PERFORMANCE ANALYSIS AND DESIGN OF ISCSI OVER WIRELESS NETWORK

GAO YAN

(*B. Eng.(Hons.), Tianjin University*)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

Acknowledgments

I am sincerely grateful to my supervisors Dr. Zhu Yaolong and Dr. Liu Zhejie for giving me the privilege and honor to work with them over the last 2 years. Without their constant support, insightful advice, excellent judgment, and, more importantly, their demand for top-quality research, this thesis would not be possible.

I would also like to thank Yan Jie, Xiong Hui, Meng Bin and Renuga Kanagavelu for giving a necessary direction to my research and providing continuous encouragement throughout my M.Eng. This work would not be possible without a long-lasting support and infinite patience of Zhou Feng, Wang Chaoyang, So Lih Weon and Xi Weiya. Also Many thanks to Prof. Guan Yong Liang from Nanyang Technological University, Sim Chin San David and Li Zhixiang for their extreme generosity in providing the abundant resources needed for completing this M.Eng.

Furthermore, I would like to thank my friends Tian Tian, Sun Dongwei, Xiang Xu and Zhang Hua for always inspiring me and helping me in difficult times.

I am also thankful to IEEE International Conference on Networks (ICON 2004) reviewers for providing their helpful comments on this work.

Last, but not least, I would like to thank my parents. Without their continuous supports this work would be simply impossible.

To

Mom and Dad

With Forever Love and Respect

Contents

Summary

With the trend that wireless network and mobile devices have become more and more prevalent, there is an increasing need to build a wireless storage system that can access information efficiently and correctly. iSCSI (internet Small Computer Systems Interface) is a protocol to enable remote storage access through the ubiquitous TCP/IP network. The performance of iSCSI over wireless network is an interesting research topic due to the impacts of the low bandwidth, unreliability and long latency of the wireless network.

This thesis focuses on the performance analysis of block level storage protocol iSCSI over wireless network and the design of new iSCSI architecture to improve the wireless storage performance and network utilization of wireless network.

First, the theoretical analysis is focused on different factors which may affect the performance of wireless storage in different layers including MAC layer, TCP/IP layer, iSCSI storage layer and file layer. The TCP layer net throughput, the packet failure pattern and the impact of multi-hop wireless network on wireless storage performance are analyzed. The analysis shows that for small I/O request, the normal single connection iSCSI's throughput should be far less than the maximum throughput in theory and it is very low compared to big I/O request. This is because the time for the initiator to wait for the status before sending the next I/O request and many frames in the lower layer that do not sufficiently use the frame size.

Then a new iSCSI architecture is proposed with the concepts of multiple virtual TCP connections in an iSCSI session and parallel working mechanism in iSCSI layer over wireless LAN 802.11. The new iSCSI design not only improves the iSCSI performance by increasing the utilization of limited wireless network

bandwidth, but also provides a better mechanism to handle the packet failure in wireless channel and the long latency issues in multi-hop wireless network.

After that, the iSCSI prototype based on the proposed multiple connection design has been developed by Linux kernel level programming on commercial PC over wireless LAN 802.11b for performance analysis. The prototype is different from single connection implementation but is compatible with iSCSI standard. Some implementation issues such as login phase, information exchange, SCSI command implementation and semaphore implementation are also explained in detail.

Finally, various experiments are conducted to test the performance of the self-developed iSCSI prototype and normal single connection iSCSI. The test results show that multiple virtual connection iSCSI design for wireless storage can achieve significant throughput improvement for small I/O request (2K $\sim$ 8K). For example, for 2K request size, the multiple connection iSCSI can achieve 112% improvement. For big I/O request (128K), the maximum throughput can reach 0.62 MB/s, which is closed to the theoretical analysis result. In order to identify the key issues of the iSCSI performance, the experiments are also conducted to test the iSCSI performance with different network parameters, different network latency and different queue length. Experiment results show that multiple connection iSCSI can achieve high performance even in multi-hop, unreliable and long latency wireless network.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Network Storage

The information is exploding. According to Berkeley's research report, the amount of digital data stored doubles every 24 months [1]. The dramatic growing demand on storage puts a lot of challenges to network storage, management and security. These challenges make the storage architecture keep changing. Nowadays the network storage technology has become a hot research field in the world. The key issues of network storage technology include the following:

- **Access:** Keeping data on-line and accessible all the time.
- **Utilization:** Sufficiently utilizing the bandwidth and storage capacity.
- **Scalability:** Easily adding more storage when needed in the future.
- **Movement:** Ensuring data mobility and data sharing.
- **Security:** Ensuring data access with security.
- **Management:** Establishing a single point from which to manage growing complexity and heterogeneity.

Currently, there are three main network storage architectures: DAS (Direct Attached Storage), NAS (Network Attached Storage) and SAN (Storage Area Network) which are shown in Figure 1.1, 1.2 and 1.3 respectively.

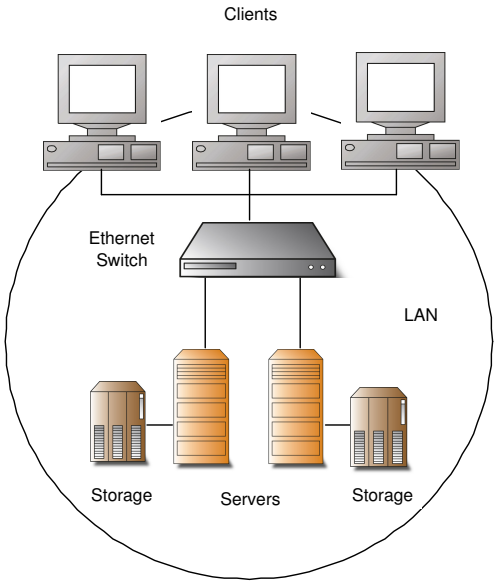

Figure 1.1: Direct Attached Storage

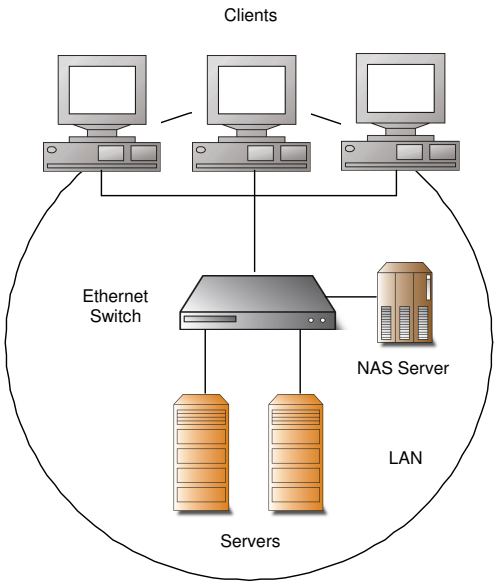

Figure 1.2: Network Attached Storage

- **DAS:** In DAS architecture, storage devices are directly attached to the server. Data are transferred using block level transfer protocol such as SCSI (Small Computer System Interface).

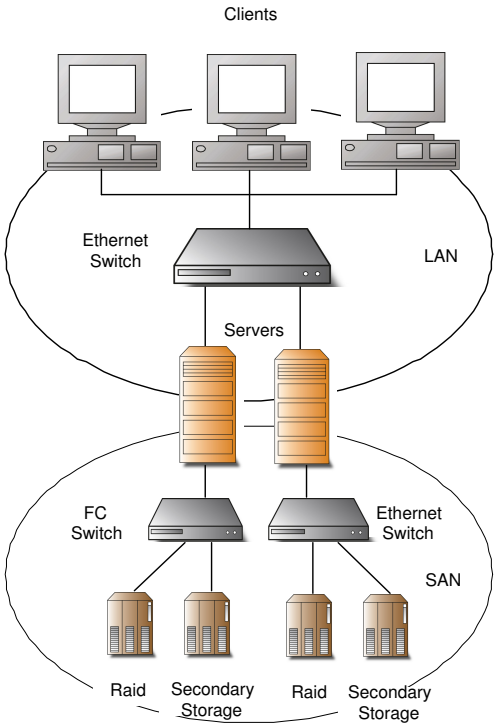

Figure 1.3: Storage Area Network

- **NAS:** NAS is a thin file server. The connection between NAS storage system and servers is based on IP network. Data are transferred using file level transfer protocol such as NFS (Network File System) and CIFS (Common Internet File System).

- **SAN:** SAN is a dedicated, high speed network among servers and storage devices. Data are transferred in block level. The current storage protocols include FC (Fiber Channel) and iSCSI (internet Small Computer Systems Interface).

Researches on network storage performance are emerging in recent years. Several papers have provided theory models and simulations [2, 3, 4, 5, 6, 7, 14] for improving the performance of the network storage. However their works are based on the communication networks such as Ethernet and FC (Fiber Channel) which are fast and reliable compared to wireless network. Until now, few research has been conducted to achieve high storage performance in wireless environment. This is a challenging topic due to the low bandwidth, long latency and unreliability of

wireless network.

1.2 Wireless Network Storage

The end user market for portable and mobile devices keeps increasing every year. The mobile devices such as notebook computer, PDA, mobile phone, tablet PC and even digital video and camera, are playing more and more important role in people's life. Compared to their desktop counterparts in wired environment, mobile devices have created new challenges [8, 9, 10, 11, 12] for data accessibility, security and transmission due to the relative low bandwidth, unreliability and the limited storage capacity. With more and more implementations of wireless network and mobile computing, there is a need to build a wireless storage system that can access information efficiently and correctly.

The objective of wireless storage is to provide a fast, stable and abundant external storage system to support a variety of wireless interfaces and reach out to as many different types of portable devices as possible for wireless users. The wireless storage can be designed in different machines based on different protocols and interfaces as shown in Figure 1.4.

Although mobile computing offers the benefits of having their work and data available at all times, there are still risks in terms of data reliability and dependability in the event of a device related failure [13]. The wireless environment's features such as low transmission rate, frequent packet loss, unreliable transmission and long latency make great impact on the network storage system and create new challenges for the performance of data accessibility. Mobile devices exhibit major differences from their desktop counterparts. They may travel across a variety of networks, each with its own bandwidth limitation and quality of service. Typically, they are designed with mobility in mind and therefore must sacrifice their performance leaving the limited connectivity due to the unreliability of wireless network and the limited data transmission rate which is due to the low bandwidth.

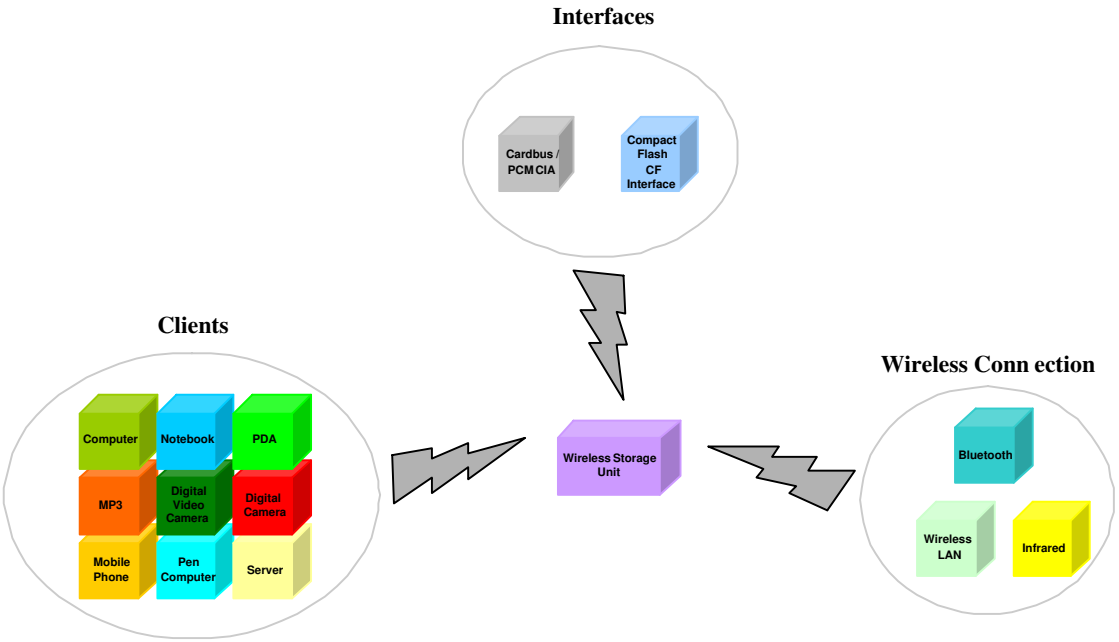

Figure 1.4: Wireless Storage

In order to solve above mentioned challenges and problems, the current research field of wireless storage can be separated to two subfields. The first one is to address accessibility problem to achieve high data accessibility under different quality of network service. The second one is to deal with transmission problem to achieve high storage performance and network utilization under wireless network's limited bandwidth.

1.3 Organization

The remainder of this thesis is organized in six chapters. The next chapter, Chapter 2, provides the background for the specific wireless storage problem addressed in this thesis. The related works in wireless storage are reviewed and the relative merit of this thesis is highlighted. IP storage and related work in iSCSI are also introduced in this chapter.

Chapter 3 presents the theoretical analysis of storage performance over wireless network. First, the lower layers such as MAC layer and TCP/IP layer are

discussed. The TCP layer net throughput, packet failure pattern and TCP performance of Multi-hop channel in wireless LAN 802.11 environment are also analyzed. Then, the analysis of upper storage layers such as block level iSCSI and file level NFS are conducted.

Chapter 4 focuses on the new iSCSI design. The basic concepts and the general iSCSI storage model are illustrated. Then, the different phases of the iSCSI design such as login phase and full feature phase are explained in detail. The new concept of multiple virtual TCP connection is presented. Finally the queuing models are used to analyze the new iSCSI architecture.

Chapter 5 focuses one some implementation issues of the multiple virtual TCP connection iSCSI design. The experiment methodology and its system setup are also illustrated.

Chapter 6 presents the performance evaluation of iSCSI and result discussion. The performances of iSCSI and NFS are compared. Next, test results of the multiple connection iSCSI are compared with that of normal single connection iSCSI in both wired and wireless environment. The results illustrate that the new iSCSI design can achieve high performance in wireless environment. The analysis of the multiple connection iSCSI performance under different network conditions and different network parameters are also conducted to identify some key issues of iSCSI performance.

Finally, the conclusion of this thesis and the discussion of future research are presented in Chapter 7.

Chapter 2

Background

In this chapter, some previous works in wireless storage are reviewed. The comparison to the work done in this thesis is highlighted to emphasize the contributions. Then iSCSI protocol and related works of iSCSI are analyzed. The analysis and design of the multiple connection iSCSI to solve wireless storage issue are also illustrated. Finally, the motivation of this work is presented.

2.1 Current Status of Wireless Storage

Although wireless storage did not attract as much attention as what did in wired environment, the pervasiveness of mobile devices and wireless network has led to the increased researches in the area of wireless storage.

Researchers have proposed a variety of solutions to deal with the first subfield of wireless storage as what is discussed in Section 1.2, which is the mobile data access, in file level. The approaches to mobile data access can be divided into three phases: preparing for disconnection, disconnected operation, and update propagation and conflict resolution, which are shown in Figure 2.1.

The main method of preparing file for disconnection is to cache part or all of the data on servers before disconnection, which can be divided into predictive

caching (also known as pre-fetching) and hoarding. Lei and Duchamp designed a system [15] by which the file system would analyze the computing activities of a user and selectively pre-fetch files. The paper [16] provided aggregating cache which can be used to reduce the number of file retrieval requests made by a caching client. In UCSC, the researchers introduced a file pre-fetch method [17] by using actions from prior events. As to the hoarding method, Kistler studied disconnected operation in Coda file system in his Ph. D. dissertation [18]. Also in paper [19, 20, 21, 32, 33, 34, 35], the authors provided various hoarding methods for mobile computers.

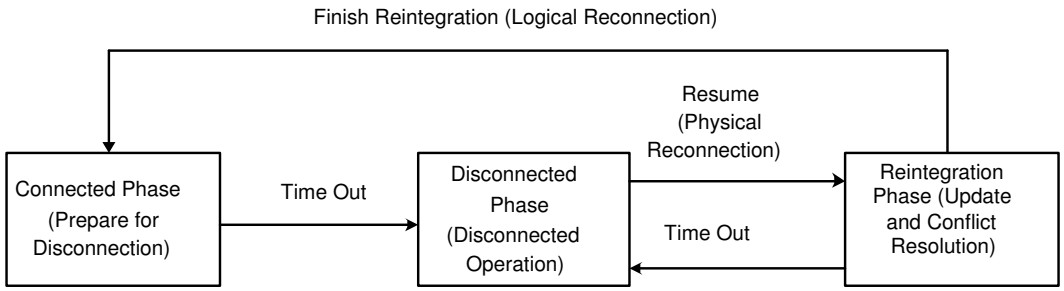

Figure 2.1: Phases for Mobile Data Access Solutions

During disconnected phase, the client should work based on the local copy of the data. Paper [22] provided a cache policy based on semantics. A research group investigated how standard CORBA mechanisms (Objects By Value and Portable Interceptors) can be used for enhancing legacy CORBA based distributed application to support disconnections [23]. The paper [24] extended the Coda file system to handle the cache misses on isolated clients in wireless environment. Cache cooperation for clustered disconnected computers [25] was proposed by Kinuko Yasuda, which resolved cache misses in a coordinated manner among the clients. Cache Cooperation allows each client to retrieve missing data from the other remote clients.

During reintegration phase, any operations during disconnected phase must be propagated to the server side, which may introduce the cache inconsistency between client and server side. If two mobile clients reside in different network partitions and they modify the same data object, data conflict will occur. Some

researches focused on the topic of conflict resolution. In Chinese University of Hong Kong, Dr. Liu developed an open platform [26] which can resolve the above mentioned problem. The paper [27] provided a propagation algorithm to support disconnected write operations for update propagation. Paper [28] introduced the parameter of conflict probability to submit the propagation to the server and transfer transactions in groups. There are other researches on propagation and conflict resolution [29, 30, 31].

The researches for mobile data access have been conducted for years. Various algorithms, systems and methods, which were mentioned above, were proposed to deal with mobile data access problem, especially for disconnected operation, in the wireless storage. In addition to the mobile data accessibility, achieving high network utilization, high data transmission rate and better storage performance are also critical issues in wireless storage because the mobile users also want their mobile data to be transferred as fast as possible. However, so far no research work has been reported in literature which exclusively focused on the second subfield of wireless storage as what is mentioned in Section 1.2, how to sufficiently utilize the limited bandwidth of wireless network to achieve high storage performance over wireless network. Nevertheless, there are some researches reported in literature which have attempted to distribute or parallelize the transmission to achieve performance enhancement either at the lower layers of wireless network or at the storage level over wired network. Kim and Lee presented a parallel transmission of DS/CDMA signal by orthogonal bases and repeated spreading - the chip-spreading OCDM [49]. The result shows that the bit error rate of the proposed system is much lower than that of the conventional DS/CDMA system which uses the maximal ratio combing. Reference [50] presented a prioritized parallel transmission MAC protocol and conducted performance evaluation in a simplified All-IP wireless WAN system. Reference [51, 52] also reported the researches on parallel transmission over mobile communication system or ATM connections. Data Storage Institute Singapore proposed a multiple addresses parallel transmission architecture for stor-

age area network to improve the storage performance [53]. Reference [54] presented a distributed parallel storage system which can achieve scalability and high data throughput over wired network. There are other researches on how to use parallel and distributed mechanism to address storage problem over wired network [55, 56]. The above mentioned solutions to address transmission problems provided us valuable clues to address transmission problem of storage data over wireless network. The current status of wireless storage makes us pay the attention to this new and valuable field of wireless storage. In order to achieve high storage performance and sufficiently utilize the limited bandwidth in wireless environment, the block level data access approach such as iSCSI [36] is the better choice than file level data access approach such as NFS [73], which will be discussed in detail in Section 3.5 and 3.6. However, so far no research work has been reported in literature focusing on the block level data storage, such as iSCSI, over wireless network.

2.2 Storage Protocol

2.2.1 IP Storage

IP storage refers to a group of technologies that allows block-level storage data to be transmitted over an IP-based network. There are two key concepts in this definition: "the use of IP" and "block-level storage". Transferring block level storage data over a network topology is not a new concept. Today's SANs use the Fibre Channel (FC) technology to do just that. The promise of the new IP storage protocols is the interconnection, as well as the complete construction of these SANs with prevalent IP enabled technologies such as Ethernet (802.3) or Wireless LAN (802.11). The use of IP to transfer data is also not a new concept. Familiar protocols such as CIFS (Common Internet File System) and NFS (Network File System) have been used to access file level storage data over IP networks for years. The difference between these protocols and the IP storage protocols lies in how the

data is accessed: at the"file level" or at the"block level". iSCSI is a block level storage protocol while NFS or CIFS is a file level storage protocol.

2.2.2 Introduction to iSCSI

The iSCSI (internet Small Computer Systems Interface) [36] protocol is based on SCSI, which is used on host computer systems to perform block data input and output with various peripheral devices. These devices include disk and tape devices, as well as printers and scanners.

In iSCSI, SCSI commands are sent over TCP/IP, which is widely used in corporate networks. Hence, iSCSI defines a means to enable end-to-end block data transfer between targets and initiators over an IP network. With the widespread use of IP based network (Ethernet and Wireless LAN) and the SCSI command set being used throughout all storage configurations, iSCSI is set to become the protocol of choice in storage solutions.

In addition to iSCSI, several other protocols have been defined to transport storage over an IP network such as FCIP (Fiber Channel over TCP/IP) [37] and iFCP (internet Fiber Channel Protocol) [38]. Whereas FCIP and iFCP are used to allow the connection of existing Fibre Channel infrastructures to each other and to IP networks, iSCSI enables the creation of SANs completely independent of Fibre Channel.

iSCSI has attracted a lot of attention recently due to the following advantages and features [48]:

- **Good scalability:** iSCSI is based on SCSI protocol and TCP/IP network, which can provide good scalability.
- **Low cost:** iSCSI can share and be compatible with existing TCP/IP networks. The user does not need to add any new infrastructure hardware.

- **Remote data transferring capability:** TCP/IP network can extend to metro area, which makes iSCSI suitable for remote backup and disaster recovery applications.

The iSCSI based storage is quite different from a traditional one. A traditional storage system is often physically restricted to a limited environment, e. g. in a data center. It also adopts a transport protocol specially tailored to this environment, e. g. parallel SCSI, Fiber Channel, etc. While in an iSCSI storage, the transport is no longer restricted to a small area. The initiator and the target can be far away from each other. The network technology in between can be diverse and heterogeneous. The network condition can be congested and dynamically changing. Thus, the iSCSI storage solution deserves more careful design, implementation and performance analysis.

2.2.3 Related Works of iSCSI

Most current iSCSI implementations use software solutions in which iSCSI device drivers are added on top of the TCP/IP layer for off-the-shelf NICs (Network Interface Cards). It may cause performance issue. Many researches and projects have been carried out to analyze and implement iSCSI. A prior research project of iSCSI - Netstation project of USC showed that it was possible for iSCSI to achieve 80% performance of direct-attached SCSI device [39]. IBM Haifa Research Lab carried out research on the design and the performance analysis of iSCSI [40, 41]. Bell Laboratories also did some test and performance study of iSCSI over metro network [42]. University of Minnesota carried out research on the performance analysis of iSCSI [43]. University of Colorado also did some tests and performance studies of iSCSI in both hardware and software [44]. A research group proposed a solution to use a log disk along with a piece of non-volatile RAM to cache the iSCSI traffic [45]. Other solutions included using a TOE (TCP/IP Offload Engine) [46] and even iSCSI adapter [47] to reduce the burden of host CPU by offloading

the processing of TCP/IP and iSCSI protocol into the hardware on the network adapter. But these hardware solutions will add extra cost compared to a software solution.

Until now, the researches and implementations of iSCSI are based on FE (Fast Ethernet) and GE (Gigabit Ethernet). No research work has been performed that focused on the performance of iSCSI over wireless environment. Because the wireless LAN 802.11 is also an IP based network, iSCSI can work properly and does not need to care about whether the low layer is run on Ethernet or wireless LAN. But due to the limited bandwidth of wireless network, how to optimize the utilization of the limited bandwidth to achieve high storage performance is an interesting topic.

2.3 iSCSI over Wireless Network

Currently, people rarely explore running block level storage such as iSCSI over wireless network because the performance would be a critical issue due to the inherent characters of wireless network.

Current iSCSI design and implementation is sensitive to the network reliability and latency. In low speed and unreliable wireless environment, the normal iSCSI may encounter serious performance problem caused by packet failure and long latency. The initiator need to wait for a long time to receive the ACK status from the target before issuing the next I/O request.

In some scenario, the small I/O request is sent frequently compared to big I/O request, more time is wasted when waiting for the ACK status. And for small I/O request, if the request size can not be divided exactly by the maximum frame size, there will be a lot of 802.11 frame that do not sufficiently utilize each frame size. Thus the storage performance problem of iSCSI over wireless network is even worse.

The key problem of wireless storage is how to optimize the storage communication mechanism based on the fundamental features of wireless network which is much more unreliable and has longer latency than wired network. This thesis addresses this issue by designing an new iSCSI architecture with multiple virtual TCP connection.

With the multiple virtual TCP connection design, the iSCSI initiator may utilize multiple virtual TCP connections to request or send data from or to the target and do not need to wait for the ACK status before sending the next I/O request. Such an iSCSI design may achieve high network utilization and better storage performance for iSCSI over wireless network.

Chapter 3

Theoretical Analysis of Wireless Network and Storage

Since the overall research field of this thesis is wireless storage, before the design and implementation of the wireless storage system, the general analysis of the characteristics of the lower layers such as MAC layer and TCP/IP layer, and the upper storage layers such as block layer and file layer is necessary. The storage system is built on top of the wireless network, thus the characteristics of wireless network such as the TCP net throughput, the packet failure pattern and the multi-hop network make great impacts on the upper storage system. The most pervasive standard of wireless LAN, 802.11b, is used for the purpose of analyzing the lower layer. The storage system in the upper layer can only be designed and implemented, which is the main focus of this thesis and will be discussed in the following chapters, to adapt the wireless environment very well with the understanding of wireless network in this chapter. The theoretical analysis of the block level and file level storage is also conducted to provide the theoretical ground. Special care has been taken to carefully choose the storage protocol to achieve high network utilization and storage performance.

3.1 Introduction of Wireless LAN 802.11

IEEE 802.11 [57] is a standard for wireless systems, it focuses on the MAC layer and PHY layer for access point based networks and ad-hoc networks. Although industry is seeking data rates as high as possible, even over 100 Mbps data rates [59, 60, 61, 62, 63], 802.11b [58] is the most pervasively used standard nowadays. The 802.11b supports DSSS (Direct Sequence Spread Spectrum) in the 2.4GHz band with bit rates of 1, 2, 5.5 and 11 Mbps. The last two bit rates are achieved through CCK (Complementary Code Keying). Different from Ethernet 802.3, IEEE 802.11 employs a CSMA/CA (Carries Sense Multiple Access with Collision Avoidance) protocol with binary exponential backoff, called DCF (Distributed Coordination Function). DCF defines a basic access mechanism and RTS (Request To Send) and CTS (Clear To Send) mechanism.

The Communication process of 802.11 is shown in Figure 3.1. A station with a packet to be transmitted monitors the channel activities until an idle period equal to a DIFS (Distributed Inter Frame Space) is detected. After sensing an idle DIFS, the station waits for a random backoff interval before transmission. The backoff time counter is decremented in terms of slot time as long as the channel is sensed idle. The counter is stopped when a transmission is detected on the channel and reactivated when the channel is sensed idle again for more than a DIFS. The station transmits its packet when the backoff time reaches zero. At each transmission, the backoff time is uniformly chosen in the range (0, CW-1), where CW is the current backoff window size. At the very first transmission attempt, CW equals to the minimum backoff window size. After each unsuccessful transmission, CW is doubled until a maximum backoff window size value is reached. After the destination station successfully receives the packet, it transmits an ACK (Acknowledgment) packet following a SIFS (Short Inter Frame Space) time. If the transmitting station does not receive the ACK within a specified ACK Timeout, or it detects the transmission of a different packet on the channel, it reschedules the packet transmission

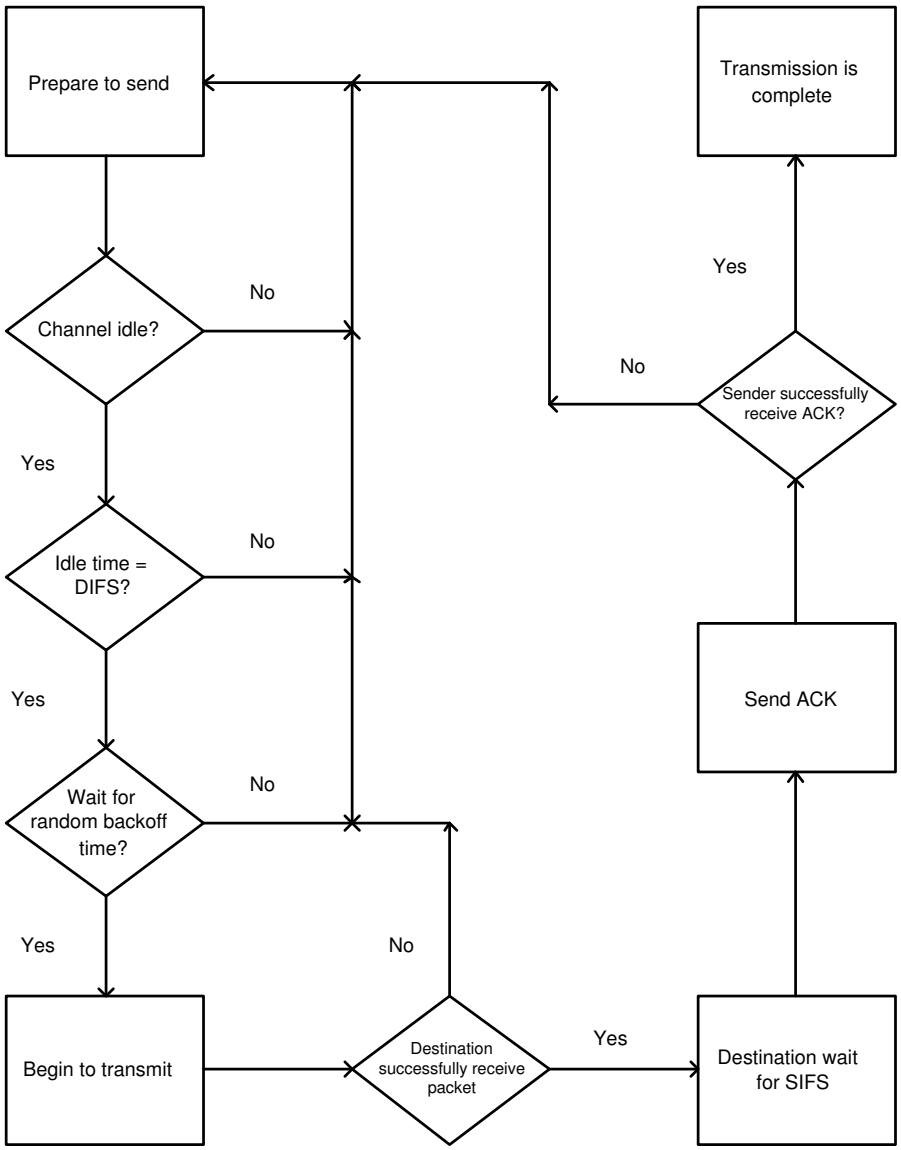

Figure 3.1: Communication Process of 802.11

according to the previous backoff rules. For more detail, please refer to [57].

3.2 TCP Layer Net Throughput Analysis of 802.11b

3.2.1 Net Throughput Introduction

The theoretical modelling to evaluate the throughput in a LAN is often based on a large number of stations. Each station provides a traffic load, which in theoretical analysis is often characterized by Poisson arrival process with a certain

average arrival rate and negative exponential packet size distribution. In practice the number of active stations is limited and gives rise to a bursty packet generation process for most applications each station gives. Therefore, the measurement of net throughput in a wireless LAN should be done by registration of the time it takes to transfer large files between the server and wireless clients. The net throughput metric is based on the file transfer time. The effective net throughput depends on the data rate, but there is a lot of overhead, for example, the preamble of the transmitted frame, the MAC header, ACK frames, transmission protocol overhead and processing delay in local and remote computer. Then the net throughput of 802.11b is far less than the raw data rate, which is 11 Mbps.

3.2.2 802.11 Overhead

IEEE 802.11 defines the frame structure and MAC scheme. Thus the overhead involved in the PHY and MAC can be simply modelled and the maximum MAC layer throughput can be calculated. The maximum MAC layer throughput reflects the effective payload transfer with a continuous stream of packets with a payload of 1,500 bytes in one direction and providing an ACK in the other direction. Figure 3.2 illustrates the overhead of 802.11. The overhead and throughput with consecutive 1,500 byte payload data frames for 802.11 can be determined which reflects the MAC layer's throughput limitation.

3.2.3 MAC Throughput Calculation

To derive the MAC throughput, the system should be at the best scenario: 1) the channel is an ideal channel without errors and 2) at any transmission cycle, there is one and only one active station which always has a packet to send. In a real channel, the real throughput should be less than net throughput. The notations of the calculation are shown in Table 3.1.

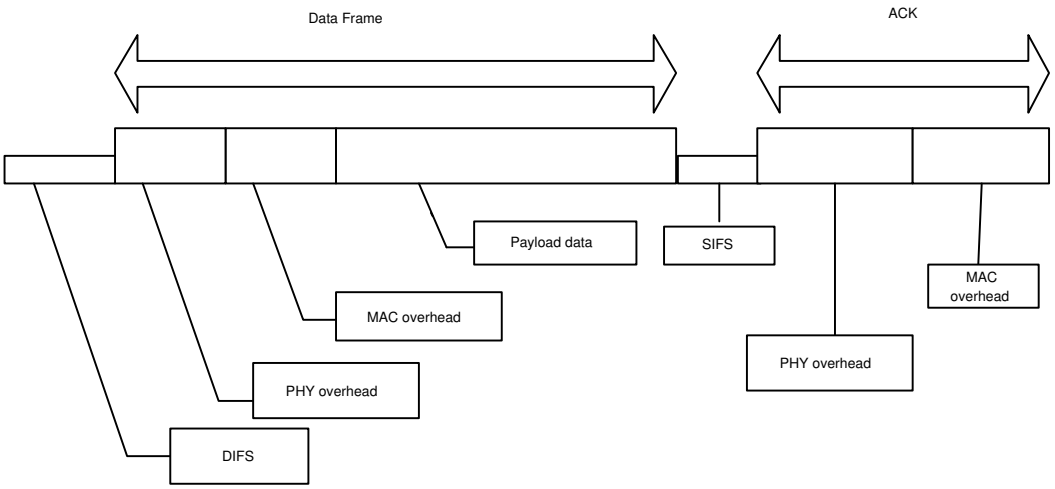

Figure 3.2: Overhead of 802.11

A transmission cycle of DCF (Distributed Coordination Function) consists of DIFS deferral, backoff time, data transmission, SIFS deferral and ACK transmission.

The average backoff time is denoted as:

$$\overline{CW} = \frac{CW_{min} \cdot T_{slot}}{2} \tag{3.1}$$

The data transmission delay is

$$T_{Delay\_DATA} = T_p + T_{PHY} + T_{MAC} + T_{DATA} \tag{3.2}$$

The ACK transmission delay is

$$T_{Delay\_ACK} = T_p + T_{PHY} + T_{MAC} + T_{ACK} \tag{3.3}$$

Then the MAC throughput can be expressed as:

$$Throughput = \frac{8L_{DATA}}{T_{Delay\_DATA} + T_{Delay\_ACK} + 2\tau + T_{DIFS} + T_{SIFS} + \overline{\overline{CW}}} \tag{3.4}$$

Table 3.1: The Notations of 802.11b MAC Throughput Calculation

T_{slot}	A slot time
T_{SIFS}	SIFS time
T_{DIFS}	DIFS time
CW_{min}	Minimum backoff window size
T_p	Transmission time of the physical preamble
T_{PHY}	Transmission time of the PHY header
L_{MAC}	MAC overhead in bytes
L_{ACK}	ACK size in bytes
T_{MAC}	Transmission time of MAC overhead
T_{ACK}	Transmission time of ACK
L_{DATA}	Payload size in bytes
L_{DATA_TCP}	Payload size with TCP hearer in bytes
T_{DATA}	Transmission time of payload
τ	Propagation delay
R_{DATA}	Data rate
R_{ACK}	Control rate

For IEEE 802.11, the transmission time equals to the ratio of the packet size and the transmission rate, then

$$T_{Delay\_DATA} = T_p + T_{PHY} + \frac{8L_{MAC} + 8L_{DATA\_TCP}}{R_{DATA}} \tag{3.5}$$

$$T_{Delay\_ACK} = T_p + T_{PHY} + \frac{8L_{MAC} + 8L_{ACK}}{R_{ACK}} \tag{3.6}$$

Thus the MAC throughput can be determined by the following formula:

Throughput=

$$\frac{8L_{DATA}}{T_p + T_{PHY} + \frac{8L_{MAC}+8L_{DATA\_TCP}}{R_{DATA}} + T_p + T_{PHY} + \frac{8L_{MAC}+8L_{ACK}}{R_{ACK}} + 2\tau + T_{DIFS} + T_{SIFS} + \overline{CW}} \tag{3.7}$$

From the 802.11 specification [57, 58], the following parameters of 802.11b can be gotten, which are shown in Table 3.2.

Usually the packet size is 1,500 bytes, but the TCP header consumes 40 bytes, thus the real payload size reduces to 1,460 bytes. From equation (3.7) with

Table 3.2: The Parameters of 802.11b

Parameter	Value
T_{slot}	20 μs
T_{SIFS}	10 μs
T_{DIFS}	50 μs
CW_{min}	31
T_p	144 μs
T_{PHY}	48 μs
L_{MAC}	34 bytes
L_{ACK}	14 bytes
τ	1 μs

parameters in Table 3.2, the MAC throughput of wireless LAN 802.11b with default packet size of 1,500 bytes can be calculated that approximately equals to 6.1 Mbps.

3.2.4 TCP Layer Net Throughput of 802.11b

The above MAC layer throughput does not encompass the overhead of transport layer (TCP). The overhead of TCP layer include the header overhead and the overhead caused by the flow control mechanism. So the total net throughput with TCP is about 15% to 20% lower than the numbers based on the MAC layer which is calculated above, ignoring the overhead from transport protocol (included in packets) and request response frames (extra short frames) [64].

From what has been discussed above, the conclusion can be drawn that the net throughput with TCP of 802.11b is around 45% to 47% (4.9 Mbps to 5.2 Mbps or 0.61 MB/s to 0.65 MB/s) with default 1,500 bytes packet size of the raw data rate which is 11 Mbps.

This theoretical calculation result of TCP layer net throughput is the actual throughput that the TCP layer can provide for the upper block layer and file layer. This calculation result is verified by the experiment that will be discussed in detail in Chapter 6. The throughput of the storage system designed and implemented in this thesis is also compared with this TCP layer net throughput calculation result in Chapter 6 to highlight the advantage of the proposed storage system which will

be discussed in the following chapters.

3.3 Packet Failure Analysis of Wireless LAN 802.11

Wireless LAN 802.11 uses CSMA protocol. In CSMA protocol, if there are multiple stations that use the protocol to share a wireless channel without a coordinating base station, the packet transmission failure may occur due to the hidden terminal problem, which is when carrier sensing is not perfect, a node may not hear another node even when they are in the same cell. In such case, the packet collision occurs due to the contending of the wireless channel from stations. This problem has been studied in [65, 66].

There are the other reason for packet transmission failure which is packet error. The packet error refers to the packet transmission failures between a pair of wireless stations, which are due to reasons other than collisions. Packet errors usually occur due to non-ideal channel condition [67]. Multi-path fading, ambient noise, partition loss and decrease SNR (Signal to Noise) may cause packet errors. Packet errors cause packet retransmission at sender station and reception errors at receiver station. When the reception error occurs, unless the receiver station receives another frame correctly after the error, the receiver station waits for T_{EIFS} (Extended Inter Frame Space) instead of T_{DIFS}. The analysis of packet error concerns unreliable packet transmission ability.

P_f is defined to be the transmission failure probability, P_c to be packet collision probability and P_e to be the packet error probability. Also $r_{transmit}$ is denoted as the rate of transmission (including failures) and $r_{success}$ as the rate of successful transmission respectively. Let $r_{collision}$ and r_{error} be the rate of collision and the rate of packet errors respectively.

Then the probability of successful transmission can be expressed as:

$$\frac{r_{success}}{r_{transmit}} = 1 - P_f \tag{3.8}$$

Multiple transmissions that collide can be counted as one collision. Approximately each collision is between just two transmissions, then $2r_{collision}$ contributes to the rate of transmission failure, therefore

$$r_{transmit} - r_{success} = 2r_{collision} + r_{error} \tag{3.9}$$

$$\frac{2r_{collision}}{r_{error}} = \frac{P_c}{P_e} \tag{3.10}$$

T_{cycle} is defined to be the average time between the starts of two payload transmissions. The transmissions colliding with each other occur at the same time, therefore $r_{collision}$ contributes to $1/T_{cycle}$. Then the following equation can be gotten

$$\frac{1}{T_{cycle}} = r_{success} + r_{collision} + r_{error} \tag{3.11}$$

From equations (3.8), (3.9), (3.10) and (3.11), $r_{success}$ can be expressed as

$$r_{success} = \frac{2(1 - P_f)}{2 - P_f + P_e} \frac{1}{T_{cycle}} \tag{3.12}$$

Finally, channel utilization is expressed by successful transmission of payload bits as:

$$S = r_{success} \times T_{payload} = \frac{2(1 - P_f)}{2 - P_f + P_e} \times \frac{T_{payload}}{T_{cycle}} \tag{3.13}$$

In the existing analytical model [68], it derived the expression of P_c as:

$$P_c = 1 - (1 - \frac{2(1 - 2P_f)}{1 - P_f - P_f(2P_f)^m} \frac{1}{W})^{n-1} \tag{3.14}$$

Where n is the number of stations in a wireless cell and W is the minimum window size.

Thus the transmission failure probability P_f is given by

$$P_f = P_c + P_e - P_c P_e \approx P_c + P_e = 1 - (1 - \frac{2(1-2P_f)}{1 - P_f - P_f(2P_f)^m}\frac{1}{W})^{n-1} + P_e \quad (3.15)$$

P_c increases as the number of stations in a wireless cell increases. As to the P_e, in [69], it is stated that if the channel condition varies over time, packet errors would have the corresponding variability and it is difficult to predict the variability of packet errors. If there are only two stations in a cell, which forms the peer to peer ad hoc network, P_c will not be there, and $P_f = P_e$. From equation (3.13), it can be seen that whether P_c or P_e increases, the channel utilization by successful transmission of payload bits S, also known as saturation throughput, will decrease. In order to improve S, both the probability of collisions and probability of packet errors need to be reduced.

Since packet collision and packet error are the intrinsic characteristics and P_c and P_e are determined by the lower MAC and PHY layers, such as CSMA protocol, non-ideal channel condition and multi-path fading and so on, in order to improve S, further researches are needed to be done in lower layers, which are not the main concern of this thesis. As to the upper storage layers such as block layer and file layer, because they are built on top of the lower layers and can only utilize the bandwidth the lower MAC and PHY layers provide for them, in order to sufficiently utilize the bandwidth, the only way is to reduce the waiting time of the transmission acknowledgement before sending the next I/O request in storage layer and send as many packets as possible to lower layer to prevent the occurrence of the time gap between 802.11 frames.

3.4 The Impact of Multi-hop Wireless Network on TCP Performance

Multi-hop network is not a new concept in wired environment, whereas relatively few researches focused on the multi-hop wireless networks. Multi-hop wireless networks have several characters different from wired networks. Firstly, in a typical wireless network that uses IEEE 802.11 MAC, packets may be dropped due to either packet collision caused by hidden terminals or packet error. Such losses directly affect TCP window adaptation. Secondly, wireless channel is a scarce, shared resource. Thus the performance of multi-hop wireless networks deserves some analysis. The IP packet forwarding and concurrent packet transmission are two key issues in multi-hop networks.

3.4.1 IP Packet Forwarding

The concept of a forwarding node, which receives packets from upstream nodes and then transmits these packets to downstream nodes, is a key element of any multi-hop network, wired or wireless, however the data forwarding operation in the wireless environment differs from the corresponding function in wired networks in a fundamental way.

In a wired network, a forwarding node typically has at least two physical network interfaces, with the forwarding functionality consisting of receiving a packet over one physical interface and subsequently sending it out over a second interface. In contrast, a node A, with a single wireless interface, may act as a forwarding node simply by retransmitting a packet that it received, over the same interface. In effect, A acts as an intermediary for two nodes that are each within the communication range of A but not directly within the range of each other.

Accordingly, packet forwarding in wireless environment does not typically imply the transfer of a packet between distinct interfaces on a single host. A

conventional implementation of packet forwarding thus involves the reception of a packet on the wireless interface, transfer of the packet up the host's protocol stack to the IP layer where a routing lookup is used to determine the IP and MAC address of the next hop, and subsequent transmission of the packet using the same wireless interface to the MAC address of the next hop. This form of forwarding suffers from two key deficiencies:

- The forwarding node is thus involved in two separate contention-based channel access attempts during the forwarding process: once to receive the packet (from the upstream node) and again to "forward" it (to the downstream node), and must thus suffer the contention resolution overhead twice.
- The same IP packet makes an unnecessary round-trip between the memory on the NIC and the hosts memory (accessed by the host software). This round-trip not only loads the processor of the forwarding node, but also suffers from additional delays in transfers between the NIC and the host operating system.

The current IEEE 802.11 DCF MAC algorithm has been designed implicitly for either receiving or transmitting a packet, but not for a forwarding operation (i.e., receiving a packet from an upstream node and then immediately transmitting the packet to a downstream node as an atomic channel access operation).

In order to improve the packet forwarding performance, a research group proposed a MPLS + DCMA solution to solve this problem [70]. But this is a wireless LAN 802.11 MAC inherent problem, the performance can not be improved by proposing the solutions in the upper file and block layer.

3.4.2 Concurrent Packet Transmission

The 802.11 MAC protocol and its variants, are primarily designed for a single-hop wireless environment, where nodes typically form a clique and communication

always takes place over a single wireless hop (often to a base station providing connectivity to the wired infrastructure). In such a "single-cell" environment, the 802.11 MAC contention resolution mechanism focuses primarily on ensuring that only a single sender - receiver node pair receives collision - free access to the channel at any single instant.

The fundamental MAC constraint in a shared wireless medium is that no receiving node can be within the transmission range of more than one simultaneously transmitting node, since such concurrent transmissions will lead to collision and incorrect reception at the receiver. While any MAC should prohibit only parallel transmissions, the 802.11 DCF MAC imposes a more rigorous constraint: any node that is a neighbor of either participant (sender or receiver) in an ongoing RTS/CTS/DATA/ACK exchange must remain quiet for the entire duration of the four-way exchange; it can neither initiate a parallel transmission (by sending an RTS) nor respond with a CTS during this period.

Solving the above mentioned problem by improving channel utilization through spatial channel reuse is highly desirable. Multiple nodes that do not interfere with each other should be encouraged to transmit concurrently.

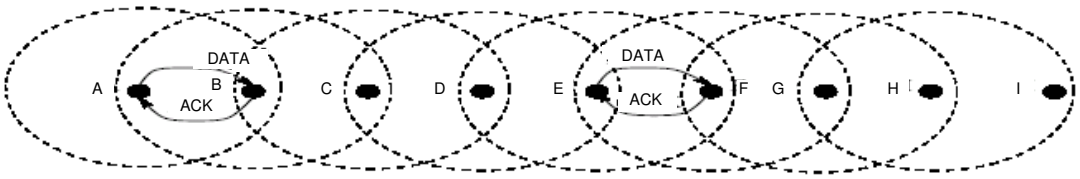

Figure 3.3: Spatial Reuse and Contention

Figure 3.3 illustrates an example of 8 hop chain. Optimal spatial reuse is achieved when nodes {A, D, G} and nodes {B, C, E} are scheduled for transmission alternatively. Node D is the hidden terminal for transmission A → B. Two adjacent nodes are about 200 m apart. The current hardware specifies that for each wireless node, its transmission range is about 250 m, its carrier sensing range is 550 m and its interference range is about 550 m. The potential sending node D is a hidden

terminal of the current transmission pair (A, B). When A and B are initiating RTS - CTS handshake, D cannot hear CTS since it is out of the 250 m transmission range of node B. Besides, D can not sense A's data transmission since A is out of D's 550 m carrier sensing range. Therefore, D may transmit to its intended receiver E at any time. When D is transmitting to E, it will cause collisions at B, since D is within the 550 m interference range for B. Therefore, hidden terminal D will cause contention loss at node B. Location dependent contention, together with multi-hop packet forwarding, also allows for spatial channel reuse. Specifically, any two transmissions that are not interfering with each other can potentially occur simultaneously; this improves aggregate channel utilization. Pairs of (A, B) and (E, F) may transmit simultaneously, but simultaneous transmissions from pairs of (A, B) and (C, D) will collide. Improving spatial reuse will result in increased TCP throughput.

From what has been discussed above, it can seen that multi-hop wireless network may make great impacts on the performance of TCP layer, which are due to the IP packet forwarding and concurrent packet transmission mechanism of wireless LAN 802.11. Some researches have proposed some solutions to reduce the impacts of multi-hop wireless networks in the lower layers [70, 71, 72]. However, for the upper layers such as file layer and block layer, since they are above TCP/IP layer, they can not control the TCP performance, which is determined by the mechanism of lower MAC layer. What storage layers should do is to send as many downstream packets as possible to reduce the waiting time for I/O request in lower layers. To do so, whenever the IP packet forwarding or spatial reuse method is further improved and the MAC layer can address the multi-hop wireless network problem more efficiently, the continuous data packets from the upper storage layer will allow the storage system to sufficiently utilize the wireless bandwidth and significantly improve the storage performance.

3.5 Block Level Storage

One approach for accessing remote data is to use an IP based storage area networking (SAN) protocol such as iSCSI. In block level storage, a remote disk exports a portion of its storage space to a client. The client handles the remote disk the same as its local disk. It runs a local file system that reads and writes data blocks to the remote disk. Rather than accessing blocks from a local disk, the I/O operations are carried out over a network using a block access protocol which is shown in Figure 3.4. Block level storage access remote data at the granularity of disk blocks. The network I/O consists of block operations (block reads and writes). In case of iSCSI, remote blocks are accessed by encapsulating SCSI commands into TCP/IP packets. The iSCSI design is discussed in detail in Chapter 4.

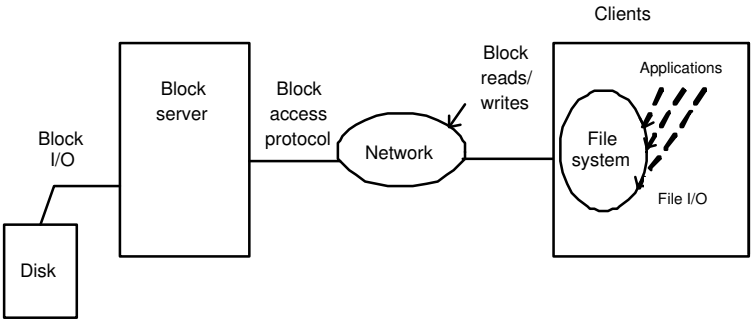

Figure 3.4: Block Level Access Storage

Figure 3.5 shows components of TCP/IP packet when it carries iSCSI protocol. The data are encapsulated into SCSI commands, to which iSCSI header is added, and iSCSI message is the part of TCP/IP message. Since iSCSI only use TCP as transmission protocol, for large data transfers in wireless environment, the iSCSI transfer is efficient.

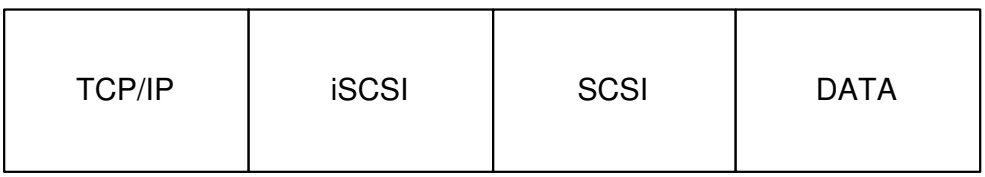

Figure 3.5: TCP/IP Packet Components for iSCSI Protocol

3.5.1 iSCSI Protocol Analysis

The basic iSCSI protocol of read command under the condition of no queue is analyzed. In the initiator, the application, which needs to read data from the storage device, issues file requests. The file system converts file requests to block requests from application to the SCSI layer. The initiator iSCSI driver encapsulates SCSI commands in iSCSI Protocol Data Units (PDUs) and sends them to the network via the TCP/IP layer. After receiving iSCSI PDUs from TCP/IP layer, the target iSCSI driver de-capsulates them. The target driver then sends response data and status back via TCP/IP.

The read command used to analyze the iSCSI protocol is shown in Figure 3.6. The average command time is

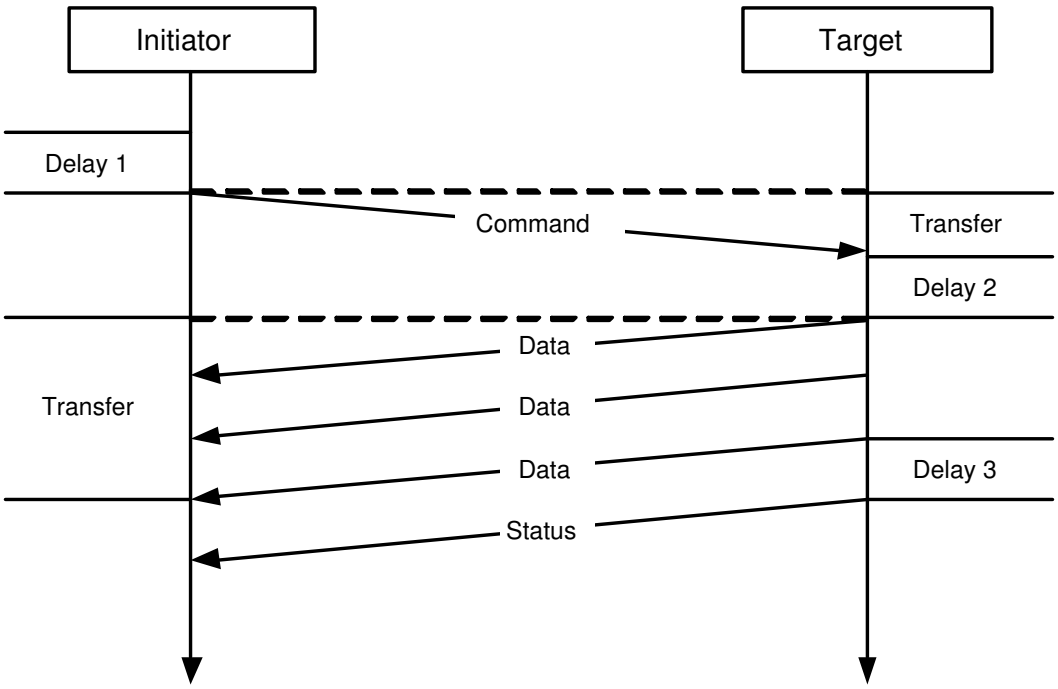

Figure 3.6: Average Read Command Time Analysis

$$T_{average\_command\_time} = T_{delay} + T_{transfer} \tag{3.16}$$

Where T_{delay} is

$$T_{delay} = T_{delay1} + T_{delay2} + T_{delay3} \tag{3.17}$$

T_{delay1} is the time gap between the time one command has finished and the next command is issued. T_{delay2} is the time gap between the time target receives one command and issues the data to be transferred. T_{delay3} is pure drive delay. Target should send all the data and then prepare the status data. $T_{transfer}$ is determined by the speed of the wireless network, thus it can not be decreased.

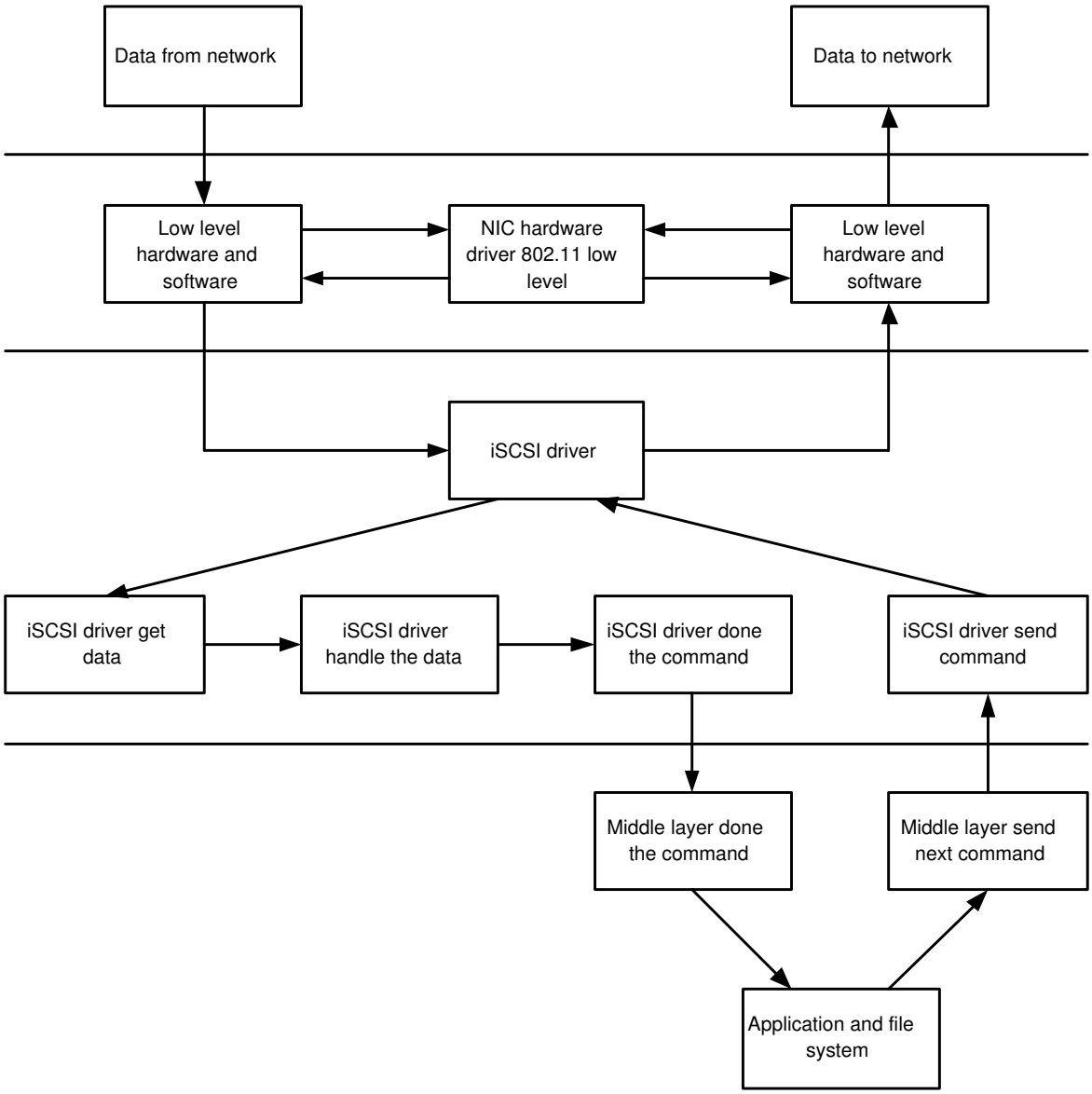

Figure 3.7: Delay 1 for iSCSI Protocol Analysis

Figure 3.7 shows every part of delay 1. From Figure 3.7, it can be seen that delay 1 should include wireless LAN low level hardware and software, iSCSI driver

and middle layer delays.

$T_{delay1} = T_{NIC\_driver}$ (receive status)

$+ T_{iSCSI\_driver}$ (receive and analyze status and send done)

$+ T_{middle\_layer}$

$+ T_{NIC\_driver}$ (send command)

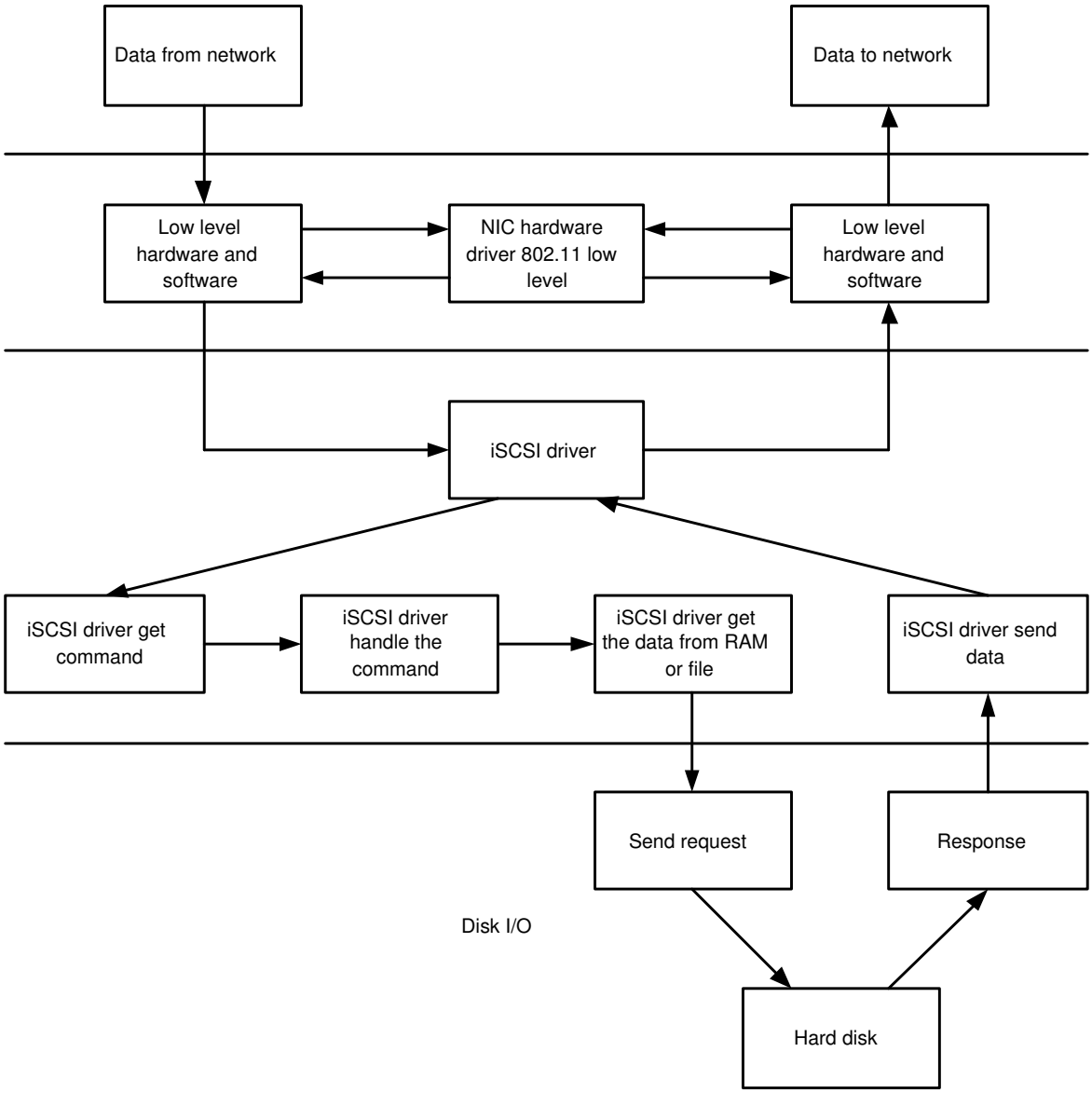

Figure 3.8: Delay 2 for iSCSI Protocol Analysis

The delay 2 is shown in Figure 3.8. Delay 2 is almost the same as delay 1. The delay in the low level network is the same except that now sending more data is needed. It will take more time, the same as iSCSI driver. If RAM I/O is used,

there is no delay of the hard disk. If the file I/O or disk I/O is used, reading data from hard disk is needed and it will take more time.

$T_{delay2} = T_{NIC\_driver}$ (receive status)

$+ T_{iSCSI\_driver}$ (receive and analyze command and send data)

$+ T_{low\_level}$ (read data from hard disk if disk I/O or file I/O)

$+ T_{NIC\_driver}$ (send data)

Although as less time as possible in RAM I/O is used, delay 2 still can not be decreased. No memory copy, just use a point to the right memory site. But delay 3 can be decreased. The last data packet with the status can be combined to let delay 3 equal to 0. The maximum delay which cause the performance worse can be analyzed. During delay 1, TCP also reply the ACK to the sender, which means the receiver also need to free the buffer to prepare more data during delay 1. Thus one big buffer to receive all the data of one command and then done the command can be defined. It should be the fastest way to the sender and receiver to transmit data in wireless environment.

3.6 File Level Storage

Traditionally, the file level storage is to simply setup a NAS [74] by using file level access protocol such as NFS (Network File System) [73].

Figure 3.9 illustrates the architecture of NAS. NAS is just another name for file serving, which was introduced to enable data sharing across platforms. With NAS, the meta-data describing how files are stored on devices is managed completely on the file server. NAS enables cross-platform data sharing but comes at the cost of directing all I/O through the single file server. The server makes a subset of its local namespace available to clients. Clients access meta-data and files on the server using a RPC (Remote Procedure Call) based protocol which is

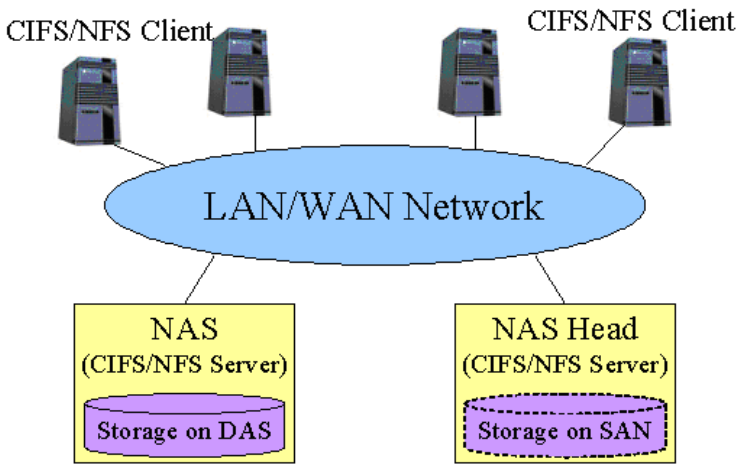

Figure 3.9: Network Attached Storage Architecture

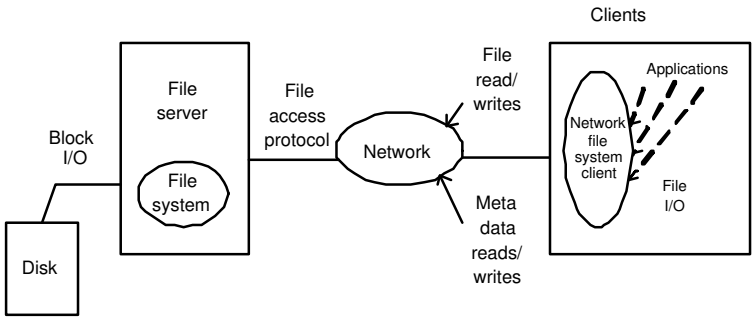

Figure 3.10: File Level Access Storage

shown in Figure 3.10.

Figure 3.11 illustrates the communication process of NFS. NFS gives all network users access to files that may be stored on different types of computers. In a client - server environment, NFS enables computers connected to the network to operate as clients to access remote files. The same computers also can act as servers by allowing remote users to access their files. In other words, NFS makes files stored on a file server accessible to any computer on a network and eliminates the need to transfer files between users.

3.6.1 Analysis of NFS

In NFS, the file system is located at the server and so is the file system cache (hits in this cache incur a network hop). NFS clients also employ a cache that can hold

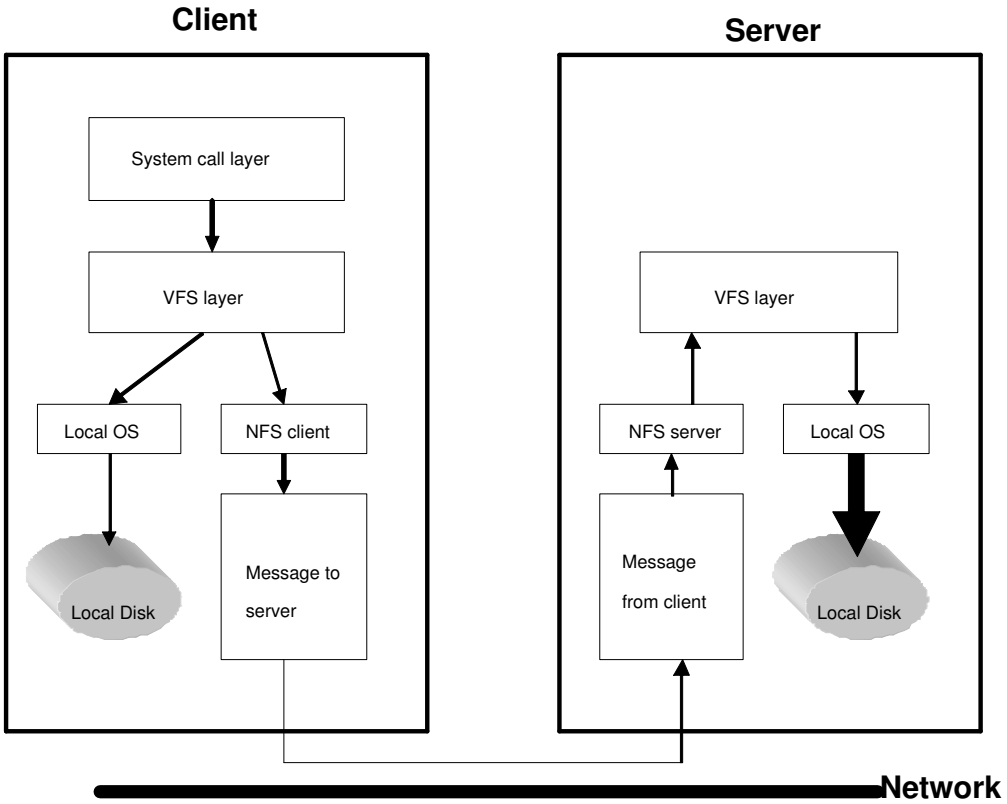

Figure 3.11: Communication Process of NFS

both data and meta-data. To ensure consistency across clients, NFS version 2 and version 3 (Although NFS version 4 can avoid this message exchange for data reads if the server supports file delegation, the most common version of NFS is 3) require that clients perform consistency checks with the server on cached data and meta-data. The validity of cached data at the client is implementation-dependent, in Linux, cached meta-data is treated as potentially stale after 3 seconds and cached data after 30 seconds. Thus, meta-data and data reads may trigger a message exchange (i.e., a consistency check) with the server even in the event of a cache hit, which will consume more times for data transferring and consume more wireless network bandwidth.

From the perspective of writes, both data and meta-data writes in NFS version 2 are synchronous. NFS version 3 and version 4 supports asynchronous data writes, but meta-data updates continue to be synchronous. Thus, depending on the version, NFS has different degrees of write-through caching. The write-through

caching may slows down performance as the system waits for the data to be written to hard disk and then send the ACK status back to client.

NFS is a stateless protocol. In other words, when a client sends a request to a server, the server carries out the request, sends the reply and then removes from its internal tables all information about the request. Between requests, no client specific information is kept on the NAS server. The advantages of stateless scheme are shown in Table 3.3.

Table 3.3: Advantages of Stateless Scheme

Advantages of stateless scheme
Fault tolerance
No open or close call needed
No server space wasted on tables
No limits on number of open files
No problems if a client crashes

For each file access, since NFS runs on the file level, it requires the exchange of several commands before data communication can occur. For example, to read a file, the client side needs to send an ACCESS command several times to check the access permission for the directories of the file. It then issues a command to look up the specified file and checks its access permission by sending an ACCESS command. These waste several round trip times for the exchange of command, consume more wireless bandwidth and downgrade the storage performance.

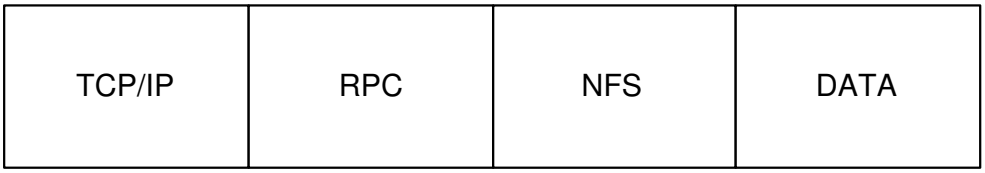

Figure 3.12: TCP/IP Packet Components for NFS Protocol

Figure 3.12 shows components of TCP/IP packet carrying NFS packets. The data are enclosed into NFS messages, which are further encapsulated into RPC messages. The latter are the part of TCP/IP packets. NFS uses RPC over TCP, due to these two transmission protocols, for large data transfers in wireless environment, the NFS transfer is not very efficient compared to iSCSI.

Advantages of file level data access provide high level abstraction that enables cross-platform data sharing as well as policy-based security, which come at the cost of more time for information exchange and consuming more limited wireless bandwidth.

Chapter 4

Multiple Virtual TCP Connection iSCSI Design

In this chapter, the new iSCSI design is discussed in detail. The explanations of the basic concepts and iSCSI storage model are given. The login phase, data transfer in full feature phase are discussed. The idea of multiple virtual TCP connection is discussed and the iSCSI design and working principle are presented in detail. A typical queuing model of the multiple connection design is established and used to illustrate the implication of the multiple connection design.

4.1 iSCSI Storage General Model

4.1.1 iSCSI Session and Connection

Figure 4.1 reflects the currently approved SCSI family. iSCSI is one of members of SCSI family. iSCSI is a storage protocol on top of the TCP/IP network similar to FCP (Fiber Channel Protocol) on top of the FC (Fiber Channel) and SBP (Serial Bus Protocol) on top of the IEEE 1394 to transfer SCSI command. SCSI commands are typically issued by a storage initiator (client) to a storage target (server). The relationship between SCSI entities is referred to as a nexus. The

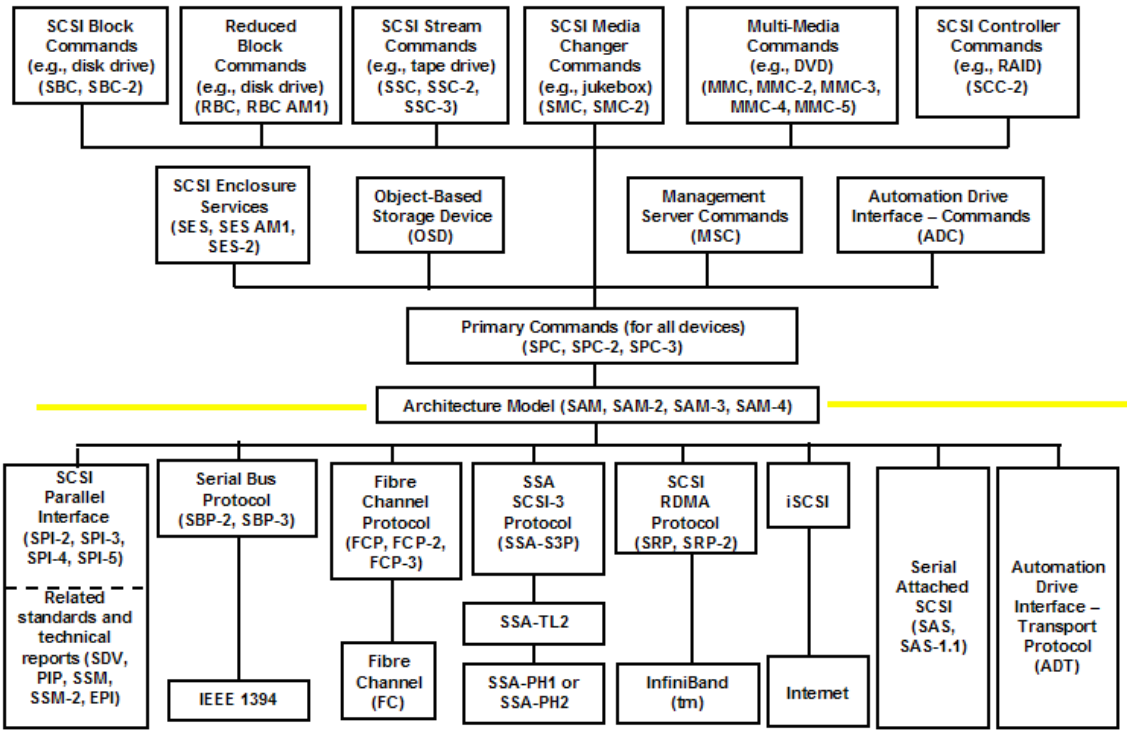

Figure 4.1: SCSI Standard Architecture

iSCSI entity corresponding to an initiator-target nexus is an iSCSI session. An iSCSI session is a collection of TCP connections between an iSCSI initiator and an iSCSI target used to pass SCSI commands and data between the initiator and the target.

Connections exist within a session, TCP connections carry control messages, SCSI commands, parameters and data. A session can encompass one or multiple TCP connections over one or more physical links connecting an initiator and a target.

There are four phases in a session, which are:

- **Phase 1 - Initial login phase:** In the initial login phase, an initiator sends the name of the initiator and target. It also specifies the authentication options. The target responds to the authentication options the target selects.
- **Phase 2 - Security authentication phase:** To ensure that each party is actually talking to its intended party, this phase is used to exchange authen-

tication information.

- **Phase 3 - Operational negotiating phase:** This phase is used to exchange certain operational parameters such as PDU length, buffer size and burst length.

- **Phase 4 - Full featured phase:** In this phase, SCSI commands and data are transferred between an initiator and a target.

4.1.2 iSCSI Storage Architecture

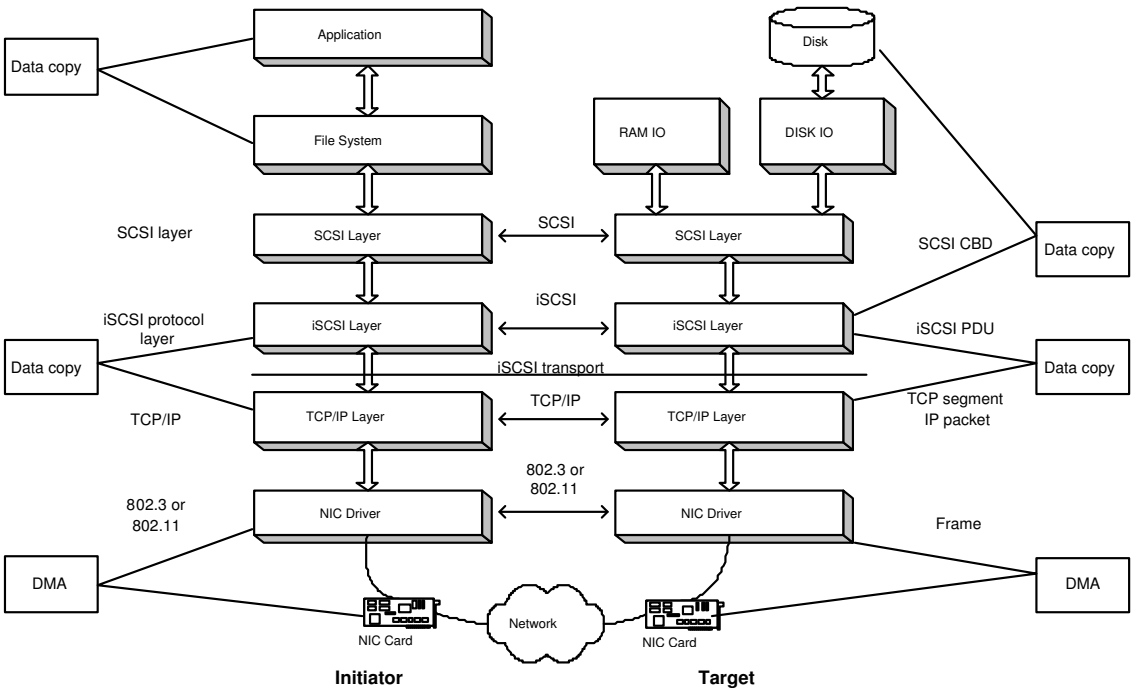

Figure 4.2: iSCSI Storage Model

Figure 4.2 shows the iSCSI storage model including an initiator and a target. iSCSI builds on top of TCP layer. For the communication between initiator and target, a session must be established. The data and command exchange occurs within the context of the session. In the initiator, the application, which needs to store and access data to or from the storage device, issues file requests. The file system converts file requests to block requests from application to the SCSI layer. A SCSI command execution consists of three phases: Command, Data and Status

Response which are shown in Figure 4.3. The initiator iSCSI driver encapsulates SCSI commands in iSCSI Protocol Data Units (PDUs) and sends them to the network via the TCP/IP layer. After receiving iSCSI PDUs from TCP/IP layer, the target iSCSI driver de-capsulates them. Then SCSI commands are mapped to the storage device. If the SCSI commands are mapped to RAM, it is called RAM I/O. If the SCSI commands are mapped to an actual magnetic storage disk, it is called disk I/O. The target driver then sends response data and status back via TCP/IP. There are two data copies and one DMA in the initiator during one I/O access and there are one data copy and one DMA for RAM I/O and two data copies and one DMA for disk I/O in the target side.

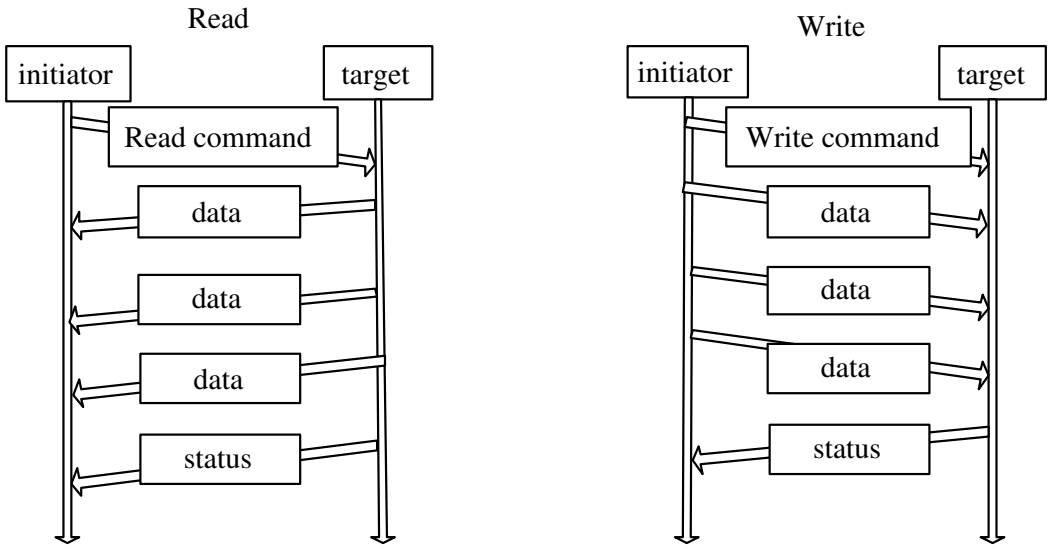

Figure 4.3: iSCSI Command Sequence

Besides iSCSI parameter such as PDU size, MaxBurstLength and First-BurstLength, factors in the lower layer such as TCP flow control algorithm, maximum frame size and MAC mechanism all significantly affect iSCSI performance.

4.1.3 iSCSI Protocol Data Units

iSCSI PDUs are defined as the communication messages between iSCSI initiator and target. Table 4.1 describes some of the most commonly used PDU types.

Table 4.1: iSCSI PDU Types

iSCSI PDU	Description
Command	Informs target of read or write request
Response	Report status of command
Data-out	Data from initiator to target
Data-in	Data from target to initiator
Ready To Transfer (R2T)	Informs initiator to send next set of data
Reject	Indicate iSCSI error condition

All iSCSI PDUs have a BHS (Basic Header Segment), which is a fixed-length 48 byte header segment. The BHS contains various parameters, including an op-code to indicate the type of iSCSI PDUs the headers encapsulate. The PDUs may also contain AHS (Additional Header Segments), which helps to transfer additional parameters, header-digest, data-digest and data segments. The optional data-digest and header-digest help to protect the integrity of the data segment and the header respectively. The data segment contains PDUs associated data. It is optional and is included usually in read and write commands to transfer data. Figure 4.4 shows the format and content of an iSCSI PDU.

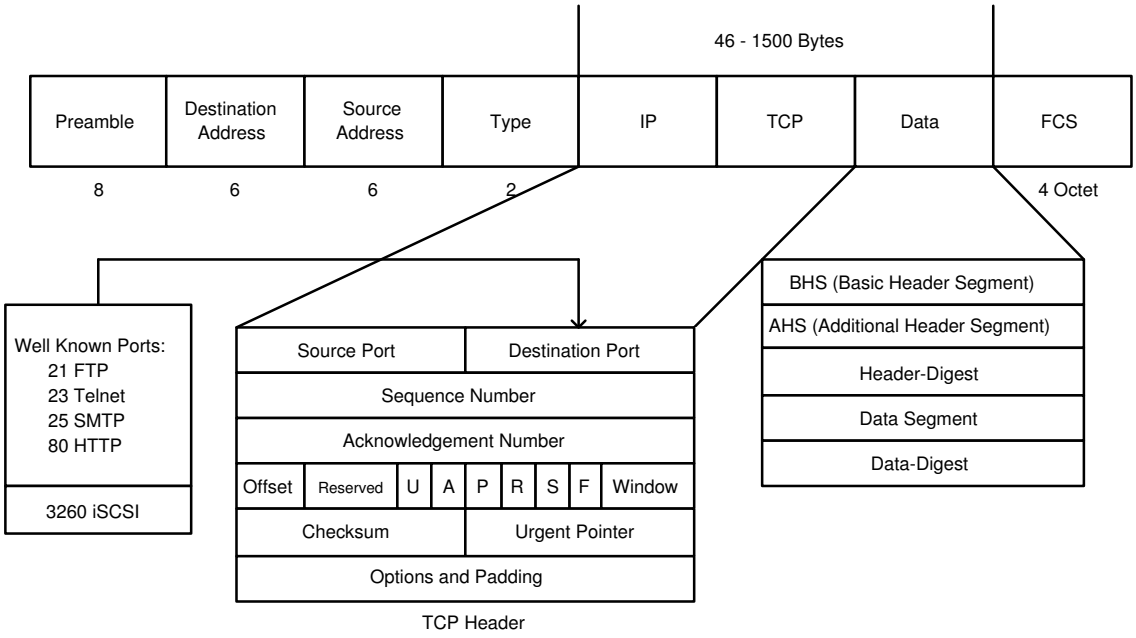

Figure 4.4: iSCSI PDU Format

The data segmentation and encapsulation is shown in Figure 4.5. The iSCSI PDUs after encapsulation are transferred to the TCP/IP layer, MAC layer and

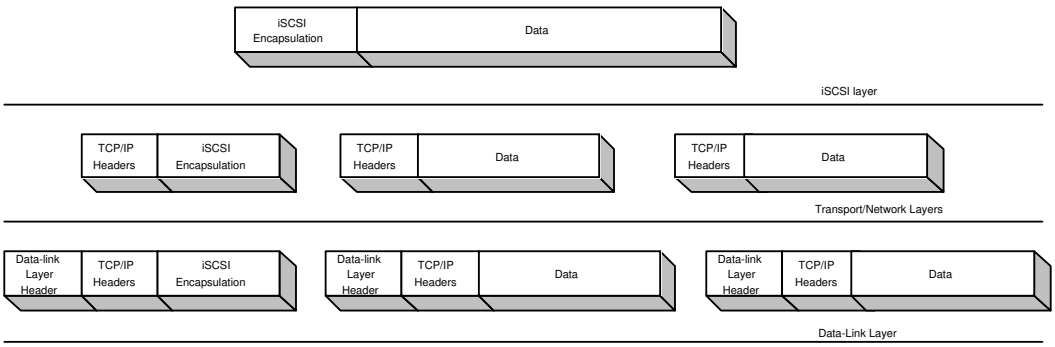

Figure 4.5: Data Segmentation and Encapsulation

PHY layer sequentially. In such process, the data requires segmentation of the initial data and the encapsulation of the data using headers such as TCP/IP header and MAC header.

4.2 iSCSI Phase Design

4.2.1 Thread Design

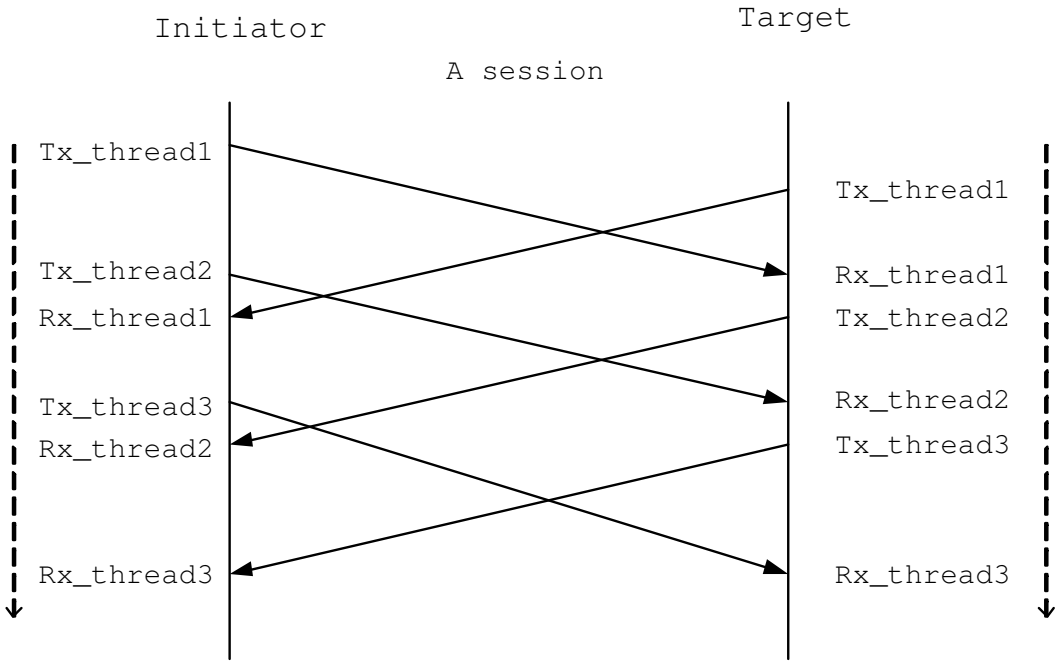

Figure 4.6: Tx_thread and Rx_thread

In programming, thread is defined as a part of a program that can execute

independently of other parts. Operating systems that support multi-threading enable programmers to design programs to execute multiple thread concurrently. In the iSCSI design, after a session has been built up, the initiator and the target can set several TCP connections whether they are across one or several physical connections. Threads run in the programs of the initiator and target. In both initiator and target, each connection spawns to two threads: tx_thread and rx_thread. They are responsible for transferring and receiving command, data and response respectively. Thus the command, data and response can be transferred by multiple threads in different connections. Figure 4.6 shows the tx_thread and rx_thread. In the multiple virtual connection design which will be discussed in Section 4.3 in detail, multiple sockets and multiple threads are used. Every tx_thread handles a sending sockets, rx_thread handle a receiving sockets.

4.2.2 Login Phase Design

There are four phases in an iSCSI session as what is discussed in Section 4.1.1. It is noted that phases 2 and 3 are optional. In the iSCSI design, phase 1 and 4 are focused, as to the necessary information and parameter exchanges, they are merged into phase 1 because the main objective to achieve high data storage performance and it is not to realize every function of iSCSI. The works in this thesis focuses on the basic functions of iSCSI.

4.2.2.1 Login Phase

Before iSCSI initiator can send SCSI commands to target, it must first establish an iSCSI session. login command is the first commands used to build session between initiator and target. After a session is setup, several connections can be built between initiator and target within a session.

During login phase, the initiator initiates the session, gets all target IP addresses from a pre-defined global session data structure. After that, initiator

builds connection to each target in each session. It sends login command PDU, as shown in Figure 4.7 through the connection and then waits for login response from the target. After login is successful, initiator creates several connections to the target (the number of connections is defined by the session parameter: MAX_CONNECTION_PER_SESSION), each connection will spawn to a tx_thread and a rx_thread, which is discussed in Section 4.2.1.

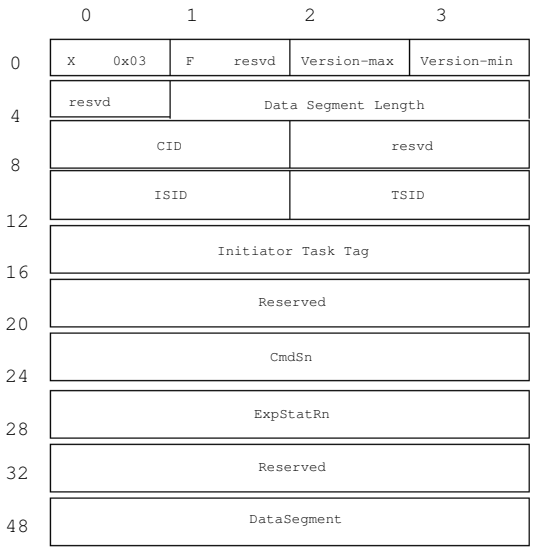

Figure 4.7: Login PDU Format

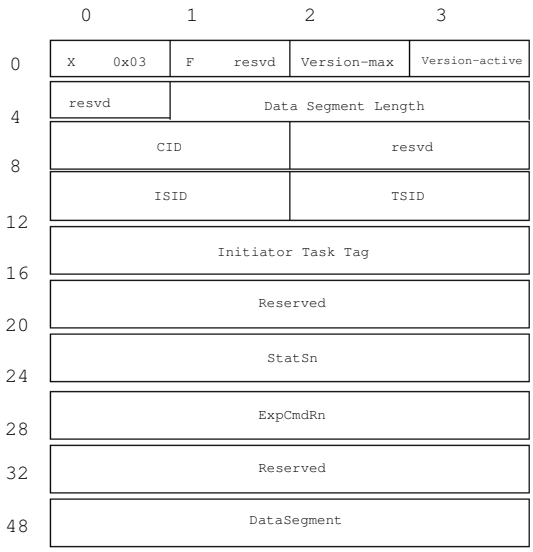

Figure 4.8: Login Response PDU Format

Target can only receive connection passively at the beginning. It creates a

socket, binds listen and sends response PDU, which is shown in Figure 4.8, with the authentication options the initiator selects. Then it creates a server thread in the target part. It can receive new connection from initiator in one session. Every connection in the target will also spawn to a tx_thread and a rx_thread. The overall process of iSCSI login command can be illustrated in Figure 4.9.

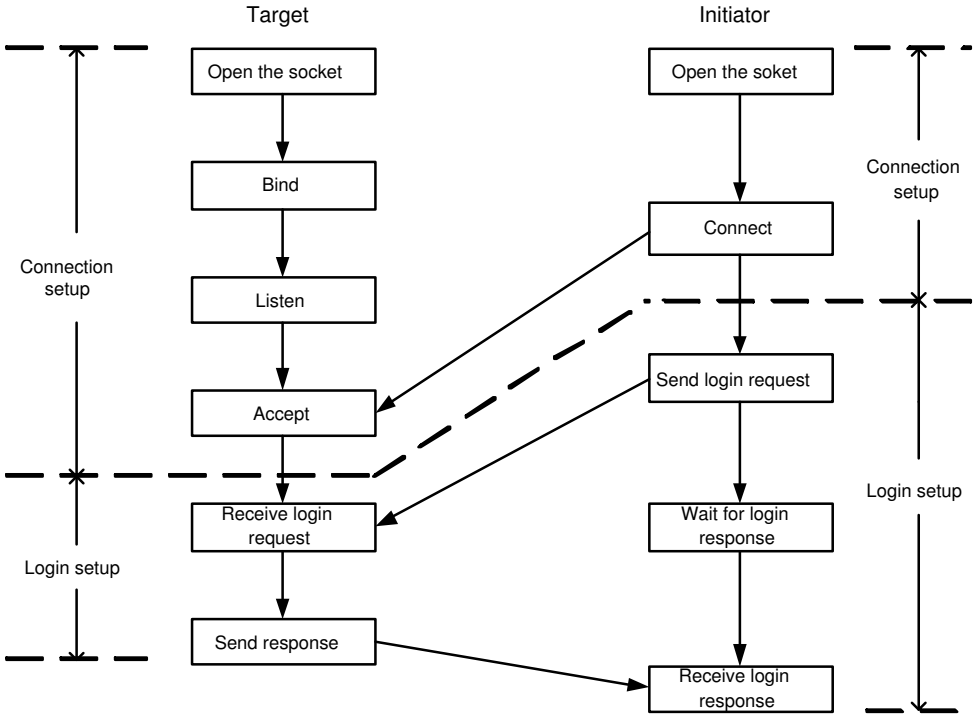

Figure 4.9: iSCSI Login Process

As what is mentioned in the beginning of this section, the necessary information exchange between initiator and target is merged into login phase. From now on, the information exchange is discussed.

4.2.2.2 Information Exchange

Information exchange is also called text command, which is to allow the exchange of information such as parameters and for future extensions according to the iSCSI draft [36]. Text command should be the second command next to login command. The status of a iSCSI session is set to several stages which are shown in Table 4.2. Only after the necessary information exchange, iSCSI can enter full feature phase.

Table 4.2: iSCSI Session Stages

iSCSI activities	iSCSI stages
Before build a sock between initiator and target	SESSION_STATUS_NOT_CONNECT
After build a sock	SESSION_STATUS_CONNECT
After initiator successfully login	SESSION_STATUS_LOGIN
After information exchange	SESSION_STATUS_FULLFEATURE
After initiator logout	SESSION_STATUS_LOGOUT

Text command involves many rounds of information exchange. Every round the initiator or target tries to do some parsing of text contents which come from other part. When initiator wants to end text exchange, it sent a text command with F bit set and target reply a text response with F bit set.

iSCSI draft [36] lists 36 kinds of parameters, because the main objective of this thesis is to achieve high storage performance, the information exchange is not the main concern, only 6 kinds of them are used in the iSCSI design, which are TEXT_MAXCONNECTIONS (maximum Connections), TEXT_TARGETNAME (target name), TEXT_INITIATORNAME (initiator name), TEXT_TARGETADD RESS (target address), TEXT_IMMEDIATEDATA (immediate data) and TEXT_D ATAPDULENGTH (data PDU length). In the model, text command is finished just in one time of text request and response. So in the first text command, F bit is set and so does the text response.

After both sides satisfy with authentication and the information exchange, the login process in the design is completed and the connection can used to pass SCSI command and data.

4.2.3 Data Transfer in Full Feature Phase Design

Once the initiator and the target authenticate each other and exchange necessary information and parameters, the initiator registers to SCSI middle layer to emulate a SCSI device. The iSCSI session is in data transfer phase, also called full feature phase.

4.2.3.1 Data Transfer

Data transfer in iSCSI can also refer to read and write operation. How we design the read and write command is discussed.

Initiator receives SCSI command from SCSI middle layer by scsi_queuecommand. Then it searches for a session according to current_session scsi_target_id. If the command is a write command for unsolicited data (immediate data), the offset is set to 0 in the initiator SCSI write buffer and the initiator prepares for data transfer. The command is queued in the current_session's pending_commands and the initiator waits for its tx_thread to send the command and data. Initiator's rx_thread receives any response including read data, R2T (Read to Transfer) and status from the target.

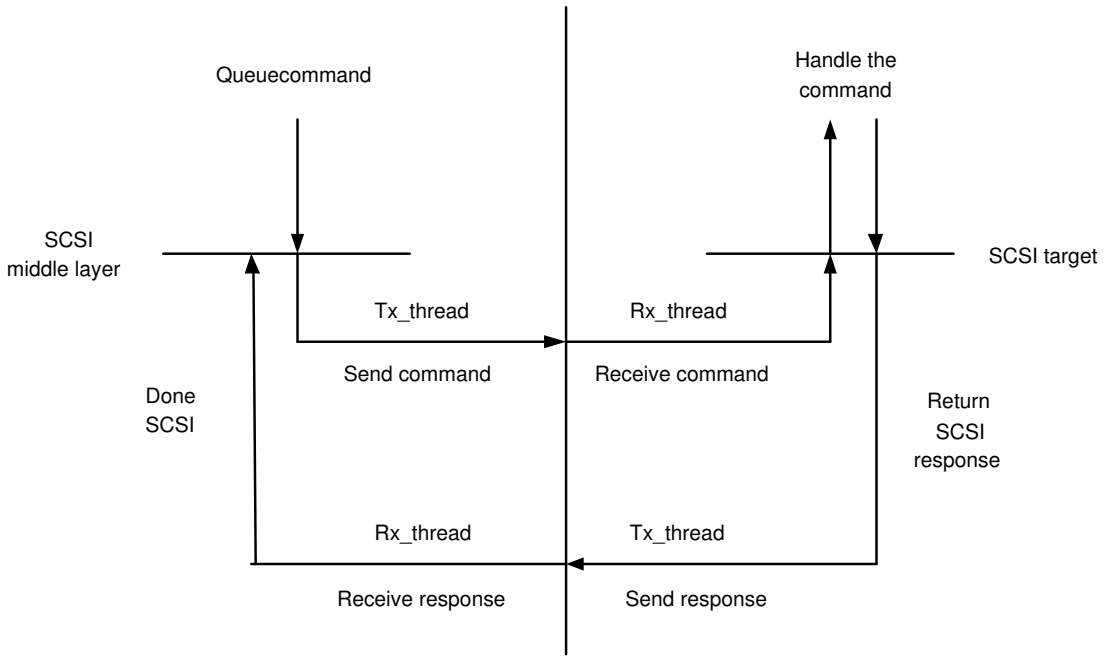

Figure 4.10: Model of Data Transfer

In the iSCSI design, the model of data transfer can be illustrated in Figure 4.10. The target receives command by rx_thread, it examines the header of iSCSI command and handles the command. Target should handle command with iSCSI target disk by using one of memory I/O, file I/O or disk I/O. When the target finishes the reading or writing process, it sets the status of command to scsi_done

and waits for the target tx_thread sending the response and status of the command. Tx_thread of target search for the command list of current_connection and current_session, check the status of the command, if the status is scsi_done, it sends the response data and status.

4.2.3.2 iSCSI Target Design

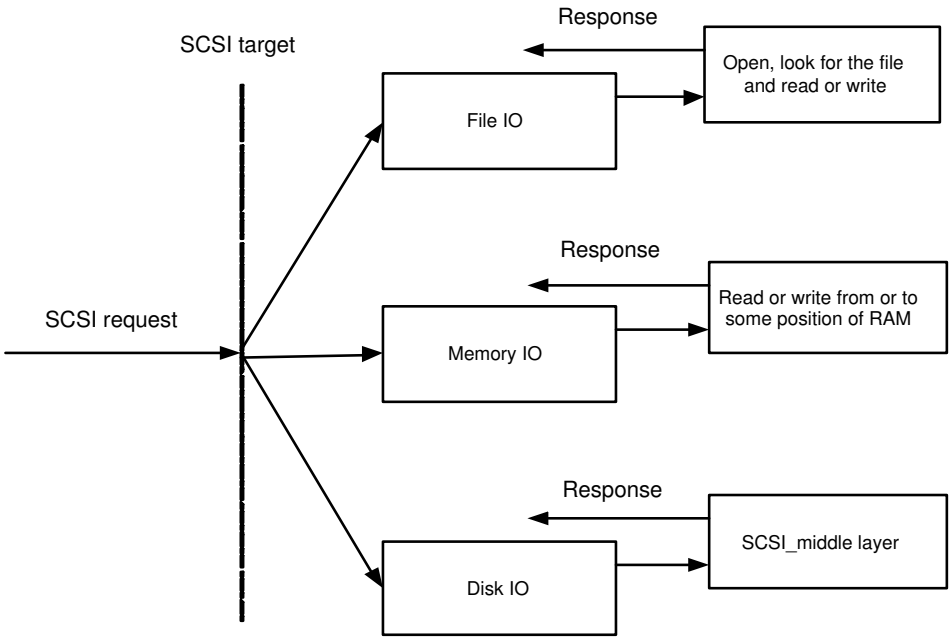

Figure 4.11: SCSI Request Types

SCSI target is in charge of handling data between buffer and disk (file I/O, memory I/O or disk I/O). The parameter of the disk is set in the login phase of the initialization process of target. These parameters include LUNs (Logical Unit Numbers), block_len and num_blocks. When initiator issue read capacity command, the target will reply the result of LUN capacity.

The SCSI request types are shown in Figure 4.11. For the file I/O and memory I/O, the target translates the read or write LBA (Logical Block Addressing) to the position in the file or memory and read or write data. For the disk I/O, SCSI requests are sent to the scsi_middle layer and replay is gotten from the middle layer.

4.3 Multiple Virtual TCP Connection Design

4.3.1 Multiple Virtual Connection Solution

Since TCP connection is used to transport iSCSI command, iSCSI initiator is connected to iSCSI target by TCP connection. It might be impossible to achieve the full bandwidth capability of the underlying physical transport by a single TCP connection. However normal iSCSI design and implementation is based on the single TCP connection, which may not sufficiently utilize the wireless network bandwidth due to the TCP window size and the round trip time of TCP ACK status. Compared to wired environment, the wireless network is low speed and unreliable, in normal iSCSI design, the initiator needs to wait for a long time before sending the next I/O request because of the packet retransmission over low speed wireless network due to the packet failure and the long latency as what is discussed in Chapter 3. Especially for small I/O, the single connection iSCSI may face more serious performance problem because the requests are sent more frequently than big I/O, more time is wasted when waiting for the ACK status. And for small I/O request, if the request size can not be divided exactly by the maximum frame size in the low layer, there will be a lot of 802.11 frames, which do not sufficiently utilize each frame size.

Due to the limited bandwidth of wireless network, it is supposed that all other system resources such as CPU, memory and the speed of hard disk are powerful enough compared to the limited bandwidth to make it possible to achieve good performance and high network utilization in wireless environment for iSCSI software solutions.

Since TCP is responsible for end to end data integrity and precedes reliable, full duplex connections and reliable service through positive acknowledgment with retransmission strategy, if the network is unreliable, the system throughput is low due to the time to wait for the acknowledgment and retransmission. Furthermore,

TCP does not have the mechanism to distribute traffic across multiple interfaces when that traffic is part of a single TCP connection. Fortunately, TCP enables hosts to maintain multiple, simultaneous connections, thus distribute traffic across multiple TCP connections. Multiple connections will provide better performance and availability. Multiple connections are more likely invisible to the upper layer since there is still single session for the application. For wireless network, the physical connection is weak and unreliable, however having multiple connections could help to reduce the risk of single TCP transmission failure. Also, the parallel transmission, resulted from the multiple connections, is obviously faster than serial, because more than one packet is sent at a time.

From what has been discussed above and in previous chapters, a new iSCSI design with the multiple virtual TCP connections in an iSCSI session and parallel working mechanism in iSCSI layer is proposed. The key idea of this new architecture actually employs multiple virtual TCP connections over one physical wireless connection and efficiently spread its traffic over multiple virtual TCP connections. The working principle will be discussed in detail in Section 4.3.3. It is supposed that the iSCSI driver can utilize the wireless channel more efficiently and the iSCSI performance would achieve significant improvement especially for small I/O with multiple virtual TCP connections.

With multiple connections, the iSCSI driver can continuously send I/O requests to the lower MAC layer via different virtual connection and do not need to wait for the ACK status before issuing the next I/O request. Also due to the continuous requests from the iSCSI layer, each 802.11 frame can be sufficiently used to carry the data. The multiple virtual connection design supposes not only to improve the iSCSI performance by increasing the utilization of limited wireless network bandwidth, but also to provide a better mechanism to handle the packet failure in wireless channel and the long latency issues in multi-hop network environment. In wireless storage, since the demand for data is not as large as in wired network, multiple connection iSCSI design that significantly improve throughput

for small I/O is very valuable for the application in wireless storage.

The multiple virtual TCP connection iSCSI design is compatible with iSCSI draft [36], because the draft states that iSCSI protocol allows multiple connections, not all of which need to go over the same network adaptor within an iSCSI session.

Although it is possible to design multiple TCP connections in one iSCSI session over several physical connections [36], it costs more with the new hardware. In addition, even though multiple connection over several physical network adapters may achieve higher storage performance, each physical connection still can not be sufficiently utilized as what is discussed above of the single connection iSCSI.

The multiple virtual connection iSCSI, which is different from general idea of single connection iSCSI or multiple physical connections, can achieve higher utilization of available bandwidth and better storage performance with minimum hardware costs over wireless network.

4.3.2 Symmetric and Asymmetric Approach

With the multiple connections, one of the issues that is concerned with is whether data should travel over the same connections as commands and control information. There are two approaches: symmetric approach and asymmetric approach. Symmetric approach is to transfer data over the same connection on which the corresponding command is sent. In this approach, all connections are treated equally. The other approach is called asymmetric approach because there would be different types of connections: control and data. The asymmetric approach is to send all SCSI commands, SCSI responses and task management information over a control channel, while all data transfers go over separate data channels. The symmetric and asymmetric models are shown in Figure 4.12.

A major concern with the symmetric model is the possibility of filling up a connection with data of some commands and then being unable to deliver a high

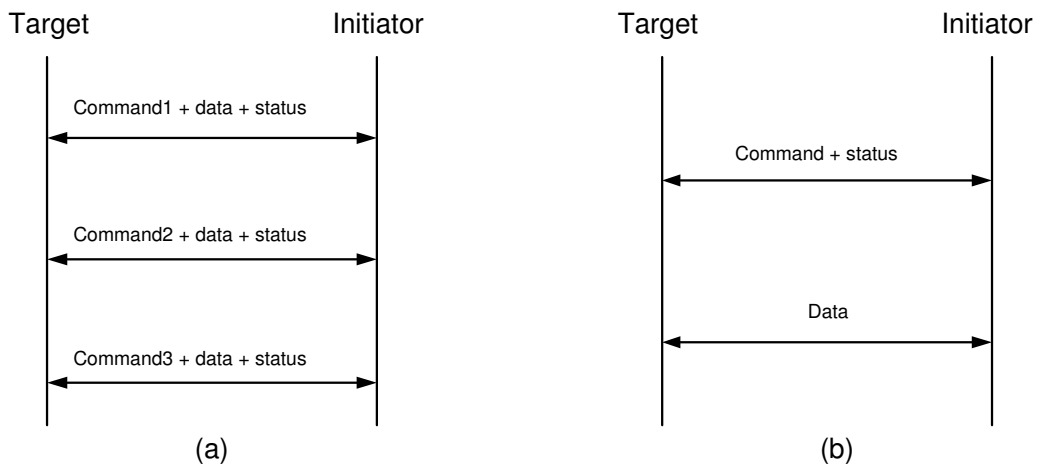

Figure 4.12: (a) Symmetric Model; (b) Asymmetric Model

priority task management request over the same channel. In the asymmetric model, only commands and task management requests are sent over the control channel. These are all relatively short messages, so connection will almost never be filled up and the control channel would not block. The major drawback of asymmetric model is that when using multiple processors and iSCSI adapters, if control channel and its data channels are handled by different processors or adapters, the handling of a command would require the interaction between different processors and iSCSI adapters. This is deemed most undesirable. Fortunately multiple virtual connection design is chosen with single processors and iSCSI adapters, thus the overhead imposed by the inter-processor or inter-adapter communication do not need to be concerned. The asymmetric model is used in this iSCSI design.

4.3.3 Working Principle

The detailed working principle is shown in Figure 4.13 . Multiple virtual TCP connections are built on one physical wireless connection (one wireless LAN adaptor in the initiator and one in the target). Half of the connections are used for sending SCSI requests from initiator to target, the other half of the connections are used for sending responses including data and status from target to initiator. One pair of transmitting thread (tx_thread) and receiving thread (rx_thread), which locates

in initiator and target respectively, are responsible for data communication within one connection.

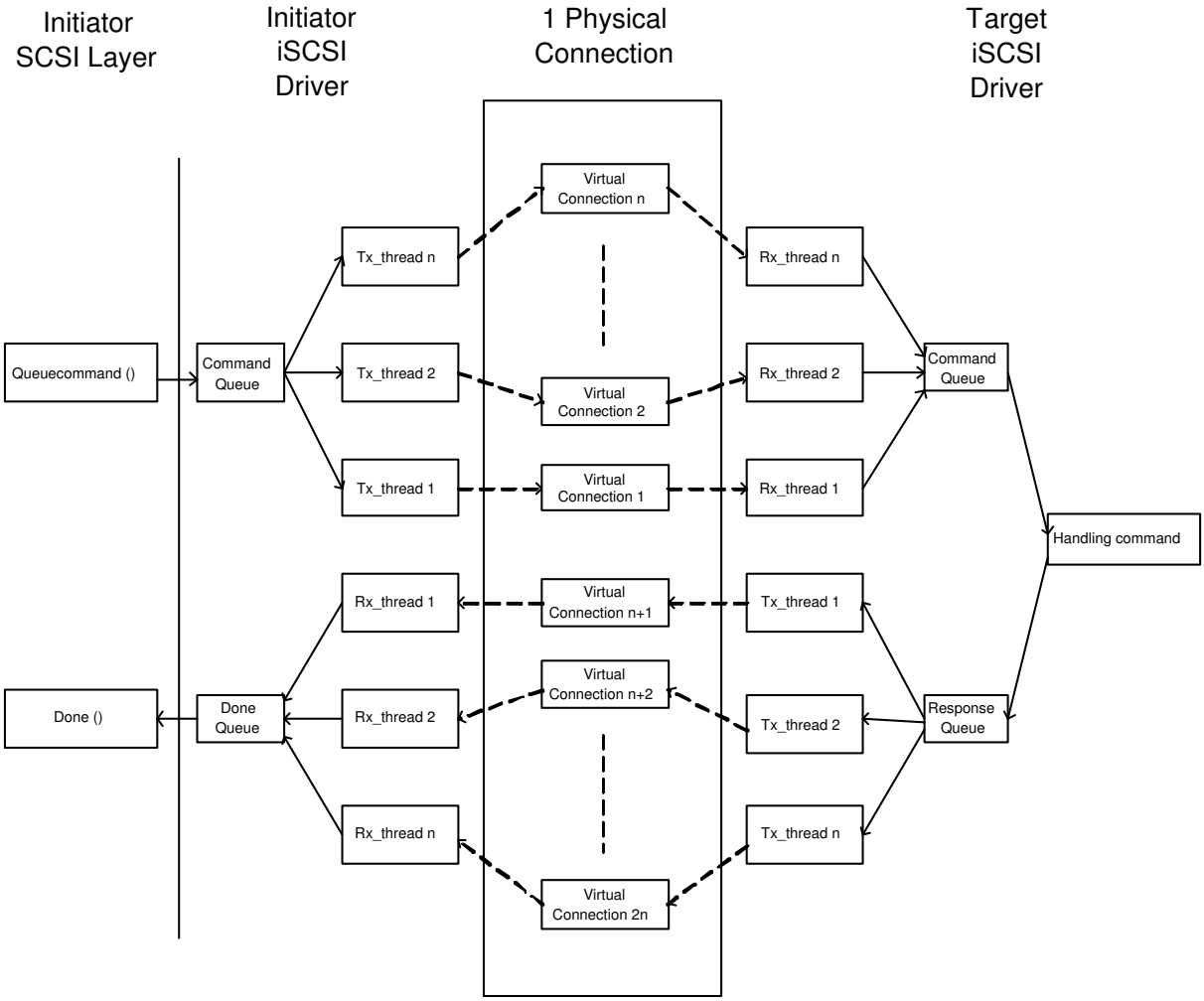

Figure 4.13: Multiple Virtual TCP Connection Architecture

The detailed communication procedure is illustrated by illustrating a typical read operation. The SCSI middle layer is a standard interface for SCSI layer to communicate with iSCSI device driver. The two main functions of SCSI middle layer are queuecommand ( ) which issues SCSI command to the iSCSI driver and done ( ) which informs SCSI middle layer that the command is finished by iSCSI driver. The iSCSI initiator driver gets read commands from SCSI middle layer. These commands are encapsulated in iSCSI request PDUs and queued in the initiator command queue.

As shown in Figure 4.13, in the initiator the transmitting threads (tx_threads)

(1, 2,....,n) of connections (1, 2,....,n) send these PDUs from command queue to the target. The target driver receives these PDUs by the receiving threads (rx_threads) (1, 2,....,n) of their corresponding connections. The target driver then de-encapsulate iSCSI request PDUs and map SCSI commands to the storage device (memory or disk). After getting data from storage device and forming iSCSI response PDUs (status or data), the target driver queues these iSCSI response PDUs in response queue. Tx_thread (1, 2,....,n) in the target send these PDUs by connection (n+1, n+2,....,2n). The initiator driver receives response through the rx_threads (1, 2,....,n) of their corresponding connections. Then the initiator driver put it to done queue to call the done ( ) function to finish the SCSI exchange.

It is noted that in the iSCSI design, the number of connections must be even (2, 4,...., 2n), because the issue of data transmission balance is considered. The transmitting thread (tx_thread) and receiving thread (rx_thread) are separated. They work independently and the number of tx_thread and the number of rx_thread is equal. In the initiator, for example, virtual connections 1 to n are used to transmit request PDUs to target and virtual connections n+1 to 2n are used to receive data and status from the target. This mechanism is a scalable design. High bandwidth utilization can be achieved by adding a pair of connections even when the wireless bandwidth become wider and wider in the future.

As multiple connections are used, synchronization becomes an important issue. The parallel transmission means several commands and data are transferred synchronously. In such a situation, each PDU must be distinguished even though the receiver'sequence of PDU is not the same as sender's sequence of PDU. According to iSCSI draft [36], the "initiator task tag" is used to record the sequence number of the iSCSI command in every iSCSI PDU. Related data and Status PDUs of the iSCSI command attach the same "initiator task tag". Even if multiple connections are used and command queue is enabled in both initiator and target, the device driver can easily find respective data or status PDUs from the pending queue. Although the sequence number of the iSCSI command is the extra

overhead which may introduce the delay and decrease the storage performance, this mechanism is necessary to solve the synchronization problem of multiple TCP connection design.

4.3.4 Queuing Model for Multiple Virtual Connections

The multiple virtual connection queuing model is established here to analyze the performance of single connection iSCSI and multiple virtual connection iSCSI, and illustrate the implication of the multiple virtual connection design on system performance. The queuing model is shown in Figure 4.14. The single connection model is a typical $M/M/1$ queuing model and the multiple connection model is a $M/M/m$ model.

For $M/M/1$ model, suppose the packet arrival rate is λ and service time is exponentially distributed with mean $1/\mu_1$. For $M/M/m$ model, suppose a communication link serving m independent Poisson traffic streams with overall rate λ and the link is divided into m separate channels with one channel assigned to each traffic stream. However, if a traffic stream has no packet awaiting transmission, its corresponding channel is used to transmit a packet of another traffic stream. The transmission times of packet on each of the channels are exponentially distributed with mean $1/\mu_m$.

For $M/M/1$ model, the average delay can be expressed as

$$T_1 = \frac{1}{\mu_1 - \lambda} \tag{4.1}$$

For $M/M/m$ model, the average delay can be expressed as

$$T_m = \frac{1}{\mu_m} + \frac{P_Q}{m\mu_m - \lambda} \tag{4.2}$$

Where P_Q denotes the queuing probability of $M/M/m$ model and can be

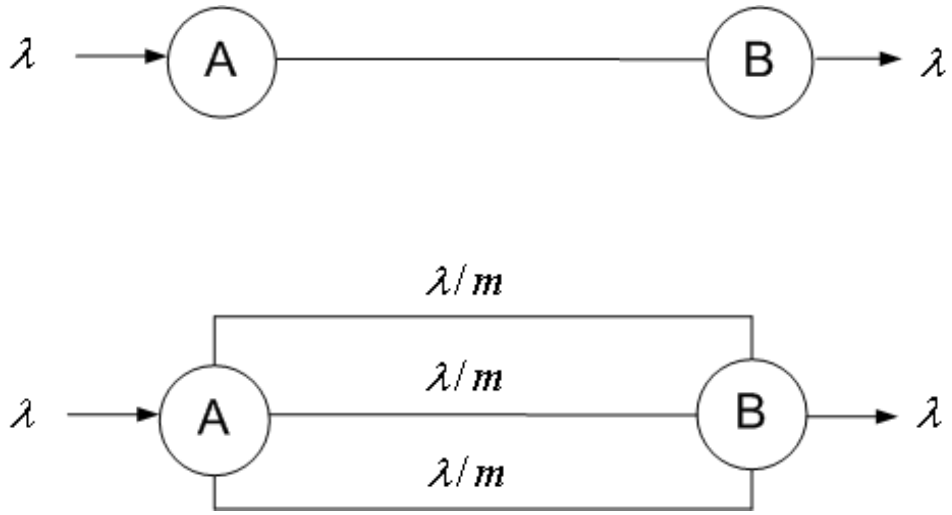

Figure 4.14: Single Connection and Multiple Connection Queuing Model

expressed as [75]

$$P_Q = \frac{p_0(m\rho)^m}{m!(1-\rho)} \tag{4.3}$$

Since the multiple virtual connections are based on one physical connection, the value of μ_m depends on the workload and the number of connections m. Here $M/M/2$ is discussed as a typical example.

In equation (4.2) when $\rho << 1$ (lightly loaded system), $P_Q \cong 0$ and

$$T_2 \cong 1/\mu_2 \tag{4.4}$$

For lightly loaded system in the multiple virtual connection over one physical link design, since one physical link can not be sufficiently used by single TCP connection, the multiple virtual connections do not need to contend for one physical link and the service rate μ_2 is approximately equal to μ_1. From equation (4.1) and (4.4), it can be seen that in lightly loaded system, $M/M/2$ model's packet delay is much less than $M/M/1$ model's packet delay.

In equation (4.2) when ρ is only slightly less than 1 (heavily loaded system), $P_Q \cong 1$, $1/\mu_2 << 1/(m\mu_2 - \lambda)$ and

$$T \cong \frac{1}{m\mu_2 - \lambda} \tag{4.5}$$

For heavily loaded system in the design, since one physical link almost can be used by single TCP connection, the multiple virtual connections usually need to contend for one physical link and the service rate μ_2 is much less than μ_1 but slightly greater than $\mu_1/2$. From equation (4.1) and (4.5), it can be seen that in heavily loaded system, $M/M/2$ model's packet delay is a little less than $M/M/1$ model's delay.

On average, the multiple connection iSCSI design is more efficient for packet transmission than single connection iSCSI. The improvement is more remarkable for lightly loaded system than heavily loaded system. In the iSCSI design, small I/O is considered as light loaded system and big I/O is considered as heavy loaded system. The experiment and test results stated in Chapter 6 will verify what we discuss here.

Chapter 5

Implementation and Experiment

In this chapter some implementation issues of the multiple virtual TCP connection iSCSI is explained. The experiment methodology and system setup are also illustrated.

5.1 Implementation Issues

The iSCSI prototype is developed in the commercial PC. All programs are based on Linux open source code (2.4.18-14).

5.1.1 Login Phase

During the iSCSI login phase, after the socket is created and bound, read and write operation can be achieved by using recvmsg() and sendmsg() functions.

```
Sendmsg(sock_fd, &msg, len);

recvmsg(sock_fd, &msg,len);
```

where msg is of the struct msghdr defined in /include/linux/socket.h

```
struct iovec

{
```

void *iov_base;

size_t iov_len;

};

The above iovec structure is filled with the data pointer and the length of the data to be passed to the kernel

memcpy_fromiovec(skb_put(skb,len), msg→msg_iov, len);

The data to be sent is filled up, the iovec structure initialized. The data from the iovec is copied and processed in the kernel.

The below lines are used to map the above iovec structure to TCP/IP

iov.iov_len = ISCSI_HDR_LEN;

iov.iov_base=login;

msg.msg_iov=&iov;

msg.msg_iovlen =1;

After the above mapping, the TCP connection can be established between the initiator and the target, then the login command PDU which was discussed in Section 4.2.2.1 can be used through the TCP connection.

5.1.2 Information Exchange

The text command and text response structures are defined totally the same as iSCSI draft [36]. Also the text parameters are set as follow:

struct text_parameter

{

char text_parameter_name[128];

unsigned int text_int_value;

int text_boolean_value;

char text_char_value[128];

};

One parameter includes a parameter name and its value. The value type can be integer, boolean or char. In the initiator part, There is a text parameter array with includes the six parameters stated in Section 4.2.2.2.

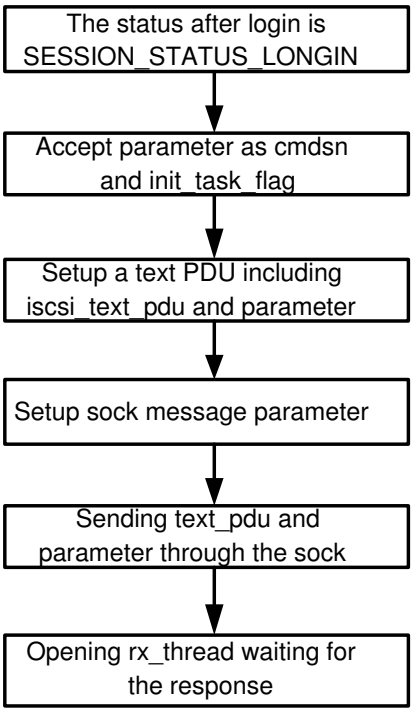

Figure 5.1: Initiator Sending Text Command

Figure 5.1 illustrates the flow chart of the initiator sending text command (parameter exchange). Figure 5.2 illustrates the flow chart of the target sending the text response to the initiator. The target will send a text response to initiator after handling all the text parameters. Then the target set the status to full feature phase. The initiator receives the text response by rx_thread and sets the status to full feature phase too.

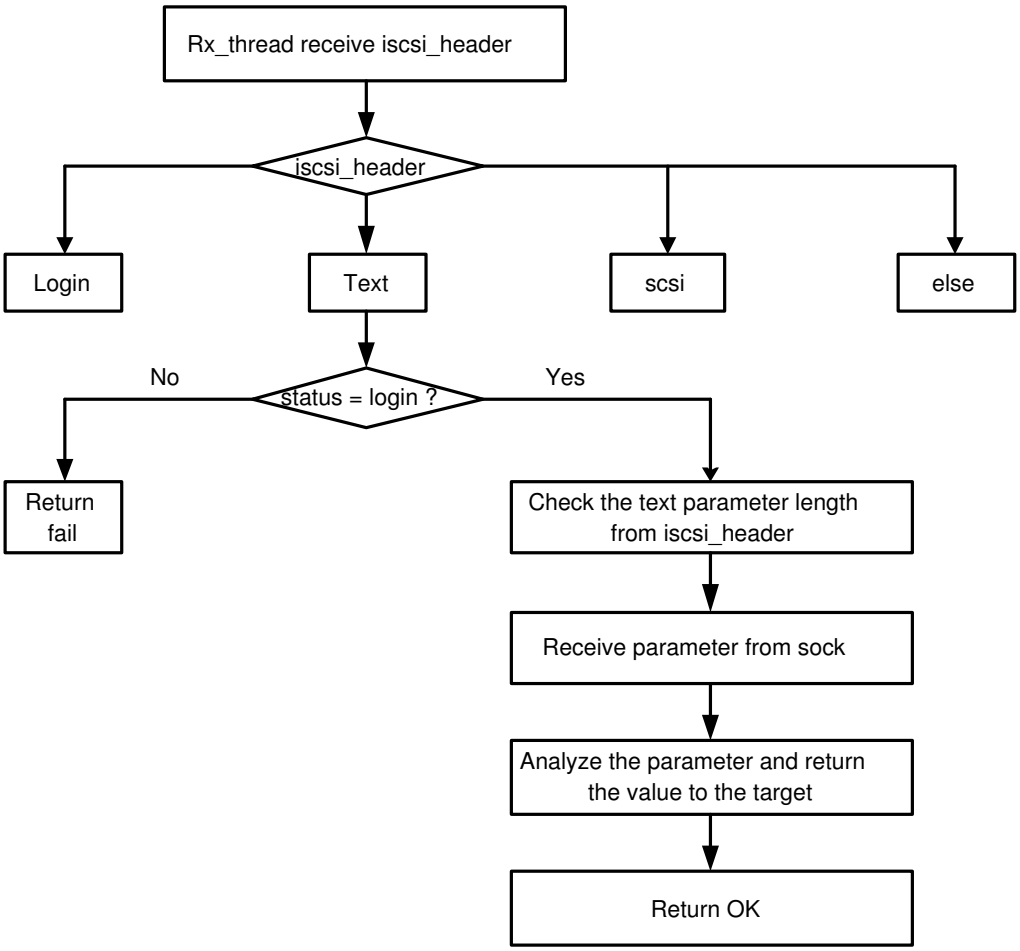

Figure 5.2: Target Sending Text Response

5.1.3 SCSI Command Implementation

Login command and text command (Information Exchange) should be the first two commands. After that, the initiator registers to SCSI middle layer, then SCSI middle layer will automatically send inquiry, test_unit_ready and read_capacity commands. These commands are discussed below:

- **INQUIRY:** The initiator typically issues several INQUIRY commands to detect which devices are on the SCSI bus. It starts at SCSI ID 0, LUN (Logical Unit Number) 0. If a device is located, the scan continues with SCSI ID 0, LUN 1, then LUN 2, and so on until either no device is located or LUN 7 has been scanned. At this point, the scan continues with SCSI ID 1, LUN 0, proceeding through to SCSI ID 7, LUN 7. The results from this

initial scan sends to upper layer. The operating system sees one LUN as one disk. The results from this initial scan may also be available to application and driver software through the use of the host's SCSI system routines.

The information returned by the INQUIRY command includes the peripheral qualifier and peripheral device type, vendor and product identification (DSI_SSIA, iSCSI) and the product revision level.

The CDB (command descriptor block) for an INQUIRY command for LUN 0 would be *12h 00h 00h 00h FFh 00h* and indicates the initiator has set aside a buffer of *FFh* bytes to receive the response.

The INQUIRY command will always return with GOOD status unless the initiator has set unsupported bits in the CDB.

A basic test of the INQUIRY data should include verification of the peripheral qualifier and peripheral device type.

- **TEST_UNIT_READY:** In the implementation this command is used to test the target's status. If the target can accept a medium-access command, e. g. a READ or a WRITE, it returns with a GOOD status. Otherwise, the command returns with a CHECK CONDITION status and a sense key of NOT READY. This response usually indicates that the target is completing power-on self-tests.

- **READ_CAPACITY:** This command has the standard structure of 10 byte commands and returns eight bytes of response. Four bytes reflect the last LBA (Logical Block Addressing) of the drive while the remaining four reflect the block length.

5.1.4 Data Transfer Operation

The read operation is discussed in Section 4.3.3. The write operation implementation is illustrated here. In the SCSI write, the R2T (Ready to Transfer) PDUs

fulfill the role of iSCSI layer flow control between target and initiator. The R2T PDUs are issued by the target device as buffers become available to receive more data. At the completion of the write, the target issues status and sense, indicating a successful transaction. Outgoing SCSI data (initiator write to target data or command parameters) will be sent as either solicited data or unsolicited data which depends on the negotiation in login phase. Solicited data are sent in response to R2T PDUs. Unsolicited data can be part of an iSCSI command PDU ("immediate data") or an iSCSI data PDU. An initiator may send unsolicited data (immediate or in a separate PDU) up to the negotiated limit (initial burst size). All subsequent data have to be solicited. The maximum size of an individual data PDU is negotiated at login phase.

During write operation, an initiator must hold data until it has received the status for the corresponding command. Even if the initiator sends immediate data or unsolicited data, the target may discard the data in case it does not have the resources to handle the data at that instant. The target may then request that the data to be resent.

A target does not need to keep a copy of the data buffers it has sent, if such data can be regenerated from the storage device. However, the target must keep around the status information until the initiator has acknowledged it. The initiator sends status acknowledge information to the target. If strict ordering between commands is needed (such as reading and writing of the same device) then the application must perform the proper synchronization by not issuing the second command until it has received the status of the first command (as in linked commands). The flow chart of the write operation in the implementation is in Figure 5.3.

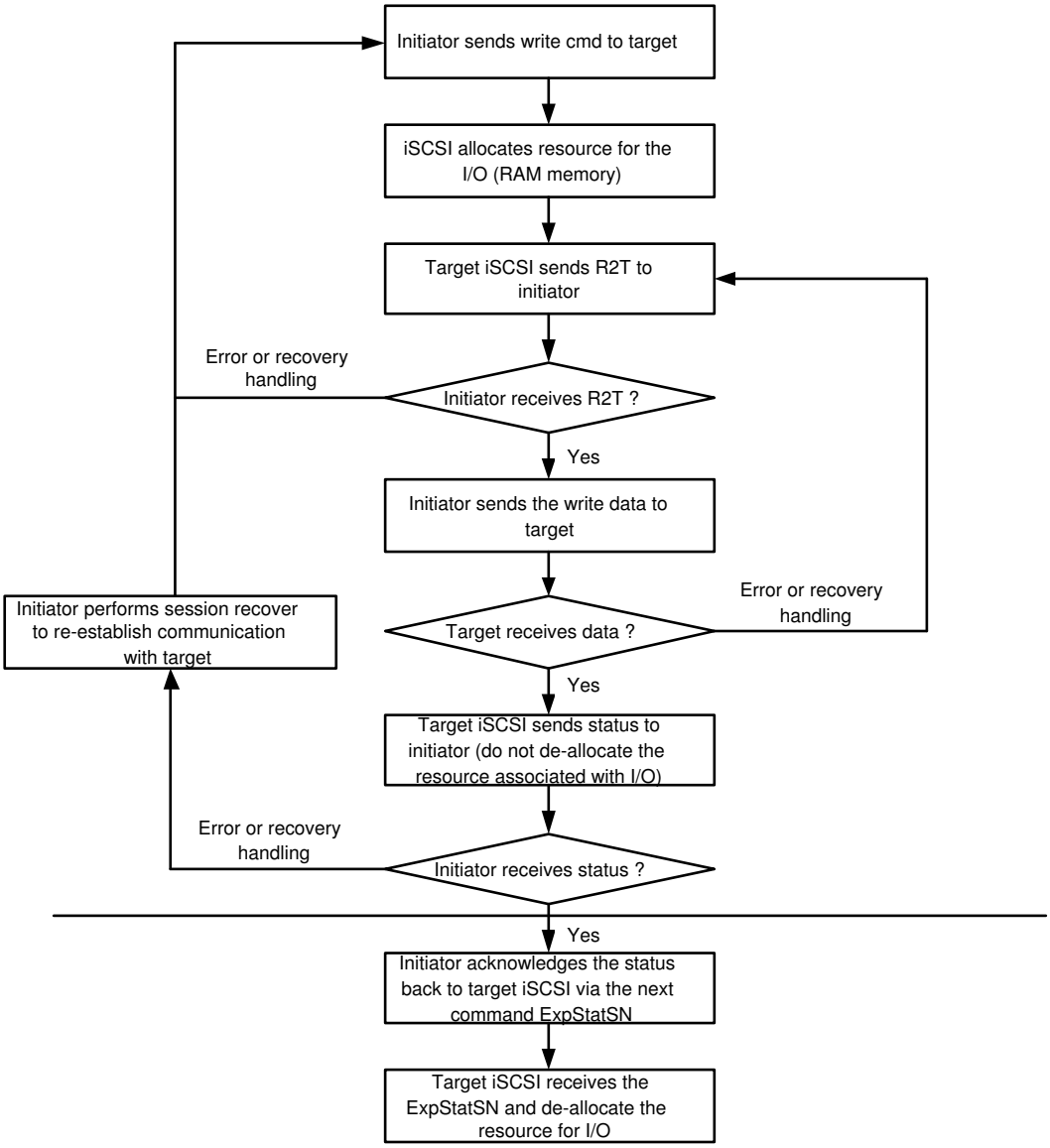

Figure 5.3: Flow Chart of iSCSI Write Operation

5.1.5 Semaphore Implementation

Since multiple threads (tx_thread and rx_thread) is used in the iSCSI design, sometimes semaphore should be used. Semaphores can be described as counters used to control access to shared resources by multiple processes or threads. They are most often used as a locking mechanism to prevent processes from accessing a particular resource while another process is performing operations on it or they are used for synchronization between threads. Semaphore has two basic operations: up and down. Up operation is an atomic operation that waits for semaphore to become positive, then decrements it by 1. Down operation is an atomic operation that

increments by 1. In Linux, up and down operating is used as P and V operation.

In the program, semaphore is used in two circumstances. One is for making a thread blocked and active under some conditions, another is for different threads to use the same special data structure.

For example, in the initiator, these two kinds of semaphore are used. Tx_sem is used between queuecommand and tx_thread (queuecommand is called by SCSI mid-layer and can be seen as another separate thread). Every time queuecommand adds a SCSI command to the end of the command queue. Then it processes a up operation to wake up tx_sem. In the tx_thread, it uses a down operation to block itself. Figure 5.4 illustrates this kind of semaphore. Another semaphore initiator_sem is needed for mutual exclusion when visiting a command queue in different threads. Figure 5.5 illustrates this kind of semaphore in the program.

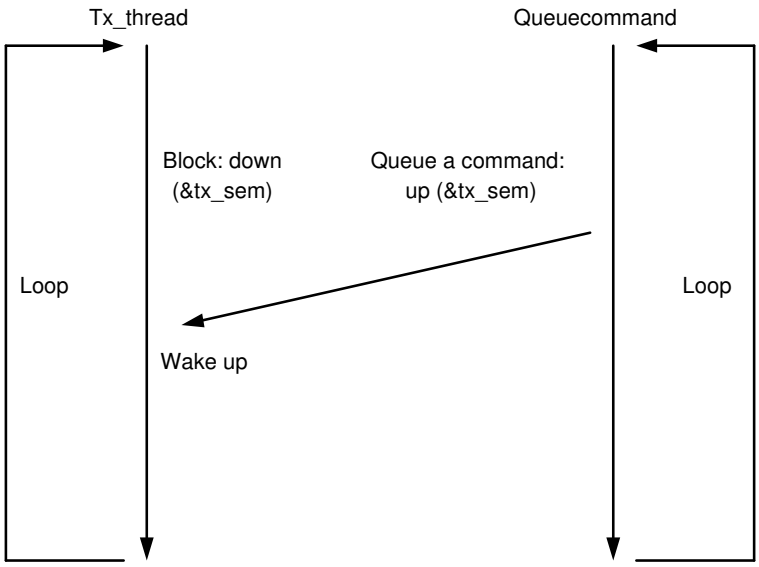

Figure 5.4: Tx_sem Semaphore

5.2 Experiment Setup

The main storage testbed used in the experiments consists of a target and an initiator connected by wireless LAN 802.11b. Figure 5.6 shows the main platform of the iSCSI experiments. Table 5.1 shows the system configuration and experimental pa-

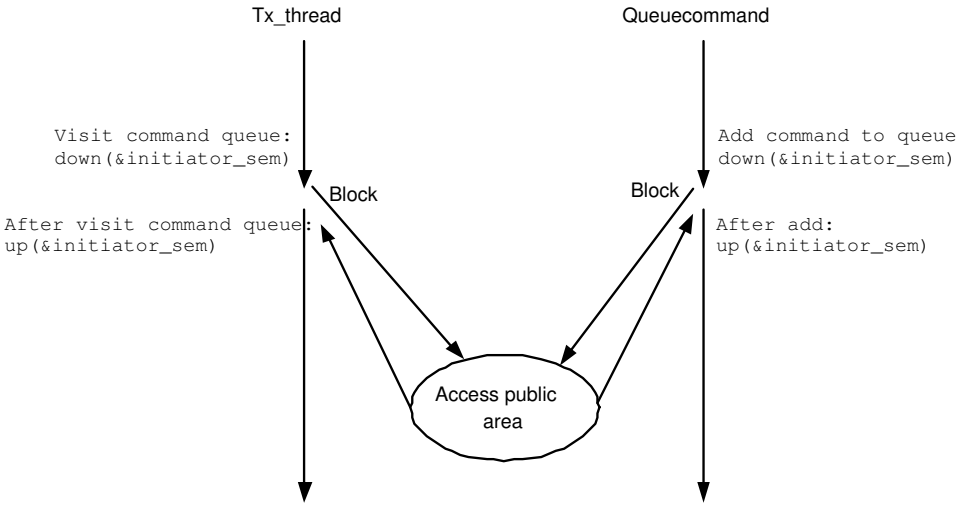

Figure 5.5: Initiator_sem Semaphore

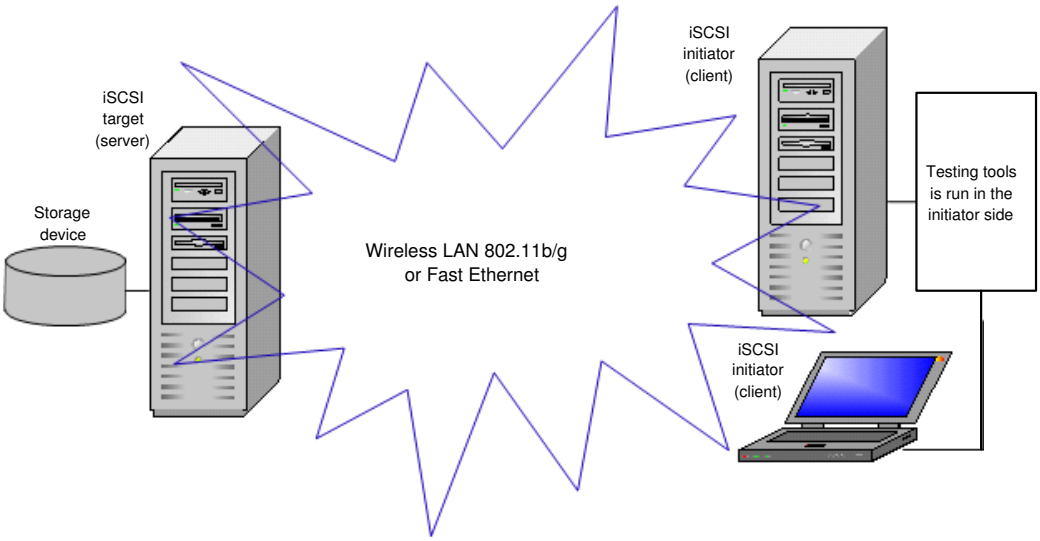

Figure 5.6: Main iSCSI Experiment Setup

rameters. The hardware configuration of the iSCSI target includes a 866MHz PIII processor, 256M RAM and a Cisco Aironet 350 series PCMCIA 802.11b Wireless LAN Adaptor. The Adaptec 39160 SCSI card is used to connect an IBM 18.2G SCSI hard disk. The initiator is implemented in a 733MHz PIII machine with 256M RAM and a Cisco Aironet 350 series PCMCIA 802.11b Wireless LAN Adaptor. In order to test the performance of iSCSI under various connection environments, for some tests, 3com 3c59x fast ethernet cards in target and initiator are used to test the iSCSI performance under wired environment, for other tests, Linksys Wireless-G notebook adapter and SMC 802.11g wireless PCI card in target and initiator respectively are used to test the iSCSI performance under 802.11g.

Table 5.1: Experiment Configuration

	Target	Initiator
Hardware	Intel PIII 866 MHz CPU 256M RAM Cisco PCMCIA 802.11b adapter 3com 3c59x FE card SMC 802.11g wireless PCI card Adaptec 39160 SCSI card IBM 18.2G hard disk	Intel PIII 733 MHz CPU 256M RAM Cisco PCMCIA 802.11b LAN adapter 3com 3c59x FE card Linksys wireless-G adapter
Software	Redhat 8.0 Linux kernel 2.4.18-14 ext 3 NFS version 3	Redhat 8.0 Linux kernel 2.4.18-14 (modified) ext3

Table 5.2: Testing Tools

Chariot 5.0	TCP layer Throughput in Windows
Netperf	TCP layer Throughput in Linux
dd command	Comparison of iSCSI and NFS (throughput and latency)
IOMeter	Multiple connection iSCSI throughput Multiple connection iSCSI IOPS Multiple connection iSCSI response time

Both machines run Redhat 8.0 with the Linux kernel version 2.4.18-14. Default Linux implementation of NFS version 3 is used for the experiment to compare the performance of iSCSI and NFS. The default file system used in the experiments is ext3 (Linux Extended File System 3). The ext3 file system resides at the client for iSCSI experiments and at the server for NFS experiments respectively. The TCP is used as the default transport protocol in the experiments.

5.3 Methodology

Table 5.2 simply lists the tools used in the experiments. The TCP network performance is tested by Netperf in Linux environment and Chariot 5.0 in Windows environment. The objective of this test is to verify the TCP layer throughput that can provide for the upper iSCSI layer and compare it with theoretical analysis results of wireless 802.11 as discussed in Section 3.2. The performance of two storage schemes: file level storage (NFS) and block level storage (iSCSI) is also compared.

Throughput, which is the sustained data transfer rate between an initiator and the storage device, and the response time, which is the time interval for an initiator from issuing an iSCSI command to the request has been served by the target, are used as two main metrics in the performance study. dd command is used as the application benchmark tool to copy to or from iSCSI disk. For example, "dd write" is "dd if=/dev/zero of=/dev/sda bs=4k count=250000" and "dd read" is "dd if=/dev/sda of=/dev/zero bs=4k count=250000". It will generate 1GB data to be written to iSCSI disk and generate 1GB data to be read from iSCSI disk respectively.

IOMeter, an industry standard I/O benchmark tool, is used to test the performance of the multiple virtual connection iSCSI with respect to throughput with different number of connections and network latency, IOPS with different queue length for small I/O. The experiments are also conducted to show the effect of lower layer parameter to iSCSI performance. Multiple virtual connection iSCSI is also tested under wired environment and 802.11g environment to further analyze its performance.

In order to reduce the effect of the storage device on the iSCSI performance under wired environment, RAM I/O mode is set in iSCSI target side in that experiment. All of the I/O data are directly mapped to the RAM rather than to the logical block of the actual magnetic disk. But in the other experiments of the iSCSI over wireless network, since the main bottleneck is the low wireless bandwidth, disk I/O mode is chosen.

In Linux kernel 2.4.18-14 or 2.4.20, if the kernel source code is not modified, the benchmark tools ("dd" and "IOMeter") will generate SCSI command, which request 128K data to SCSI layer. In the small I/O test, the kernel source code in /usr/src/linux-2.4.18-14/drivers/block/ll_rw_blk.c needs to be modified to allow the initiator to generate 2K $\sim$ 32K small I/O request to SCSI layer. It is also noted that no matter what test is conducted, the whole request size must be set much bigger than initiator's memory to avoid the impact of local cache.

Chapter 6

Performance Evaluation and Result Discussion

In this chapter, the performance of the prototype is evaluated and analyzed. The TCP layer net throughput and the comparison of the results of iSCSI and NFS are also discussed in this chapter. The multiple connection iSCSI results are compared with normal single connection iSCSI results in both wired and wireless environment. The analysis is extended to the multiple connection iSCSI performance under different network conditions, different network parameters to identify some of the key issues of iSCSI performance. It is noted that all experiments are conducted with disk I/O if no specific illustration.

6.1 TCP Layer Throughput Result

Since TCP layer throughput is a very important factor that affect the performance of iSCSI as what is discussed in Section 3.2, the TCP performance of the system in Windows and Linux operating system is tested by Netperf and Chariot 5.0

respectively. Figure 6.1 is the test result from Chariot 5.0. The sample time is 26s, the maximum throughout is 5.195 Mbps, the minimum throughput is 3.582 Mbps and the average throughput of the TCP layer is 4.903 Mbps which accord with the theoretical analysis in Section 3.2.

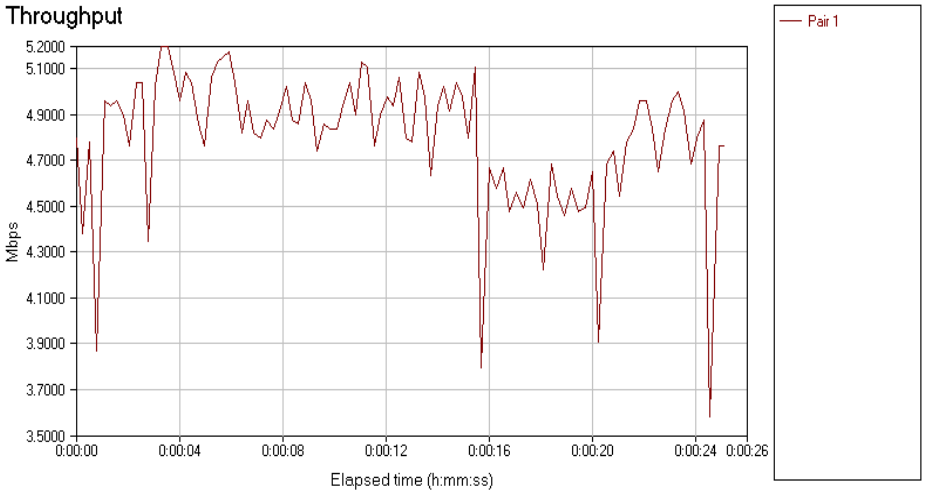

Figure 6.1: TCP Layer Throughput in Windows (P2P)

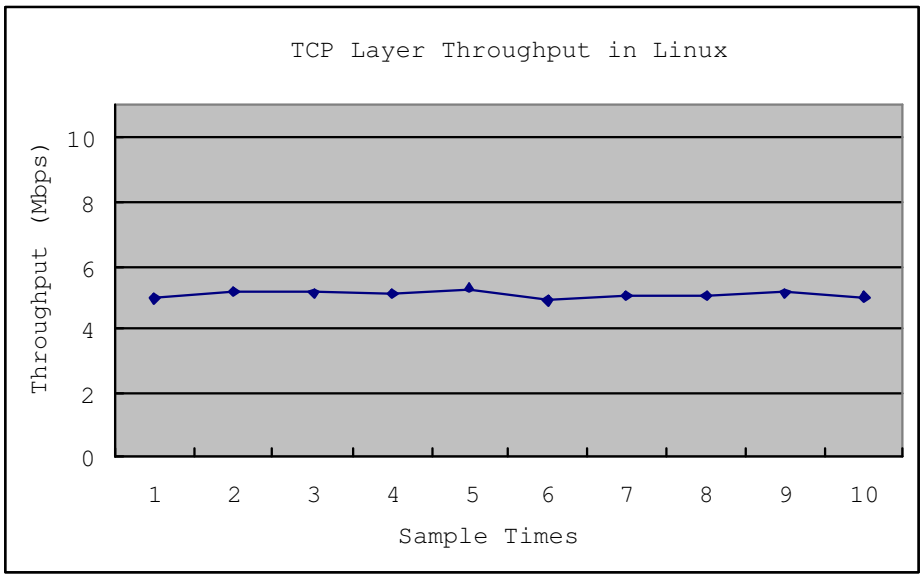

Figure 6.2: TCP Layer Throughput in Linux (P2P)

Figure 6.2 shows the results under Linux environment. The TCP throughput is tested 10 times and the average throughput is 5.06 Mbps which is also in the range of the theoretical analysis.

The TCP performance test is also conducted by using Linksys wireless access

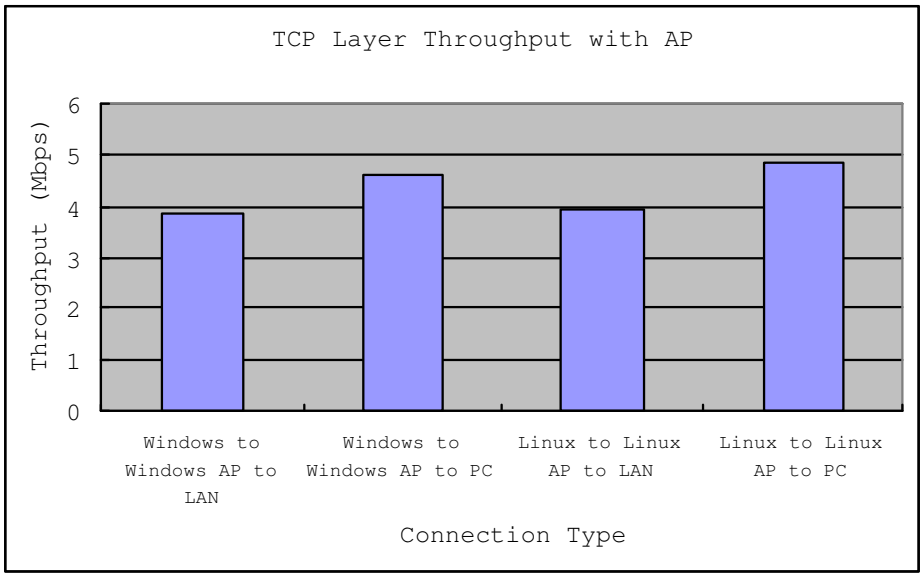

Figure 6.3: TCP Layer Throughput with Access Point

point router connected to LAN or directly to the other computer. The results are shown in Figure 6.3. All test results are less than the results with P2P connection. This is because connecting to access point adds one more hop to the communication link, the packet loss and the overhead definitely increase. From what has been discussed above, it can be seen that the theoretical calculation in Section 3.2 that TCP layer net throughput of 802.11b is around 45% to 47% (4.9 Mbps to 5.2 Mbps or 0.61 MB/s to 0.65 MB/s) with default 1,500 bytes packet size of the raw data rate 11 Mbps accords with the test results (for Windows: 4.903 Mbps or 0.61 MB/s and for Linux: 5.06 Mbps or 0.63 MB/s), thus it is verified by the experiment test results. These results will also be used to compare with throughput of the multiple connection iSCSI tested in block level to highlight the contribution of this thesis.

6.2 Performance Comparison of iSCSI and NFS

Figure 6.4 shows the read throughput comparison of normal single connection iSCSI and NFS. The throughput of iSCSI always outweighs NFS. For small I/O request, 4K for example, normal single connection iSCSI throughput is almost 50% higher than NFS and for big I/O request (128K), iSCSI throughput is still about 4%

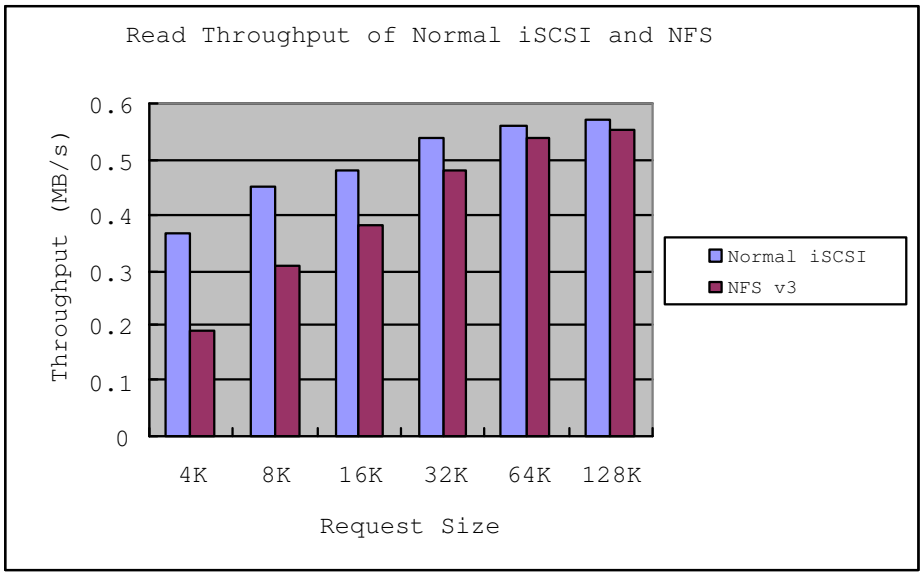

Figure 6.4: Read Throughput of Normal iSCSI and NFS version 3

higher than NFS. This is because to access the raw device from block level such as iSCSI can achieve higher throughput than access storage device from file level such as NFS as what is discussed in Section 3.5 and 3.6.

Figure 6.5 displays the latency on the access of file objects with the iSCSI and NFS schemes respectively. The test is conducted to transfer a 1G file between the client (initiator) and the server (target). For the write operations, the iSCSI significantly outperforms the NFS. As what can be seen, the read operations also have a similar trend. The lower data transfer latency of iSCSI is mainly because the asynchronous operation in initiator ext3 file system of iSCSI. In addition, NFS is a stateless protocol and runs on the file level, it requires the exchange of several commands before data communication can occur which consumes more time.

The test results show that iSCSI based storage can reach high bandwidth utilization and provide higher storage performance than NFS based system. In order to achieve higher throughput and network utilization, the iSCSI gain an advantage over NFS as what is discussed in Section 3.5 and 3.6.

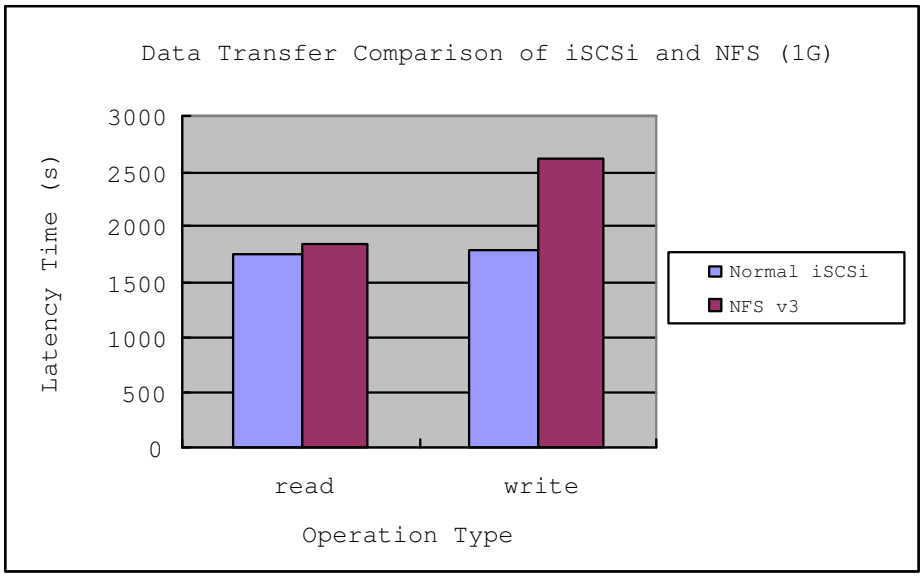

Figure 6.5: Data Transfer Time Comparison of Normal iSCSI and NFS version 3

6.3 Normal iSCSI Test Result Analysis

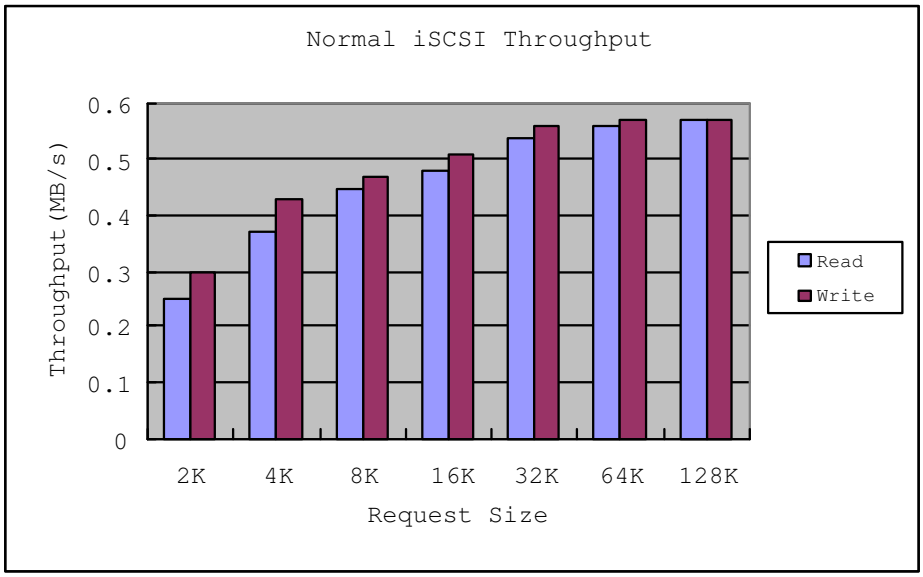

Figure 6.6: Normal Single Connection iSCSI Throughput

Figure 6.6 shows the normal single connection iSCSI throughput by using wireless LAN 802.11b. The I/O request size range from 2K to 128K and the low layer 802.11 frame size is set as default size of 1,500 bytes. From Figure 6.6 it can be seen that for small I/O request (2K $\sim$ 8K), the throughput is far less than the maximum throughput in theory (0.61 MB/s $\sim$ 0.65 MB/s or 4.9 Mbps $\sim$ 5.2

Mbps) as what is discussed in Section 3.2 and it is less half compared to big I/O request. This is because the occurrence of the time gap between the consecutive 802.11 frames, the waiting time for the status before sending the next I/O request in iSCSI layer and the overhead imposed by frequent small I/O request.

The other experiment is conducted by manually opening several ports in target side to form several targets in one machine with the same IP address using the normal iSCSI and establishing several connections between single initiator and target. One wireless LAN 802.11b physical channel is used for all of these connections. Figure 6.7 and Figure 6.8 shows the read and write test results from the experiment.

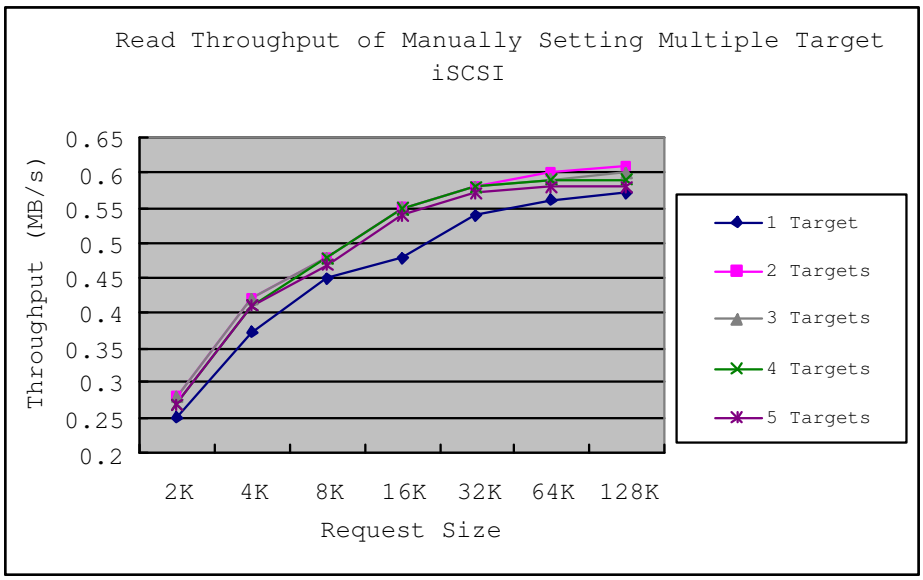

Figure 6.7: Read Throughput of Manually Setting Multiple Target iSCSI 802.11b

From the figures, it can be seen that whether for read throughput or write throughput, two targets can achieve 10% throughput improvement for small I/O and for big I/O, throughput is 0.58 MB/s which is still less than theoretical analysis result as what is discussed in Section 3.2. However, the throughput of two targets has the similar trend as normal single connection iSCSI. For small I/O request, the throughput is still quite low and far less than the maximum throughput the wireless channel can achieve. This is because that each connection is for one target and although multiple connection within the same channel can further utilize the

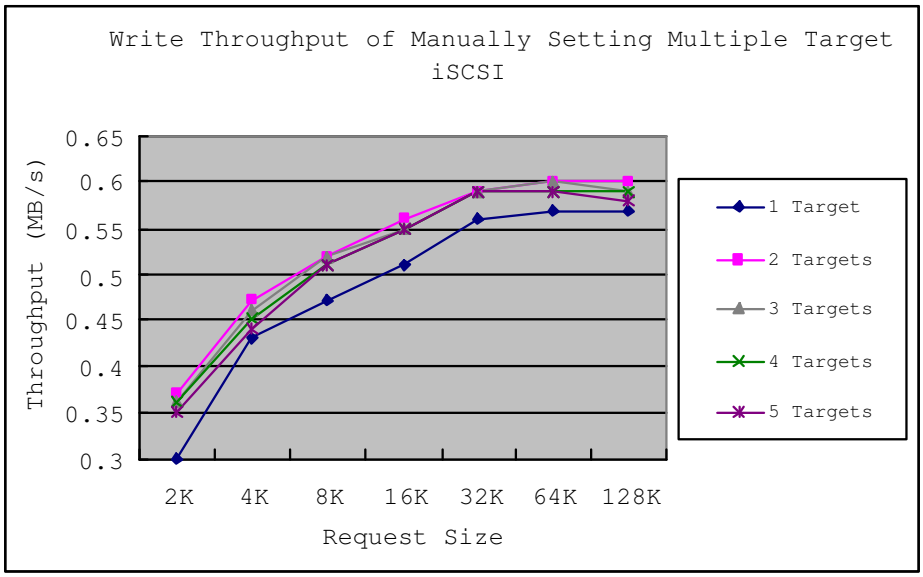

Figure 6.8: Write Throughput of Manually Setting Multiple Target iSCSI 802.11b

time gap between the consecutive frames of the low layer in a certain extent, the initiator still need to wait for the status before issuing the next I/O request. In addition, since more connections may add more overheads to the system due to the handover among different independent connections and threads, the throughput of 3, 4 and 5 connections is lower than the throughput of 2 connection.

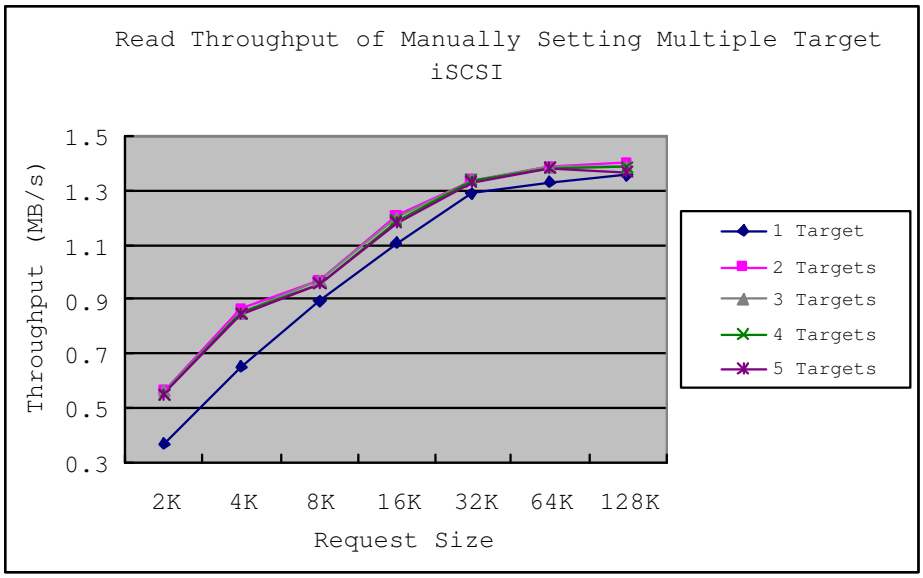

Figure 6.9: Read Throughput of Manually Setting Multiple Target iSCSI 802.11g

The same experiment is also conducted under wireless LAN 802.11g, the test results are shown in Figure 6.9 and Figure 6.10. The test results have the same

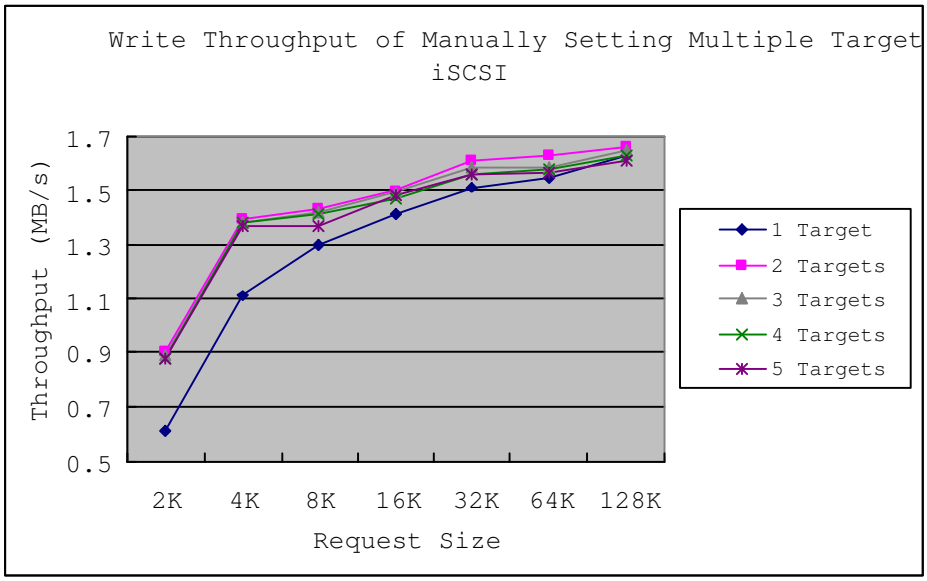

Figure 6.10: Write Throughput of Manually Setting Multiple Target iSCSI 802.11g

trend as the test results under 802.11b. Thus in order to achieve high storage performance, multiple virtual connection within one session should be adopted.

6.4 Multiple Connection iSCSI over Wireless Network Result Analysis

6.4.1 Comparison of Normal iSCSI and Multiple Connection iSCSI

Figure 6.11 and Figure 6.12 show the read and write throughput comparison of multiple connection iSCSI with normal iSCSI in disk I/O mode by using wireless LAN 802.11b. The I/O request size range from 2K to 128K and the frame size is set as default size of 1,500 bytes. From Figure 6.11 and Figure 6.12, it can be seen that for small I/O (2K $\sim$ 8K), the normal single connection iSCSI's throughput is far less than the maximum throughput in theory (0.61 MB/s $\sim$ 0.64 MB/s) and it is very low compared to big I/O request. However for small I/O request (2K $\sim$ 8K), the multiple connection iSCSI can achieve a significant throughput improvement

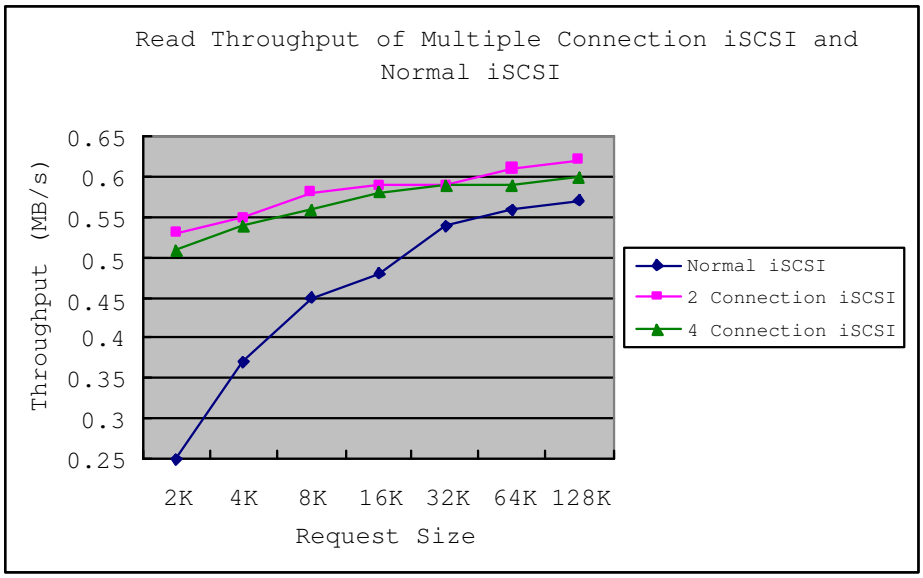

Figure 6.11: Read Throughput of Multiple Connection iSCSI and Normal iSCSI

compared to normal single connection iSCSI. For 2K request size, for example, the throughput is improved from 0.25 MB/s to 0.53 MB/s, which is about 112% improvement. The wireless environment is very unreliable. In normal iSCSI, the initiator needs to wait for a long time before sending the next I/O request because of the packet retransmission for the ACK status. Especially for small I/O, the requests are sent frequently compared to big I/O, so much time is wasted when waiting for the status. And for small I/O request, if the request size can not be divided exactly by the maximum frame size in the low layer, there are a lot of 802.11 frames, which do not sufficiently utilize each frame size.

In multiple TCP connection iSCSI design, the above mentioned problems can be solved. By using the multiple virtual TCP connections in an iSCSI session and the parallel working mechanism, the iSCSI driver can utilize the wireless channel more efficiently, which means the iSCSI driver can continuously send I/O requests to low layer via different connections and does not need to wait for the ACK status before sending the next I/O request. Also due to the continuous requests from the iSCSI layer, each 802.11 frame can be sufficiently used to carry the data. In wireless storage, since the demand for data is not as large as in wired network, multiple connection iSCSI design that significantly improve throughput for small

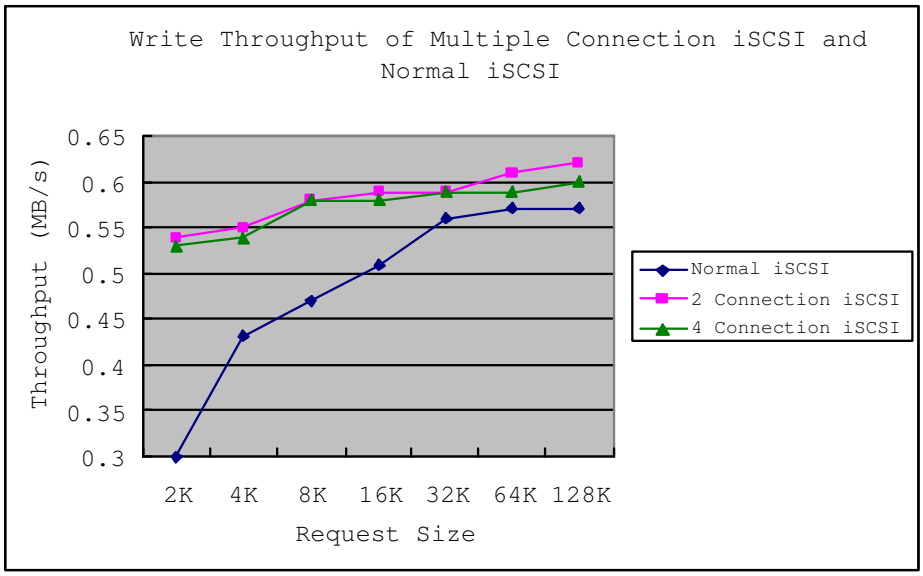

Figure 6.12: Write Throughput of Multiple Connection iSCSI and Normal iSCSI

I/O is very valuable for the application in wireless environment.

For big I/O request (128K), there are also some improvements. The iSCSI maximum throughput can reach 0.62 MB/s, which is very close to the the TCP layer net throughput tested in Section 6.1 (0.63 MB/s for Linux) and in the range of the theoretical analysis result stated in Section 3.2. The maximum throughput of the upper iSCSI layer, which is very close to both the theoretical and experiment TCP layer net throughput, verifies that the system overhead within block level in multiple connection iSCSI is very small. For the system, by using 2 connections, it can achieve the maximum throughput. Adding more connections, the test results show that the system performance reduces due to imposing more overheads on the system.

The test is also conducted over 100 Mbps FE (Fast Ethernet) wired network. The results are shown in Figure 6.13. It can be seen that the multiple connection iSCSI cannot achieve such a significant throughput improvement compared to normal single connection iSCSI over wired network. In addition, the performance of multiple connection iSCSI increases with the request size, which is the same trend as normal iSCSI. This is because wired network is reliable, the overhead caused by retransmission is small, the normal iSCSI can already achieve a relatively high

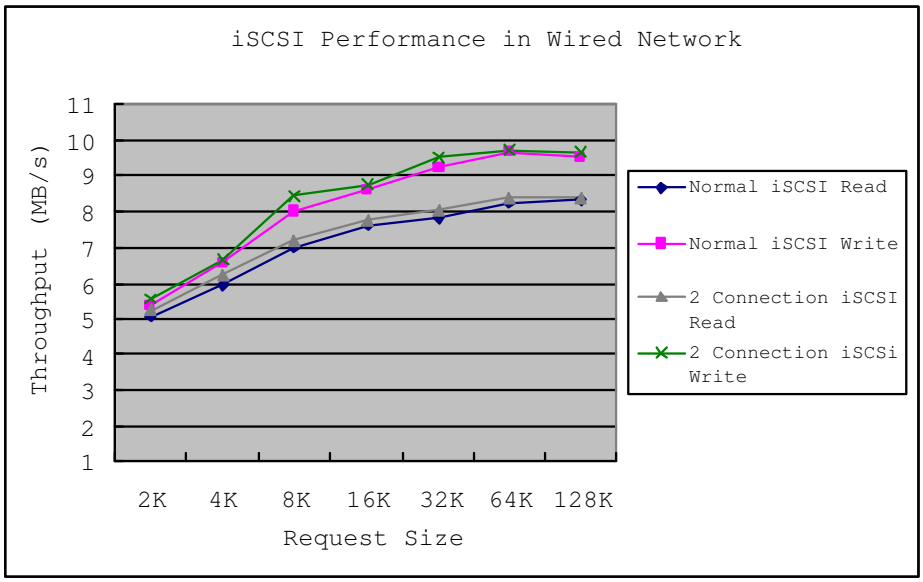

Figure 6.13: iSCSI Performance over Wired Network

throughput. What's more, the overhead caused by the header and the driver to process a SCSI command, such as checking the command, preparing data and preparing status is the inherent overhead of the system and cannot be reduced, thus the multiple connection iSCSI cannot achieve a great improvement.

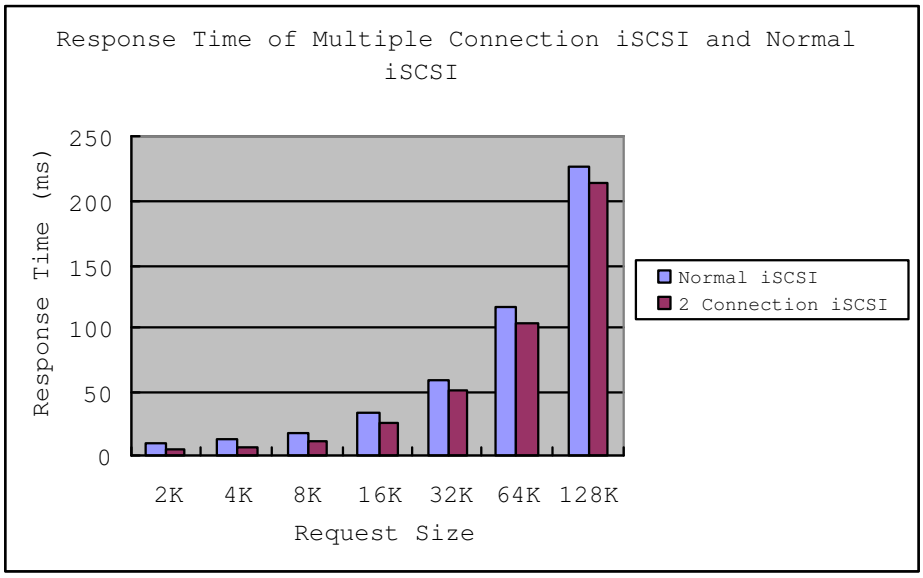

Figure 6.14: Response Time of Multiple Connection iSCSI and Normal iSCSI

The average response time of normal iSCSI and multiple connection iSCSI is tested over wireless LAN 802.11b. The test results are shown in Figure 6.14. The response time of multiple connection iSCSI is always lower than the response

time of normal single connection iSCSI. For small I/O, the response times are only half of the normal iSCSI. As to the big I/O, the response times are also a little lower (6%) than normal iSCSI. This is also because the multiple connection iSCSI design. The upper layer does not need to wait for the status sent back before it can issue the next I/O request, which save the time and make the response times lower than normal single connection iSCSI.

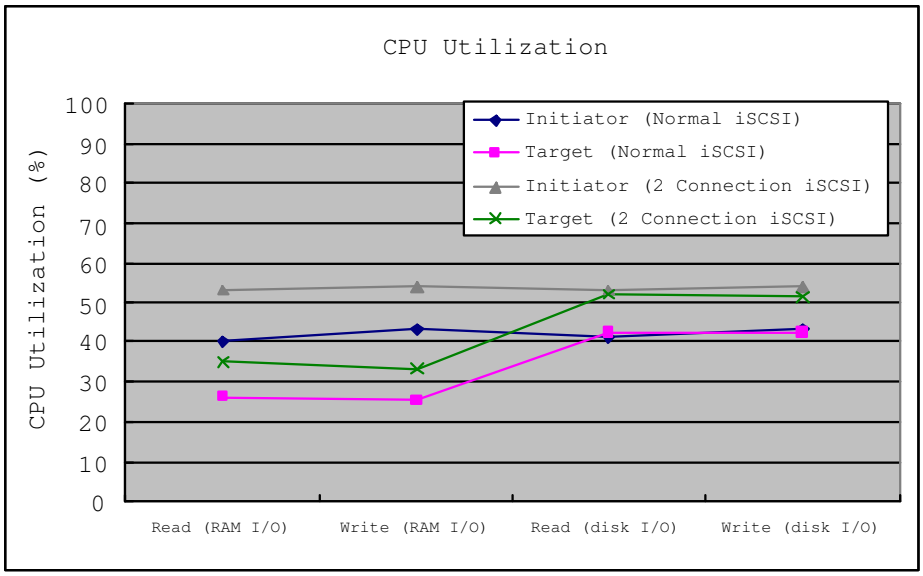

Figure 6.15: CPU Utilization of Multiple Connection iSCSI and Normal iSCSI

Further experiment is conducted to analyze the CPU utilization of big I/O (128K) request. In this experiment, the read and write operation is tested with disk I/O mode and RAM I/O mode. The test results are shown in Figure 6.15. It can be seen that if disk I/O mode is used, the initiator side and the target side CPU utilizations is almost the same for normal iSCSI and 2 connection iSCSI respectively.While if the RAM I/O is used, whether it is normal iSCSI or 2 connection iSCSI, the CPU utilization in the initiator side is always higher than that of the target side. This is because the initiator needs to handle one more data copy in memory than that of the target as shown in Figure 4.2. Also Figure 6.15 shows that the CPU utilization of the multiple connection iSCSI is only a little higher than normal iSCSI. For example, in initiator side, when disk I/O mode and read operation is used, the CPU utilizations of normal iSCSI and multiple connection

iSCSI is 41% and 53% respectively. Because in GE (Gigabit Ethernet) environment, the CPU utilization can almost reach 100%, a conclusion can be drawn that the system's main bottleneck is still on the low wireless bandwidth itself.

6.4.2 iSCSI Throughput with Different Network Latency

It is commonly supposed that network latency has a great impact on the iSCSI performance, especially in wireless environment. The experiment is conducted to test the network latency's impact on the multiple connection iSCSI to illustrate that this new iSCSI design can still achieve good performance in long latency and unreliable wireless environment.

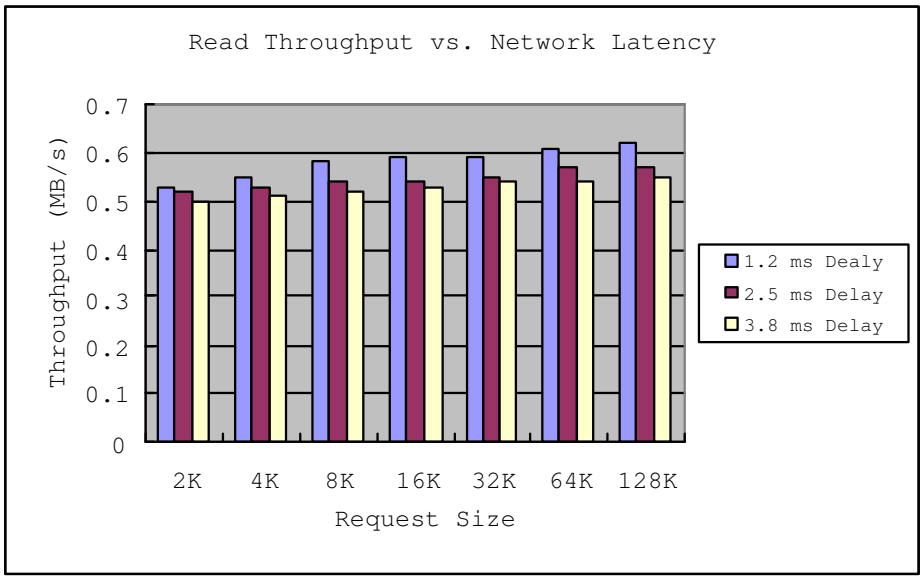

Figure 6.16: Read Throughput vs. Network Latency

Figure 6.16 shows the read throughput with different network latency over wireless LAN 802.11b. The network latency in this experiment is estimated by the ping command. The 1.2 ms delay refers to peer to peer ad-hoc network while the 2.5 ms delay and 3.8 ms refers to the communication between initiator and target with one hop and two hops respectively. From Figure 6.16, it can be seen that although the network latency is significantly increased, the throughput of the multiple virtual connection iSCSI can still achieve good storage performance. For

big I/O request, e. g. 128K, although the delay is doubled (2.5 ms)or tripled (3.8 ms), the throughput is only 8% or 11% less than peer to peer network scenario (1.2 ms). For small I/O request, e. g. 2K, the decrease is only 2% and 5% for 2.5 ms delay and 3.8 ms delay respectively. Normal iSCSI is also tested with the network latency of 2.5 ms, the throughput decrease is around 15% and 30% for small I/O request and big I/O request respectively.

The above mentioned achievement for long latency wireless network is due to the multiple virtual TCP connection design and the architecture that can support long queue. In wireless environment, the multi-hop channel and the unreliable signal strength can delay the data transmission and make the network latency quite long. The experiment verifies that the multiple connection iSCSI design is effective enough to achieve high throughput in such scenario.

6.4.3 The Impact of Network Parameters on iSCSI Performance

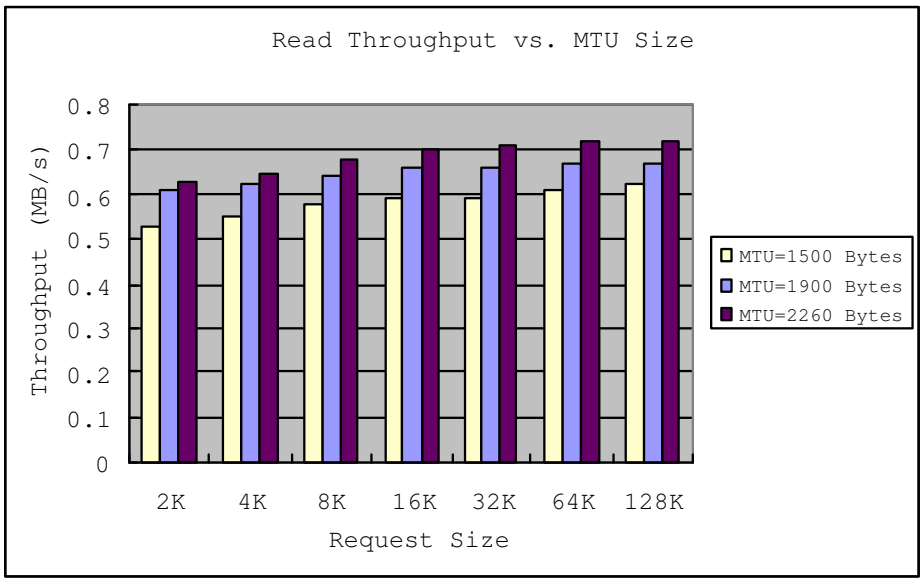

Figure 6.17: Read Throughput vs. MTU size

The packet size's effect on the iSCSI transmission performance over wireless LAN 802.11b is discussed. The experiment is designed working on different packet

size to demonstrate the influence of packet size on the multiple connection iSCSI data transmission performance. Figure 6.17 shows the achieved throughput of 2 virtual connection iSCSI with different MTU (Maximum Transmission Unit) size.

Cisco Aironet 350 series PCMCIA 802.11b Wireless LAN Adaptor's maximum MTU size is 2,260 bytes, thus the maximum packet size used in the experiment is 2,260 bytes. From Figure 6.17, it can be seen that for large MTU size, e. g. 2,260 bytes, the throughput for 128K request is 0.72 MB/s, while for default MTU size, e. g. 1,500 bytes, the throughput for 128K is only 0.62 MB/s. A conclusion can be drawn that the iSCSI performance is increased as the MTU size is increased all the time. This is mainly because the big frame can carry more data, which make the payload size greater in proportion to TCP header overhead than that of small frame size and the big frame can decrease the frequency of interruption of wireless adapter for the same size of data transmission.

6.4.4 The Impact of Queue Length on I/O Rate

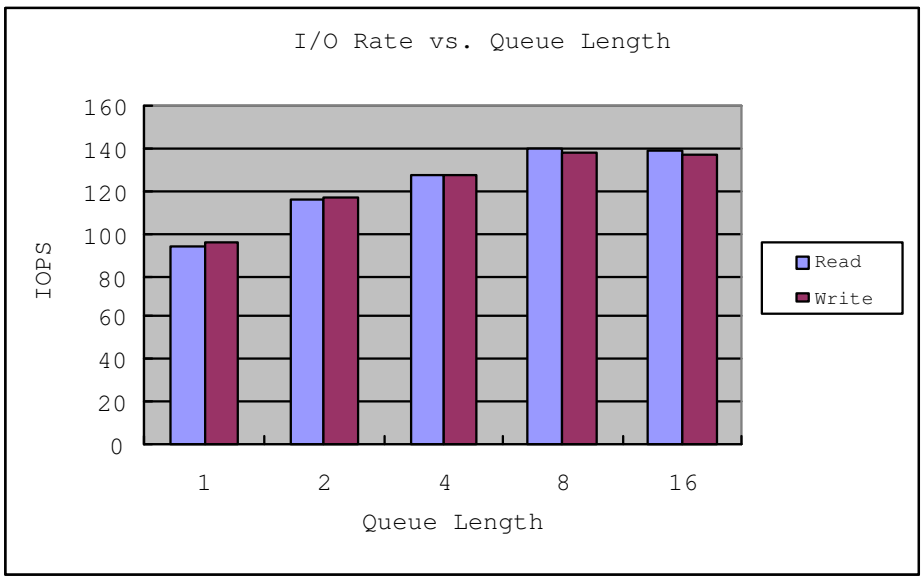

Figure 6.18: I/O Rate for Small I/O on Different Queue Length

The performance of the multiple connection iSCSI for small I/O request is tested. Since the queue depth is an important parameter that affects the iSCSI

performance for small I/O, the multiple connection iSCSI is designed to support different queue length. The impacts of the queue length on iSCSI performance is summarized in Figure 6.18. The test is conducted with the request size of 4K bytes and the MTU (Maximum Transmission Unit) size of 1,500 bytes.

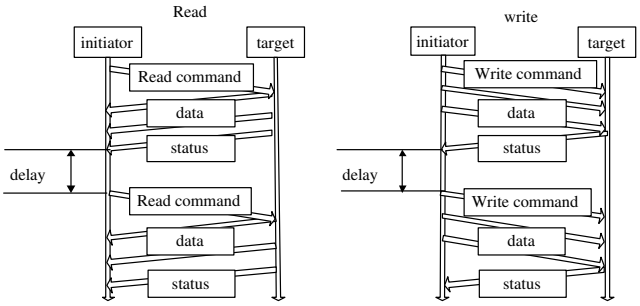

Figure 6.19: No Queue Model

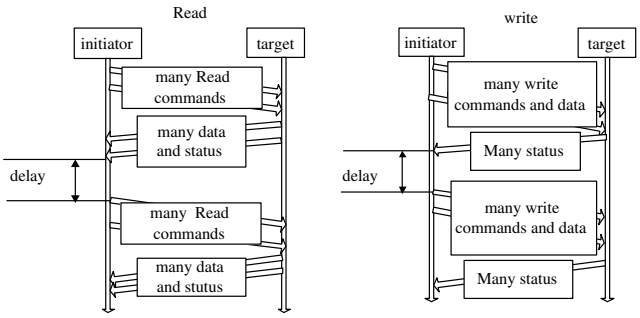

Figure 6.20: Queue Model

When the queue length equals to 1, the iSCSI performance is poor and the I/O rate is only about 94 IOPS. This is because all iSCSI commands, data and status are transmitted one by one. Figure 6.19 shows that the big delay appears between two iSCSI commands. When the queue length is more than 1, the multiple iSCSI commands can be issued continuously. The network bandwidth utilization increases. When the queue length is greater or equals to 8, the I/O rate can reach around 140 IOPS. The big delay appear only after a group of iSCSI commands finished, which is shown in Figure 6.20. The test results show that the I/O rate increases with the queue length until the queue length equals to 8 when I/O rate reach is peak value of around 140 IOPS. Further increasing the queue length can not increase the I/O rate anymore. This is because when queue length is 8, the

bandwidth has already been sufficiently used by iSCSI and it may also be restricted by interruption response frequency of wireless adapter.

Chapter 7

Conclusion and Future Works

This chapter presents the conclusion and highlights the contributions of this thesis based on the performance evaluation and result analysis conducted in previous chapters. Possible future work is briefly introduced here.

7.1 Conclusion

This thesis presents a new iSCSI design based on the concepts of multiple virtual TCP connections and parallel working mechanism over wireless LAN 802.11. The various experiments are conducted to identify some key issues of iSCSI performance over wireless network.

The background for the specific wireless storage problem is presented in Chapter 2. The theoretical analysis, which covers the overhead from the lower MAC layer to storage block layer and file layer, is presented in Chapter 3 in order to achieve high performance of wireless storage.

Multiple virtual TCP connection iSCSI design and the implementation issues are the topics of Chapter 4 and Chapter 5. The general iSCSI storage model is discussed to identify the working mechanism of iSCSI. Then the multiple virtual TCP connection iSCSI design is illustrated with respect to the comparison of symmetric

and asymmetric approach. After that the working principle of new iSCSI architecture and queuing models are discussed. Some implementation issues such as login phase, information exchange, SCSI command implementation and semaphore implementation are explained in Chapter 5.

The multiple connection iSCSI design and normal iSCSI are tested over wireless LAN 802.11 by using two popular benchmark tools, dd command and IOMeter. The experiment results are compared, analyzed and evaluated in Chapter 6. In order to identify the key issues of the iSCSI performance, the experiments are also conducted to test the iSCSI performance with different network parameters, different network latency and different queue length.

The main contributions of this thesis are as follows:

1. The characteristics of wireless LAN 802.11 are analyzed with respect to the TCP layer net throughput, packet failure pattern and the multi-hop channel impact on TCP performance. The storage level protocol such as block level iSCSI and file level NFS are also discussed in detail. The key performance problem of wireless storage is that the unreliability and packet retransmission of wireless network serious affects the storage performance. Especially for small I/O requests, the performance is even worse due to the more frequently requested ACK storage communication mechanism over unreliable wireless network.

2. An iSCSI design is proposed with the concepts of multiple virtual TCP connections in an iSCSI session and parallel working mechanism in iSCSI layer over wireless LAN 802.11. The new iSCSI design not only improves the iSCSI performance by increasing the utilization of limited wireless network bandwidth, but also provides a better mechanism to handle the packet failure in wireless channel and the long latency issues in multi-hop wireless environment.

3. The iSCSI prototype based on the multiple connection design has been devel-

oped by Linux kernel level programming on commercial PC. The prototype is different from single connection implementation but is compatible with iSCSI standard [36].

4. Various experiments are conducted to test the performance of the self-developed iSCSI prototype and normal single connection iSCSI. The test results show that multiple virtual connection iSCSI design for wireless storage can achieve significant throughput improvement for small I/O request and some throughput improvement which is close to the theoretical analysis result for big I/O request. The iSCSI can achieve high performance even in multi-hop, unreliable and long latency wireless network.

The main results are as follows:

1. TCP layer throughput in Windows and Linux is 4.903 Mbps (0.61 MB/s) and 5.06 Mbps (0.63 MB/s) respectively, which accords with the theoretical analysis result and verifies the correctness of the theoretical calculation (4.9 Mbps to 5.2 Mbps or 0.61 MB/s to 0.65 MB/s) in Section 3.2.

2. For small I/O request (2K $\sim$ 8K), the multiple connection iSCSI can achieve significant throughput improvements compared to normal single connection iSCSI. For 2K I/O request, for example, the throughput is improved from 0.25 MB/s to 0.53 MB/s, which is about 112% improvement.

3. For big I/O request (128K), the maximum throughput of iSCSI can reach 0.62 MB/s, which is very close to the experiment results of TCP layer net throughput (0.63 MB/s for Linux) in Section 6.1 and in the range of theoretical analysis of TCP layer net throughput (4.9 Mbps to 5.2 Mbps or 0.61 MB/s to 0.65 MB/s) in Section 3.2. This result verifies that the system overhead within iSCSI is very small.

4. For small I/O request (2K $\sim$ 8K), the response times are only half of the response time of normal single connection iSCSI.

5. For big I/O request (128K), the response times are also 6% lower than normal single connection iSCSI.

6. In initiator side, when disk I/O mode is used, for read operation, CPU utilization increases only from 41% to 53% although the storage performance has been significant improved.

7. When network latency is doubled, the throughput decrease is only about 2% for 2K small I/O request and 8% for 128K big I/O request. When network latency is tripled, the throughput decrease is about 5% for 2K small I/O request and 11% for 128K big I/O request.

8. The throughput increases with MTU size increase. When MTU is equal to 2,260 bytes, for big request size (128K), the throughput can reach 0.72 MB/s. When default MTU size 1,500 bytes is used, the throughput is 0.62 MB/s for big request size (128K).

9. The multiple connection iSCSI supports different queue length. When queue length is equal to 8, the IOPS can reach its peak value of 140.

10. For small I/O request, 4K for example, normal single connection iSCSI throughput is almost 50 % higher than NFS and for 128K big I/O request, iSCSI throughput is still about 4% higher than NFS.

7.2 Future Works

In this thesis, the main concern is to achieve high storage performance and network utilization. However, in wireless environment, network security is also an important issue needed to be solved.

When storage devices are directly attached to host machines, the data on the storage devices can be considered secure by its being inaccessible to the outside world. With iSCSI attached storage devices, this is no longer the case. A security

problem may arise if sensitive storage data is accessed over a general data network, especially in wireless network. Thus in order to perfect the design of multiple connection iSCSI, wireless security and authentication issues of iSCSI deserve to be studied and considered.

Bibliography

[1] Technical Report of University of California, Berkeley, "How Much Information? 2003," *http://www.sims.berkeley.edu/research/projects/how-much-info-2003/printable_report.pdf*, 2003.

[2] J. Lu, X. L. Lu, H. Han and Q. S. Wei, "A cooperative asynchronous write mechanism for NAS," *ACM SIGOPS Operating Systems Review*, Volume: 36, Issue: 3, 2002.

[3] I. Ari, M. Gottwals and D. Henze, "SANboost: automated SAN-level caching in storage area networks," *IEEE Proceedings of the International Conference on Autonomic Computing*, Pages: 164 - 171, 2004.

[4] B. Gordon, S. Oral, G. Li, H. Su and A. George, "Performance analysis of HP AlphaServer ES80 vs. SAN-based clusters," *IEEE Proceedings of the 2003 IEEE International conference on Performance, Computing and Communications*, Pages: 69 - 76, 2003.

[5] J. C. Namgoong and C. I. Park, "Design and implementation of a fibre channel network driver for SAN-attached RAID controllers," *IEEE Eighth International Conference on Parallel and Distributed Systems*, Pages: 477 - 483, 2001.

[6] C. Y. Wang, F. Zhou, Y. L. Zhu, T. C. Chong, B. Hou and W. Y. Xi, "Simulation of fibre channel storage area network using SANSim," *The 11th IEEE International Conference on Networks*, Pages: 349 - 354, 2003.

[7] M. Mesnier, G. R. Ganger and E. Riedel, "Object-based storage," *IEEE Communications Magazine*, Volume: 41, Issue: 8, 2003.

[8] T. S. Rappaport, "Wireless personal communications: trends and challenges," *IEEE Antennas and Propagation Magazine*, Volume: 33, Issue: 5, 1991

[9] T. Phan, L. Huang and C. Dulan, "Challenge: integrating mobile wireless devices into the computational grid," *ACM Proceedings of the 8th annual international conference on Mobile computing and networking*, Pages: 271 - 278, 2002.

[10] J. P. G. Sterbenz, R. Krishnan, R. R. Hain, A. W. Jackson, D. Levin, R. Ramanathan and J. Zao, "Survivable mobile wireless networks: issues, challenges and research directions," *Proceedings of the ACM workshop on Wireless security*, Pages: 31 - 40, 2002.

[11] S. Shakkottai and T. S. Rappaport, "Research challenges in wireless networks: a technical overview," *IEEE The 5th International Symposium on Wireless Personal Multimedia Communications*, Volume: 1, Pages: 12 - 18, 2002.

[12] M. Satyanarayanan, "Fundamental challenges in mobile computing," *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*, Pages: 1 - 7, 1996.

[13] L. L. Burge, S. Baajun and M. Garuba, "A ubiquitous stable storage for mobile computing devices," *Proceedings of the 2001 ACM symposium on Applied computing*, Pages: 401 - 404, 2001.

[14] I. Georgiev and I. I. Georgiev, "Networks: An information-interconnectivity-based retrieval method for network attached storage," *ACM Proceedings of the first conference on computing frontiers on Computing frontiers*, Pages: 268 - 275, 2004.

[15] H. Lei and D. Duchamp, "An Analytical Approach to File Prefetching," *Proceedings of the USENIX Annual Technical Conference*, Pages: 275C288, 1997.

[16] A. Amer and D. D. E. Long, "Aggregating caches: A mechanism for implicit file prefetching," *IEEE Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Pages: 293 - 301, 2001.

[17] T. M. Kroeger and D. D. E. Long, "Predicting File System Actions from Prior Events," *Proceedings of the USENIX Annual Technical Conference*, Pages: 319-328 , 1996.

[18] J. J. Kistler, "Disconnected operation In a distributed file system," *Ph. D. dissertation*, Carnegie-Mellon University, 1993.

[19] Y. Saygin, O. Ulusoy and A. K. Elmagarmid, "Association rules for supporting hoarding in mobile computing environments," *IEEE Tenth International Workshop on Research Issues in Data Engineering*, Pages: 71 - 78, 2000.

[20] A. Helal, A. Khushraj and J. Zhang, "Incremental hoarding and reintegration in mobile environments," *IEEE 2002 Symposium on Applications and the Internet*, Pages: 8 - 11, 2002.

[21] G. H. Kuenning, W. Ma, P. Reiher and G. J. Popek, "Mobility, Modeling and Management: Simplifying automated hoarding methods," *Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, Pages: 15 - 21, 2002.

[22] A. M. Keller and J. Basu, "A Predicated-based Caching Scheme for Client-Server Database Architectures," *VLDB Journal*, Volume: 5, Issue: 1, Pages: 35-47, 1996.

[23] D. Conan, S. Chabridon and G. Bernard, "Disconnected operations in mobile environments," *IEEE Proceedings of the International Parallel and Distributed Processing Symposium*, Pages: 192 - 199, 2002.

[24] H. Inamura, "Extending the Coda File System to Handle Cache Misses on Isolated Clients," *IEEE Symposium on Reliable Distributed Systems*, Pages: 336 - 340, 1998.

[25] K. Yasuda, "Cache cooperation for clustered disconnected computers," *IEEE Ninth International Conference on Parallel and Distributed Systems*, Pages: 457 - 464, 2002.

[26] J. C. S. Lui, O. K. Y. So and T. S. Tam, "NFS/M: an open platform mobile file system," *IEEE 18th International Conference on Distributed Computing Systems*, Pages: 488 - 495, 1998.

[27] I. R. Chen and N. A. Phan, "Update propagation algorithms for supporting disconnected operations in mobile wireless systems with data broadcasting," *IEEE 23rd International Conference on Distributed Computing Systems*, Pages: 784 - 789, 2003.

[28] Y. S. Lu and X. K. Shao, "Improve performance of disconnected operation through submitting by probability and transferring transactions in groups," *IEEE International Conference on Computer Networks and Mobile Computing*, Pages: 502 - 505, 2003.

[29] A. Kahol and S. Khurana, "A strategy to manage cache consistency in a disconnected distributed environment," *IEEE Transactions on Parallel and Distributed Systems*, Volume:12, Issue:7, Pages: 686 - 700, 2001.

[30] Y. W. Lee, "Operation-based Update Propagation in a Mobile File System," *Ph. D. dissertation*, The Chinese University of Hong Kong, 2000.

[31] A. P. Sistla, O. Wolfson and Y. X. Huang, "Minimization of communication cost through caching in mobile environments," *IEEE Transactions on Parallel and Distributed Systems*, Volume: 9, Issue: 4, Pages: 378 - 390, 1998.

[32] D. M. Huizinga and H. Sherman, "File hoarding under NFS and Linux," *Proceedings of the 1998 ACM symposium on Applied Computing*, Pages: 409 - 415, 1998.

[33] U. Kubach and K. Rothermel, "Exploiting location information for infostation-based hoarding," *ACM Proceedings of the 7th annual international conference on Mobile computing and networking*, Pages: 15 - 27 , 2001.

[34] U. Kubach and K. Rothermel, "An adaptive, location-aware hoarding mechanism," *IEEE Fifth IEEE Symposium on Computers and Communications*, Pages: 615 - 620, 2000.

[35] S. Burklen, P. J. Marron and K. Rothermel, "An enhanced hoarding approach based on graph analysis," *IEEE International Conference on Mobile Data Management*, Pages: 358 - 369, 2004.

[36] J. Satran et al., "iSCSI (Internet SCSI) Specification, Internet Draft," *http://www.ietf.org/internet-drafts/draft-ietf-ips-iscsi-20.txt*, 2003.

[37] M. Rajagopal et al., "Fibre Channel over TCP/IP (FCIP)," *IETF draft-ietf-ips-fcovertcpip-12.txt*, 2002.

[38] C. Monia et al., "iFCP A Protocol for Internet Fibre Channel Storage Networking," *IETF draft- ietf-ips-ifcp-13.txt*, 2002.

[39] R. V. Meter, G. G. Finn and S. Hotz, "VISA: Netstation's virtual Internet SCSI adapter," *ACM Proceedings of the eighth international conference on Architectural support for programming languages and operating systems*, Volume: 32,33, Issue: 5,11, Pages: 71 - 80, 1998.

[40] P. Sarkar and K. Voruganti, "IP Storage: The Challenge Ahead," *Proceedings of tenth conference on Mass Storage Systems and Technologies*, 2002.

[41] K. Z. Meth, "iSCSI Initiator Design and Implementation Experience," *Proceedings of tenth conference on Mass Storage Systems and Technologies*, 2002.

[42] W. T. Ng, B. Hillyer et al., "Obtaining High Performance for Storage Outsourcing," *Proceedings of the Conference on File and Storage Technologies*, Pages: 145C158, 2002.

[43] Y. P. Lu and D. H. C. Du, "Performance study of iSCSI-based storage subsystems," *IEEE Communications Magazine*, Volume: 41, Issue: 8, Pages: 76 - 82, 2003.

[44] S. Aiken, D. Grunwald, A. R. Pleszkun and J. Willeke, "A performance analysis of the iSCSI protocol," *Proceedings. 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies*, Pages: 123 - 134, 2003.

[45] X. B. He, Q. Yang and M. Zhang, "A caching strategy to improve iSCSI performance," *Proceedings of the 27th Annual IEEE conference on Local Computer Networks*, Pages: 278 - 285, 2002.

[46] Alacritech, White paper, "Delivering high performance storage networking," *http://www.alacritech.com/html/storagewhitepaper.html*.

[47] Adaptec, White paper, "Building SANs with iSCSI, ethernet and adaptec," *http://www.graphics.adaptec.com/pdfs/buildingsanwithiscsi-21.pdf*.

[48] K. Z. Meth and J. Satran, "Features of the iSCSI protocol," *IEEE Communications Magazine*, Volume: 41, Issue: 8, Pages: 72 - 75, 2003.

[49] B. H. Kim and B. G. Lee, "Parallel transmission of DS/CDMA signal by orthogonal bases and repeated spreading: the chip-spreading OCDM," *IEEE Global Telecommunications Conference*, Volume: 3, Pages: 1815 - 1819, 1996.

[50] Z. H. Jiang and M. C. Zhou, "Prioritized parallel transmission MAC protocol and its performance in a simplified all-IP wireless WAN," *IEEE International Conference on Networking, Sensing and Control*, Volume: 2, Pages: 1224 - 1228, 2004.

[51] S. Takahashi, K. Takahashi, H. Masui, K. Kage and T. Kobayashi, "Performance of parallel transmission applied to broadband mobile communication systems in urban multipath environments," *IEEE VTS 50th Vehicular Technology Conference*, Volume: 2, Pages: 1028 - 1032, 1999.

[52] S. Servetto, K. Ramchandran, K. Nahrstedt and A. Ortega, "Optimal segmentation of a VBR source for its parallel transmission over multiple ATM connections," *IEEE International Conference on Image Processing*, Volume: 2, Pages: 5 - 8, 1997.

[53] B. Meng, B. T. Patrick and T. C. Chong, "Design and implementation of multiple addresses parallel transmission architecture for storage area network," *20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies*, Pages: 67 - 71, 2003.

[54] Q. M. Malluhi and W. E. Johnston, "Coding for high availability of a distributed-parallel storage system," *IEEE Transactions on Parallel and Distributed Systems*, Volume: 9, Issue: 12, Pages: 1237 - 1252, 1998.

[55] T. Kitamura, Y. Oue, K. Ohnishi and M. Shimizu, "Dynamic access load balancing on the parallel secondary storage," *IEEE Second Aizu International Symposium on Parallel Algorithms/Architecture Synthesis*, Pages: 316 - 323, 1997.

[56] T. Sterling, D. J. Becker, M. R. Berry, D. Savarese and C. Reschke, "Achieving a balanced low-cost architecture for mass storage management through multiple fast Ethernet channels on the Beowulf parallel workstation," *IEEE 10th International Parallel Processing Symposium*, Pages: 104 - 108, 1996.

[57] "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification," *IEEE 802.11 WG*, 1999.

[58] "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification: High-Speed Physical Layer Extension in the 2.4 GHz Band," *IEEE 802.11b WG*, 1999.

[59] T. K. Tan, "Call for interests on 802.11a higher rates extension," *IEEE 802.11 Interim Meeting*, 2002.

[60] S. Hori, Y. Inoue, T. Sakata and M. Morikura, "System capacity and cell radius comparison with several high data rate WLANs," *IEEE 802.11 Plenary Meeting*, 2002.

[61] S. Coffey, "Suggested criteria for high throughput extensions to IEEE 802.11 systems," *IEEE 802.11 Plenary Meeting*, 2002.

[62] V. K. Jones, R. DeVegt and J. Terry, "Interest for HDR extension to 802.11a," *IEEE 802.11 Interim Meeting*, 2002.

[63] M. Tzannes, T. Cooklev and D. Lee, "Extended data rate 802.11a," *IEEE 802.11 Plenary Meeting*, 2002.

[64] A. Kamerman and G. Aben, "Net throughput with IEEE 802.11 wireless LANs," *IEEE Wireless Communications and Networking Conference*, Volume: 2, Pages: 747 - 752, 2000.

[65] K. C. Huang and K. C. Chen, "Interference analysis of nonpersistent CSMA with hidden terminals in multicell wireless data networks," *Sixth IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Volume: 2, Pages: 907 - 911, 1995.

[66] H. S. Chhaya and S. Gupta, "Performance modeling of asynchronous data transfer methods of IEEE 802.11 MAC protocol," *Wireless Networks*, Volume 3, Issue 3, Pages: 217 - 234, 1997.

[67] T. S. Rappaport, "Wireless Communications: Principles and Practice," Prentice Hall, 2002.

[68] Y. C. Tay and K. C. Chua, "A capacity analysis for the IEEE 802.11 MAC protocol," *Wireless Networks*, Volume 7, Issue 2, Pages: 159 - 171, 2001.

[69] J. Yeo, A. Agrawala, "Capacity and Variability Analysis of the IEEE 802.11 MAC Protocol," *CS-TR-4475 and UMIACS-TR-2003-45*, University of Maryland in College Park, 2003.

[70] A. Acharya, A. Misra and S. Bansal, "Mobile Ad Hoc Networks: A label-switching packet forwarding architecture for multi-hop wireless LANs," *Proceedings of the 5th ACM international workshop on Wireless mobile multimedia*, Pages: 33 - 40, 2002.

[71] X. G. Guo, S. Roy, W. S. Conner, "Spatial reuse in wireless ad-hoc networks," *IEEE 58th Vehicular Technology Conference*, Volume: 3, Pages: 1437 - 1442, 2003.

[72] F. J. Ye, S. Yi, B. Sikdar, "Improving Spatial Reuse of IEEE 802.11 Based Ad Hoc Networks," *IEEE Global Telecommunications Conference*, Volume: 2, Pages: 1013 - 1017, 2003.

[73] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and Implementation of the Sun Network File System," *In Proceedings of the Summer 1985 USENIX Conference*, Pages: 119 - 130, 1985.

[74] G. A. Gibson and R. Van Meter, "Network Attached Storage Architecture," *Communication of the ACM*, Volume: 43, Issue: 11, Pages: 37 - 45, 2000.

[75] D. Bertsekas and R. Gallager, "Data Networks," *Prentice Hall*, 1992.