Introducing A Flexible Data Transport Protocol for Network Storage Applications

Patrick Beng T. KHOO and Wilson Yong H. WANG
MCSA Group, NST Division – Data Storage Institute
Affiliated to the National University of Singapore
Funded by the Agency for Science, Technology And Research Singapore
DSI Building, 5 Engineering Drive 1, 117608 Singapore
Tel: +65 8748413 Fax: +65 7772053
Email: patrick@dsi.nus.edu.sg
Web: http://nst.dsi.nus.edu.sg/mcsa/

Abstract

The purpose of this paper is to demonstrate that alternative solutions to current methods exist for network storage. We would like to introduce one such alternative, a new protocol that we call HyperSCSI. This protocol is used for the transmission of Small Computer Systems Interface (SCSI) family of protocols across a network and multi-technology device support. In this paper, we will outline some of the key features and basic technical details of HyperSCSI. We have also developed several fully functioning disk array prototypes using a variety of hardware and storage devices as well as conducted benchmarks and performance tests on this. A performance comparison between this new protocol and iSCSI and NFS is also included here.

1. The Problem

Research has been ongoing for ways to transport data over networks for storage applications for quite some years. While we pursued efforts in developing network storage technologies, we came across the following issues and concerns.

- High cost of Fibre Channel SANs – Implementing and managing FC-based SANs is quite expensive. Even if hardware costs were to come down (and we expect them to do so), ultimately the “hidden” costs of systems, infrastructure, manpower and software implementation and maintenance is still very high.
- TCP/IP SAN performance is still not good enough without hardware acceleration – TCP/IP is inherently slow compared to FC-based storage technologies. Special hardware for TCP/IP represents higher costs and a more difficult upgrade path for users.
- FC-based SANs cannot do Storage Wide-Area Networks – FC is an inherently local communications technology, and if one needs to go wide-area, the best method is to use the ever present IP due to its wide-spread availability. Ongoing efforts such as FCIP, iFCP and iSCSI are in line with this idea.
- Interoperability is weak at times – Vendors are also often stuck on the interoperability of various storage products and systems. The fundamental issue is that vendors need to differentiate their solutions in order to compete. However, this often results in interoperability issues or worse, vendor lock-in for customers.

- Security is lacking – Normal TCP/IP and Fibre Channel do not provide serious security for data transport. FC does have LUN Masking, but this is mostly a function of the FC switch, not the storage device itself. IPsec does provide security to IP-based applications, but adds yet another layer of complexity to an already difficult solution.
- Inability of existing storage technologies to apply to new areas – Existing network storage methods do not take non-traditional applications and areas into account. An example of this is in home and personal network storage and using simple infra-red, Bluetooth or wireless LAN for small data access or transport.
- Difficulties in scaling – Existing systems scale upwards through new higher bandwidth standards. This is often slow due to the standards process. Furthermore, the scaling of capacity is difficult due to the continuous need to build and implement larger and larger disk systems that are generally not modular enough.

Based on this background, we set about designing, developing and testing a new network storage protocol that we hope will address these and other network storage issues. We would like to present some of the results from our research and development efforts that began in June 2000 in this paper.

2. Existing Solutions

Recent efforts in network storage have expanded to include development of alternatives to pure Fibre Channel as the primary method for network storage. These efforts include iSCSI, FCIP, SST and many others. Below are descriptions of a few of these efforts.

2.1. Fibre Channel over TCP/IP (FCIP)

Fibre Channel Over TCP/IP (FCIP) describes mechanisms that allow the interconnection of islands of Fibre Channel storage area networks over IP-based networks to form a unified storage area network in a single Fibre Channel fabric. FCIP relies on IP-based network services to provide the connectivity between the storage area network islands over local area networks, metropolitan area networks, or wide area networks [1]. What this means is that FCIP is designed to encapsulate Fibre Channel over a TCP/IP-based network for the purposes of connecting dispersed FC-based SANs.

2.2. iSCSI

The iSCSI Internet Draft describes a transport protocol for SCSI that operates on top of TCP [2]. iSCSI enables the use of SCSI devices over a TCP/IP-based infrastructure. Other areas considered include Naming and Discovery, Boot and Security. It is important to note that iSCSI is the only protocol currently in the process of standardisation that allows for the construction of native end-to-end Ethernet SANs [3].

2.3. Internet Fibre Channel (iFCP)

iFCP specifies an architecture and gateway-to-gateway protocol for the implementation of Fibre Channel fabric functionality on a network in which TCP/IP switching and routing elements replace Fibre Channel components. The protocol enables the attachment of existing Fibre Channel storage products to an IP network by supporting the fabric services required by such devices [4]. The purpose here seems quite clear, that is to implement Fibre Channel fabric architectures over a TCP/IP-based network, thus allowing FC devices to connect and run FC natively over a TCP/IP-based infrastructure.

2.4. Metro FCP (mFCP)

mFCP is a UDP-based implementation of the iFCP over metro- and local-scale IP networks. These networks are provisioned to have latency, reliability, and performance levels comparable to that of a Fibre Channel network. Storage devices use the Fibre Channel SCSI mapping in FCP for data transport and error recovery. mFCP leverages these existing mechanisms to facilitate high-performance interconnection of Fibre Channel- based storage devices over suitably provisioned IP networks. As in the case of iFCP, Fibre Channel frames may be transported natively over such a network without Fibre Channel switching and routing elements [5].

2.5. Internet Storage Name Service (iSNS)

iSNS provides a generic framework for the discovery and management of iSCSI and Fibre Channel (FCP) storage devices in an enterprise-scale IP storage network. iSNS is an application that stores iSCSI and FC device attributes and monitors their availability and reachability in an integrated IP storage network. Due to its role as a consolidated information repository, iSNS provides for more efficient and scalable management of storage devices in an IP network [6]. iSNS is meant to be used with iSCSI, FCIP, iFCP and such protocols for the hosts or servers to locate and use storage devices over a large network infrastructure such as the Internet.

2.6. SCSI on Scheduled Transfer Protocol (SST)

The SCSI on STP standard defines a transport protocol within the SCSI family of standards. The physical interconnects to which the SST protocol may attach are not defined within this standard, but rather, are any interconnects or other protocols on which the basic ST protocol may operate [7]. SST defines a mapping to carry SCSI traffic on top of an STP-based infrastructure.

2.7. Basic Technologies

The above technologies are built on top of a basic set of storage technologies. There are two such basic command sets today, ATA/IDE and SCSI. Based on these two command sets, other derivative technologies have been developed. See Table 1 for a pictorial representation of these technologies.

Base Command Set	**ATA/IDE**	**SCSI**
Derivative / New Developments	ATA 133 Serial ATA	SCSI-320 Universal Serial Bus IEEE 1394 “FireWire” Fibre Channel SSA
Network Storage Developments		iSCSI iFCP FCIP SST

Table 1: Storage Technologies

At this point, we turn our attention to our development efforts of the HyperSCSI protocol. Further in this paper, we will present a few ideas for thought regarding HyperSCSI and various other technologies.

3. The Approach

The first thing we decided on was to standardise on using the Small Computer Systems Interface (SCSI). It is the predominant mechanism for various storage and even non-storage devices. The question then turned quickly to how we could make SCSI “network-enabled”. This gave rise to our idea of “HyperSCSI”.

We found that the requirements of local network storage (SAN) and wide-area network storage (SWAN) are quite different. As such, we provided the capability to spilt HyperSCSI protocol into multiple modes of operation. Two such modes are currently being developed, one for local access, Local HyperSCSI over Ethernet (HS/eth), and the other for wide-area connectivity, Wide-Area HyperSCSI over IP (HS/IP). The basic protocol structure is essentially the same, thus allowing devices to speak local or wide-area storage seamlessly. This has allowed us to adopt IP-based networking technologies for wide-area applications where it is needed but bypassing IP entirely and putting the protocol directly onto Ethernet itself for optimum local area communications. This model also allows us to eventually develop HyperSCSI for other technologies, such as Asynchronous Transfer Mode (ATM) for high speed Telco / ISP environments and Wireless LAN for home or personal network storage.

Furthermore, since we are designing a low-level protocol, some of the intelligence or command and control functions can be passed on to higher layers or the clients to adapt and handle. This allows us to design a protocol that is lightweight and efficient, while leveraging the intelligence and capabilities of both the storage system and host machine to mutual benefit. For example, we allow device, security and compression options as well as storage virtualisation technologies to be implemented on either the storage system, host machine or both as the needs arise. In addition, packetisation and virtualisation options of HyperSCSI allow us to implement N-channel communications technologies in order to use “scale-out” methods of bandwidth and capacity increases with fault tolerance and reliability. Figure 1 shows a Local HyperSCSI packet on Ethernet (HS/eth). A wide-area HyperSCSI (HS/IP) packet is essentially the same, but built on IP instead of directly on the Ethernet.

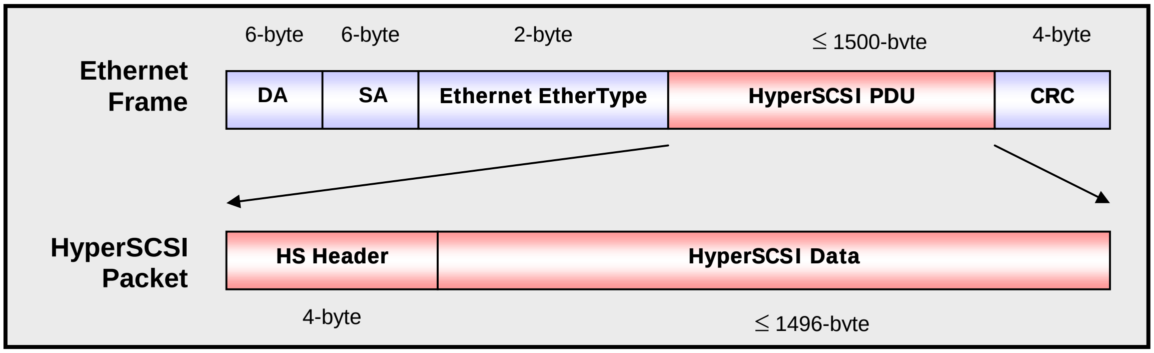


Figure 1: HyperSCSI Packet

Finally, more advanced functions and capabilities were built into the HyperSCSI protocol to support other requirements like dynamic management, dynamic flow control and in-band management capabilities. Manufacturers, system integrators and technology companies are not left out in the cold either. To enable the protocol to be interoperable, and yet be able to support vendor-specific or implementation-specific functions, a special set of dynamically negotiated device options has been designed into the protocol. These options can be negotiated at connect time and depending on the configuration of the clients and servers, be enforced, supported or ignored. Thus, HyperSCSI can provide a minimum level of connectivity for interoperability operations and while supporting advanced vendor-specific or implementation-specific functions. Our initial encryption methods demonstrate this function in action. Other possible device specific options include read-only access, removable media locking and data compression.

4. HyperSCSI Operation

The HyperSCSI protocol comprises of various packet structures. These structures are categorised by classes and then by specific types. Packets of a specific class and type may also have more than one function depending on the context of the communication. These packets are responsible for transmitting the SCSI data and commands as well as managing the connection and communication channel. Table 2 illustrates some of the packets in the HyperSCSI protocol.

HyperSCSI Packet	Description
HyperSCSI Command Block Encapsulation Class	
HCBE_REQUEST	HyperSCSI command block encapsulation request
HCBE_REPLY	HyperSCSI command block encapsulation reply
HyperSCSI Connection Control Class	
HCC_DEVICE_DISCOVERY	Client issues this packet to discover storage devices on the network
HCC_ADN_REQUEST	Authentication challenge and device operation negotiation request
HCC_ADN_REPLY	Authentication and device operation negotiation reply
HCC_DISCONNECT	Termination of HyperSCSI connection
HyperSCSI Flow Control Class	
FC_ACK_SNR	Flow control set-up and acknowledgement request
FC_ACK_REPLY	Acknowledge reply

Table 2: HyperSCSI Operations

5. Typical HyperSCSI Connection Flow Sequence

Figure 2 illustrates a typical sequence of the communication stages between a client and server using the HyperSCSI protocol. The various stages of the connection flow sequence are described below.

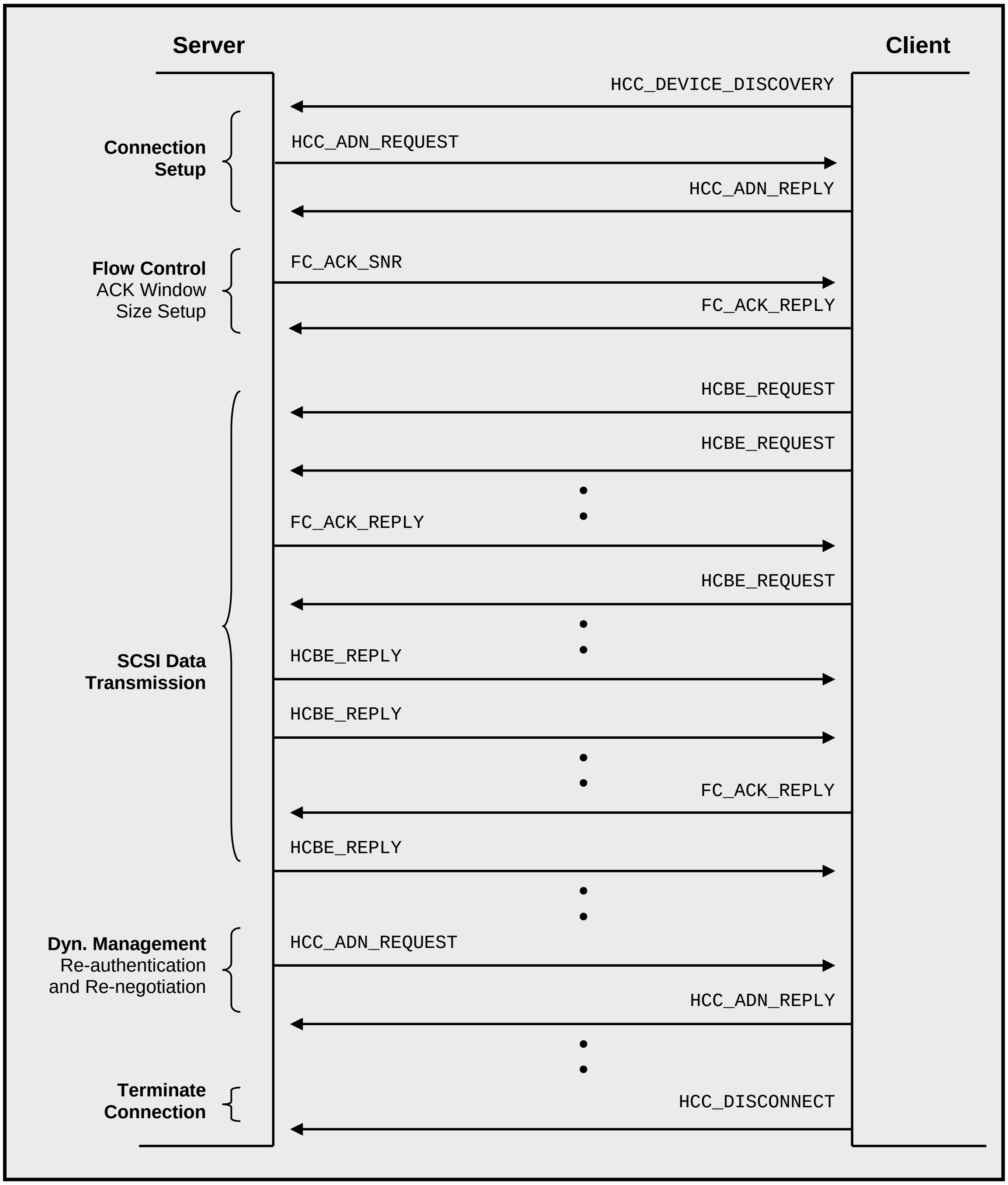


Figure 2: Typical HyperSCSI Connection Flow Sequence

5.1. Connection Setup

The HyperSCSI connection setup is a three-step handshaking procedure between a HyperSCSI client and server pair. Typically, in a storage network, the host machine (HyperSCSI client) is responsible for locating and initiating connections to storage devices (HyperSCSI servers). During this process, the HyperSCSI client issues a HCC_DEVICE_DISCOVERY via Ethernet broadcast or IP packet, to locate devices on the network. For IP-based situations, neither broadcast nor multicast methods are used. Instead, a client must specify an IP address (or DNS name) and a HCC_DEVICE_DISCOVERY packet is sent over IP directly to the server. Further information about device discovery is covered in section 6.2. Once the HyperSCSI server receives this packet, it checks the client address for authentication purposes and transmits the HCC_ADN_REQUEST packet back to the HyperSCSI client. In order for the HyperSCSI client to establish a connection with the HyperSCSI server, it must then send the correct response through a HCC_ADN_REPLY command and add the ID numbers of the devices that it has access to into its own registry. If the server successfully authenticates the HCC_ADN_REPLY, the connection is accepted and the HyperSCSI client can now send commands to the server. Within the HCC_ADN request and reply method, authentication challenges, encryption key exchanges, device specific option negotiations and other information supporting N-channel communications such as server/client IDs and network addresses are also provided and exchanged.

5.2. Flow Control and ACK Window Size Setup

An ACK mechanism has been adopted to support flow control of data between an HyperSCSI client and server pair. The ACK window size refers to the number of packets that the transmitter may continuously send before waiting for an acknowledgement. This window size must be negotiated and agreed upon before data flow can take place and is set by the requestor through an FC_ACK_SNR command. This packet is issued as a separate message and typically, the server will be the one to issue this command so that the server has the ability to balance loads or priorities across multiple clients, although this does not mean that the client may not issue one either. Once the FC_ACK_SNR has been received, the new status will be acknowledged to the requestor with an FC_ACK_REPLY. If the requestor receives the acknowledgement, it assumed that the window size is accepted and packet transmission using the new window size can begin. The ACK window size can be set based on traffic loads, or buffer capacities and can be set at start-up or changed dynamically during run time. This allows for different window sizes to be dynamically set by clients and servers to fit changing performance, reliability or QoS requirements. For example, under bad network environments, windows sizes can be reduced, while under optimum situations, window sizes can be increased for better performance. However, we are still studying algorithms for the detection of network congestion and updating of the window size during run time. The basic protocol supports this capability and we will include this portion when it is complete. Transmission windows used here are neither fixed nor sliding in nature, but rather utilises a moving window scheme similar to credit-based schemes used in Fibre Channel, but measured in windows rather than individual packets. In addition, the FC_ACK_REPLY is also used to acknowledge the correct reception of a window to the requestor and synchronises the data flow between an HyperSCSI client and server pair. In this case, it functions as an indicator of the receiver status for normal HyperSCSI data transmission. If the transmitter does not receive the correct FC_ACK_REPLY packet within a timeout period, it will re-transmit all the packets in the window in question again. Another retransmission scheme supported is by using the FC_ACK_SNR to query the receiver's status. The transmitter can then use the FC_ACK_REPLY results to re-calculate the next packet to be transmitted. With these two schemes, re-transmits can be conducted selectively or by ACK windows, thus giving users a high level of flexibility in controlling the flow of data and commands.

5.3. HyperSCSI Data Transmission

When there is a SCSI request from the local OS SCSI upper layer of the host machine, the HyperSCSI client software is responsible for converting the OS-specific SCSI command block together with any relevant data (as in a write command) into a platform independent HyperSCSI Command Block (HCB). The client then encapsulates and fragments the HCB into one or more HCBE_REQUEST packets that it sends to the HyperSCSI server. SCSI command blocks and user data will therefore be transmitted together in the same packet. The HyperSCSI server receives the data stream, re-assembles the HyperSCSI command block and relevant user data, converts it back to an OS-specific SCSI command block and passes it to the relevant hardware for execution. When the result of this SCSI request is ready, the HyperSCSI server will send the result together with the requested data back to HyperSCSI client by issuing the HCBE_REPLY packet stream in a similar manner as the request. The HyperSCSI client reassembles the HyperSCSI command block and converts it back to a OS-specific SCSI command block before passing it on to the local OS SCSI upper layer. During this transmission, flow control mechanisms are in effect through the use of FC_ACK_REPLY commands as described in section 5.2.

5.4. Dynamic Management

During a HyperSCSI connection, the HyperSCSI server will regularly (timer-based) issue a HCC_ADN_REQUEST command for three purposes, re-authentication of clients and key-exchange for security, re-negotiation of device options (if permitted), and as a form of "keep-alive". Through this method, servers not only poll the client's status, but also check its identity. Furthermore, if HyperSCSI encryption options are turned on for data transmission, the HCC_ADN_REQUEST and HCC_ADN_REPLY uses an authenticated exchange mechanism to update and change encryption keys. This scheme also allows a device's options to be modified dynamically. For example, a device which does not have encryption enabled may turn it on during this time so that the communication will be secured from this point onwards. To enable such remote management functions, an encrypted Management Command Stream is used to transfer management commands from a client to a server or vice-versa. This MCS also allows adding or removing clients, requesting the change of device options, changing access passwords and device access permissions. The MCS is implemented within a valid HyperSCSI connection, thus only authenticated HyperSCSI clients and servers can use this in-band management mechanism.

5.5. Connection Termination

The HyperSCSI client can close a connection by sending an HCC_DISCONNECT command to the HyperSCSI server. The server will then remove this client from its connection list and close the connection. Servers do not need to acknowledge disconnect requests from clients because SCSI connections are host-target based. Unlike TCP/IP connections, which are full-duplex and can be closed by both clients and servers, SCSI connections can only be terminated (gracefully) by clients. If a server were to terminate a connection, it implies that service has been lost (or a hard disk has crashed). Servers do not keep connection information forever, and will drop relevant connections if "keep-alives" (as outlined in section 5.4) to a particular client should fail for some reason. Through the use of hashing, encryption and security methods (see section 6.3), connections are protected from denial of service attacks from hackers arbitrarily using the HCC_DISCONNECT command.

6. Feature Comparison

There are many points to consider when making comparisons of HyperSCSI features to other technologies. In the area for security for example, HyperSCSI makes use of sequence numbers, hashing, SCSI command identifiers, digital keys and other mechanisms to secure a connection, similar in some areas to IP and SCSI. A point to note has been that where possible, we have tried to adapt good ideas and mechanisms from other technologies for use in HyperSCSI. A good reference is the six manipulation functions used in any data transport protocol [8]. Thus, while differences exist, similarities will definitely show up as well in any comparison with HyperSCSI. Presented in Figure 3 are some ideas for consideration.

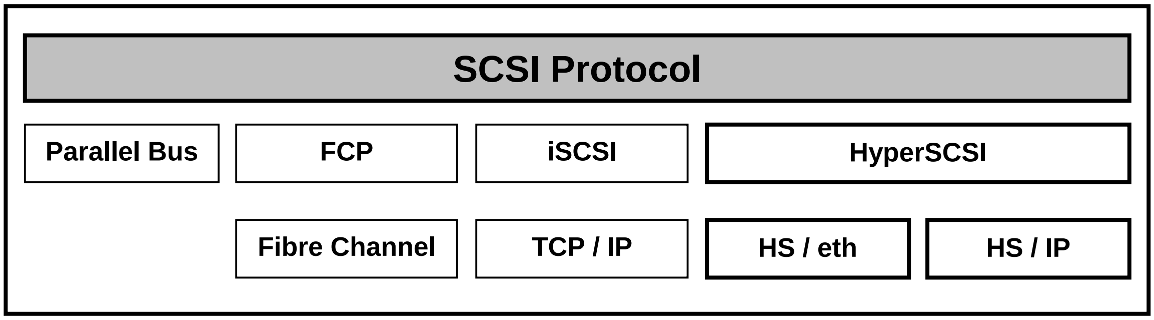


Figure 3: Protocol Stack Comparison

6.1. Storage Device Management

As it turns out, this is a key aspect of network storage that is often neglected. Proprietary enterprise management software or dedicated SAN management software from vendors or switch manufacturers is often required to properly manage the storage devices. Fibre Channel devices, switches and arrays often have an additional Ethernet port and IP address for access from the management software. HyperSCSI provides an in-band management mechanism that allows properly authenticated (and permitted) clients and servers to manage each other's settings and properties. Some device and management options can even be modified and updated dynamically during a connection.

6.2. Device Discovery Mechanisms

To identify and locate storage devices, Fibre Channel has World Wide Name (WWN) while iSCSI/FCIP/iFCP use iSNS. Such mechanisms are complex and add another hindrance to achieving ease of use and even plug-and-play networking. For this purpose, HS/eth uses standard a broadcast device discovery mechanism to dynamically locate targets on the network. If a server is configured to allow a particular client to attach, it will respond appropriately, else the discovery request is ignored. Thus the only configuration users have to be concerned about is granting permissions, rather than setting up complex name servers of some type. This is particularly useful in a plug-and-play wireless personal storage network environment. HS/IP on the other hand, leverage standard DNS mechanisms to “locate” a server across the network. We do not endorse the idea of “broadcast / multicast to find out who’s out there on the Internet” as a means to locating storage resources. Storage being a key and critical resource should be managed as securely as possible, especially if it is on a public or private IP-based network. If protocols can be routed, physical security of the storage network is less assured. As such, administrators should know before hand the IP or DNS address of the client and server, configure them accordingly and not have such information “discovered” for security reasons. This also means that there is no single point of failure like having iSNS servers or requiring expensive switches with additional intelligence built-in. HyperSCSI clients will then attempt to connect to the server address given to it, and no other. The only configuration that users need to worry about in the end is granting permissions.

6.3. Security

All three TCP/IP based encapsulation methods iSCSI, iFCP and FCIP provides for and requires the use of IPsec for securing the TCP/IP connection. Certainly, this is a step forward when considering that Fibre Channel’s main security mechanism is LUN masking which is implemented mostly on the switch. However, using IPsec implies securing the entire connection. This is different from the more flexible LUN masking method that FC uses to allow the user to secure individual LUNs as the case may be. HyperSCSI thus supports security options to be specified by individual devices (or LUNs) instead of at the connection level. Of course, iSCSI for example, only supports one LUN per connection, while HyperSCSI can have multiple devices in a single connection, as outlined in section 6.4. It should also be noted that like Fibre Channel, HS/eth (which does not use IP at all and is not routable) would require physical access to the network in order to hack it. HyperSCSI also allows for security to be modularised into different levels of requirements such as hashing, encryption or none at all, thereby giving even more options to secure (or not) the device and/or the connection.

6.4. Multiple Device Access

iSCSI uses one or more TCP connections to make up a single session and requires that across all connections within a session, an initiator sees only one “target image”. All target identifying elements, like LUNs, are the same [9]. While this makes sense in a pure SCSI environment, where a single host bus adapter would see a single target to have one “target image”, this may not be true in a network storage environment where usually disk arrays of one or more targets may be “exported” to the initiator. HyperSCSI on the other hand allows a single connection to have access to as many SCSI devices (or LUNs) as supported by both the initiator and target. This single connection can then be established as a virtual channel over multiple physical links to form a redundant trunk. Devices that may require multiple LUN access includes optical jukeboxes and tape libraries.

6.5. Optimising Performance

One of the most controversial aspects of performance for network storage are the overheads of TCP/IP. Industry analysts have noted that the TCP/IP stack is very CPU intensive and without complex optimisation techniques like hardware accelerators, interrupt coalescing, checksum offloading, and so on [10], the only practical application for iSCSI is to extend current Fibre Channel SAN-to-LAN connectivity into the realm of SAN-to-MAN/WAN connectivity [11]. If every implementation were to require TCP/IP implemented in hardware, it would be no different than requiring all devices to have Fibre Channel hardware built-in. HyperSCSI can bypass TCP/IP entirely to build a storage network similar to (and capable of replacing) Fibre Channel architectures, but using plain old Ethernet instead. For wide area implementations, HyperSCSI does in fact also support the use of IP-based infrastructure for building Storage Wide-Area Networks through HS/IP, a strategy which is no different from Fibre Channel. It should also be noted that while HS/eth reduces overheads partly by eliminating certain checksums (ie. header checksum), IPv6 also does away with the header checksum. IPv6 designers felt that the risk was acceptable given that data link and transport layers check for errors [12]. Another key point of HyperSCSI is its reliance on state tables so that information about a connection does not have to be retransmitted over and over again. Such information includes SCSI host/target information, device options and HyperSCSI sequence numbers. This is also similar in idea to STP’s architecture of setting up the receiving buffer and related information before transmitting data [13]. This is also a security benefit since the capture of a single packet is unlikely to reveal much information about the connection itself. For HS/IP, only one IP port is required, since each client can access multiple devices through a single connection, unlike iSCSI (see section 6.4).

6.6. Flow Control Issues

Fibre Channel is often touted as the best solution for network storage due to its high speed packetised but dedicated channel for storage. iSCSI on the other hand relies upon TCP/IP for flow control and packet transmission and can leverage TCP/IP’s sliding windows as a counter to the idea of packetisation being less efficient compared to dedicated channels. To provide the best of both worlds, HyperSCSI adopts a moving window mechanism but makes the window size dynamic. A balance is provided in that the window size does not fluctuate like TCP/IP’s sliding windows, but can and does change dynamically in the middle of a connection. Since this window size is dynamically controlled by clients and servers, algorithms for determining the window size can be adopted to find the optimal window size during run-time, thus adapting to network congestion. This is particularly evident in HS/eth implementations. HS/IP of course leverages standard IP-based methods for flow-control issues. In addition, retransmission can be implemented either using a selected retransmission scheme or a simpler window retransmit scheme. This can be decided based on the implementation environment, thus giving users a wide degree of flexibility and performance tuning options.

6.7. Simplicity, Interoperability and Diversity

HyperSCSI is designed from the ground up to be simpler for users to implement and yet capable of achieving interoperability without sacrificing diversity. For this purpose, negotiable device options allow for vendor-specific or implementation-specific features to be supported. If different vendor devices with different supported device options were to try to connect to each other, the worst case is expected to be a basic connection with no additional features or functions. When used in conjunction with the varied SCSI-3 command set and the Management Command Stream, this becomes quite a powerful value-added option for vendors and users alike.

7. Development Progress

We have implemented and tested HyperSCSI under various conditions over Fast Ethernet and Gigabit Ethernet. The results so far have proven to be most encouraging. Today, HyperSCSI on Gigabit Ethernet achieves a quick 96% of the local physical disk performance compared to iSCSI's 82% for block level access. The results are even better when considering file system level tests. Using a straightforward file copy test, HyperSCSI can reach 88% of the local physical disk performance, iSCSI managed 43% while NFS only succeeded to match 39%. Not only that, it can be seen that HyperSCSI provides a more reliable and predictable performance level similar to that of the local physical disk than iSCSI or NFS and is less dependent on caching to achieve performance. One might wonder why iSCSI performance is not as good as expected. Seeing how iSCSI performance seems to closely track NFS performance, we hypothesise that the TCP/IP overhead is the differentiating factor between iSCSI and HyperSCSI performance. The following charts highlight some of the performance measurements that we have conducted.

The results illustrated in Figure 4 represent results from five different tests, two of which were raw block level reads (hdparm and dd) and the other three represent data access above the file system level. These tests were done on the same hardware and the same OS for all three technologies and both the client and server. We used two AMD Athlon 1.2GHz SMP machines with 32-bit 33MHz PCI busses, 266MHz 256MB DDR RAM running RedHat Linux 7.1 using the standard Linux kernel version 2.4.16, one of which was the client and the other was the server. Both machines had 3Com 3C985B-SX Gigabit Ethernet NICs, connected over a cross connect fibre-optic cable with jumbo frames, and the server used an Adaptec 39160 U160 SCSI controller. The server exported 8 IBM UltraStar U160 9.1GB 10k RPM drives configured in RAID 0. For the tests using a file system, Linux Ext2 was used as the file system. We used NFS version 2 over UDP from the RedHat Linux RPM version 0.3.1-5. The iSCSI version we used was version 6 from Intel, while the HyperSCSI version was 110-011226. The destination for the cp test was /dev/null while the Iozone version used was 3.71. We would like to draw attention, not to the absolute numbers of MB/s, but rather to the performance comparisons between iSCSI, NFS and HyperSCSI.

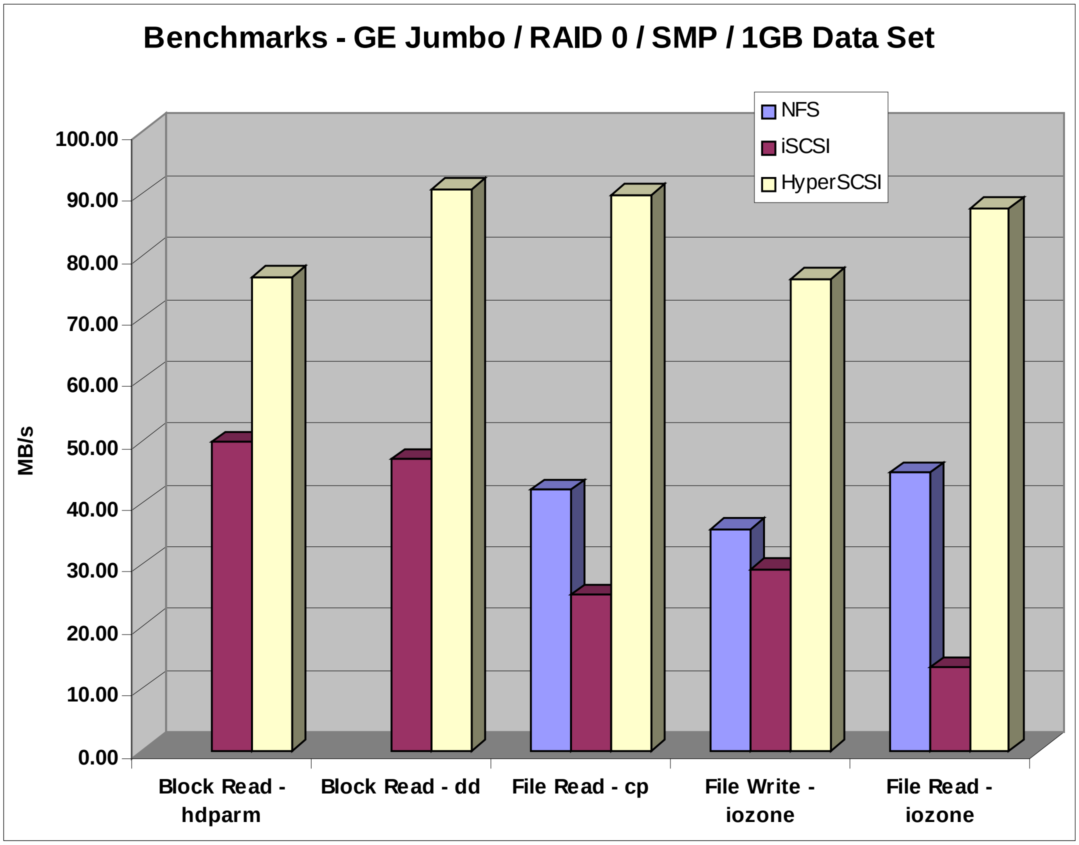


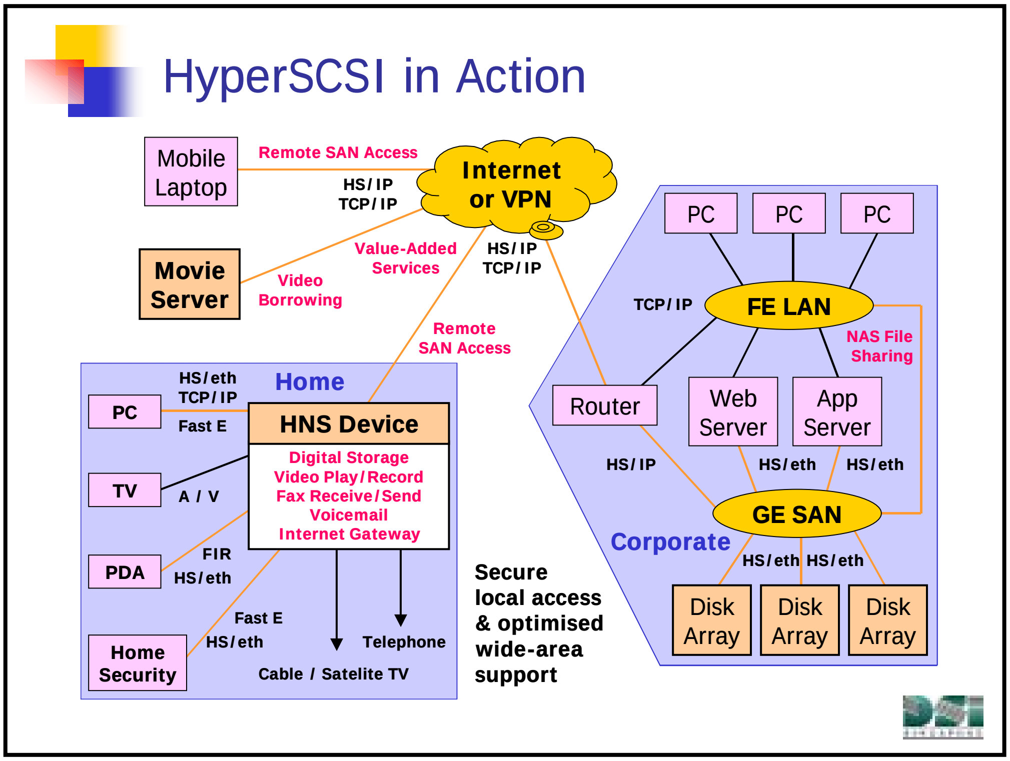
Figure 4: HyperSCSI Block and File Access Performance Comparison

Feature-wise, the HyperSCSI reference implementation already supports standard SCSI hard drives, IDE hard drives, software RAID / virtualised drives, optical disks (like DVDROM and CDRW), USB devices (like Iomega Zip Disk) and SCSI tape drives (like HP DAT40). We have even successfully used HyperSCSI is to write CDs remotely over our own live corporate LAN. File systems like Microsoft's FAT16/FAT32, SGI's XFS, IBM's JFS and Linux Ext2/Ext3 have all been successfully tested on HyperSCSI drives. HyperSCSI clients and servers have been successfully implemented on Linux, while client versions on Windows 2000 and Solaris 8 is currently in development. Encryption schemes that have already been implemented include 64-bit Blowfish and 128-bit Rijndael. HyperSCSI has been assigned its own IEEE Ethertype Number, and will soon receive a registered IP port for HS/IP implementations.

Areas that are currently under development (at the time of writing of this document) include aspects of the Management Command Stream, the Transmission Pause / Resume, various hashing and security related options, HS/IP implementation and Windows 2000 and Solaris 8 versions of the Linux client. With continued optimisations and bug-fixes of the reference implementation, we expect raw block data read speeds for a RAID0 subsystem of 8 drives on normal frame Gigabit Ethernet to exceed 100MB/s in early

2002. Another effort underway is the testing of several N-channel communications schemes for HyperSCSI. A Peer Round-Robin scheme is likely to be used in the final implementation. The documentation of HyperSCSI specifications is also critical in order to allow other organisations to adapt and build their own HyperSCSI solutions. Currently there are three documents in the HyperSCSI specifications, HyperSCSI Protocol Specifications, HyperSCSI Security Specifications and HyperSCSI Management Command Stream Specifications. A Quick Reference Manual, Reference Implementation Source Code Documentation, and various introductory documents like this one will also be provided. These documents will be available on our website when completed.

8. HyperSCSI Applications and Conclusion


HyperSCSI in Action

Secure local access & optimised wide-area support

Figure 5: HyperSCSI in Action

We believe that HyperSCSI provides an opportunity to address various concerns and open up new possibilities for network storage. The Local HS/eth protocol allows the construction of high-speed Ethernet based SANs while the use of Wide-Area HS/IP permits mobile devices like laptops to access the corporate SAN directly (bypassing servers if need be). Storage devices can support SAN or NAS or both access methods simultaneously through the use of a single network interface. Home devices will also be able to access storage directly with simple plug-and-play methods over Fast Ethernet or Wireless LAN using HyperSCSI's device discovery schemes. HyperSCSI has also been designed with the future in mind. It supports more than 32,000 different device options that will allow vendors to introduce a wide variety of vendor-specific capabilities and technologies, without sacrificing interoperability. The protocol also allows each single

HyperSCSI connection to handle 64 simultaneous in-transit SCSI commands, each with SCSI command block sizes up to 512KB. These SCSI command block sizes can be further increased six-fold by using Gigabit Ethernet jumbo frames, thus providing an even higher level of performance.

In conclusion, we believe that HyperSCSI is a relatively simple technology that can provide users with performance, security, scalability and flexibility, thus making it a viable alternative solution for network storage applications.

For more information on HyperSCSI, please visit our website at http://nst.dsi.nus.edu.sg/mcsa/

9. Acknowledgements

The authors would like to thank various people who have helped tremendously in the writing of this paper. First and foremost of course is Rod Van Meter who provided valuable feedback and reviews of our work, and in so doing helped us produce a better paper. In addition, various people have contributed tremendously in the development and testing of HyperSCSI, including Alvin Koy, Ng Tiong King, Vincent Leo, Don Lee, Premalatha Naidu, Wang Hai Chen and of course, Wei Ming Long, who carried out a lot of work on the testing environment. Thanks also to Jimmy Jiang, Lalitha Ekambaram and Yeo Heng Ngi, who have contributed much to the development of HyperSCSI on Win2K. This has been a team effort and our thanks and appreciation goes out to all that have helped in one way or another.

References

[1] IPS Working Group Internet Engineering Task Force, "Fibre Channel over TCP/IP (FCIP) Internet Draft", November 2001

[2] IPS Working Group Internet Engineering Task Force, "iSCSI Internet Draft", November 2001

[3] Brent Ross, Adaptec Inc, "IP Storage and iSCSI – The SAN Fabric of Choice", Proceedings of Storage Area Networks Conference 2001

[4] IPS Working Group Internet Engineering Task Force, "iFCP – A Protocol for Internet Fibre Channel Storage Networking Internet Draft", November 2001

[5] IPS Working Group Internet Engineering Task Force, "mFCP – Metro FCP Protocol for IP Networking Internet Draft", May 2001

[6] IPS Working Group Internet Engineering Task Force, "Internet Storage Name Service Internet Draft", November 2001

[7] American National Standard of Accredited Standards Committee ANSI NCITS T11.1 Technical Committee, “Information Technology – SCSI on Scheduled Transfer Protocol (SST) Work Draft”, July 2001

[8] David D Clark and David L. Tennenhouse, “Architectural Considerations for a New Generation of Protocols”, ACM SGICOM 1990 Symposium

[9] IPS Working Group Internet Engineering Task Force, “iSCSI Internet Draft”, November 2001

[10] Jeffrey S. Chase, Andrew J. Gallatin and Kenneth G. Yocum, “End System Optimisations for High-Speed TCP”, IEEE Communications Magazine, April 2001

[11] Roy Levine, “IP-based Storage: Benefits and Challenges”, Infostor, March 2001

[12] Andrew Conry-Murray, “Internet Protocol Version 6”, Network Computing, December 2001

[13] American National Standard of Accredited Standards Committee ANSI NCITS T11.1 Technical Committee, “Information Technology – Scheduled Transfer Protocol Revision 4”, October 2000

[14] Network Working Group Internet Engineering Task Force (IETF), “Internet Protocol Version 6 (IPv6) Specification RFC 2460”, December 1998

[15] American National Standard of Accredited Standards Committee ANSI NCITS T10 Technical Committee, “SCSI-3 Architecture Model (SAM)”, 1996, Revised 2001

[16] American National Standard of Accredited Standards Committee ANSI NCITS T11 Technical Committee, “Fibre Channel Protocol (FCP)”, 1996, Revised 2001

[17] Information Sciences Institute University of Southern California, “Internet Protocol DARPA Internet Program Protocol Specification RFC 791”, September 1981

[18] Information Sciences Institute University of Southern California, “Transmission Control Protocol DARPA Internet Program Protocol Specification RFC 793”, September 1981