces the cost of infrastructure and no additional skilled people need to maintain it, so it also reduces the maintenance cost.

1.3 Organization of Thesis

The organization of remaining part of this report is as follows: chapter 2 provides the introduction to SCSI, chapter 3 provides the introduction to iSCSI, chapter 4 provides the literature review, chapter 5 provides the modeling of iSCSI Read, Write and Throughput, chapter 6 provides the test setup and procedures, chapter 7 provides results and analysis and chapter 8 provides the conclusion and future work.

CHAPTER 2
INTRODUCTION TO SCSI

Small Computer System Interface (SCSI) is an intelligent bus interface, which defines standards for hardware specifications, like cabling length, signaling and bus types. It also defines standards for commands which enable to use the bus in an optimized way. According to IEEE draft [26] SCSI is defined as "A local I/O (Input/Output) bus that can be operated over a wide range of data rates." The main criterion behind the design of this interface is to provide device independence to the host computer. This means, devices like tape drives, optical media drives, disk drives, CD ROM drives etc. could be added to host computer without any additional hardware or software installed. This is a high speed interface, which interconnects computer and its peripherals. This interface is mostly used in storage industry for higher data transfers between computers, controllers and disk drives.

2.1 Initiator - Target

There are two types of devices connecting the SCSI bus, they are Initiator and Target. SCSI Initiator initiates I/O (Input/Output) command requests to be carried out. Target responds to the requested I/O commands or in other words it processes the requested commands. It can be observed as a Client-Server analogy, where the initiator is the client, making request for different commands to the server and Target is the server that executes different commands requested by client (Initiator). Each SCSI target is again sub-divided in to Logical Unit Numbers (LUN), and the number of LUN's per target is defined according to the SCSI standard. In SCSI-2 standard it can support up to a maximum of eight LUNs i.e. LUN-0 to LUN-7. SCSI-3 has a 64-bit field for LUN representation which accommodates 2^{64} LUN's. Each LUN is again sub-divided in to

Logical Block Address (LBA), which hides the physical details of the device like track, cylinder, heads and sectors.

Initiator and target can be interconnected in three ways as listed below and shown in below figure 2.1.

a. Single Initiator and Single Target.

b. Single Initiator and Multiple Targets.

c. Multiple Initiator and Multiple Targets.

Figure 2.1a Single Initiator and Single Target

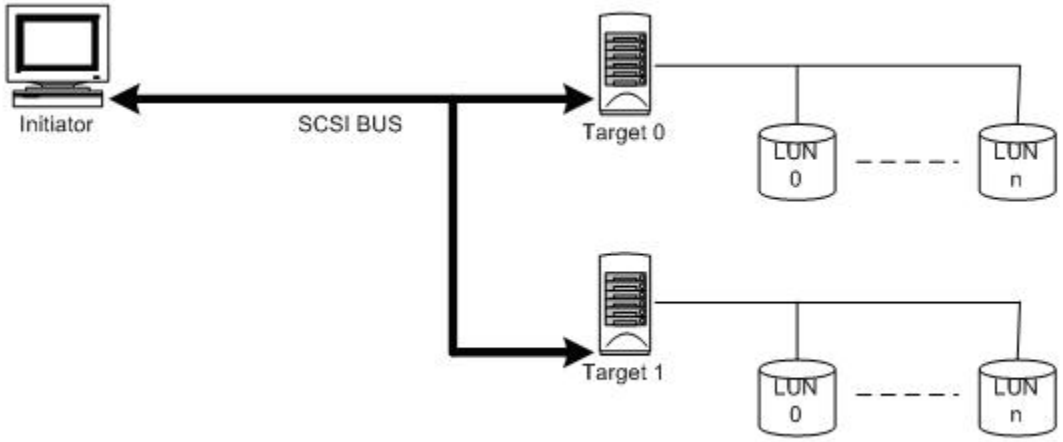

Figure 2.1b Single Initiator and Multiple Targets

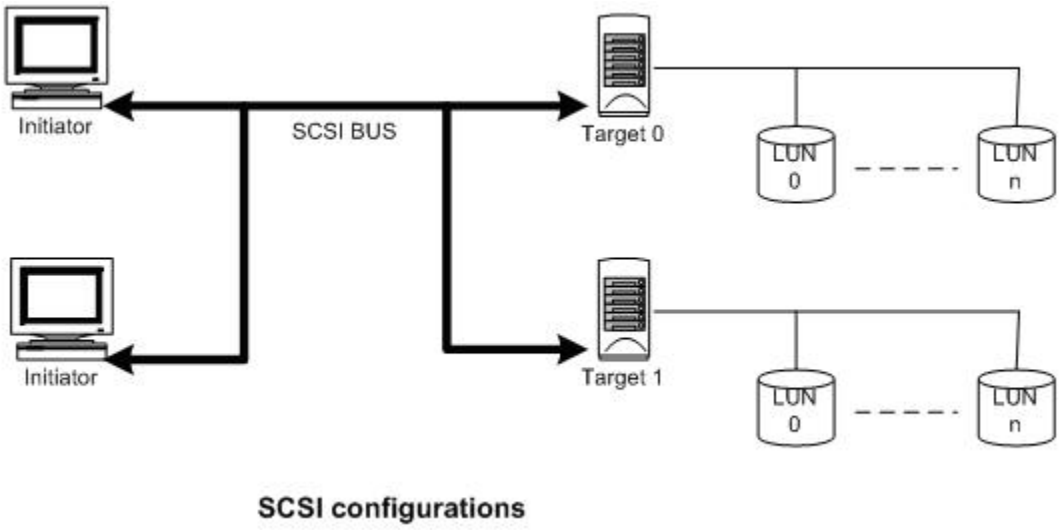

Figure 2.1c Multiple Initiators and Multiple Targets

2.2 SCSI device addressing

SCSI Identifier (ID) is used to address each device uniquely on the SCSI bus. The number of devices that can be connected to the bus depends on the SCSI bus standard [3]. Initially, SCSI began with eight data lines and SCSI ID's couldn't be encoded. It could support a maximum of 8 devices. The priority of the SCSI device depends on the SCSI ID's. The priority increases with the SCSI ID in SCSI-1 and SCSI-2 standards [17]. The mechanism varies slightly with SCSI-3 standard. The priorities are listed for different standards in the following table 2.1.

2.3 Evolution of SCSI

2.3.1 SCSI-1

SCSI-1 defines the fundamental features and transport modes of SCSI. The devices working under this standard use eight bit data bus, which is also called Narrow SCSI. This standard can address a maximum of eight devices on the bus and has data transfer rates of 5 Mhz, which gives a maximum throughput of 5MBps. This standard supports only single-ended transmission with passive termination. It is a basic standard which is obsolete today.

SCSI VERSIONS	SCSI ADDRESS	SCSI ID				PRIORITY	
		D31 D24	D23 D16	D15 D8	D7 D0		
SCSI 2&3	7	- - - - - - - -	- - - - - - - -	- - - - - - - -	1 - - - - - - -	1	Highest Priority
	6	- - - - - - - -	- - - - - - - -	- - - - - - - -	- 1 - - - - - -	2	
	5	- - - - - - - -	- - - - - - - -	- - - - - - - -	- - 1 - - - - -	3	
	4	- - - - - - - -	- - - - - - - -	- - - - - - - -	- - - 1 - - - -	4	
	3	- - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - 1 - - -	5	
	2	- - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - 1 - -	6	
	1	- - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - 1 -	7	
	0	- - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - 1	8	
SCSI 3 ONLY	15	- - - - - - - -	- - - - - - - -	1 - - - - - - -	- - - - - - - -	9	
	14	- - - - - - - -	- - - - - - - -	- 1 - - - - - -	- - - - - - - -	10	
	13	- - - - - - - -	- - - - - - - -	- - 1 - - - - -	- - - - - - - -	11	
	12	- - - - - - - -	- - - - - - - -	- - - 1 - - - -	- - - - - - - -	12	
	11	- - - - - - - -	- - - - - - - -	- - - - 1 - - -	- - - - - - - -	13	
	10	- - - - - - - -	- - - - - - - -	- - - - - 1 - -	- - - - - - - -	14	
	9	- - - - - - - -	- - - - - - - -	- - - - - - 1 -	- - - - - - - -	15	
	8	- - - - - - - -	- - - - - - - -	- - - - - - - 1	- - - - - - - -	16	
	23	- - - - - - - -	1 - - - - - - -	- - - - - - - -	- - - - - - - -	17	
	22	- - - - - - - -	- 1 - - - - - -	- - - - - - - -	- - - - - - - -	18	
	21	- - - - - - - -	- - 1 - - - - -	- - - - - - - -	- - - - - - - -	19	
	20	- - - - - - - -	- - - 1 - - - -	- - - - - - - -	- - - - - - - -	20	
	19	- - - - - - - -	- - - - 1 - - -	- - - - - - - -	- - - - - - - -	21	
	18	- - - - - - - -	- - - - - 1 - -	- - - - - - - -	- - - - - - - -	22	
	17	- - - - - - - -	- - - - - - 1 -	- - - - - - - -	- - - - - - - -	23	
	16	- - - - - - - -	- - - - - - - 1	- - - - - - - -	- - - - - - - -	24	
	31	1 - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -	25	
	30	- 1 - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -	26	
	29	- - 1 - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -	27	
	28	- - - 1 - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -	28	
	27	- - - - 1 - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -	29	
	26	- - - - - 1 - -	- - - - - - - -	- - - - - - - -	- - - - - - - -	30	
	25	- - - - - - 1 -	- - - - - - - -	- - - - - - - -	- - - - - - - -	31	
	24	- - - - - - - 1	- - - - - - - -	- - - - - - - -	- - - - - - - -	32	Lowest Priority

Table 2.1 SCSI address, ID's and arbitration priorities

2.3.2 SCSI-2

The basic goal behind SCSI-2 design was to improve the transfer rates, performance, reliability, add features to the interface and standardize the SCSI commands. It uses 16 bit or 32 bit data bus, which is called as Wide SCSI. In this standard, data transfer rates have been improved to 10 MHz, which is known as Fast SCSI. This standard improves the throughput to 20MBps and 40MBps. It improves the SCSI addressing capability to 16 to 32 devices depending

on the data bus length. It also enhances the transport method with active termination, which uses differential signaling for improving the cabling lengths.

2.3.3 SCSI-3

SCSI-3 is defined to improve the standards of SCSI-2. It has technically evolved to improve different parameters in previous versions of SCSI. SCSI-3 is of two kinds, SCSI-3 Parallel interface and SCSI-3 serial interface. SCSI-3 Parallel interface is an extension to SCSI-2 and it is also called as SCSI Parallel Interface (SPI). In this standard, the transfer rates have been improved to 20 MHz which has been denoted as FAST/20 or Ultra SCSI, 40 MHz as FAST/40 or Ultra-2 SCSI and 80 MHz as FAST/80 or Ultra-3 SCSI. This improvement in speeds resulted in better throughput which is way beyond the SCSI-2 standards. SCSI-3 Serial standard was designed to give high performance, provide network capability, span for lager distances from meters to kilometers, provide extended addressability, improve data rates and provide high reliability.

2.4 SCSI Bus Signaling

SCSI bus signaling can be classified in to two categories i.e. Control signals and Data signals. Control Signals are used for controlling different activities on the bus i.e. SCSI phase control, hand shaking etc. Data signals are used for transmission of data and messages between Initiator and Target. These signals are also used for gaining bus control in arbitration phase and for device selection in the selection phase.

SCSI signals are defined as follows.

i. **BSY (BUSY):** This signal indicates that the bus is being used. It is used by both initiator and the target
ii. **ATN (Attention):** This signal is used by the initiator for signaling target to change its phase to message out and obtain message bytes.

iii. **REQ (Request):** This signal is used by the target to indicate that a byte is ready to be transferred.

iv. **ACK (Acknowledge):** This signal is used by initiator to indicate acknowledgement for transferred a byte.

v. **RST (Reset):** This signal is used by both initiator and target to indicate reset condition.

vi. **SEL (Select):** This signal is used by initiator to select a target and it is used by target to reselect an initiator.

vii. **MSG (Message):** This signal is used to indicate message phase and it is driven by target.

viii. **C/D (Command/Data):** This signal is driven by target, which indicates command or data information on the data bus.

ix. **I/O (Input/Output):** This signal is driven by the target to indicate the direction of data on the data bus with respect to initiator.

x. **DB (7-0,P) (Data Bus):** The data bus can carry eight bits, with additional parity bit. Most significant bit has the highest priority and least significant bit has the lowest priority during arbitration.

2.5 Phase control signals

MSG, C/O and I/O signals are called phase control signals. Depending on the state of the signals, the SCSI phases are varied. The following table 2.2 will explain it in brief.

MSG (Message)	*C/D (Command/Data)*	*I/O (Input/Output)*	*Phase Name*
0	*0*	*0*	*Data Out*
0	*0*	*1*	*Data In*
0	*1*	*0*	*Command*
0	*1*	*1*	*Status*
1	*0*	*0*	*Not Defined*
1	*0*	*1*	*Not Defined*
1	*1*	*0*	*Message Out*
1	*1*	*1*	*Message In*

Table 2.2 Phase control of SCSI

2.6 SCSI BUS Phases

SCSI bus phases can be classified into eight different phases:

1. **BUS FREE Phase:** This phase indicates that the SCSI BUS is available for connections and currently no I/O processes are running on the bus. This condition can be detected by the device on the bus by two signals i.e. SEL and BSY in false state.
2. **Arbitration Phase:** SCSI devices need to gain control of the bus before starting an I/O process or to resume the I/O process. Gaining bus control involves the following:
   a. SCSI device needs to wait for 'BUS FREE' state to start arbitration.
   b. Ones the SCSI bus is free, device start arbitration by asserting the BSY signal and its SCSI ID.
   c. After asserting the BSY and SCSI ID signal, the device waits for arbitration delay amount of time.

d. If more than one device arbitrates for the bus, the device with the highest priority will win the arbitration.

3. **Selection Phase:** In this phase, Initiator will select the target for I/O process. Initiator needs to select the target before it starts the I/O process. The SCSI device which wins the arbitration will assert both BSY and SEL signals. This process indicates a selection phase to rest of the devices. The SCSI device will select the Target by asserting the Target ID logically 'OR'ed with Initiator ID. The Target will detect its selection when the SEL signal and its ID bits are asserted and while BSY and other I/O's are in a false state.
4. **Reselection:** Reselection is done by targets to select the initiators. To perform certain I/O operations, Target may disconnect from Initiator to make the bus free so that it can be used by other devices. Once the Target is ready to perform I/O operations, it will reselect the Initiator.
5. **Command Phase:** In this phase, the Target requests commands from Initiator.
6. **Data Phase:** In data phase, the data is transferred between Initiator and Target. Data from Initiator to Target is known as **DATA OUT** and data from Target to Initiator is called as **DATA IN**. **DATA IN** and **DATA OUT** are referred with respect to the Initiator.
7. **Status Phase:** In this phase, Target provides the status to Initiator as soon as the commands issued by the Initiator are executed.
8. **Message Phase:** In message phase, protocol level messages are exchanged between Initiator and Target to negotiate various SCSI parameters, which need to be followed in the later phases. Messages from Initiator to Target are called as **MESSAGE-OUT** and messages from Target to initiator are called as **MESSAGE-IN**.

2.7 Flow diagram of SCSI protocol

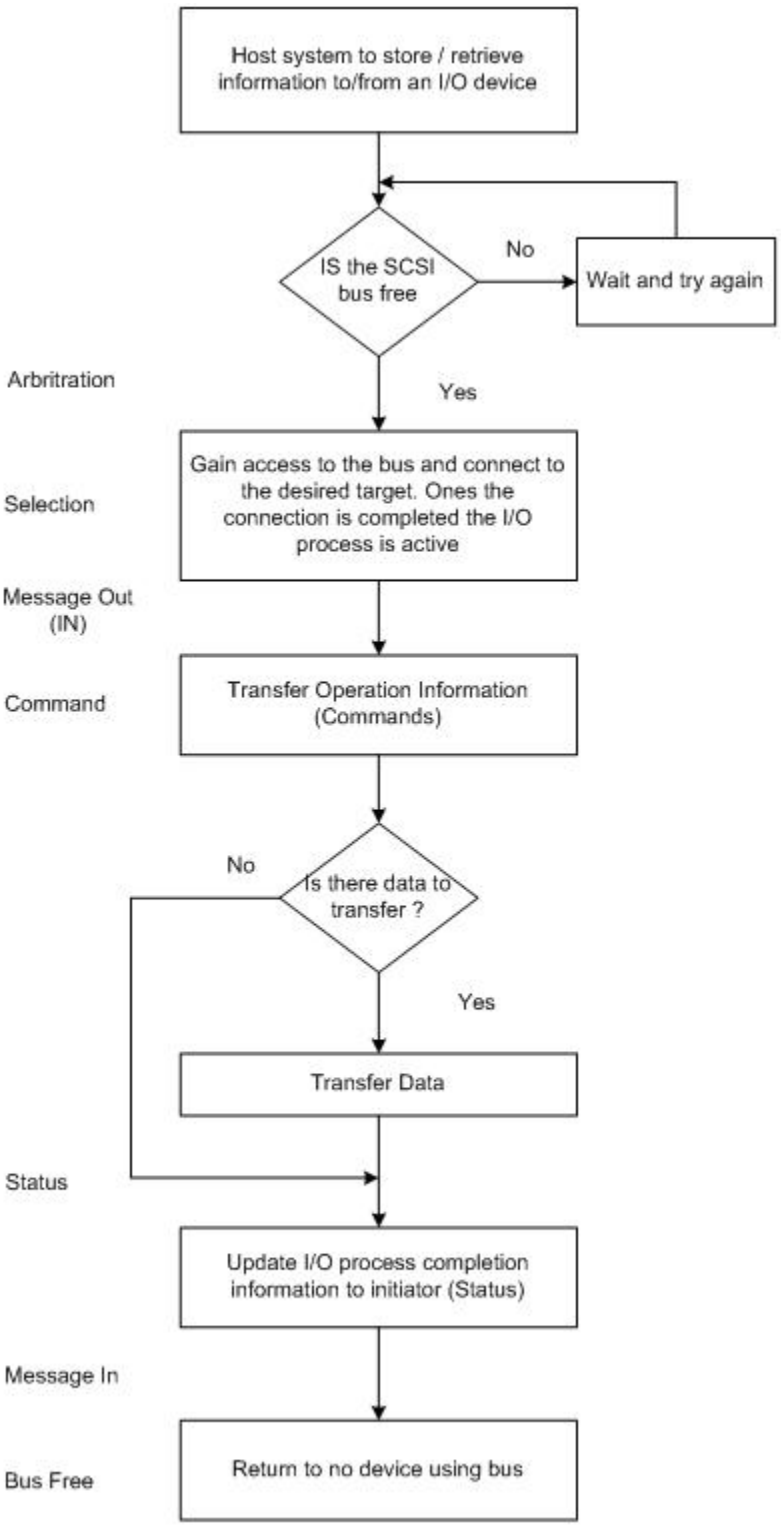

Figure 2.2 Flow of SCSI protocol

CHAPTER 3
INTRODUCTION TO iSCSI

iSCSI stands for internet over SCSI [1,13, 14], which encapsulates SCSI commands over TCP/IP. As in SCSI, iSCSI provides communication between initiator and target over IP. It can be rather interpreted as a client/server program, where client is the initiator and server is the target. Initiator issues the commands, where as Target executes the commands and gives the response to initiator. As in the client server topology, a single server can be connected to many clients. iSCSI uses the benefits of TCP protocol, as it works on top of it with different physical mediums. TCP is a well tested and standardized protocol, which is in practice for long time. As some of the key features of TCP are, it is connection oriented protocol, which establishes a connection between two endpoints before sending data, it provides reliable data transfer between two end points, as all the sent data is acknowledged, and it also provides in order delivery of data and congestion control. iSCSI using TCP/IP is least bothered to implement the features of TCP/IP, rather use them for the functionality. Figure 3.1 will illustrate the protocol model of iSCSI. The following sessions will discuss about the iSCSI in depth for analysis and modeling.

3.1 Naming and Addressing

iSCSI nodes (initiator and target) are named using two different standards [25], they are

- eui (enterprise unique identifier)
- iqn (iSCSI qualified names)

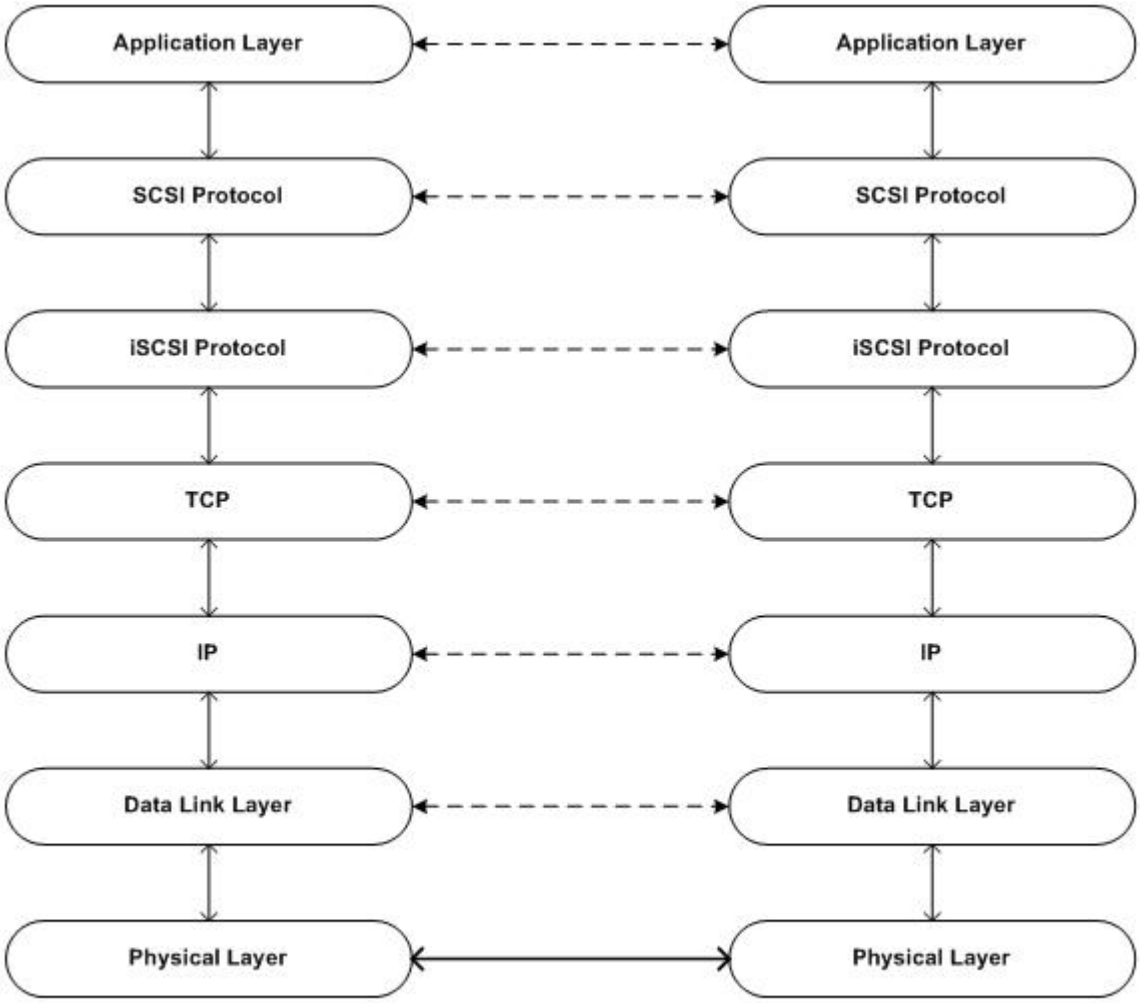

Figure 3.1 iSCSI Protocol Model

3.1.1 Enterprise Unique Identifier (EUI)

EUI is a string constructed from Institute of Electrical and Electronics Engineers (IEEE) EUI format (EUI-64). EUI is a 64 bit value used to identify iSCSI nodes uniquely in the world. It is a combination of two fields, company ID and manufacturer ID. The most significant 24 bits is company ID and following 40 bits are manufacturer chosen ID. IEEE assigns the company ID for the manufacturers.

Example: eui.wush98675497ocks

In the given example the first 24 bits (wush98) is the company ID and the last 40 bits (675497ocks) is the manufacturer selected extension identifier.

3.1.2 iSCSI qualified name (IQN)

This naming convention starts with string “iqn” following by the reverse Domain Name Service (DNS) name.

Example: iqn.2006-04.com.wichita.state.ece:anrc

In the above example “iqn” is followed by the DNA registration year and month, followed by the reverse Fully Qualified Domain Name (FQDN).

3.2 Session Establishment

Session establishment starts with login phase and exchanges various session wide parameters before reaching full featured phase, where exchanged parameters will be used for the entire session. Following explanation will give an in-depth explanation of session establishment. The following is the sequence for login process

- Establish the TCP/IP connection
- Authenticate the iSCSI endpoints (Initiator and Target).
- Negotiate the parameters to be used by the iSCSI endpoints.

iSCSI session is started with opening a single TCP/IP connection between initiator and target, later it can expand to multiple connections depending on the capabilities of initiator and target, which are exchanged during login phase. Each connection negotiates its own unique

parameters. In this login process, the leading (first) login connection is the one that establishes the session-wide values which are discussed later in this section.

The login process is a combination of requests and responses between initiator and target. Until the login process is complete and the full-feature phase is reached, no other functions are permitted. There will be one or more login requests, each followed by a login response from the target until the target sends the final response.

There are two types of login sessions:

- Normal
- Discovery

3.2.1 Normal Phase

In this phase, the initial login negotiate on the session-wide values, which are Initiator name, Target Name, Session type, Authentication method, Target Portal Group Tag (TPGT). Following the initial login, all the secondary logins must have the same Initiator name and Target name as of the first connection.

3.2.2 Discovery Phase

In this phase initiator discover the available targets nodes in the given portal group (IP address with port number). Initiator uses “target = all” key value pair in the login phase to specify it is in discovery phase. The rest remains the same as in normal phase.

There are three phases through which the session establishment (or additional connection establishment) progresses, they are (in sequence of implementation)

- Security Negotiation phase(SNP)
- Login Operational Negotiation phase(LONP)
- Full Featured phase(FFP)

The login process will go through the exchange of iSCSI operational parameters in login operational negotiation phase (LONP) as well as security authentication processes (SNP), which are intended to ensure that the initiator is valid and authorized. On the completion of the login i.e. when full feature phase is reached, the connection of a session can carry SCSI commands and data between the initiator and the target. If desired, additional connections can also be started, so that an iSCSI session can be made up of multiple connections. In all login requests and responses there is a field called CSG (current stage) and NSG (next stage), which will specify the phase that the session is currently in and what phase is desired next. The login command is an "immediate command" requesting immediate action and must be operated immediately.

In general initiator can request a transition from one phase to another whenever it is ready. However, a target can respond with a transition only after it is offered one by the initiator.

3.3 Connection Establishment

In iSCSI, initiator establishes its first connection with the target by sending a Login Request Protocol Data Unit (PDU). In this PDU the initiator places the appropriate values for Connection Identifier (CID) (equal to zero), initiator session ID (ISID), initiator name, target name and other things. The iSCSI target responds accordingly and, upon last leading login response of the session it includes the unique (nonzero) target session identifier handle (TSIH). The TSIH must be unique within the target node for each session with the same initiator name. After the TSIH is returned to the initiator, the session is considered established and reached full-

feature phase (FFP). The initiator must then use the same TSIH when starting subsequent connections with the target with in the same session.

Every session between initiator and target has a unique Session Identifier (SSID). When a target node sees that an initiator is issuing a login for an existing SSID (with a nonzero TSIH), it assumes that the initiator is attempting to start a new connection or re-establish an old connection. In case multiple connections are started, the initiator may send commands on any free connection. The data and status responses related to a command must flow on the same connection on which the command travels. This is known as connection allegiance.

3.4 iSCSI data

- **Write Data:** is the data sent from Initiator to Target.
- **Read Data:** is the data received from Target to Initiator.

3.4.1 Write Data

In write, Data is sent from Initiator to Target in the following three ways:

Immediate data: is the data which can be sent to the target along with the command, without any acknowledgement. This data is limited to the First Burst Length.

Unsolicitated data: is the Data-Out PDU sent from initiator to the target without any explicit request for data from target like Ready-To-Transfer (R2T) PDU. This data is limited to Maximum Burst Length.

Solicitated data: is the Data-Out PDU sent from initiator to target with explicit request from target like R2T PDU. This data is limited to Maximum Burst Length.

3.4.2 Read Data

In read, data is sent from the target to the initiator within a Data-In PDU's. The amount of data that can be sent or received in a transaction is limited by the following values (negotiated during login)

First Burst Length is the maximum number of data bytes that can be delivered unsolicited from the iSCSI initiator to the iSCSI target during execution of a single SCSI command. This is a session-wide parameter, which applies to all the unsolicited data i.e. immediate data, unsolicited Data-Out PDUs, or a combination of both.

Max Burst Length is the maximum data payload in a sequence of Data-In or solicited Data-Out PDUs. A sequence consists of data PDUs that all have the same task tags (initiator or target) and is terminated by a "finished" flag (F bit) in the ending data PDU.

Max Recv Data Segment Length is the maximum amount of data that can be included in any iSCSI PDU. It is a connection- specific and direction-specific value.

The limiting factor on solicited data is the smaller of Max Recv Data Segment Length and Max Burst Length. For unsolicited data, the limiting factor is the smaller of Max Recv Data Segment Length and First Burst Length. Immediate date it limited by Max Recv Data Segment Length. The Ready to Transfer (R2T) PDUs can be sent one after the other, without waiting for the corresponding response from initiator, up to the negotiated limit in the value known as MaxOutStandingR2T.

3.5 Sequencing

iSCSI does not have any concept of acknowledgements, instead uses a set of sequences called sequence numbers on both initiator and target side which implicitly acknowledges the flow between both sides. There are two types of sequences:

- Connection specific sequencing
- Session specific sequencing

3.5.1 Connection specific sequencing

Each connection will have sequence numbers that it uses to keep track of the flow within the connection, these are

Status Sequence Number (StatSN): a counter maintained per connection that verifies the number of status response PDUs sent from the target to the initiator.

Data Sequence Number (DataSN): Sequence numbers dealing with the Data-In (from target), Data-Out (from initiator) that flow between the initiators and the targets.

Request to Send Sequence Number (R2TSN): These PDU's flow from target to initiator carrying their own sequence number called R2TSN. This sequence number is unique only within the scope of the command to which it applies. The Data-In/Out PDU and R2T PDU sequence numbers are all associated with commands, so they start counting at zero for each command.

3.5.2 Session specific sequencing

Session specific sequence numbers will be unique throughout the session, they are

Command Sequence Number (CmdSN): Since commands are sequenced across all connections within the session, this counter applies across the entire session (and any multiple connections

therein as well). Commands such as read and write are sequenced by iSCSI initiators which enables the target to deliver them in order to the SCSI level of processing in the target, regardless of the connection on which the command traveled.

Apart from this, iSCSI protocol maintains expected values for all the above counters.

Expected Command Sequence Number (ExpCmdSN): Target maintains ExpCmdSN, which is used to acknowledge the initiator issued commands. It always maintains one greater than the present command sequence number.

Expected Status Sequence Number (ExpStatusSN): Initiator maintains ExpStatusSN. The various PDUs that travel from the initiator to the target will return a value in a field called ExpStatSN. This field is generally one greater than the last StatSN value received from the target.

Maximum Command Sequence Number (MaxCmdSN): This value is given by the target to initiator, to tell the maximum number of commands that target can process at a given time. The combination of ExpCmdSN and MaxCmdSN are included in every PDU sent from the target to the initiator and can be used to provide a "windowing" technique for the acceptance of new commands.

Expected Data Sequence Number (ExpDataSN): Target acknowledges to initiator in response to received data-out PDU's. It is always one greater than the last Data Sequence Number (DataSN) sent for the corresponding command. The initiator can then determine if target has received all the data it has sent.

Sequencing is a means of sending acknowledgments of received commands or responses without having the overhead and latency of individual acknowledgments for each PDU. The SCSI Response PDU, which is sent by the target to acknowledge the completion of a command, will not only contain its own Status Sequence Number (StatSN) but also will contain the ExpCmdSN, thus informing the initiator of not only the status of the specific command but also the safe arrival of all the commands up to the ExpCmdSN. The target in fact can send the SCSI Response in the ending Data-In PDU without having to send a separate response PDU. This technique is called "phase collapse," because the normal SCSI response phase is collapsed into the PDU that ends the data transfer.

If there are no pending response messages for a period of time, the responder should generate a NOP (no-operation) message either NOP-In (from target) or NOP-Out (from initiator) and specify the next expected sequence number. This approach is especially useful given that an error seldom occurs that is not detected and fixed by TCP/IP. Therefore, network bandwidth is not consumed by acknowledgments that are almost never needed.

3.6 Command Ordering

Commands can be issued ones the connection reaches full- feature phase. After initial login, for all non-immediate commands, the CmdSN is increased by one.

If the difference between next CmdSN and the last received ExpCmdSN is large, than initiator thinks that some or all of its commands are not received at target side (either due to error or because of link failure) and then begin by resending the commands starting with the command that has CmdSN equal to the returned ExpCmdSN. This process continues until the ExpCmdSN matches with what the initiator thinks should be. Thus, iSCSI requires the iSCSI layer to deliver

the commands to the SCSI layer in order. In case a "hole" (Missing commands, which are tracked using CmdSN) exists in the CmdSN sequence, the commands that have a CmdSN higher than the "expected value of the hole" cannot be delivered to the SCSI layer until the missing commands are received by iSCSI.

3.7 Command Windowing

In iSCSI, buffer management which is a form of resource management plays a vital role since iSCSI storage controllers have more than one connection per sessions and the initiator can send both immediate and unsolicited data to the target. Therefore, iscsi uses the combination of ExpCmdSN and MaxCmdSN to define a window that controls the arrival of commands at the target i.e. the absolute difference between the ExpCmdSN and the MaxCmdSN (plus 1) will determine the actual number of commands the target can service for a particular initiator. When the buffer pool is being depleted at target side than the target should dynamically reduce the absolute difference between the ExpCmdSN and the MaxCmdSN (plus 1). As the absolute difference becomes smaller, fewer commands can be sent. Thus, when the window gets small (say one unit) then initiator must wait for response after sending each command to ensure that the window is still open. If the window size becomes zero, the initiator must stop sending commands and must wait until buffers are freed up by the completion of previous commands and the MaxCmdSN value is raised (relative to the ExpCmdSN), which will raise the command window size from zero and permit more commands to flow. Therefore MaxCmdSN value should be monitored continuously in order to prevent the oversubscription of commands to the available target buffers. For this reason, MaxCmdSN is always returned on the same PDUs with ExpCmdSN so that initiator can calculate the absolute difference between them and determine the command window size.

3.8 Retransmission of data and status

Since iSCSI data is transmitted over internet, the chances of data/status being corrupted is more which might results in retransmissions. In normal operation, TCP/IP will perform all error recovery and retransmission, but when both target and initiator are operating with CRC digest, only then iSCSI application can tell that it has detected an error which was undetected by TCP/IP error detection.

In case of data digest error, the initiator can ask for the data to be retransmitted by issuing Selective Negative Acknowledgement (SNACK) without waiting for response from appropriate side. Similarly on target side, it can request for retransmission of missing data or command by issuing a Reject PDU and then sending a recovery R2T PDU.

In case of header digest error i.e. header part of data or status PDU being corrupted, the initiator have to wait for the status response PDU to detect that a Data-In or Status (SCSI Response) PDU is missing. As specified above, initiator will request the retransmission of missing data or status (response) PDU using SNACK PDU. On target side, if it finds that header part of received PDU is corrupted then it simply drops the PDU and can have it resent via the R2T (called a recovery R2T).

3.9 Error Recovery Levels

There are three general classes of errors in iSCSI implementation, they are

- **Protocol error:** This is generally a program error and requires session restart. Here error recovery is done by the SCSI layer.
- **CRC error:** This is further classified as header digests (error in header segment of PDU) and data digest (error in data segment of PDU) error. It can be recovered by

retransmitting the data or response PDU or by reissuing the command PDU, depending on missing type. This error recovery is done by the iSCSI layer. Sometimes error recovery is not possible and implementation will retort to protocol error.

- **TCP/IP or link failure error:** This type of error can often be recovered by restarting another connection and shifting command and data allegiance to it from the failed connection. Here error recovery is done by the iSCSI layer. Sometimes recovery is not possible and implementation will retort to protocol error.

iSCSI classifies the above three mentioned recovery techniques as:

- **Error Recovery Level 0:** Recovery only by session restart (also known as session failure recovery or session recovery)
- **Error Recovery Level 1:** Recovery by reissuing commands, data, or status (also known as CRC failure recovery)
- **Error Recovery Level 2:** Connection failure recovery

In iSCSI, implementation that supports an error recovery level greater than 0 also supports the error recovery levels below it. If no error recovery levels are negotiated during login than by default it will be Error Recovery Level 0.

3.9.1 Error Recovery Level 0

In this error recovery level, if any error is detected (on target or initiator side), the initiator SCSI layer will restart the session i.e. all the executing and queued tasks (both iSCSI and SCSI tasks on both sides) will be aborted. And all the TCP/IP connections in that session will be closed and will be cleaned. After everything things are completed, all the connections

which were previously closed will be logged on as if they did in the first time. However, if one or more SCSI tasks at the target SCSI layer cannot be aborted, than those tasks will undergo normal processing. Initiator need to give simulated responses to the SCSI layer. For restarting the session, all the SCSI persistent reservations are maintained. Only the appropriate persistent reserve commands will affect these reservations (ISID, target node name, and TPGT) since, persistent reserve is associated with the I-T-L nexus.

3.9.2 Error Recovery Level-1

Error recovery level 1 is used to recover CRC-detected errors in iSCSI PDU with header digest and data. In case this error is not recovered, error recovery level 0 is performed and session is restarted.

3.9.2.1 Recovery from data digest errors

If the error was detected in the data part of the PDU via the data CRC digest error (header passed the CRC correctness test), the initiator can ask for the data to be retransmitted by issuing SNACK without waiting for response from appropriate side. Similarly on target side, it can request the retransmission of missing data or command by issuing a Reject PDU and then sending a recovery R2T PDU.

3.9.2.2 Recovery from header digest errors

If header part of data or status PDU is being corrupted, the initiator has to wait for the status response PDU to detect that a Data-In or Status (SCSI Response) PDU is missing, then initiator will request for the retransmission of missing data or status (response) PDU using SNACK PDU. On target side, if it finds that header part of received PDU is corrupted then it

simply drops the PDU and allows the initiator to notice a hole in the ExpCmdSN and cause it to resend the missing PDU.

3.9.2 Error Recovery Level 2

This type of error recovery level is used to recover from the connection failure and restoring the tasks therein. However in case not able to recover from connection failure, implementation will retort to Error Recovery Level 0.

Initiator side: This type of error recovery can be done when

- Initiator detects a TCP connection failure.
- Initiator receives an Asynchronous Message PDU from the target saying that one or all the connections in a session will be dropped.
- Initiator receives an Asynchronous Message PDU from the target requesting a connection logout.

In case of TCP connection failure which can be identified by either TCP/IP failure or gap in the command/response sequence which cannot be filled or failed iSCSI NOP, initiator issues a logout request PDU for the failed connection from another active connection within the session provided more than one connection is active. This logout request must contain a reason code of removed for recovery, which means that target must shut down the failed connection and prepare all its commands for connection allegiance on either a new connection or on any existing (active) connection in the session. For this initiator sends a series of Task Management Function Request PDUs, each of which tells the target to change the allegiance of a suspended task from the failed connection to the connection on which the Task Management Function Request PDU is issued.

Once the entire tasks are moved on to new connection (connection allegiance is complete), the tasks will continue as if they had originally been issued on the new connection. In case a new connection is not started, than the initiator may reinstate the suspended tasks across a number of the other active connections within the session, so that workload is balanced. Once, the new allegiance is established, all unacknowledged status and data will be resent automatically by the target. In case no status or data exist, the initiator may need to retry the command.

The task management commands are marked as an "immediate command" requesting immediate action and focus and must be operated on right away. The allegiance of the commands on the old connection, even though it has the same CID, still needs to be transferred to the new connection as soon as it enters into full-feature phase.

In this level of error recovery two values, DefaultTime2Wait and DefaultTime2Retain (default 3 seconds) which are negotiated during login phase, they are described as follows:

- If the connection goes away unexpectedly, the initiator has to wait DefaultTime2Wait (in seconds) before it can attempt to reconnect. This gives the target a chance to notice that the link is down, do the required cleaning and prepare for a reconnection.
- After this period, the initiator has additional DefaultTime2Retain (in seconds) to reestablish connections (if desired) and perform all task allegiances due to failure of the original connection. If not accomplished by this time, the target may abort and clean up all tasks and states (except persistent reserves).

Target recovers this error as follows

- The target tells initiator that one or all the connections in a session will be dropped by sending an Asynchronous Message PDU.
- The target sends a logout request (Asynchronous PDU) to the initiator asking to drop the one or all the connections in a session.

In all the above ways, a failing connection is reestablished without the SCSI layer or application being disrupted.

CHAPTER 4

LITERATURE REVIEW

Different performance analyses have been done in iSCSI considering various perspectives. In [9], author presented throughput with different network conditions like distance, packet loss, bandwidth and TCP window size. But this paper fails to present the throughput under dynamic traffic conditions with varying read and write operations, Further; no models for iSCSI operations have been presented.

In [8], the author presented iSCSI performance comparing file level and block level access. In this analysis, latency of read and write operations for different block sizes were analyzed under different load conditions. Response time of iSCSI was compared with SMB and NAS respectively. This paper lacks analyzing the performance under various real time scenarios like distance, packet loss, errors and dynamic traffic conditions etc. Analyses were done with a maximum block size of 64KB, which is lesser than max burst length. In real time, performance varies if block size is greater than the max burst length, which was observed in this research.

Performance evaluation of iSCSI has been done in [10] while analyzing various parameters with local disk and iSCSI disk. Analyses of various parameters like throughput, average response time and CPU utilization with respect to block size have been done. The authors in [11] performed analysis in a local lab for various iSCSI parameters like iSCSI read and iSCSI write and throughput on local disk and iSCSI disk. The authors also analyzed the throughput with respect to block size involving different network delays and also efficiency with respect to file size and block size. In this analysis authors did not take into account the real time parameters like distance, packet loss, network traffic etc. which have considerable impact on the overall performance. Also, models have not been proposed with respect to their analysis.

In [12] modeling of iSCSI write was done along with the performance analysis of iSCSI write and write throughput considering different parameters like RTT, processing time, delay and packet loss. But this paper lacks modeling of iSCSI read and performance analysis of iSCSI read and read throughput.

In all the above mentioned papers performance analysis without considering all the real time parameters like packet loss, network delay, dynamic traffic conditions and distance were presented. Also, none of the papers presented the model for iSCSI read operation and generalized throughput.

In this paper authors came up with the models for iSCSI Read, iSCSI Write and iSCSI throughput while considering iSCSI error recovery levels, which is the first of its kind. Also, various parameters like read time, write time and throughput were analyzed considering dynamic traffic, network delay, packet loss and distance.

In real time, security must be taken in to consideration as the data travels across the networks, and may be affected by various threats. In this thesis, security considerations were taken in to account and the performance of iSCSI read, write and throughput with IPSec in place were also analyzed. While analyzing the establishment of session with different volume sizes, it was observed that session establishment time varies with volume size.

CHAPTER 5

MODELING OF iSCSI

In this research, we present analytical models for iSCSI Read, iSCSI Write and throughput. This modeling consists of detailed study of iSCSI protocol [25] and analyzing every part of the protocol considered iSCSI level errors. Following sessions will explain each of the parameters in brief.

5.1 iSCSI Read

When initiator issues a read command with given Logical Block Address (LBA) and length, first target buffers the requested data in terms of Maximum Burst Length in to its buffers and then starts sending the data to the initiator.

When target sending the data to the initiator, it forms each iSCSI PDU of size Maximum Receivable Data Length (of initiator, which is the buffer size of the network interface card) and sends these PDU's until maximum burst length is completed. Target buffers the data for every maximum burst length in to its buffers, so it takes some time to buffer, which is known as Target processing time. Once the target completes sending requested data to the initiator, it sends a response to the initiator. This overall process can be modeled and explained as follows.

Total Read Time (T_R) can be give as sum of total transfer time (T_{TTT}), total processing time (T_{PT}) and total iSCSI error recovery time ($T_{iSCSI\_E\_R}$). Each of these times are analyzed and modeled individually and explained as follows.

$$T_R = T_{PT} + T_{TTT} + T_{iSCSI\_E\_R} \quad \text{---------- (1)}$$

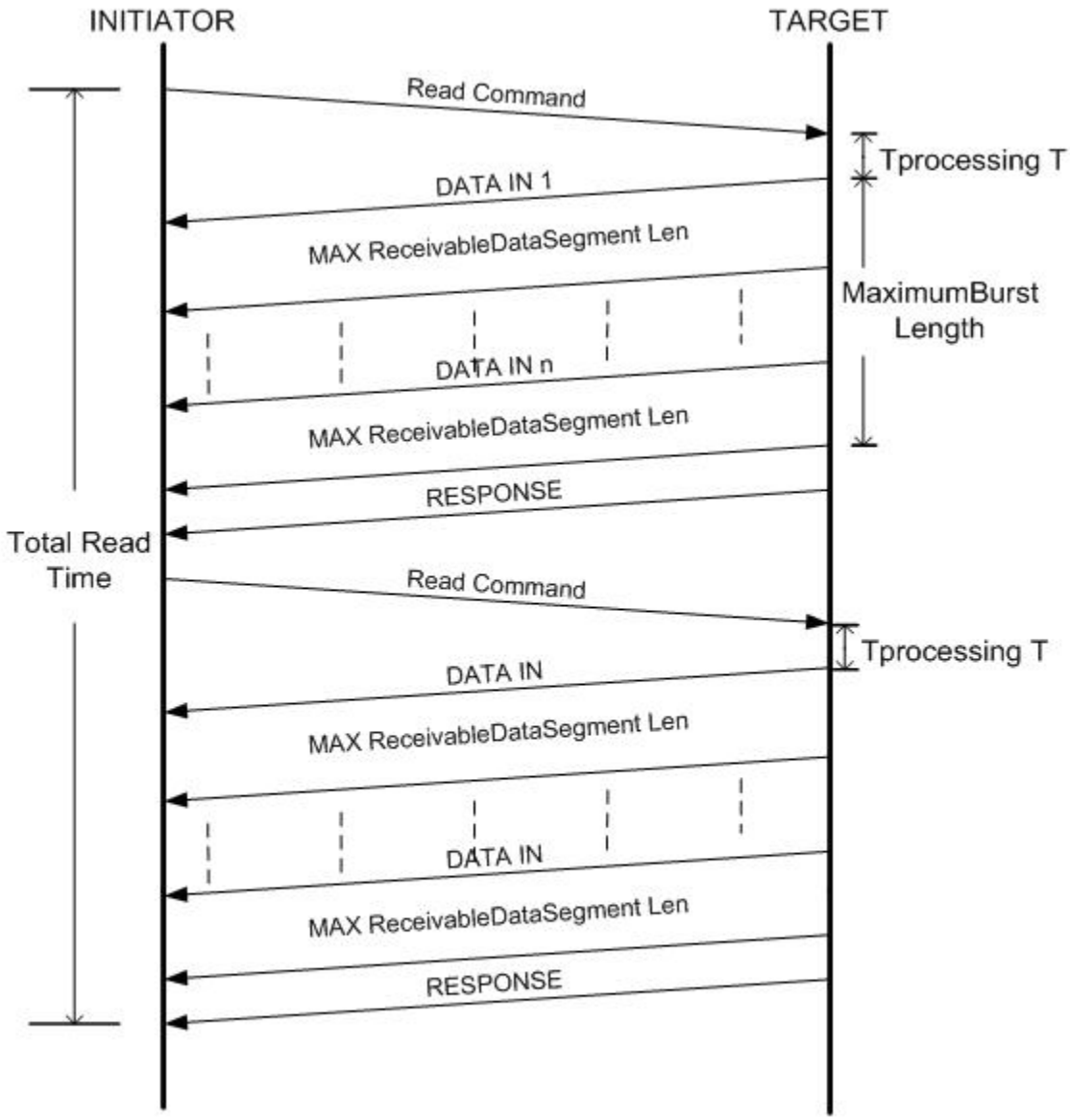

Figure 5.1 Timing Diagram of READ Command

5.1.1 Total Processing time

Processing time mainly involves two parameters, Target processing time and Initiator processing time. When initiator issues a read command, target needs to buffer the required data in terms of Maximum Burst length. For every Maximum burst length, target takes time to buffer data, which is known as Target processing time. When the requested data is received by initiator,

it takes time to buffer that data, known as initiator processing time. As observed, initiator processing time (T_{IP}) is very small compared to target processing time (T_{TP}), as the buffering operations at initiator are carried in parallel to receiving data from target. It can be given as.

$$T_{PT} = N \times (T_{IP} + T_{TP}) \text{ ---------- (2)}$$

Where N is the total number of Maximum Burst's generated, will be explained in equation (3).

5.1.2 Total Transfer time

Transfer time is the time taken for transmitting actual data between target and initiator. It is sum of three parameters, time taken to transmit total data over TCP, time taken to acknowledge the data transmitted and total TCP retransmissions time due to errors. Each of them is modeled as follows.

Total transfer time can be given as the total no of TCP packets generated for the given data times half the round trip time. Total number of TCP packets generated is given as the product of total number of Maximum bursts generated and TCP packets generated for each maximum burst length. In addition to this, for each response we need to consider one half RTT, which is given in the equation (5).

Total number of Maximum Bursts generated and total number of TCP packets generated is represented by N and M respectively and given as

$$N = Max\left(1, \frac{TotalDataSize}{MaximumBurstLength}\right) \quad \text{(3)}$$

$$M = Min\left(\frac{MaximumBurstLength, TotalDataSize}{TCPMSS}\right) \quad \text{(4)}$$

Where TCPMSS is the TCP Maximum Segment Size.

∴ Total data transfer time (T_{DT}) is given as

$$T_{DT} = N(M+1) \times \frac{RTT}{2} \quad \text{---------- (5)}$$

5.1.3 Total time taken to acknowledge transmitted data by TCP

This is the total time (T_{ACK}) taken to acknowledge total data transmitted by Target to the Initiator, this parameter totally depends on the TCP window size, if the TCP window size is 64KB, initiator can acknowledge after receiving 64KB of data from target. This window size is dynamic, varies in the session time due to TCP errors. We considered the average TCP window size during the session for approximate calculations.

$$T_{ACK} = \left(\frac{TotalDataLength}{AverageTCPWindowSize}\right) \times \left(\frac{RTT}{2}\right) \text{---------- (6)}$$

5.1.4 Total time taken for TCP retransmit due to errors

This is the total time involved in retransmitting data for TCP errors. This is a value taken for a given probability. Let in the session time, the probability of occurring an error while n_{th} read operation is say P_n. Then probability of occurring an error in nth read operation is given by the product of probability P_n and the number of packets transmitted from target to initiator in nth read operation.

Total number of TCP retransmissions is

$$Y_i = m \times P_i \quad \text{..... (7)}$$

Where

Y is total numbers TCP retransmissions for n reads and is given by

$$Y = Y_1 + Y_2 + \text{----} + Y_n$$

$$Y = \sum_{i=1}^{n} Y_i \quad \text{..... (8)}$$

'm' is the total no of packets per read.

Pi is the probability of loosing ***n*** packets in $\mathbf{i_{th}}$ read.

Total time involved in retransmissions ($\mathbf{T_{TCP\_RE}}$) error data is given as product of packets lost per read and round trip time.

$$T_{TCP\_RE} = Y \times RTT \text{ ---------- (9)}$$

Therefore Total transfer is given by

$$T_{TTT} = T_{DT} + T_{ACK} + T_{TCP\_RE} \text{ ---------- (10)}$$

5.1.5 Total time taken to recover iSCSI Errors

iSCSI errors can be classified in to three types, they are Protocol errors (Level 0), CRC detected errors (Level 1) and Connection failure errors (Level 2). These errors are explained in detail in chapter 3. Total time taken to recover iSCSI level errors ($\mathbf{T_{iSCSI\_E\_R}}$) is given by sum of time taken to recover protocol errors ($\mathbf{T_{PE}}$), time taken to recover CRC detected errors ($\mathbf{T_{CRC\_DE}}$) and time taken to recover connection failure errors ($\mathbf{T_{CFE}}$).

$$T_{iSCSI\_E\_R} = T_{PE} + T_{CRC\_DE} + T_{CFE} \text{ ---------- (11)}$$

5.1.5.1 iSCSI protocol error (level 0)

In this error recovery, if a protocol error is detected, current session is terminated and a new session is restarted. All the executing and queued commands are aborted. When this kind of error is occurred while read operation is going on, application layer should keep track of service request and re-request the service to SCSI layer. So total read operation is again performed. Total session error recovery time ($\mathbf{T_{PE}}$) can be minimum of half the round trip time for Target to receiving read command from Initiator or maximum of total read time because chances of

occurring this error at the time of finishing the read command. Error recovery level 0 is explained briefly in figure 5.2.

$$T_{PE} = \min of \frac{RTT}{2} and \max of \text{Re}adTime \text{ ---------- (12)}$$

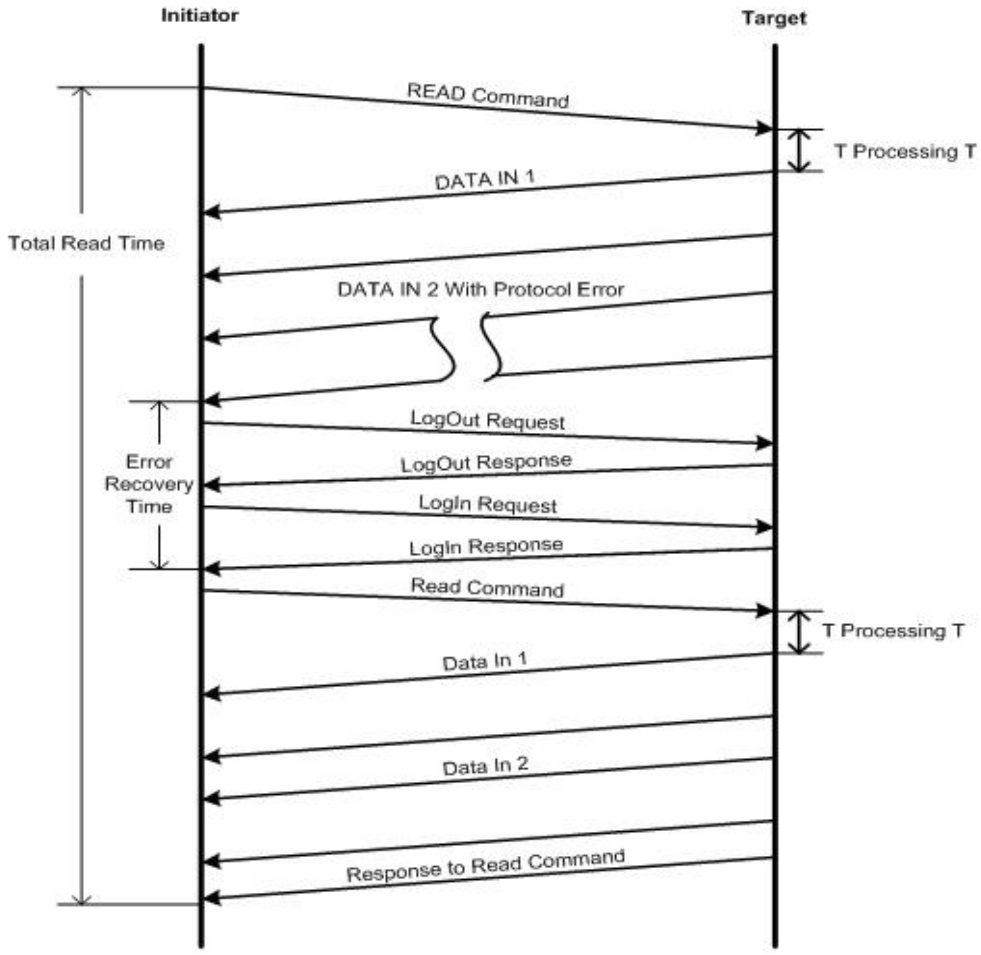

Figure 5.2 iSCSI Error Recovery Level 0

5.1.5.2 iSCSI CRC Digest Error (Level 1)

This error will occur due to two reasons, Header digest error or Data Digest error. If we consider this error occurred in read operation at target, if the header digit error is with the read command received by the target, target simply discards the packet. Initiator will notice this by

looking at the holes (missing command sequence numbers at target) in expected command sequence number from the target and re-issues the command. If this header digest error was detected in the Data-In PDU at initiator, initiator simply drops the packet and proceeds further, if it has more Data-In PDU's, it will notice that it is missing one of the Data-In PDU by means of data sequence number and request's target that particular Data-In PDU with a SNACK PDU. If the last Data-In PDU is missing and followed by a response, initiator will detect the missing Data-In PDU through expected Data sequence number and expertly requests for that PDU with SNACK PDU. If Response PDU is missing, initiator will wait for a default application defined time and issues NOP OUT and in response it receives NOP-IN from target with current status sequence number and expected command sequence number, by looking at status sequence number, initiator will come to know it is missing response for previous PDU and issue a SNACK PDU for the Response PDU. Error recovery level 1 data digest error and header digest errors are explained briefly in figure 5.3 and 5.4 respectively.

When Data digest error is detected at initiator, initiator will issue a SNACK PDU requesting particular Data-IN PDU, target will respond with that particular Data-In PDU. Total time taken to recover CRC detected errors ($\mathbf{T_{CRC\_DE}}$) is sum of time taken to recover header digest errors ($\mathbf{T_{HDE}}$) and time taken to recover data digest errors ($\mathbf{T_{DDE}}$).

$$T_{CRC\_DE} = T_{HDE} + T_{DDE} \quad \text{---------- (13)}$$

$$T_{HDE} = T_{DDE} = T_{MRDL} + \frac{RTT}{2} \quad \text{---------- (14)}$$

$$T_{MRDL} = \left(\frac{Max\,\mathrm{Re}\,cDataLength}{TCPMSS} \right) \times \frac{RTT}{2} \quad \text{---------- (15)}$$

In case of Header digest or Data digest errors, it takes half RTT time for sending SNACK PDU from initiator to the target and takes a maximum receivable data length ($\mathbf{T_{MRDL}}$) of transfer time.

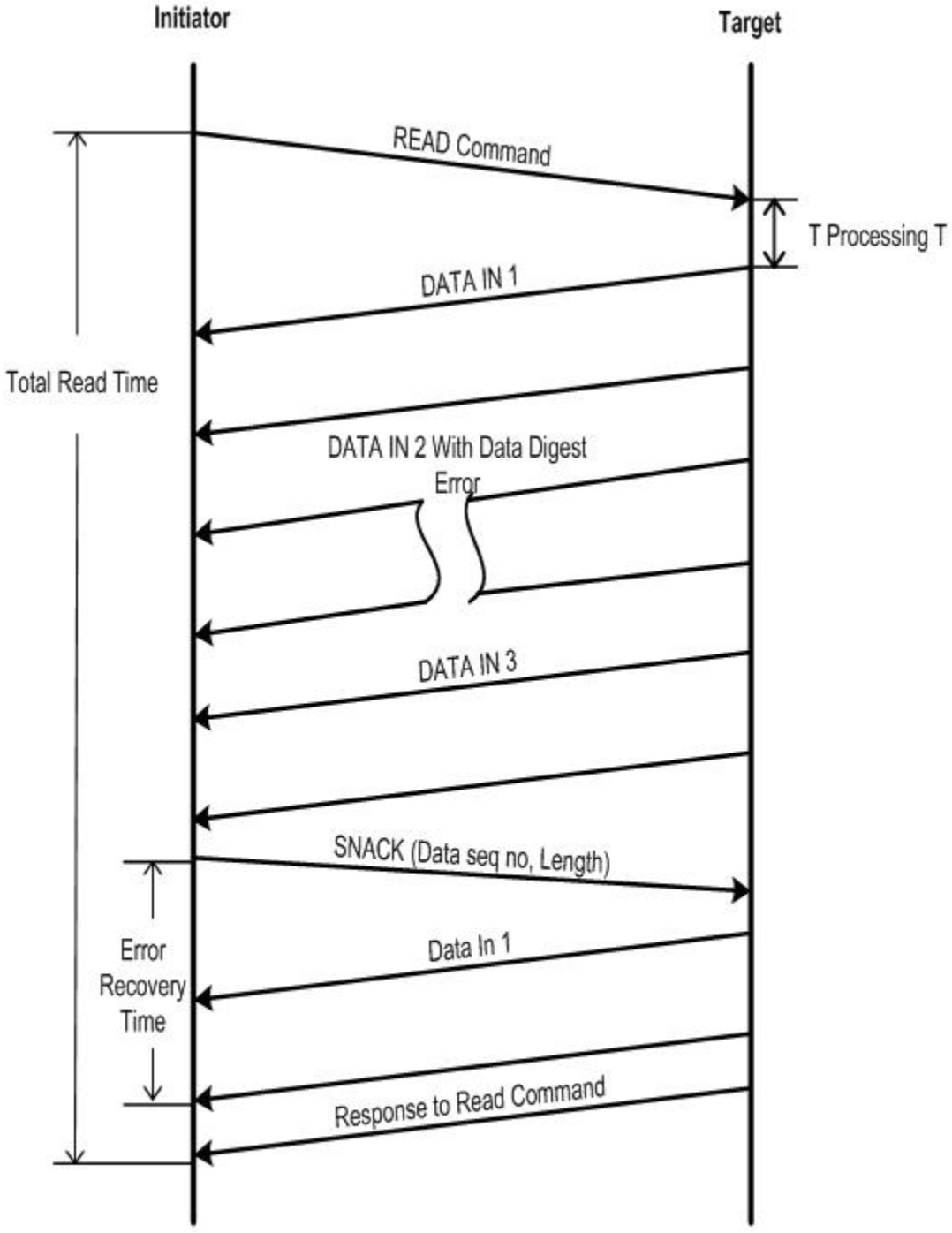

Figure 5.3 iSCSI Error Recovery Level 1 (Data digest error)

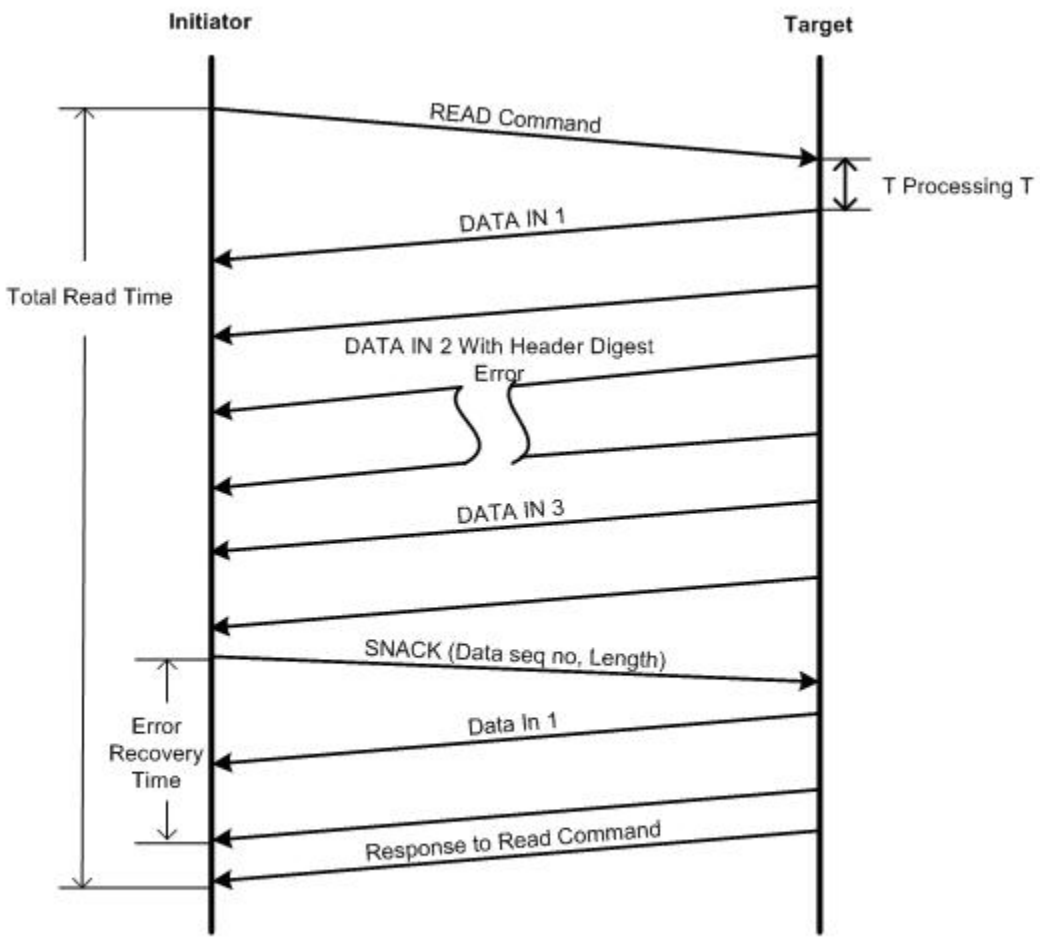

Figure 5.4 iSCSI Error Recovery Level 1 (Header Digest Error)

5.1.5.3 iSCSI Connection failure error (Level 2)

This error is detected when initiator detects a TCP connection failure or target requests initiator for Log-Out or informing about connection drop information. When this kind of error is occurred while read operation is going on, initiator will first try to logout the dropped connection from other active connections and waits for DefaultTime2Wait period to allow target to cleanup and then it may re-login for new connection with in DefaultTime2Retain period or simply use the existing connections. Then task management activities will recover the existing and

outstanding tasks from the old connection to connections on which the task management PDU's were issued. Error recovery level is explained briefly in figure 5.5.

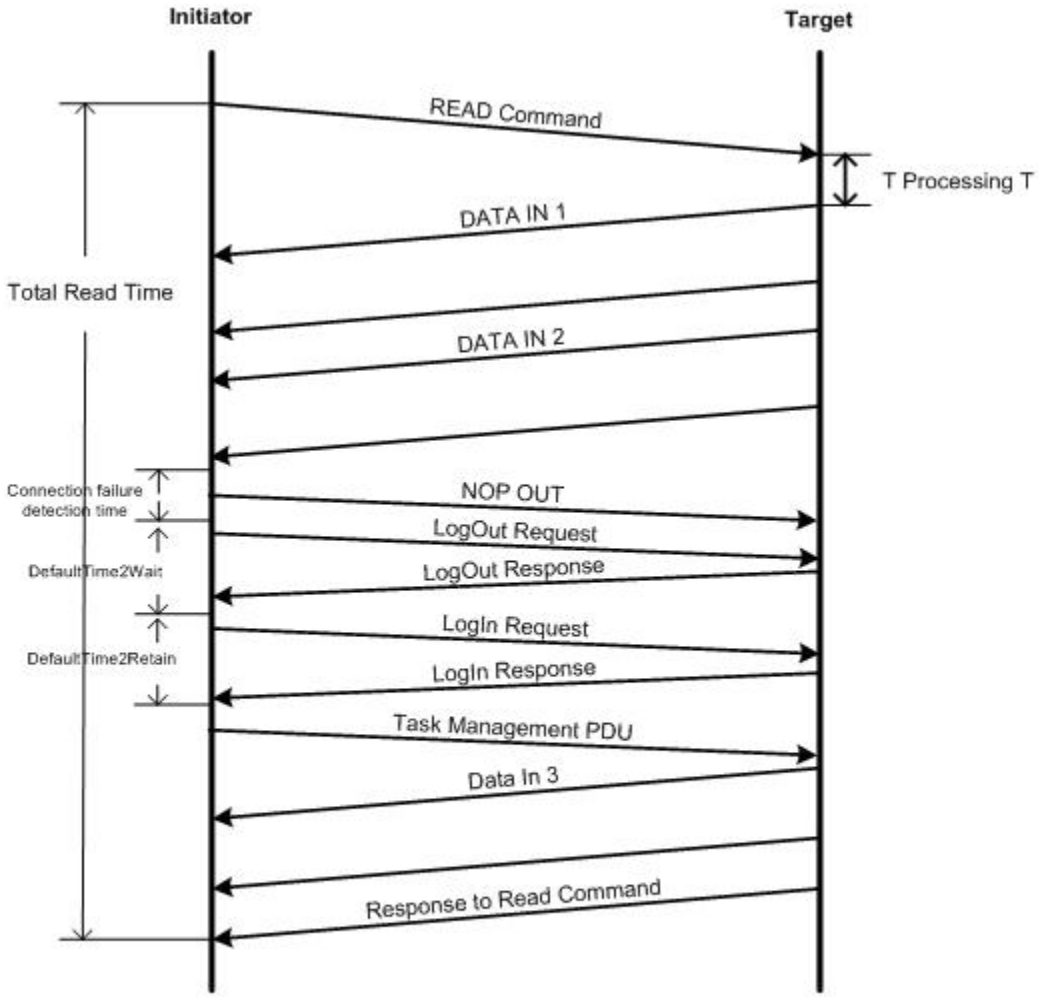

Figure 5.5 iSCSI Error Recovery Level 2

Total time taken to recover connection failure errors (T_{CFE}) is given as sum of time taken to detect connection failure (T_{CF}), Default time to wait (T_{DT2W}) and Default time to retain (T_{DT2R}).

$$T_{CFE} = T_{CF} + T_{DT2W} + T_{DT2R}. \quad \text{------ (16)}$$

T_{CF} is the time taken by initiator to detect that the TCP/IP connection is failed. It depends on the implementation of the application.

T_{DT2W} is the minimum amount of time that initiator should wait before attempting to reconnect. Generally this parameter is negotiated in the login phase.

T_{DT2R} this is the maximum amount of time that initiator has to reestablish connections after waiting for DefaultTime2Wait. Generally this parameter is negotiated in the login phase.

5.2 iSCSI Write

Initiator issues the write command with immediate data of first burst length, and waits for the target response with R2T for sending the remaining data. This form of data transfer is known as solicitude data transfer. When ever initiator receives R2T, it sends maximum burst length of data in the form of iSCSI PDU, of size maximum receivable data length. Target buffers maximum burst length of data and write it to the specified Logical Block Address (LBA) of the specified Logical Unit Number (LUN), this process takes T processing amount of time. Initiator buffers the required data in to its buffers before sending it to the target, this time is almost covered in target processing time, but depending on the system specification, it may take more than target processing time. Once all the data is sent, target responds to the initiator with good status. The overall process is modeled and explained briefly in figure 5.6.

Total Write Time (T_W) can be give as sum of total transfer time (T_{TTT}), total processing time (T_{PT}) and total iSCSI error recovery time ($T_{iSCSI\_E\_R}$). Each of these times are analyzed and modeled individually and explained as follows.

$$T_W = T_{PT} + T_{TTT} + T_{iSCSI\_E\_R} \quad \text{---------- (17)}$$

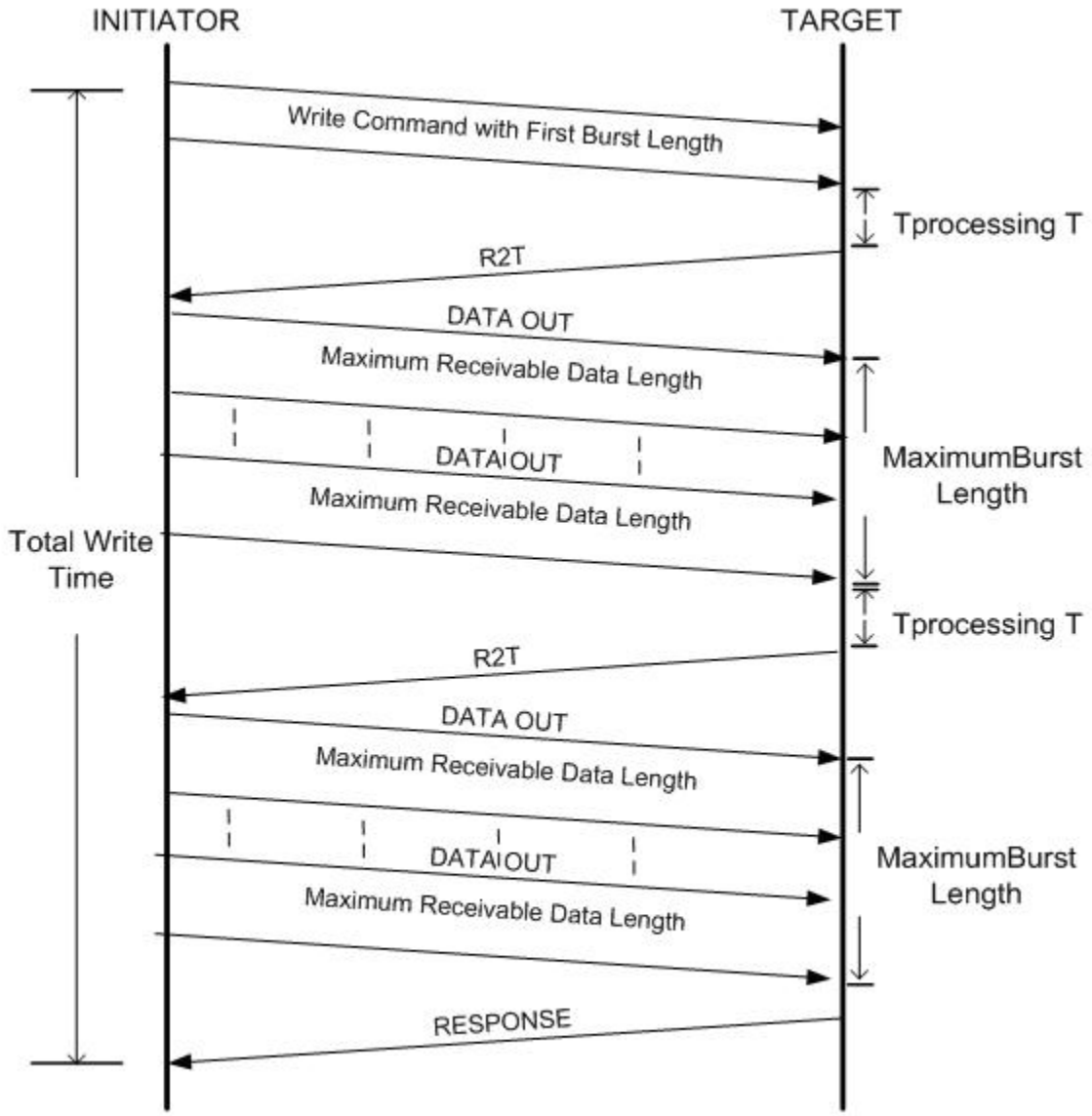

Figure 5.6
Timing diagram of Write Command

5.2.1 Total Processing time

Processing time mainly involves two parameters, Initiator processing time (**TIP**) and Target processing time (**TTP**). When initiator issues a write command, it needs to buffer the required data before sending it to the Target, which is known as Initiator processing time. As target receives the data, it first buffers maximum burst length of data and then writes it to the specified LBA of specified LUN, it takes target processing amount of time.

$$T_{PT} = N \times (T_{IP} + T_{TP}) \text{ ---------- } (18)$$

Where N is the total number of Maximum Burst's generated, will be explained in equation (21).

5.2.2 Total Transfer time

Transfer time is the time taken for transmitting actual data between initiator and target. It is sum of three parameters, time taken to transmit total data over TCP (T_{DT}), time taken to acknowledge the data transmitted (T_{ACK}) and total TCP retransmissions time due to errors (T_{TCP_RE}). Each of them is modeled in the iSCSI Read session, but a little change with the total data transfer time over TCP varies a bit, will be explained. Remaining parameters like time taken to acknowledge the data transmitted and total TCP retransmissions time due to errors are similar as in Read command. Total transfer is given by

$$T_{TTT} = T_{DT} + T_{ACK} + T_{TCP\_RE} \text{ ---------- } (19)$$

$$T_{DT} = \left(\left(\frac{FirstBurstLength}{TCPMSS} \right) + N(M+1) \right) \times \frac{RTT}{2} \text{ ---------- } (20)$$

Where N is the total number of maximum bursts generated and given by

$$N = \left(\frac{TotalDataSize - FirstBurstLength}{MaximumBurstLength} \right) \text{ ---------- } (21)$$

And M is the total number of TCP packets generated and as given in equation (4).

5.2.3 Total time taken to acknowledge transmitted data by TCP

This is the total time taken to acknowledge total data transmitted by initiator to target. This parameter is explained briefly in the equation (6) of read session. And this can be given as follows.

$$T_{ACK} = \left(\frac{TotalDataLength}{AverageTCPWindowSize} \right) \times \left(\frac{RTT}{2} \right) \text{ ---------- } (22)$$

5.2.4 Total time taken for TCP retransmit due to errors

This parameter is explained briefly in the equation (9) of iSCSI read session and given as follows.

$$T_{TCP\_RE} = Y \times RTT \text{ ---------- (23)}$$

5.2.5 Total time take to recover iSCSI Errors

iSCSI errors can be classified in to three types, they are Protocol error (Level 0), CRC detected errors (Level 1) and Connection failure error (Level 2). Total time taken to recover iSCSI level errors is given by Total time taken to recover iSCSI level errors ($T_{iSCSI\_E\_R}$) is given by sum of time taken to recover protocol errors (T_{PE}), time taken to recover CRC detected errors (T_{CRC_DE}) and time taken to recover connection failure errors (T_{CFE}) as given in equation (11).

$$T_{iSCSI\_E\_R} = T_{PE} + T_{CRC\_DE} + T_{CFE} \text{ ---------- (24)}$$

5.2.5.1 iSCSI protocol error (level 0)

In this error recovery, if a protocol error is detected, current session is terminated and a new session is restarted. All the executing and queued commands are aborted. When this kind of error is occurred while write operation is going on, application layer should keep track of service request and re-request the service to SCSI layer. So total iSCSI Write operation is performed again. Total time taken to recover protocol error (T_{PE}) is given and explained as in equation 12.

$$T_{PE} = \min of \frac{RTT}{2} and \max of \text{Re}adTime \text{ ---------- (25)}$$

This error is clearly described in the following figure 5.7.

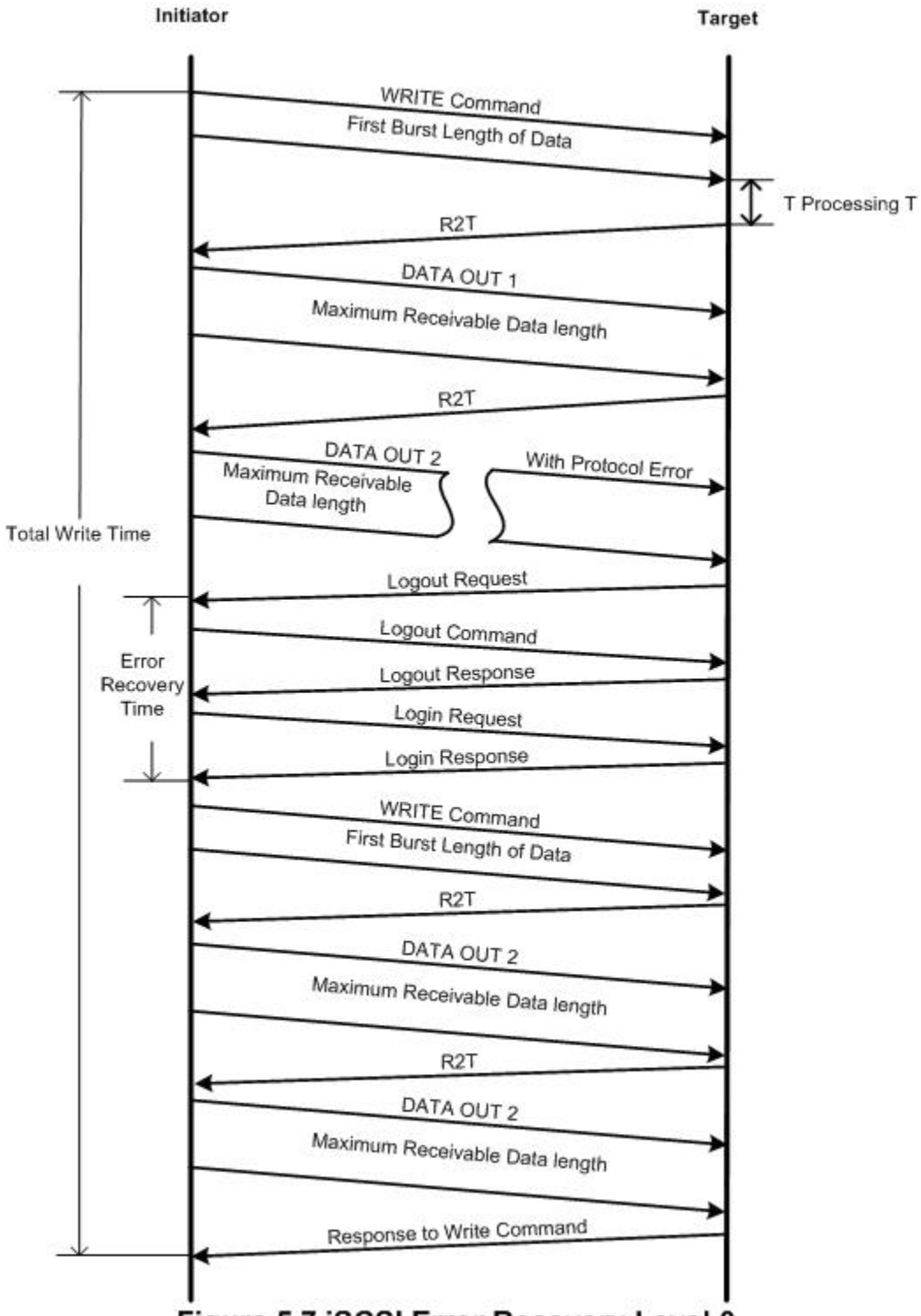

Figure 5.7 iSCSI Error Recovery Level 0

5.2.5.2 iSCSI CRC Digest Error (Level 1)

This error will occur due to two reasons, Header digest error or Data Digest error. The following timing diagram will illustrate this error clearly. If this error is detected in the write operation at target, target simply discards the iSCSI PDU. Initiator still waiting for the R2T will issue a NopOut PDU to check the status of the target, as target responds with the NopIn PDU. By checking the Target Task Tag (TTT), initiator will re-issue the Data Out PDU. In case if the header digest error detected at initiator with the R2T, initiator simply drops the PDU and waits for application defined time, after that it issues the NopOut and in response it receives NopIn from target. Initiator by looking at TTT from NopIn, will issue a SNACK PDU for R2T. Target will respond with R2T for the SNACK PDU. In case of data digest error with the data out PDU's, target detects this error and will issue the Reject PDU followed by the R2T PDU to recover the data with data digest error. This error recovery level 1 is explained briefly in figure 5.8 and 5.9. Header and Data digest errors are given as in equation (13) and (15).

Total time taken to recover CRC detected errors (T_{CRC_DE}) is sum of time taken to recover header digest errors (T_{HDE}) and time taken to recover data digest errors (T_{DDE}).

$$T_{CRC\_DE} = T_{HDE} + T_{DDE} \quad \text{---------- (26)}$$

$$T_{HDE} = T_{DDE} = T_{MRDL} + RTT \text{ ---------- (27)}$$

$$T_{MRDL} = \left(\frac{Max\,\mathrm{Re}\,cDataLength}{TCPMSS} \right) \times \frac{RTT}{2} \quad \text{---------- (28)}$$

In case of Header digest or Data digest errors, it takes RTT time for sending Reject PDU and R2T PDU from initiator to the target and takes a maximum receivable data length (T_{MRDL}) of transfer time.

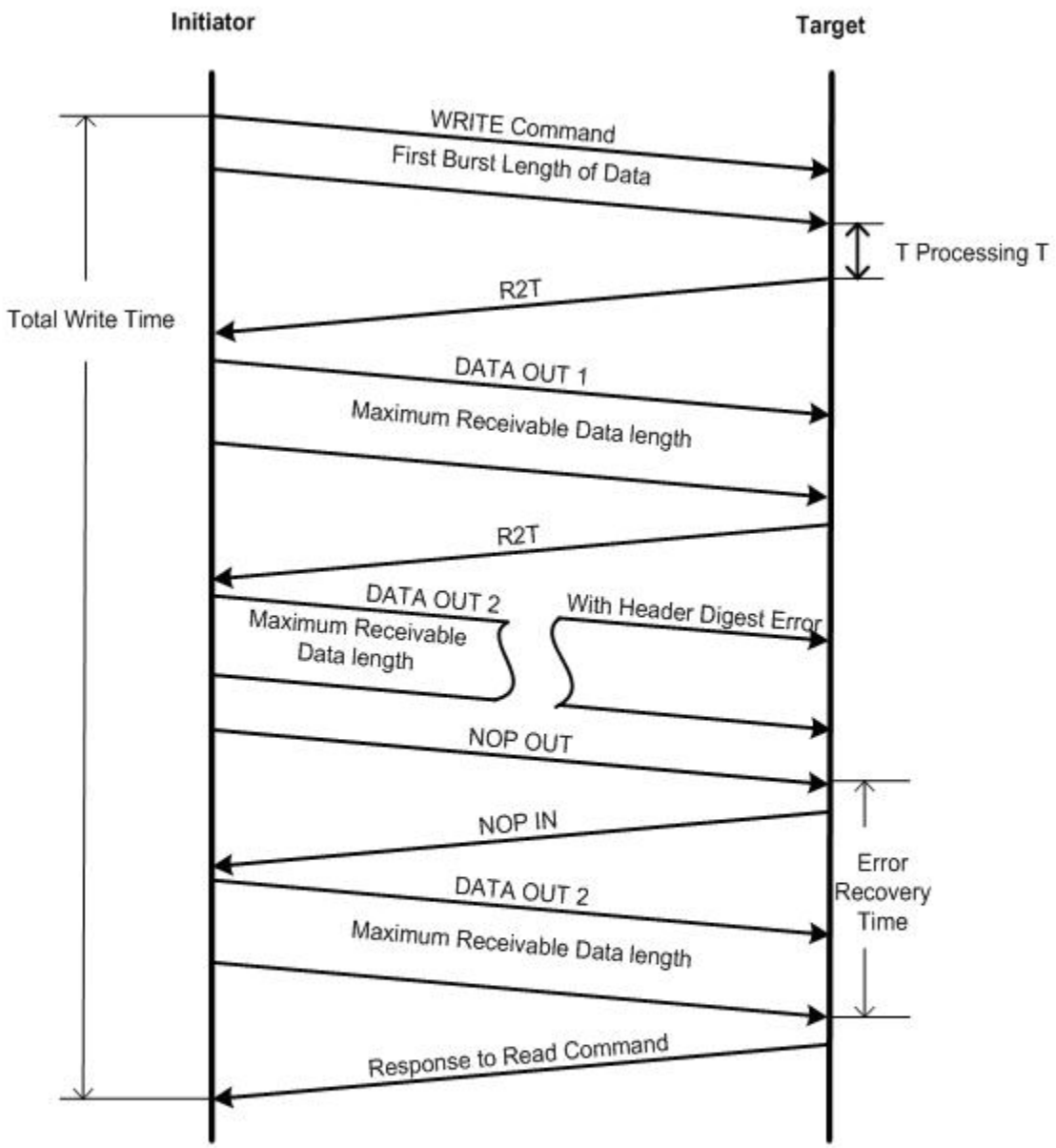

Figure 5.8 iSCSI Error Recovery Level 1 (Header Digest Error)

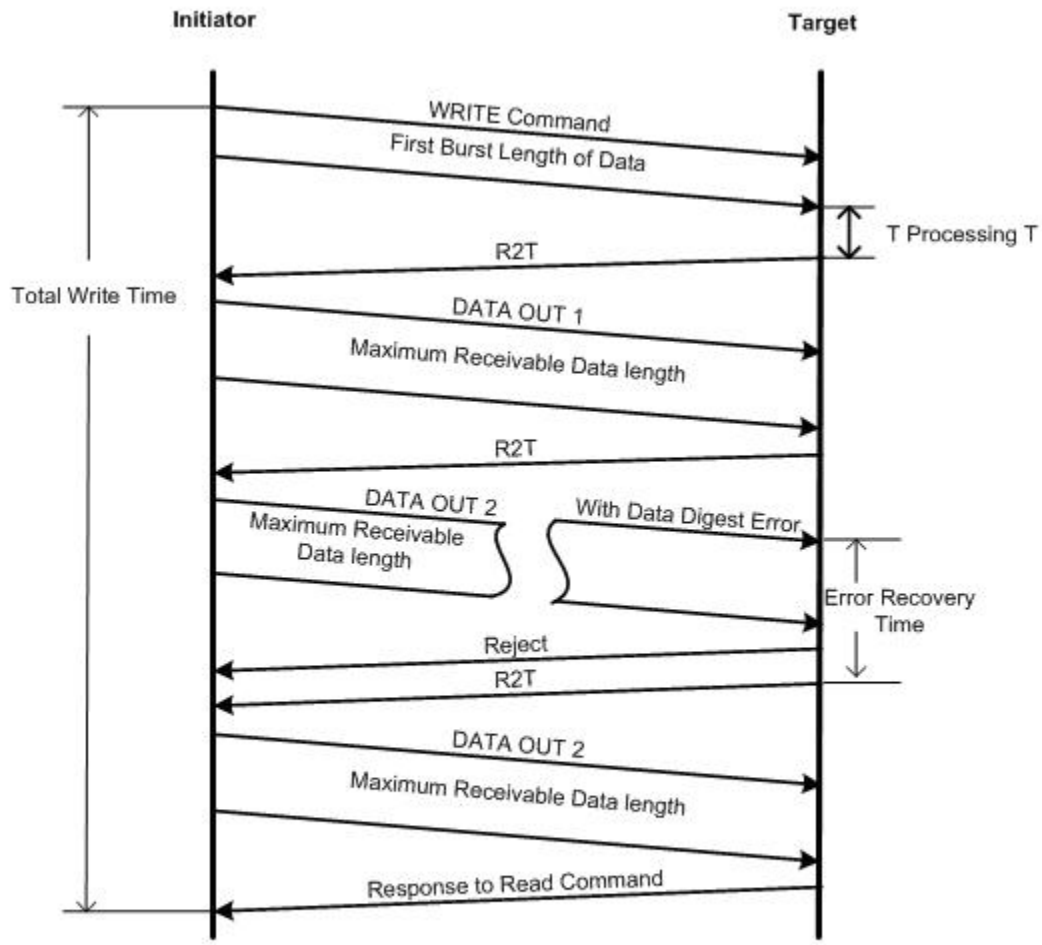

Figure 5.9 iSCSI Error Recovery Level 1 (DATA Digest Error)

5.2.5.3 iSCSI Connection failure error (Level 2)

This error is detected when initiator detects a TCP connection failure or target requests initiator for Log-Out or informing about connection drop information. When this kind of error is occurred while write operation is going on, initiator will first try to logout the dropped connection from other active connections and waits for DefaultTime2Wait period to allow target to cleanup and then it may re-login for new connection with in DefaultTime2Retain period or simply use the existing connections. Then task management activities will recover the existing

and outstanding tasks from the old connection to connections on which the task management PDU's were issued.

Total time taken to recover connection failure errors (T_{CFE}) is given as sum of time taken to detect connection failure (T_{CF}), Default time to wait (T_{DT2W}) and Default time to retain (T_{DT2R}) as in equation (16) and explained briefly in figure 5.10.

$$T_{CFE} = T_{CF} + T_{DT2W} + T_{DT2R} \quad \text{---------- (29)}$$

T_{CF} is the time taken by initiator to detect that the TCP/IP connection is failed. It depends on the implementation of the application.

T_{DT2W} is the minimum amount of time that initiator should wait before attempting to reconnect. Generally this parameter is negotiated in the login phase.

T_{DT2R} this is the maximum amount of time that initiator has to reestablish connections after waiting for DefaultTime2Wait. Generally this parameter is negotiated in the login phase.

5.3 Throughput of iSCSI

Throughput in terms of peer to peer communications is defined as the total data transferred for unit time. In iSCSI it can be defined as total data transferred for unit time. It varies with iSCSi read and write depending on various parameters like processing times involved at initiator and target. In general it can be given as

$$Throughput = \frac{TotalDataSize}{TotalTransferTime} \quad \text{---------- (30)}$$

Total Data Transfer Time (T_{DTT}) can be give as sum of total transfer time (T_{TTT}), total processing time (T_{PT}) and total iSCSI error recovery time ($T_{iSCSI\_E\_R}$), which are explained in equations (2), (10), (11), (18) and (19).

$$T_{DTT} = T_{PT} + T_{TTT} + T_{iSCSI\_E\_R} \quad \text{---------- (31)}$$

Transfer time depends on the read or write operation, which gives the read throughput or write throughput.

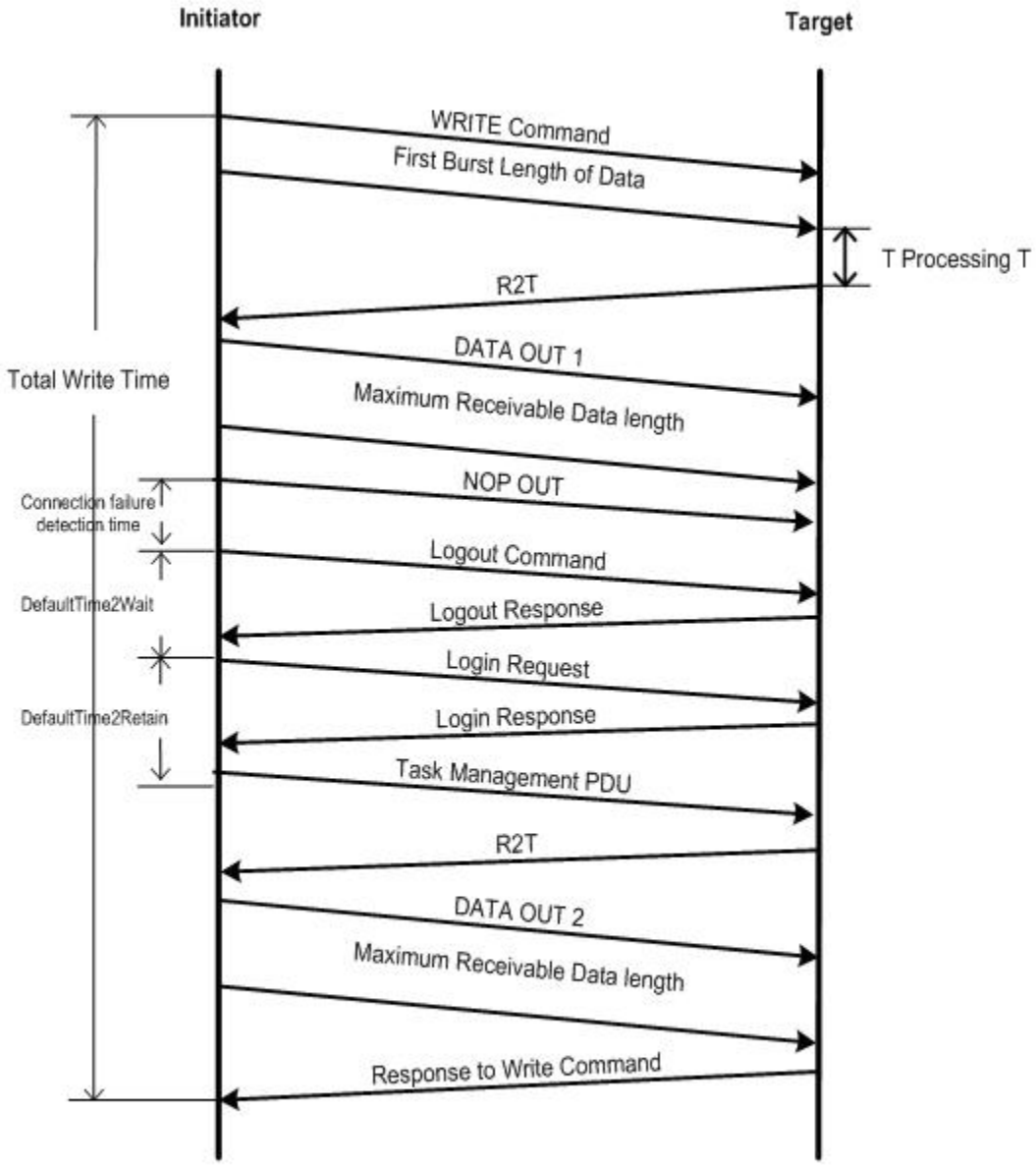

Figure 5.10 iSCSI Error Recovery Level 2

CHAPTER 6

TEST SETUP

6.1 Test Objectives

The test objective is to analyze various parameters of iSCSI like Read, Write and throughput using ethereal traces and model each of the parameter considering different realistic conditions like round trip time, data traffic conditions and iSCSI level errors, validate each of modeled parameter using IO meter results. The test bed is shown in figure 6.1 and consists of iSCSI initiator and target, traffic load generators on the internetwork.

Test bed was made under windows and Linux environments. In windows, we used Microsoft iSCSI initiator [5] as initiator and StarWind from Rocket Division Software [4] as the target on Microsoft server 2003 machine. In Linux, iSCSI Enterprise Target (IET) was installed in a Fedora Core 5 machine with a 2.6.16 kernel. The Initiator was built using a Fedora Core 5 source iSCSI Initiator in a Fedora Core 5 machine with a 2.6.16 kernel and Microsoft iSCSI Initiator in a Microsoft server 2003 machine. For traffic generation we used IP load [6] between initiator and target. IP Load is a program which generates a large number of IP frames to the specified destination. It uses both TCP and UDP ports to generate traffic. User has an option to specify the amount of data and the rate at which it should be presented on the network to the destination.

IO meter [7] is a benchmark used to analyze various parameters like average read response time, average write response time, I/O per second, %of CPU utilization and throughput etc. IO meter consists of two modules, iometer and dynamo. Iometer is a graphical user interface (GUI), used to configure workload, operating parameters start and stop tests. It gives commands

to dynamo and collects the results in to an output file. Dynamo is a workload generator, which performs I/O operations under iometer commands and returns the results to it.

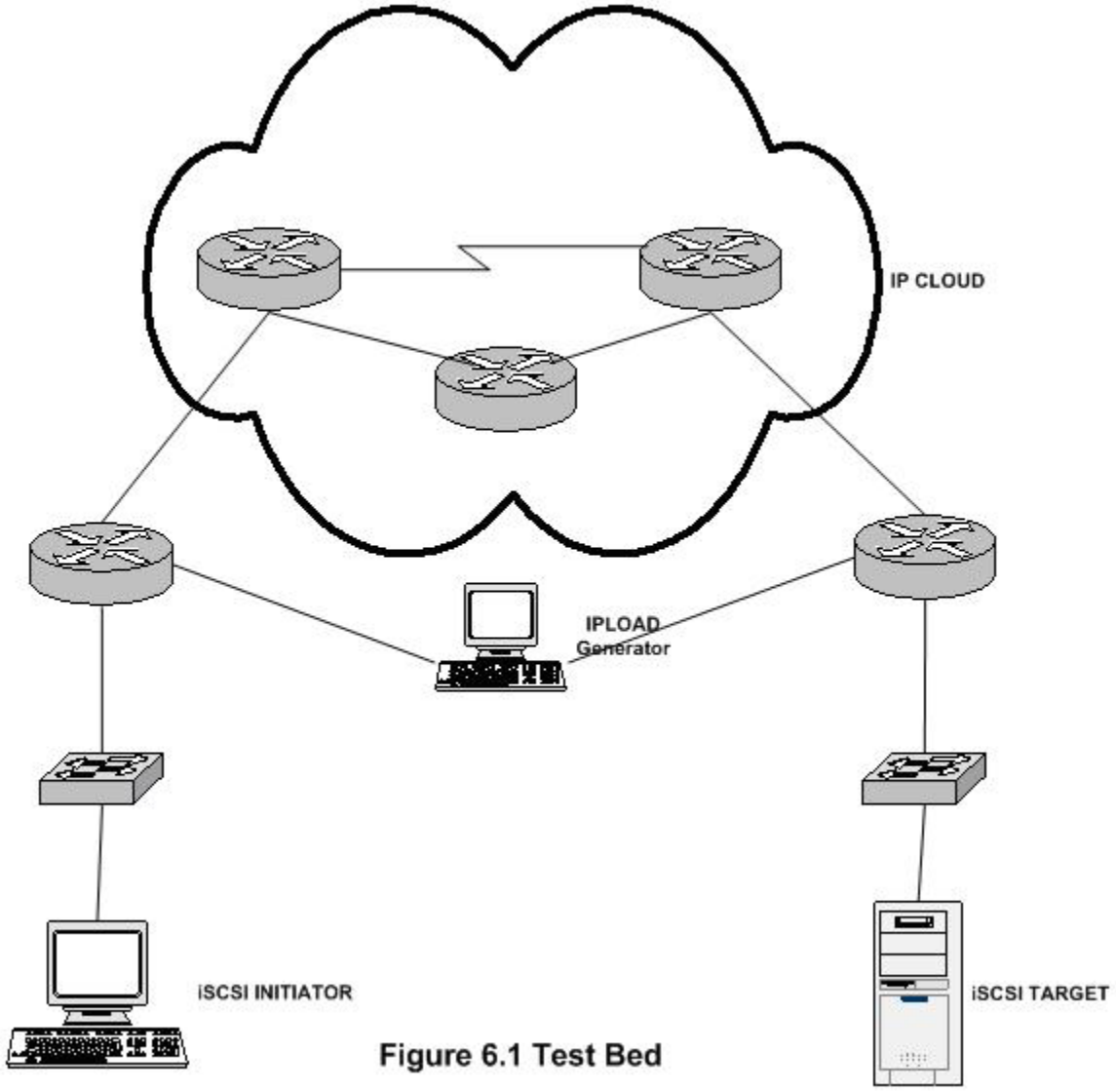

Figure 6.1 Test Bed

6.2 Test scenarios

In the test scenario, simulations have been performed under four different test conditions.

- Test bed with no other traffic,
- Test bed with other traffic as background traffic,

- Test bed with security in picture
- Test bed under different target volume sizes.

We also simulated scenarios considering iSCSI level errors and changing maximum burst length in Matlab. Each of the scenarios will be explained in detail followed by results and analysis. Test bed with no other traffic is basically a dedicated network between initiator and target, in this scenario the bandwidth between initiator and target is the minimum bandwidth of the links between them. We have considered the best and the worst cases, where the best case is using or gigabit Ethernet and the worst case is using serial link with 1.544 Mbps. In these scenarios we performed different tests running IOmeter on the initiator and performing I/O operations in the iSCSI disk, which is connected using Microsoft initiator. By varying different test conditions like read or write block size, specifying sequential or random read or write conditions and the test runtime to analyze various parameters. In this test our main goal is to acquire parameters like read response time, write response time and throughput. These parameters are validated with the modeled parameters. In the other test bed, traffic is introduced in to picture and analyzed all the parameters which were done in the previous scenario. In the same way, all the parameters are analyzed by introducing security into picture. IPSec transport mode with esp-3des encryption provides the best security to the data on the network. This involves some amount of processing time for encryption and decryption at both initiator and target. In the real world, sending data on the network without encryption is unsafe. All the previous parameters are analyzed with and with out security. iSCSI errors are modeled and considered to analyze the parameters like read response time, write response time and throughput using Matlab.

Session establishment time is the time taken by initiator to login to the target and gets basic volume information of the Logical Unit Number (LUN) of a given target, to show the LUN as if it is directly connected to the initiator. This involves series of commands and responses between initiator and target. Initiator firsts Login to the target by negotiating various protocol specific parameters. Once initiator and target reach the full featured phase. Initiator issues various commands to know about the LUN information of the target and reads the volume information form target. Once it receives the basic volume information, target shows up as being directly connected to initiator.

In this process of analysis, we observed various interesting things like session establishment time varies with various volume sizes. We analyzed this by using different volume sizes like, 512MB, 1GB, 2GB, 80GB, and 160GB on the target.

CHAPTER 7

RESULTS AND ANALYSIS

Initially author went through the IETF draft [1] for iSCSI and IETF draft for SCSI [26] and then came up with timing diagrams for the Read, and Write operations for modeling. For simulations of the prototype, author considered real time scenarios like the topology mentioned in the chapter 6, simulations were performed considering different parameters like block sizes of read or write, volume sizes (1GB, 2GB, 4GB, 40GB, and 160GB) of the target, Round Trip Times (RTT), and also introduced IPSec in the topology to get more realistic results with security. The following sections will provide the in detail explanation for simulations.

7.1 Session Establishment Time

In the research, we found an interesting thing with session establishment time, which varies with different volume sizes. Then we came to know that session establishment time not only includes the time taken for exchanging login parameters, but also includes time to read the volume information. The basic login session, which includes exchange of login parameters, is almost same for every target irrespective of the volume size, apart from this in the Session Establishment Time, the initiator reads the entire volume information of the target, which obviously depends on the volume size of the target device. In our analysis we considered different volume sizes and we analyzed the Session Establishment for each of them.

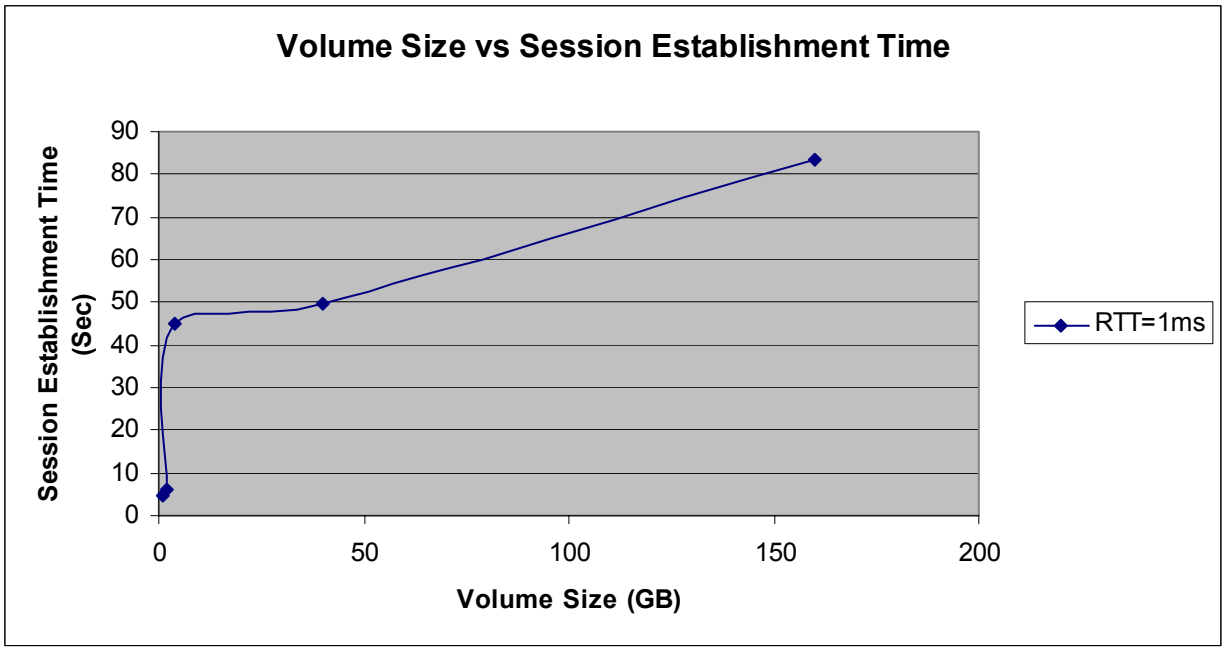

Figure7.1: Volume Size vs Session Establishment Time

From the figure 7.1 shows session establishments with respect to volume size. It can be analyzed from the results that the session establishment time increases with volume size. As per the observed, the smallest volume of 1GB takes around 4.55 Sec to complete session establishment, where as for the biggest volume size of 160 GB taking around 83.49 Sec.

The table 7.1 shows the Session Establishment Time with different Volume Size.

Volume Size(GB)	*Session Establishment Time (Sec)*
1	*4.55*
2	*6.32*
4	*45.179*
40	*49.89*
160	*83.49*

Table 7.1 Session Establishment Times ver Volume size

7.2 Read and Write Operation time

Figure 7.2, 7.3 demonstrates read and write times for the different block sizes, from this we can analyze that the read and write times were constantly increasing with block size. It was also observed that read and write operation time is increased with increasing RTT values.

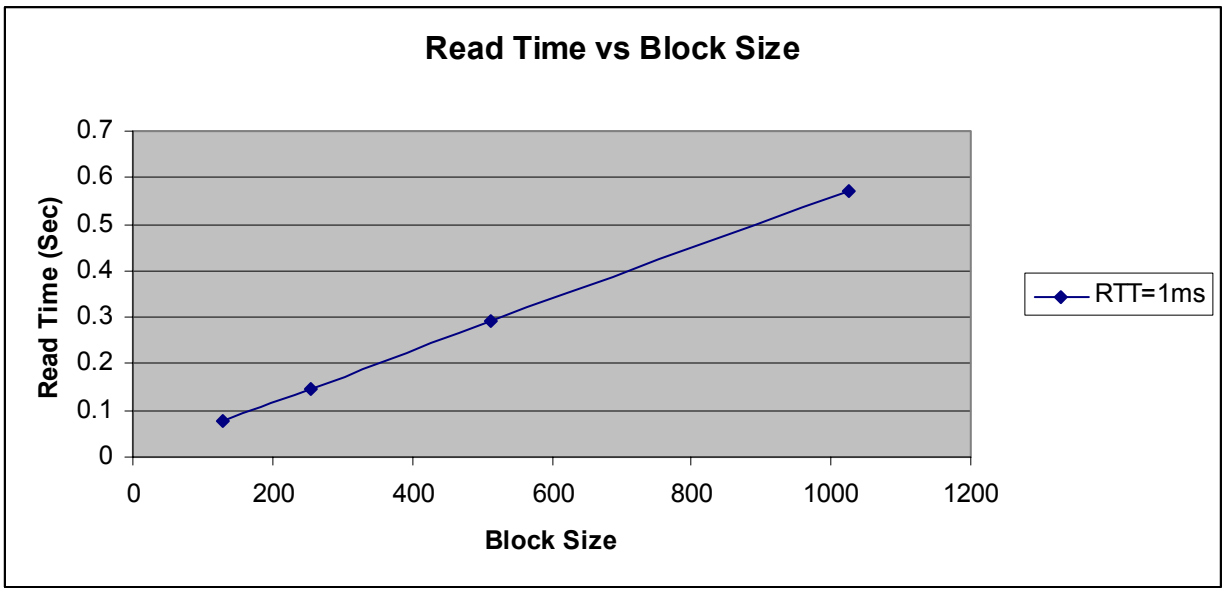

Figure7.2: Read Time vs Block Size

Figure 7.4, 7.5 analyses the results of read and write times with and with out encryption, which clearly shows that the read and write operation times drastically increases with introducing security in to picture. This is due to encryption time involved at initiator and decryption time at target. This encryption and decryption process is done for each and every IP packet that is generated due to these read and write operations, which effects the total time taken for the operations. It is clear from figure 7.4 and 7.5 that the total time taken for read and write operations increased by ten times.

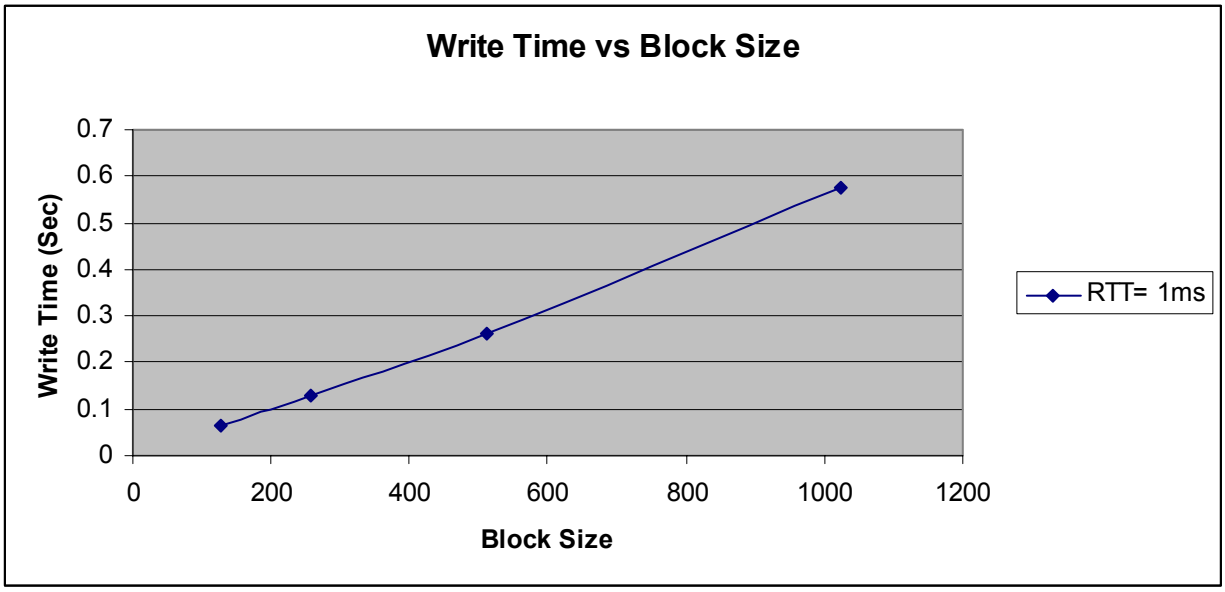

Figure7.3: Write Time vs Block Size

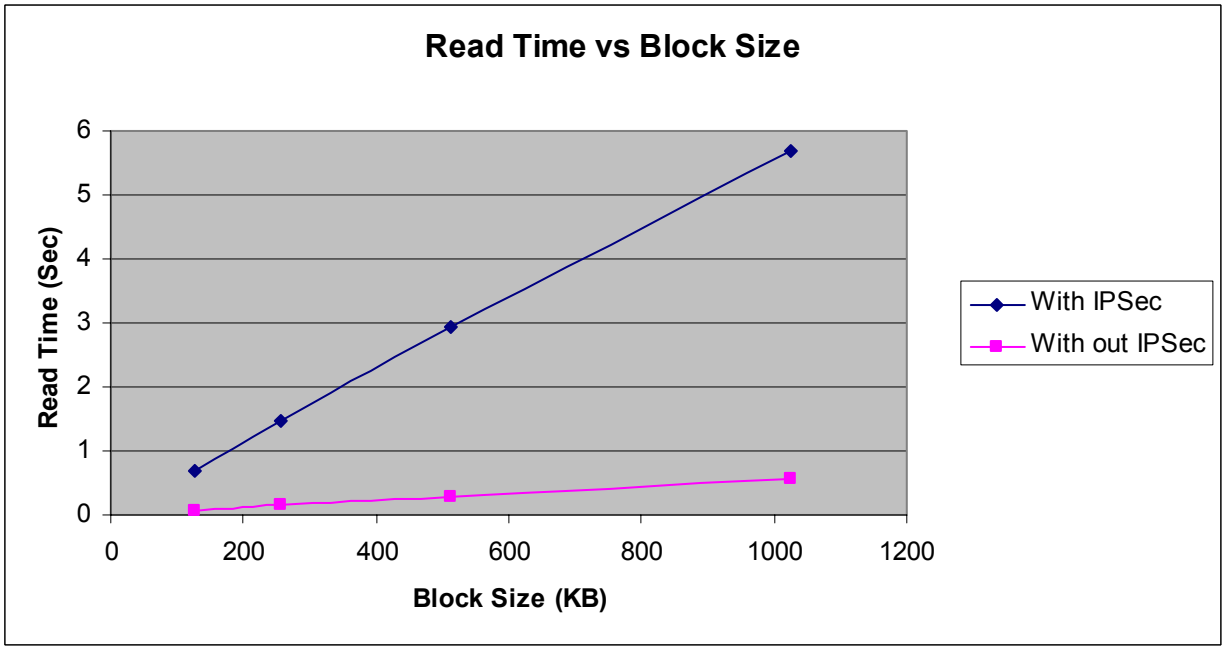

Figure7.4 : Read Time (with IPSec) vs Block Size

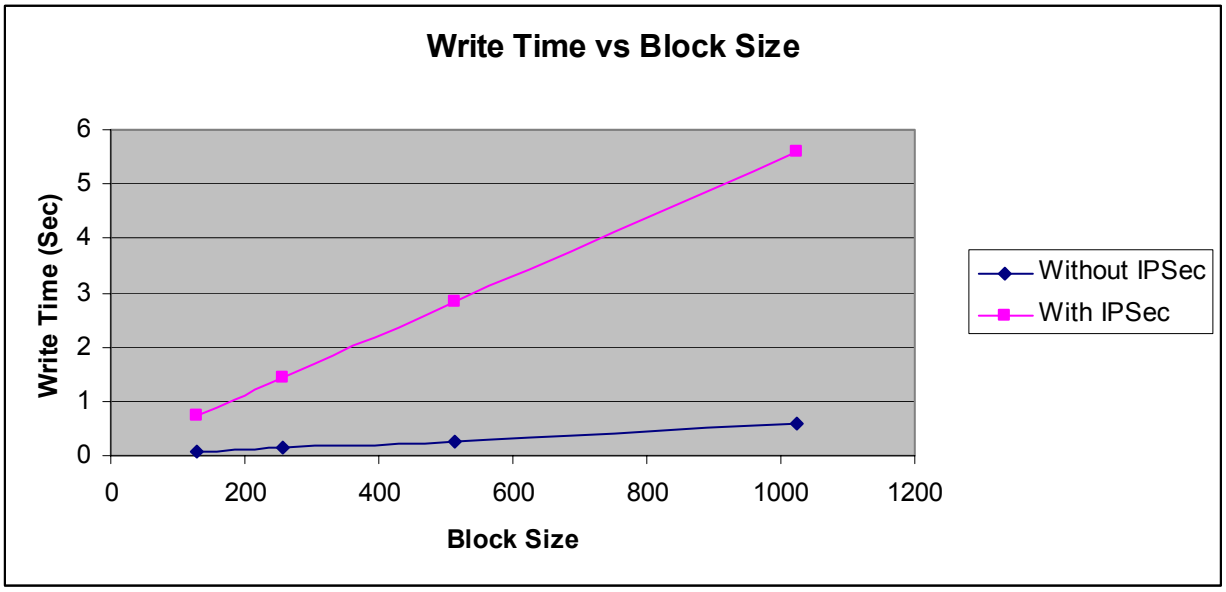

Figure7.5: Write Time (with IPSec) vs Block Size

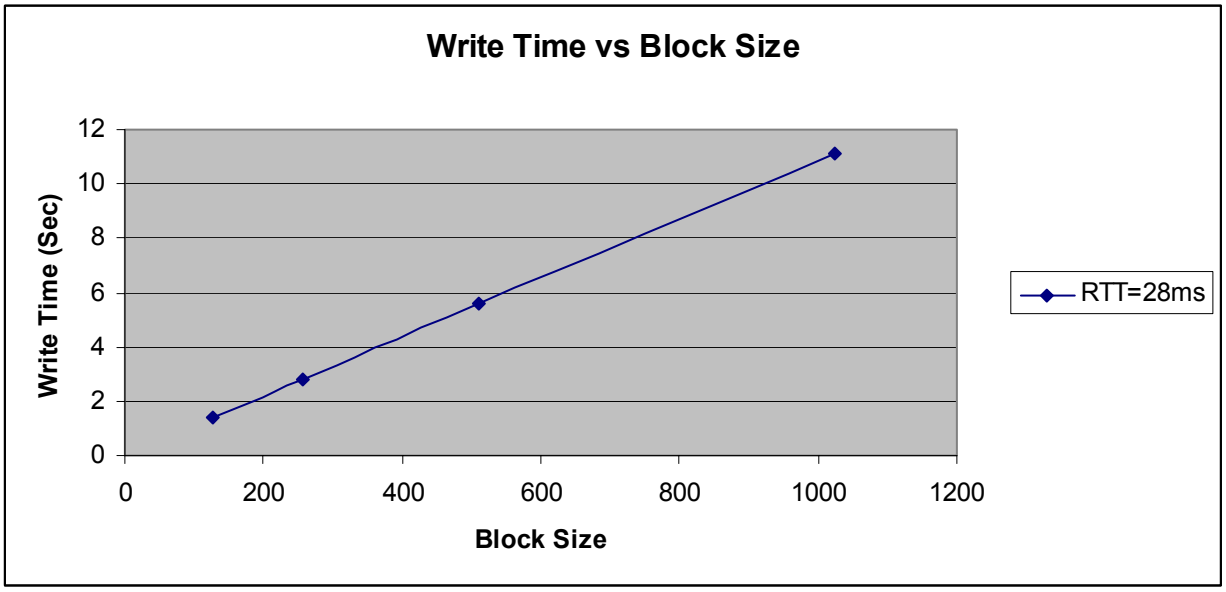

Figure7.6: Write Time vs Block Size (RTT=28ms)

When internet is used to transport iSCSI traffic, queuing delay of the packet increases because of other traffic, which in turn increases RTT. In this scenario IPload is used to generate TCP/UDP traffic flowing through the cloud, between initiator and target. In order to measure the

Read/Write times under real time IP traffic conditions; IP load was used to increase the RTT from 1ms to 28ms. Thus, the following figures 7.6 and 7.7 were obtained. The figures indicate that there has been a substantial increase in the Read/Write times when compared with dedicated networks. The results observed using the IOmeter were validated and verified with the mathematical model.

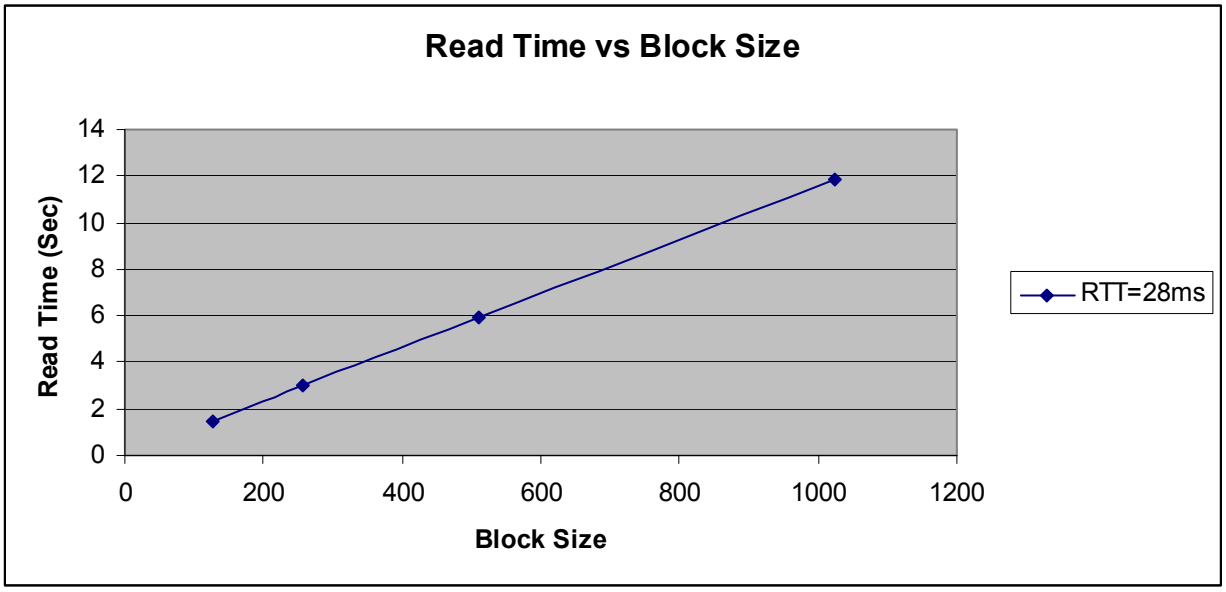

Figure7.7: Read Time vs Block Size (RTT=28ms)

Figure 7.8 analyses the read time with respect to maximum burst length. It was seen that read time drastically decreases with increase in maximum burst length from the simulations, it is because for each read command, initiator can read a maximum burst length of data, so for a given amount of data (1MB) the no of read commands and responses vary with different maximum burst lengths.

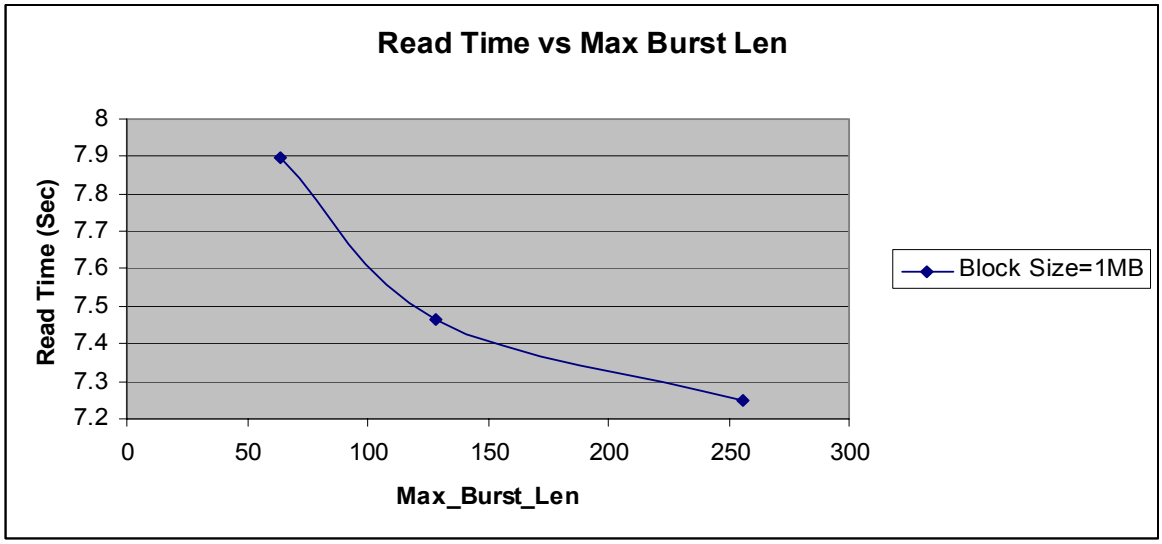

Figure7.8: Read Time vs Max_Burst_Len

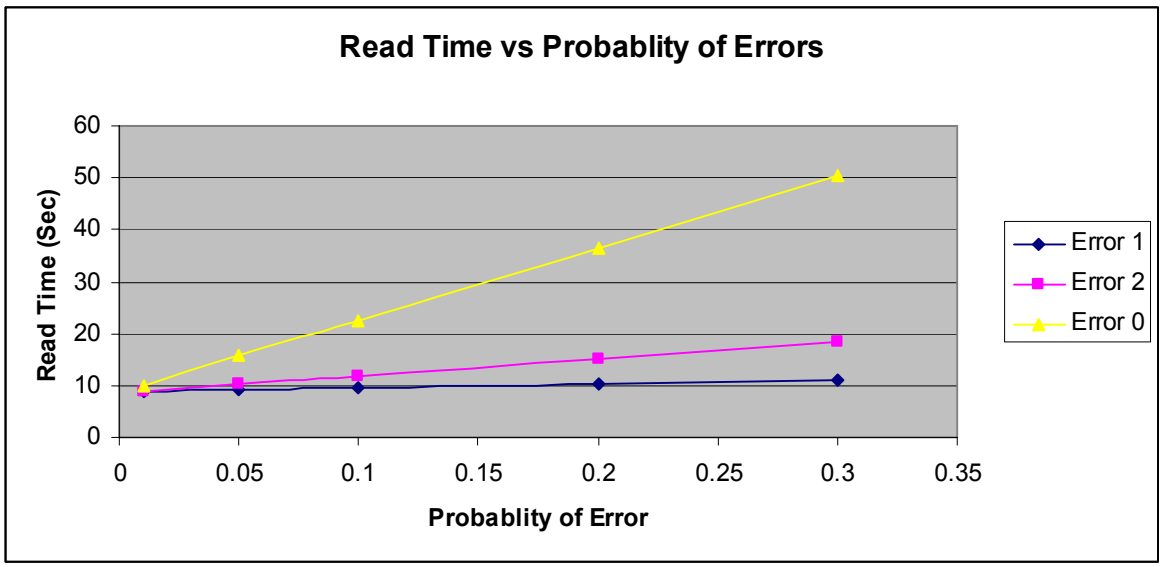

Figure7.9: Read Time vs Probability of Errors

Figure 7.9 shows the read response time with different levels of error recovery at different probabilities [29]. It was observed that time taken to recover iSCSI Errors using recovery Level 0 takes the maximum amount of time, considering the worst case (i.e. Full read time). Error recovery level 2 takes less time than recovery level 0, though it recovers from the worst cases like connection failures. Level 1 error recovery takes the least amount of time, which recovers the CRC errors.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 Conclusion

In this research, we modeled various iSCSI parameters, like iSCSI READ, iSCSI Write, throughput and analyzed them under various test scenarios like, under dynamic traffic conditions, by taking distance in to consideration, by considering TCP level errors in to consideration, for the first time we even considered the iSCSI level errors for analysis. Using this model one can analysis iSCSI parameters for their deployment model, before they actually deploy it. This gives approximate results before they decide to deploy. In this research we also considered security in to consideration, where the data is protected on the network. Using these results, we can say that parameters like iscsi read time, write time increase with security compared with out security. One can use different level of security depending on their requirements; we have performed these tests considering highest security. We also noticed that session establishment time vary with volume sizes, and performed tests with different volume sizes to analyze the session establishment times.

8.2 Future work

Other internet storage technologies like FCIP and iFCP are also used to interconnect san islands. So modeling of critical parameters like throughput, read time and write time in these technologies will enable one to analyze their requirements by comparing different technologies at one place. This will give a good idea for new deployers to select the technology for their requirements, before they deploy.

REFERENCES

LIST OF REFERENCES

[1] iSCSI Draft,
http://tools.ietf.org/id/draft-ietf-ips-iscsi-20.txt

[2] Trio demonstrates 'wire-speed' iSCSI SAN
http://www.iscsistorage.com/trioart.htm

[3] SCSI Architecture
www.t10.org/scsi-3.htm

[4] Starwind software
http://www.soft14.com/Authors/Rocket_Division_Software_4404.html

[5] Microsoft iscsi initiator
http://www.microsoft.com/downloads/details.aspx?familyid=12cb3c1a-15d6-4585-b385-befd1319f825&displaylang=en

[6] IPload
http://www.bttsoftware.co.uk/ipload.html

[7] iometer
http://www.iometer.org/doc/downloads.html

[8] Performance Evaluation of Commodity iSCSI-based Storage Systems.
Yingping Lu and David H.C. Du, University of Minnesota.
ieeexplore.ieee.org/iel5/9687/30571/01410745.pdf

[9] Storage area Network Extension Solution and Their Performance Assessment.
Radha Telikepalli, Tadeusz Drwiega, and James Yan, Nortel Networks
ieeexplore.ieee.org/iel5/35/28662/01284930.pdf

[10] Performance Evaluation of Commodity iSCSI-based Storage Systems
Dimitrios Xinidis and Angelos Bilas Institute of Computer Science (ICS)
Michail D. Flouris Department of Computer Science.
www.ics.forth.gr/carv/scalable/publications/iscsi_msst05.pdf

[11] A Performance Analysis of the iSCSI Protocol
Stephen Aiken, Dirk Grunwald, Andrew R. Pleszkun, Jesse Willeke, *University of Colorado*
ieeexplore.ieee.org/iel5/10777/33943/01620259.pdf

[12] Modeling and Performance Evaluation of iSCSI Storage Area Networks over TCP/IP-based MAN and WAN networks
C. M. Gauger, M. K¨ohn, S. Gunreben, D. Sass and S. Gil Perezy University of Stuttgart.

http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1589695

[13] iSCSI: The Universal Storage Connection By John L. Hufferd

[14] "iSCSI Protocol Concepts and Implementation," Cisco Systems.

[15] IP Storage Networking: Straight to the Core By Gary Orenstein

[16] Designing Storage Area Networks: A Practical Reference for Implementing Fibre Channel and IP SANs, Second Edition By Tom Clark

[17] Basics of SCSI
www.mskl.de/CONTENT/61/basics_of_scsi.pdf

[18] Reducing Storage Costs with an iSCSI SAN by Adaptec, Inc.

[19] Storage Area Network Architectures Technology White Paper by Heng Liao Technical Advisor.

[20] Storage Networking Protocol Fundamentals By James Long

[21] IP Storage Networking: IBM NAS and iSCSI Solutions by Rowell Hernandez
Keith Carmichael, Cher Kion Chai, Geoff Cole.

[22] Introduction to Storage Area Networks by Jon Tate Fabiano Lucchese Richard Moore

[23] Information technology -Small Computer System Interface – 2 by Lawrence J. Lamers Maxtor Corporation.

[24] Rfc 3720. Internet Small Computer Systems Interface (iSCSI) by J. Satran, K. Meth, IBM C. Sapuntzakis, Cisco Systems, M. Chadalapaka, Hewlett-Packard Co, E. Zeidner, IBM

[25] iSCSI : Networked Storage Without the Pain by: Robert C. Gray

[26] IETF Draft for SCSI
draft-ietf-ips-scsi-mib-02.txt

[27] Adaptec-Storage Area Networks, storage solutions white paper
http://www.adaptec.com/en-US/_whitepapers/tech/fcstor/san_sol_wp.htm

[28] Dot-Hill paper- Storage Area Networks: The Superior Storage Solution
By Omer Barazza and Ted Uhler.

[29] iSCSI error probability
http://www.pdl.cmu.edu/mailinglists/ips/mail/msg03573.html

[30] FCIP Internet Draft-Definitions of Managed Objects for FCIP
http://tools.ietf.org/id/draft-ietf-ips-fcip-mib-03.txt

[31] iFCP Internet Draft-iFCP - A Protocol for Internet Fibre Channel Storage Networking
http://tools.ietf.org/id/draft-ietf-ips-ifcp-03.txt