UNIVERSIDADE FEDERAL DE SANTA CATARINA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

INTERNET SMALL COMPUTER SYSTEM INTERFACE: UM ESTUDO DE CASO PARA STORAGE AREA NETWORKS

João Pedro de Carvalho Rocha Dias Costa
Ricardo Delcastanher

Trabalho de conclusão de curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciências da Computação em 06 de dezembro de 2004.

Florianópolis – SC
2004 / 2.

João Pedro de Carvalho Rocha Dias Costa
Ricardo Delcastanher

INTERNET SMALL COMPUTER SYSTEM INTERFACE: UM ESTUDO DE CASO PARA STORAGE AREA NETWORKS

Orientador:

_____

Prof. Mário A. Ribeiro Dantas

Banca Examinadora:

_____

Prof. João Bosco da Mota Alves

_____

Prof. João Bosco Mangueira Sobral

"Me dê o zero e o um que eu tiro o tudo e o nada lá de dentro."

Julio Felipe Szeremeta

AGRADECIMENTOS

Aos nossos professores, que nos ensinaram a buscar o conhecimento.

Aos nossos familiares, por nos terem dado o suporte para chegarmos até aqui.

E à turma 99.2, “A Pior Turma de Todos os Tempos”.

RESUMO

Nota-se atualmente que existe uma grande preocupação no desenvolvimento de processadores mais poderosos, de memórias maiores e mais rápidas que são capazes de processar muita informação e gerar muitos dados. Essa quantidade de dados fica cada vez maior e então começam a surgir problemas de como armazenar essa grande quantidade de dados nos dispositivos de armazenamento disponíveis atualmente. Grandes empresas estão migrando, ou já estão utilizando, as Redes de Armazenamento de Dados (*Storage Area Networks* ou SANs) para suprir suas crescentes necessidades. O estudo presente é sobre um tipo de implementação mais barata de SAN, chamada de iSCSI (*internet Small Computer Systems Interface*). A outra implementação, a mais cara, é conhecida como *Fibre Channel* SAN. No estudo também são mostradas as características desta implementação e os motivos que levaram o iSCSI ter sido implementado desta maneira. Foi apresentada uma comparação entre os dois tipos de implementação de uma SAN, utilizando várias placas adaptadoras de rede e de barramento. Os resultados mostram que o iSCSI é uma solução viável para pequenas empresas, já que pode ser implementado sobre redes *Gigabit Ethernet* e pode funcionar em conjunto com uma implementação *Fibre Channel*.

ABSTRACT

Nowadays there is a great preoccupation in the development of better processors, bigger and faster memories that can process large amounts of information and generates big quantities of data. This data gets bigger and bigger over the time and then comes the problem in how to store them in the storage devices available today. Big companies are migrating or are actually using Storage Area Networks or SANs to fulfill their increasing needs of storage. The present study is about a cheaper implementation of a SAN, it's name is iSCSI (internet Small Computer Systems Interface). The other more expensive implementation is known as Fibre Channel SAN. It is also showed the characteristics of an iSCSI implementation and the reasons why they were made. A comparison between the two types of implementation was also made, using different network cards and host bus adapters. The results show that the iSCSI is a viable solution to small business because it can be done over Gigabit Ethernet Networks and it can work in conjunction with a Fibre Channel implementation.

SUMÁRIO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

CAPÍTULO I

1. INTRODUÇÃO

Para resolver a demanda cada vez mais crescente de armazenamento de grandes quantidades de dados, bem como um rápido acesso a estes dados, *Storage Area Network* ou SANs têm se tornado comum nas empresas. *Storage Area Network* representa um modelo de armazenamento distribuído onde vários *switches* conectam os servidores da SAN aos seus dispositivos de armazenamento. Elas permitem virtualização e compartilhamento de dados, bem como facilidade de backup e recuperação.

Até hoje, *Fibre Channel* [13] tinha sido o principal protocolo escolhido para implementar *Storage Area Networks*. Entretanto, o protocolo iSCSI [3], com o uso de placas *Ethernet* comuns, tem se tornado uma alternativa possível.

No capítulo II é dada uma introdução ao armazenamento de dados, mostrando as unidades que quantificam a quantidade de espaço utilizado por estes armazenadores. Também são citados os dispositivos de armazenamento acoplados diretamente na máquina, como as tecnologias ATA [9], SCSI [5] e RAID [8]. Seguidos dos dispositivos de armazenamento ligados à rede, e posteriormente das redes de armazenamento. Estas redes de armazenamento englobam as tecnologias *Infiniband* [14], *Fibre Channel* e iSCSI. Outras tecnologias importantes descritas no capítulo II são os *clusters* [15] e os *grids* [16].

O protocolo iSCSI é abordado no capítulo III. Primeiro é feita uma abordagem geral dos conceitos e características deste padrão como: camadas, sessões, numeração, ordenação, *Protocol Data Units* (Unidades de Dados do Protocolo), *login*, nomeação, estado persistente, sincronização de mensagens e manobrabilidade, descoberta, integridade dos dados, segurança, colocação direta de dados e recuperação. Algumas partes do padrão foram mais aprofundadas, como os nomes iSCSI, negociações e segurança.

No capítulo IV aborda-se as características de performance dos adaptadores iSCSI e HBAs *Fibre Channel* quando usados para conectar um servidor à uma SAN. Um número de diferentes *workloads* sintéticos para *IOMeter* foram usados no teste, incluindo *workloads* gerais como processamento OLTP e servidor *web*, como também

workloads de grandes e pequenas quantidade de transferências de dados. Cada *workload* foi avaliado utilizando medidas de performance como: largura de banda, número de operações de entrada e saída por segundo e utilização da CPU. É mostrado que soluções iSCSI baseadas em *Fast Ethernet* ou *Gigabit Ethernet* podem oferecer boa performance para certos tipos de aplicações a um custo muito baixo se comparado com a tecnologia *Fibre Channel*. Também é feita uma descrição de um *driver* iSCSI para o sistema operacional Linux.

A conclusão, bem como as propostas para trabalhos futuros, estão inseridas no capítulo V.

1.1. Motivação

A Internet já é conhecida como um dos principais meios para troca de informação, e vem se tornando também a principal solução para troca de arquivos. Não bastasse a “troca” de arquivos, existe também a necessidade de se deixarem expostos milhares de arquivos para serem acessados de todas as partes do mundo. Dependendo do tipo de serviço, estes arquivos podem facilmente ultrapassar a barreira do 1 *Terabyte*.

A partir deste problema surge um mercado para as empresas prestadoras de serviço de hospedagem para a Internet, que podem optar por máquinas com armazenamento local utilizando discos rígidos muito caros, ou, como mostra este documento, utilizar de armazenamento distribuído.

A opção de se utilizar armazenamento distribuído não só baixa os custos, como também tende a aumentar a segurança e a performance. Os dados não precisam estar geograficamente no mesmo lugar e poderão ser acessados de qualquer computador conectado à Internet. Caso todo o espaço disponível esteja sendo usado, para aumentar sua capacidade basta acrescentar uma máquina com tecnologia de armazenamento local barata a esta rede.

CAPÍTULO II

2. ARMAZENAMENTO DE DADOS

Como no final da década de 80 ainda hoje encontramos problemas ao armazenar dados. A diferença está na quantidade e no tipo de acesso a esses dados. Enquanto no século passado os dados eram requeridos apenas localmente, hoje, além da quantidade e tamanho (*megabytes*) dos dados serem maiores, existe a necessidade constante de acessá-los de qualquer lugar e de qualquer dispositivo. O que não mudou é a necessidade de se ter controle sobre o sistema, utilizando boas técnicas de gerenciamento, fazendo-o trabalhar para o usuário [6].

Planejar um sistema de disco rígido significa estabelecer um sistema que possa ser acessado por usuários de todos os níveis de habilidades, que permita a adição de novas aplicações e que antecipe a necessidade de organizar eficientemente os arquivos dados [6]. No armazenamento distribuído, esta premissa torna-se mais difícil de ser colocada em prática, devido à existência de diversos dispositivos de armazenamento semelhantes interligados por uma rede.

No princípio, um computador era conectado diretamente ao seu setor de armazenamento, e nenhum outro computador podia acessar os dados guardados lá. Os aplicativos eram executados em um *mainframe*. À medida que a computação cliente-servidor foi desenvolvida, os aplicativos passaram a ser executados em servidores dedicados, cada um com seu próprio armazenamento, e logo depois, esses aplicativos precisaram compartilhar dados. Conforme a capacidade dos sistemas de *disk array* cresceu, um único *array* podia suprir as necessidades de armazenamento de vários servidores. Desse modo, nasceu o armazenamento em rede.

A quantidade de informação que pode ser armazenada em discos é medida em *bytes*. Um *byte* é um caractere de informação, como um caractere numérico ou uma letra isolada. Para representar maior capacidade de armazenamento de dados utiliza-se o *kilobyte* que equivale a 1.024 *bytes*. Este valor se deve à matemática binária utilizada pelo computador, que é 2 elevado a 10ª potência. O *kilobyte* pode ser abreviado em KB ou Kbytes. Outra unidade de medida maior é um *megabyte*, que pode ser abreviado por MB. Um *megabyte* representa 1.000 *kilobytes* ou 1.024.000

bytes [7]. Confira na tabela 1.1 outras medidas de unidades de armazenamento de dados utilizadas atualmente.

Unidade de medida	Número de caracteres	Espaço
1 byte	1	8 bits
1 Kilobyte (Kb)	1.024	1024 bytes
1 Megabyte (Mb)	1.048.576	1024 Kb
1 Gigabyte (Gb)	1.073.741.824	1024 Mb
1 Terabyte (Tb)	1,099511628 xe12	1024 Tb

Tabela 2.1: Medidas de unidades de armazenamento de dados [7].

A organização do disco rígido é vital para a eficiente recuperação dos dados armazenados. O atual desafio para o profissional de computação é conseguir realizar isto dentro dos novos paradigmas de armazenamento que surgem com a interligação dos computadores e, consequentemente, dos seus discos rígidos.

2.1. Direct Attached Storage (DAS)

Trata-se provavelmente da solução mais comum de armazenamento em disco rígido utilizado em quase todos os tipos de computadores conhecidos. A limitação depende da capacidade do computador e do número máximo de discos rígidos que o mesmo pode suportar. A maior parte dos computadores suporta dois, sendo que alguns suportam até quatro discos. Atualmente, a capacidade máxima de cada disco é de aproximadamente 300 GB. Deste modo, a capacidade total de disco rígido é aproximadamente de 1,2 TB (*Terabyte*) [8].

2.1.1. Advanced Technology Attachment (ATA)

A *Advanced Technology Attachment* é uma implementação no disco que integra o controlador no próprio disco. Existem várias versões, desenvolvidas pelo Comitê SFF (*Small Form Factor Committee*).

Conhecido como IDE (*Integrated Development Environment*), o ATA suporta um ou dois discos de interface 16 bits. O ATA-2, também chamado de ATA rápido ou EIDE (*Enhanced IDE*), suporta modos PIO 3 e 4, mais rápidos que o PIO 0, 1 e 2 do ATA, multipalavra no modo DMA, *logical block addressing* (LBA) e transferências de bloco. Existe também uma pequena revisão do ATA-2, conhecida como ATA-3.

O Ultra-ATA, também chamado de Ultra DMA, ATA 33 ou DMA-33, suporta multipalavra DMA rodando a até 33MBps. Com velocidade de até 66MBps, o ATA/66 é uma versão proposta pela Quantum, e suportada pela Intel, que utiliza cabos IDE ATA de 80 vias, 40 vias a mais que os usados em outras implementações. Esta modificação foi criada para evitar a perda de desempenho e foi aproveitada para os ATA/100, versão atualizada do ATA/66, e ATA/133, a versão mais atual.

Existem ainda variações como PATA (*Parallel ATA*), que usa uma sinalização (troca de dados) paralela, e SATA (*Serial ATA*), que usa sinalização serial. Esta última é uma evolução da primeira, criando uma conexão ponto a ponto entre os dispositivos. A taxa inicial de transferência é de 150MBps, daí o nome de SATA 150. Uma das vantagens é o cabo de dados (cabo serial) de dimensão muito reduzida em relação aos antigos cabos para ATA [9].

2.1.2. Small Computer System Interface (SCSI)

O SCSI é uma interface paralela para ligar periféricos, usada inicialmente pela Apple Macintosh e Unix. Atualmente, é estendida a qualquer computador, mas devido ao seu valor mais alto em relação a outras implementações, este disco rígido é usado em locais onde os dados precisam de mais segurança, velocidade de busca e funcionamento 24 horas por dia e 7 dias por semana, como em servidores, por exemplo. Os discos SCSI, em sua maioria, necessitam de placas adaptadoras que possuam a controladora padrão SCSI. Hoje, poucas placas mãe têm uma controladora SCSI nativa *onboard* [5].

Por existir a possibilidade de se instalar vários SCSI em apenas uma porta original, esta implementação é chamada de Sistema Entrada e Saída ao invés de interface. Atualmente, sua velocidade chega a 320MB/s, além do *hot swap*, e conta

com discos de alta confiabilidade e de tempo de vida muito superiores aos já citados discos ATA [9].

Apesar do SCSI ser padronizado, existem milhares de variantes, em capacidades, desempenho e portas (conectores), o que algumas vezes geram incompatibilidades entre as mesmas.

2.1.3. Redundant Array of Inexpensive Disks (RAID)

Traduzido como disposição redundante de discos baratos, o RAID conta com vários discos rígidos combinados para aumentar a performance. Num nível mais complexo, o RAID pode ser usado também para melhorar a confiabilidade do equipamento através de espelhamento ou paridade [8].

Existem diferentes configurações de RAID visando ganhos para determinadas tarefas. O RAID 0 permite obter a melhor performance possível, sacrificando parte da confiabilidade. Todos os discos passam a ser acessados como se fossem um único disco. Ao serem gravados, os arquivos são fragmentados nos vários discos, permitindo que os fragmentos possam ser lidos/gravados ao mesmo tempo. Usando RAID 0 a performance chega a um patamar próximo da velocidade de todos os discos rígidos somada.

Já em um sistema RAID 1, temos dois discos onde o segundo armazena uma cópia fiel dos dados do primeiro, mesmo que um dos discos tenha problemas, o sistema continua intacto, funcionando como se nada tivesse acontecido. Combinando o RAID 1 com técnicas de espelhamento, um dos discos rígidos armazena dados, enquanto outro armazena uma cópia fiel dos mesmos dados, usados sempre em um número par. Este sistema é conhecido como RAID 10. Controladoras RAID IDE possuem, atualmente, limitação de quatro discos simultaneamente. Já as controladoras SCSI, além de combinar mais discos, possibilitam a troca de um dos discos com o computador ligado.

O RAID 2 pode oferecer maior consistência dos dados se houver queda de energia na escrita, pois armazena informação ECC (*error correcting code*), que é a informação de controle de erros incorporado internamente em todas as unidades de disco mais novas. O RAID 3 usa um sistema de paridade para manter a integridade dos dados. Num sistema com cinco discos, os quatro primeiros em sistema RAID 0

servirão para armazenar dados, enquanto o último armazenará os códigos de paridade. Nem RAID 2 e nem RAID 3 são suportados pelos *drivers* de RAID por software no Linux.

O modo RAID 4 é semelhante ao RAID 3, mas a forma como os dados são gravados nos demais discos é diferente. São divididos em blocos, pedaços bem maiores do que no RAID 3. Com isto, é possível ler vários arquivos ao mesmo tempo. O sistema RAID 5 é baseado no uso de paridade, espalhados entre os discos, para garantir a integridade dos dados caso um dos discos rígidos falhe [8].

É possível ainda combinar os modos 3 e 1. Esta implementação é conhecida como RAID 53 e pode ser implementada em sistemas com pelo menos cinco discos rígidos. Os dois primeiros discos formam um sistema RAID 3, os dois seguintes formam um sistema RAID 0 e o último armazena códigos de paridade de todos. Este é um modo pouco usado por não existir 100% de garantia de recuperação de todos os dados caso um dos discos falhe [9].

E, finalmente, o padrão RAID 6 que é relativamente novo e suportado por apenas algumas controladoras. É semelhante ao RAID 5, porém usa o dobro de bits de paridade, garantindo a integridade dos dados caso até dois dos discos falhem ao mesmo tempo.

2.2. Network Attached Storage (NAS)

No NAS, o sistema de arquivos organiza blocos de armazenamento em objetos que são convenientes para os aplicativos que lidam com eles. Este é responsável por alocar espaço e evitar complicações nas solicitações de acesso a arquivos. No lado do servidor, um cliente traduz as solicitações de Entrada e Saída de arquivos dos aplicativos em mensagens de rede, e as envia para o dispositivo NAS, para que sejam executadas.

2.2.1. Network File System (NFS)

O NFS é o sistema de arquivos em rede padrão do Linux, e foi desenvolvido pela Sun. Qualquer operação com arquivos executada por um programa em uma

máquina, é enviada pela rede para outro computador. Esse procedimento emula um ambiente em que todos os arquivos encontram-se no mesmo equipamento onde ele está sendo executado. Isso torna o compartilhamento de informações muito simples, já que não requer nenhuma modificação nos programas utilizados [11].

2.3. Storage Area Network (SAN)

SAN trata-se de uma rede especificamente concebida não só para movimentar grandes volumes de dados, devido sua largura de banda, mas também para melhorar e tornar mais eficiente a distribuição dos recursos de armazenamento pelos vários sistemas clientes [12]. Utiliza-se de conceitos já implementados em outras soluções e conhecidos por conversores de protocolo, concentradores, comutadores e outros, conforme vistos na figura 2.1.

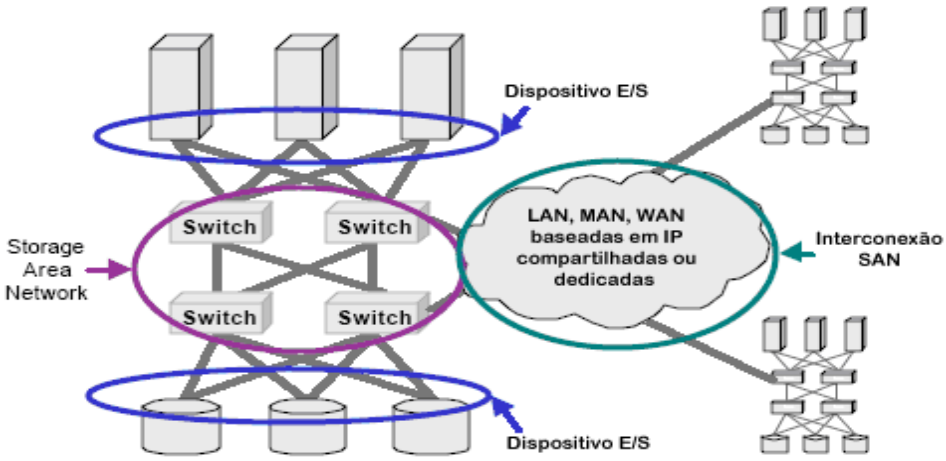


Figura 2.1: Segmentos de uma SAN

Em uma boa implementação, o principal objetivo é desviar das redes LAN/WAN para a SAN um volume de dados bastante considerável, evitando a sobrecarga e, em conseqüência, o mau desempenho das redes. Isto consegue ser alcançado graças à introdução de equipamentos FC-AL (*Fibre Channel Arbitrated Loop*).

As SANs oferecem, de uma forma geral, uma conexão total, permitindo assim a qualquer sistema cliente ceder uma conexão a qualquer outro dispositivo presente. Este fator garante melhor distribuição de recursos de armazenamento pelos

utilizadores, gestão centralizada dos recursos facilitada, utilização mais eficiente de todas as unidades de armazenamento, maior desempenho através de uma maior largura de banda e uma maior flexibilidade nas distâncias entre todos os componentes da SAN com o recurso do *Fibre Channel*. [12]

2.3.1. Fibre Channel (FC)

Fibre Channel é o padrão de conexão mais rápido até o momento disponível comercialmente. Sua velocidade chega a 400 MB/s, podendo atingir até 1 GB/s, e suportando até cento e vinte e seis dispositivos conectados.

É o único padrão que inclui a fibra óptica como meio de transmissão, além do fio de cobre. Graças a este material, um cabo pode atingir dez quilômetros de extensão, ao contrário dos outros padrões onde os cabos freqüentemente não podem ter mais do que cinco metros. Aceita diversos esquemas de conexão e freqüentemente é chamado pelo nome do esquema de conexão mais conhecido, FC-AL (*Fibre Channel-Arbitrated Loop*). É indicado para ambientes de rede que necessitem altíssima performance para armazenamento massivo de dados. [13]

2.3.2. Internet Small Computer Systems Interface (iSCSI)

O iSCSI é um padrão de armazenamento de dados em redes baseadas em IP [2], desenvolvido pela IETF (Força Tarefa de Engenharia para Internet), servindo para ligar instalações de armazenamento de dados [1]. O iSCSI é utilizado para facilitar transferências de dados pelas *intranets* e para gerenciar armazenamento à grandes distâncias. Isto é possível porque ele se utiliza de comandos SCSI sobre redes IP. O esquema na figura 2.2 ilustra os fatores que permitem o armazenamento sobre IP.

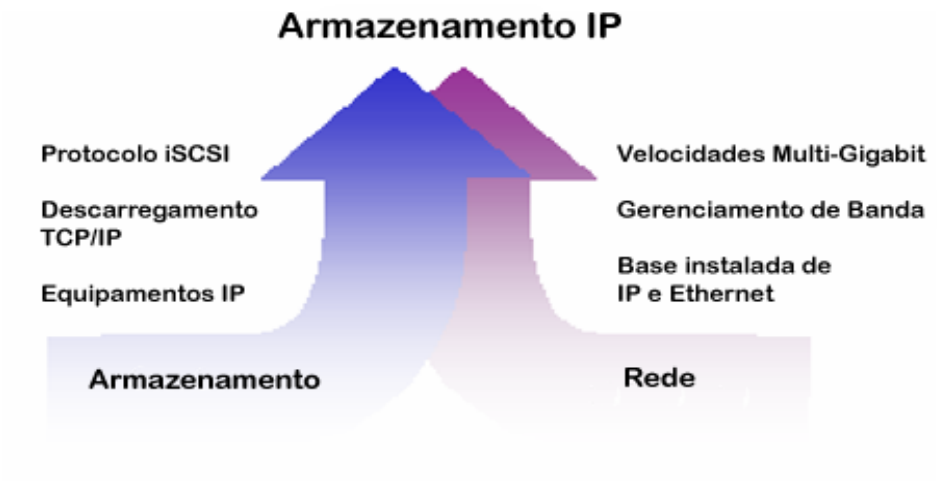


Figura 2.2: Fatores que permitem armazenamento sobre IP.

O protocolo iSCSI é uma das tecnologias que podem ser utilizadas nas *Storage Area Networks*, pois ele aumenta a capacidade e performance de transmissão de dados armazenados, utilizando uma rede IP já existente. Por isso pode ser utilizado para transmitir dados sobre Redes Locais (LANs), Redes Distribuídas Geograficamente (WANs) ou através da Internet, permitindo armazenamento e acesso aos dados independente da sua localização.

O iSCSI foi a tecnologia escolhida para ser implementada neste trabalho e será explicada com detalhes no capítulo III.

2.3.3. Infiniband

O principal objetivo do Infiniband é interligar servidores e dispositivos de armazenamento localizados a curtas distâncias. Funcionando como uma opção mais rápida às redes *Ethernet* ou a servidores de bancos de dados, ele possibilita acessar um dispositivo de armazenamento externo, sem nenhum gargalo, como se fosse um dispositivo local. Facilidade esta que abre muitas possibilidades nos servidores de alto desempenho, *clusters* e fazendas de servidores. [14]

O Infiniband é um barramento serial, que oferece 2.5 *Gigabits* por segundo por par de cabo, onde um cabo envia e outro recebe dados. Como a comunicação é bidirecional, temos 312 MB/s em cada sentido, totalizando um barramento total de 625 MB/s, em um cenário semelhante à transmissão *full-duplex* em uma rede *Ethernet*.

Esta tecnologia ainda não tem uso comercial amplamente divulgado, mas é bem provável que a sua especificação final seja utilizada também para interligar componentes internos dos computadores, substituindo o PCI ou funcionando como um barramento complementar, aumentando bastante a flexibilidade [14].

2.4. Cluster

Cluster é o nome dado a um sistema montado com mais de um computador de forma que pareça que eles sejam um computador só, fazendo com que todo o processamento da aplicação seja distribuído. Com isso, é possível realizar processamentos que até então somente computadores de alta performance eram capazes de fazer [15].

Cada computador de um cluster é denominado nó ou nodo. Todos devem ser interconectados, de maneira a formarem uma rede que permita o acréscimo ou a retirada de um nó sem interromper o funcionamento do cluster. O sistema operacional usado nos computadores deve ser de um mesmo tipo, pois existem particularidades em cada sistema operacional que poderiam impedir seu funcionamento.

Os custos de implementação e manutenção de um cluster costumam ser menores do que a aquisição/manutenção de computadores poderosos, e algumas vezes, o processamento é até mais rápido e eficiente. [15]

2.5. *Grid*

A idéia de usar o tempo ocioso de computadores em ambientes geograficamente distribuídos já é utilizada no projeto SETI, o qual pode ser considerado o embrião da computação em grade (*grid computing*). Um *grid* pode ser definido como uma forma de gerenciar racionalmente recursos computacionais distribuídos (tanto de hardware como de software). Através do *grid* podemos agregar os recursos da computação tradicional e da computação de alto desempenho, além de dar subsídios para a realização de projetos de aplicações temáticos baseados em ambientes de Computação Distribuída [16].

A integração de centros de super-computação, e suas aplicações, e usuários em malhas computacionais já é uma realidade em vários países, como no projeto *Caltech*, por exemplo. Iniciado em 1997, já conta com infra-estrutura de informação, software de *Grid*, ambientes escaláveis em clusters, ferramentas de softwares, gerenciamento de dados, visualização e códigos de aplicação. Oferece suporte *on-line* 24 horas por dia e 7 dias por semana, disponibilizando diversas ferramentas de monitoração, de verificação de desempenho e coordenação de atualizações de software distribuídos.

Toda a infra-estrutura citada é usada também em áreas como biologia, genômica, neurociência, projeto de aeronaves, física de altas-energias, astrofísica e em ambientes inteligentes e móveis.

No Brasil, instituições de fomento de pesquisa propiciaram o surgimento de centros de processamento de alto desempenho para aplicações científicas nacionais de relevância. Junto com a RNP (Rede Nacional de Pesquisa), instituições como FINEP, CNPq entre outras, dão subsídio para a interligação destes centros a velocidades de comunicação cada vez maiores. Entretanto, estes centros atuam de forma não integrada, e necessitam de uma forma mais colaborativa de trabalho para um maior benefício da ciência e tecnologia nacional. O projeto *Grid* Computacional Nacional surgiu como uma iniciativa de integrar e racionalizar o uso de tais centros de maneira escalável e eficiente propiciando uma capacidade crescente de processamento. [16]

CAPÍTULO III

3. iSCSI

O iSCSI é um padrão de armazenamento de dados em redes baseadas em IP com a finalidade de interligar instalações de armazenamento de dados. O iSCSI é utilizado para facilitar transferências de dados pelas *intranets* e para gerenciar armazenamento à grandes distâncias. Isto é possível porque ele se utiliza de comandos SCSI sobre redes IP.

O protocolo iSCSI é uma das tecnologias que vêm sendo cada vez mais utilizadas nas *Storage Area Networks*, pois ele aumenta a capacidade e performance de transmissão de dados armazenados, utilizando uma rede IP já existente. Por isso pode ser utilizado para transmitir dados sobre Redes Locais (LANs), Redes Distribuídas Geograficamente (WANs) ou através da Internet, permitindo armazenamento e acesso aos dados independente da sua localização.

Quando uma solicitação é feita por um usuário ou aplicação, o sistema operacional gera o comando SCSI apropriado e a requisição de dados, que então passam por um encapsulamento e, se necessário, passam por procedimentos de encriptação [1]. Antes de os pacotes de IP serem transmitidos por uma conexão *Ethernet* é adicionado a eles um cabeçalho. Quando um pacote é recebido, ele é descriptografado (se previamente criptografado) e desmontado, separando o pedido e os comandos SCSI. Os comandos SCSI são enviados ao seu controlador e de lá para o dispositivo de armazenamento SCSI. Devido o fato de o iSCSI ser bidirecional, o protocolo também pode ser usado para retornar dados em resposta à requisição original. Diferentemente do *Fibre Channel* sobre IP, o iSCSI pode rodar sobre uma rede *Ethernet* já existente.

3.1 Conceitos SCSI

O SAM-2 (*SCSI Architecture Model-2*) descreve em detalhes a arquitetura da família SCSI de protocolos de entrada e saída. Esta seção serve para familiarizar o leitor com a arquitetura SCSI.

No nível mais alto, o SCSI é uma família de interfaces que requisitam serviços de dispositivos de entrada/saída, incluindo discos rígidos, *drives* de fita, CD e DVD, impressoras e scanners. Na terminologia SCSI um dispositivo entrada/saída é chamado de “unidade lógica” (UL).

SCSI é uma arquitetura cliente-servidor. Clientes são chamados de iniciadores. Eles emitem comandos SCSI que requisitam serviços de componentes e unidades lógicas de um servidor, que é chamado de alvo. O dispositivo servidor na unidade lógica aceita comandos SCSI e os processa. Na figura 3.1 observa-se uma rede iSCSI padrão com iniciadores e alvos [5].

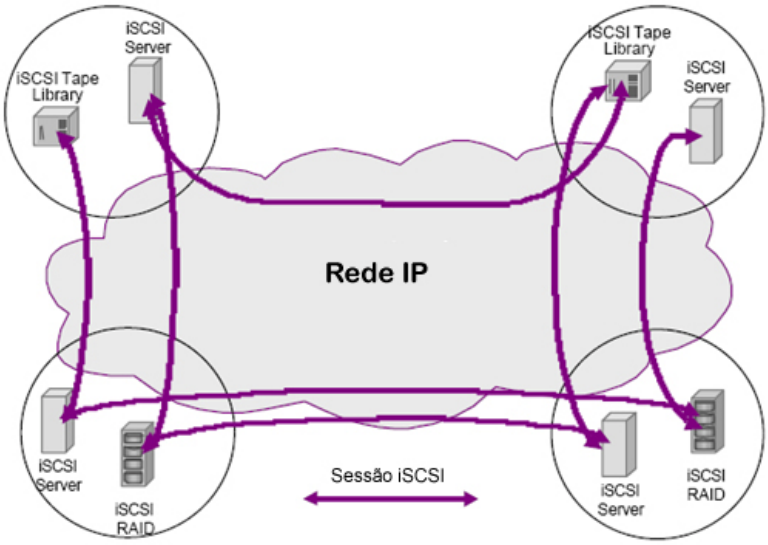


Figura 3.1: Uma rede iSCSI com iniciadores (servidores) e alvos (recursos de armazenamento).

Uma operação de transporte SCSI mapeia o protocolo SCSI cliente-servidor a uma interconexão específica. Iniciadores são uma extremidade do transporte SCSI e o alvo a outra extremidade. Um alvo pode conter múltiplas unidades lógicas. Cada unidade lógica tem um endereço no alvo chamado de Número da Unidade Lógica (NUL).

Uma tarefa SCSI é um ou mais comandos SCSI ligados. Algumas unidades lógicas dão suporte a múltiplas tarefas pendentes (na fila), mas esta fila é gerenciada pela unidade lógica. O alvo usa a “etiqueta de tarefa” presente no iniciador para distinguir entre tarefas. Em qualquer momento apenas um comando de uma tarefa pode estar sendo executado.

Cada comando SCSI resulta em uma fase de dados (opcional) e uma fase de resposta (obrigatória). Na fase de dados a informação pode viajar de um iniciador para um alvo (ex.: *WRITE*), de um alvo para um iniciador (ex.: *READ*), ou nas duas direções. A figura 3.2 exemplifica uma negociação *write*. Na fase de resposta o alvo retorna o *status* final da operação, incluindo possíveis erros [5].

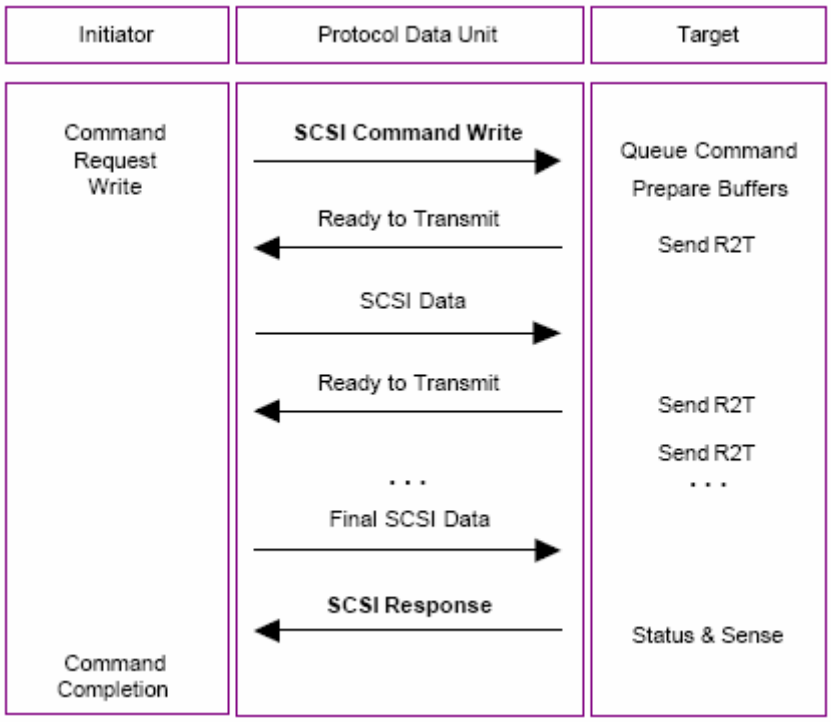


Figura 3.2: Exemplo de um *write* iSCSI.

Blocos Descritivos de Comandos (BDCs) são as estruturas de dados usados para encapsular os parâmetros do comando que um iniciador envia à um alvo. O conteúdo e a estrutura de um BDC são definidos pelo SAM-2 e padrões SCSI de tipos específicos dos dispositivos.

3.2 Conceitos e Características do iSCSI

Algumas das decisões de projeto do protocolo iSCSI foram fortemente influenciadas pela percepção de como o iSCSI seria eventualmente usado [4]. O iSCSI foi projetado para permitir que implementações eficientes de *hardware* e *software* pudessem acessar dispositivos E/S que estivessem sobre qualquer rede IP.

Também foi desenvolvido para uma grande variedade de ambientes e aplicações, incluindo espelhamento, acesso a armazenamento e *backups* locais e remotos. Foi presumido que adaptadores de aceleração TCP/IP e até adaptadores iSCSI (*Host Bus Adapters*) iriam se tornar dominantes, e seria muito desejável definir protocolos para permitir implementações de adaptadores de alta performance. Vários mecanismos foram incluídos para superar vários problemas já antecipados, como por exemplo a capacidade de manter alta largura de banda apesar da grande perda de pacotes. Foi tomado muito cuidado para não limitar a aplicação do iSCSI aos discos, vários mecanismos foram fornecidos para diferentes tipos de dispositivos SCSI.

O protocolo iSCSI é um mapeamento do modelo de invocação remoto SCSI [5] sobre o protocolo TCP (figura 3.3). Os comandos SCSI são carregados por requisições iSCSI bem como as respostas e *status* SCSI são carregados por respostas iSCSI. Por estar sendo usado por algumas décadas, o protocolo TCP tem a vantagem de ser o mais utilizado e o mais compreendido, coisa que outros protocolos como o SCTP não tem, apesar de possuírem muitas das características do TCP [4]. Algumas das características do protocolo TCP que são utilizadas pelo iSCSI estão listadas à seguir:

I. Os dados são entregues em ordem e de forma confiável;
II. Efetua retransmissão automática dos dados que não tiveram confirmação de recebimento;
III. Em uma rede congestionada, aplica controle de fluxo e de congestionamento, necessários para evitar a sobrecarga da mesma;
IV. Funciona sobre uma grande variedade de topologias e mídias físicas.

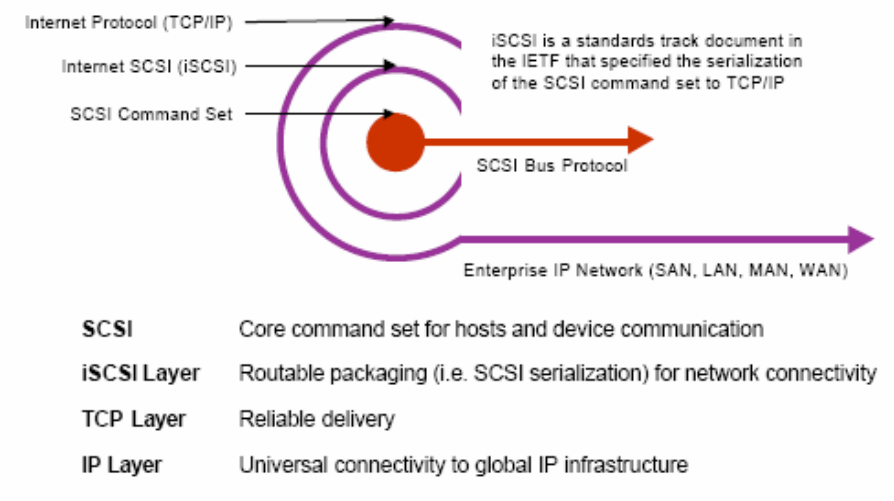


Figura 3.3: Participação de SCSI e IP para armazenamento em rede.

Da mesma maneira que protocolos similares fazem, o iniciador e o alvo dividem suas comunicações em mensagens. Essas mensagens são chamadas de “unidades de dados do protocolo iSCSI” (UDP iSCSI). Um dado associado a um comando pode ser enviado como uma mensagem única, do iniciador para o alvo, e a resposta, juntamente com algum dado, pode ser enviada a partir dos alvos.

A direção da transferência iSCSI é definida respeitando o iniciador. Transferências de saída são transferências de um iniciador para um alvo, enquanto que transferências de entrada são transferências de um alvo para um iniciador. Uma tarefa iSCSI é uma requisição iSCSI na qual uma resposta é esperada [3].

3.2.1 Camadas e Sessões

O modelo conceitual apresentado a seguir é usado para especificar as ações do iniciador e do alvo e a maneira como elas se relacionam com UDPs transmitidas e recebidas:

a) A camada SCSI monta/recebe BDCs (Blocos de Descrição de Comandos) e os passa/recebe com os parâmetros de execução de comandos restantes [5] para/de;
b) É a camada iSCSI que monta/recebe UDPs iSCSI e os repassa/recebe para/de um ou mais conexões TCP; o grupo de conexões forma uma

sessão iniciador-alvo. A figura 3.4 mostra a localização da camada iSCSI em relação às outras camadas.

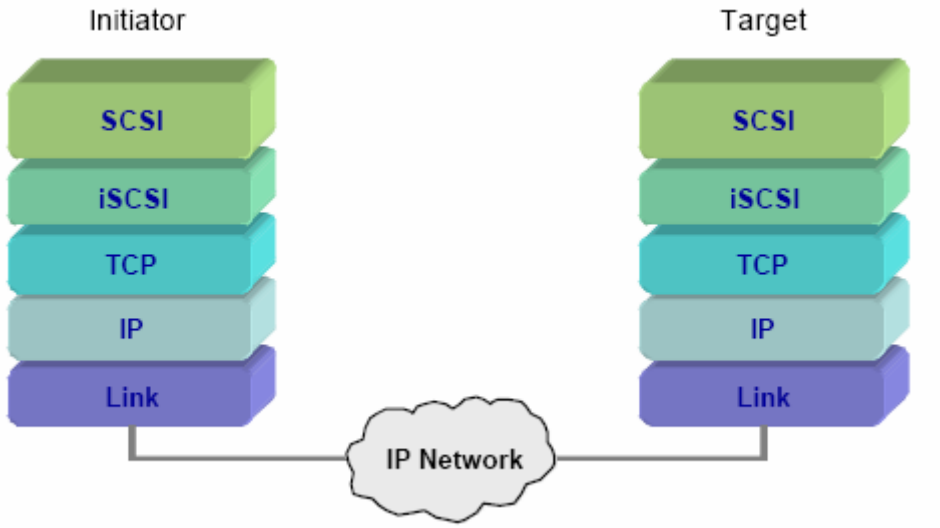


Figura 3.4: Camadas do protocolo iSCSI no iniciador e no alvo.

A comunicação entre o iniciador e o alvo ocorre sobre uma ou mais conexões TCP. As conexões TCP carregam mensagens de controle, comandos SCSI, parâmetros, e dados juntos de UDPs iSCSI. O grupo de conexões TCP que liga um iniciador com um alvo forma uma sessão [3]. Uma sessão é definida por um identificador de sessão que é composto por uma parte do iniciador e outra parte do alvo.

As conexões TCP podem ser adicionadas ou removidas de uma sessão. Cada conexão de uma sessão é identificada por um identificador de conexão (IDC).

O iniciador enxerga uma “imagem do alvo” através de todas as conexões de uma sessão. Todos os elementos identificados pelo alvo, como o número da unidade lógica (NUL), são os mesmos. O alvo também enxerga uma “imagem do iniciador” através de todas as conexões de uma sessão. Elementos identificadores do iniciador, como a Etiqueta de Serviço do Iniciador, são globais através da sessão independentemente da conexão na qual ela foi recebida ou enviada.

Alvos e iniciadores iSCSI suportam pelo menos uma conexão TCP e uma sessão pode suportar várias conexões. As conexões TCP de uma sessão iSCSI podem trafegar pelas mesmas mídias físicas ou por diferentes [4]. Apesar de uma única

conexão TCP ser suficiente para estabelecer uma comunicação entre um iniciador e um alvo, é geralmente mais vantajoso se utilizar de múltiplas conexões, pois:

I. Nem sempre é possível usar toda a largura de banda de uma interconexão física utilizando uma única conexão TCP;
II. Em máquinas multiprocessadas pode ser mais vantajoso permitir que *threads* distintas rodem em diferentes processadores, utilizando diferentes conexões TCP;
III. Podem-se espalhar múltiplas conexões TCP sobre todas as interconexões físicas existentes entre o iniciador e o alvo, agregando toda a largura de banda.

Para fins de recuperação de erros, iniciadores e alvos que suportam uma única conexão ativa em uma sessão devem ser capazes de suportar duas conexões durante a recuperação.

3.2.2 Numeração e Ordenação iSCSI

O iSCSI utiliza esquemas de numeração de s*tatus*, comandos e esquemas de seqüenciamento de dados.

A numeração de comandos ocorre na esfera da sessão e é usada para a entrega ordenada de comandos sobre múltiplas conexões. Também pode ser usado como um mecanismo para controle de fluxo de comandos sobre uma sessão. A numeração de *status* é feita por conexão. Na presença de erros de comunicação, permanentes ou transitórios, a numeração é usada para permitir a detecção e recuperação dos *status* que estão faltando [3].

O seqüenciamento de dados é feito por comando ou por parte de um comando (seqüência engatilhada R2T) e é usado para detectar dados e/ou UDPs R2T faltantes, que são causados por *header digest errors* (erros de condensação de cabeçalhos).

Os campos nos UDPs iSCSI fazem a comunicação dos números de seqüência entre o alvo e o iniciador. Nos períodos em que o tráfego numa conexão é

unidirecional pode-se utilizar UDPs *NOP-Out/In* para fazer a sincronização de contadores de ordenação de *status* e comandos do iniciador e do alvo.

A abstração da sessão iSCSI é equivalente ao *I_T nexus* do SCSI. A sessão iSCSI fornece uma entrega de comando ordenada indo do iniciador para o alvo SCSI. Para mais informações sobre as considerações de projeto do modelo de sessão iSCSI, veja [CORD].

O iSCSI executa a entrega ordenada de comandos dentro de uma sessão. Todos os comandos (UDPs) em trânsito, do iniciador para o alvo, são numerados. O iSCSI considera que uma tarefa está instanciada no alvo quando este responde cada pedido emitido pelo iniciador [3]. Um conjunto de operações de gerenciamento de tarefas, como por exemplo *abort* e *reassign*, podem ser executados em qualquer tarefa iSCSI.

Algumas tarefas iSCSI são tarefas SCSI, e muitas atividades SCSI estão relacionadas a tarefas SCSI [5]. Em todos os casos, uma tarefa é identificada pela sua Etiqueta de Tarefa do Iniciador durante toda a sua existência.

O número do comando é carregado pela UDP iSCSI com o comando CmdSN (*Command Sequence Number*) e a numeração está presente em todas as sessões. Todas as UDPs que são emitidas carregam este número. Comandos com entrega imediata são marcados com uma bandeira (*flag*) e devem também carregar seu CmdSN atual. O CmdSN não avança após o envio de um comando marcado com entrega imediata. A numeração de comandos começa com o primeiro pedido de *login* na primeira conexão de uma sessão e os números são incrementados em 1 para cada comando não-imediato emitido depois disso.

Se a entrega imediata é usada com comandos de gerenciamento de tarefas, esses comandos podem alcançar o alvo antes das tarefas na qual eles supostamente deveriam ser usados. Entretanto o CmdSN deles serve como um marcador das suas posições na cadeia de comandos. O iniciador e o alvo devem se assegurar que os comandos de gerenciamento de tarefas ajam como especificado no padrão SCSI. O número de comandos usado para entrega imediata não é limitado e sua entrega para execução não é reconhecida no esquema de numeração. Comandos imediatos podem ser rejeitados pela camada do alvo iSCSI devido à falta de recursos. Um alvo deve ser capaz de dar conta de pelos menos um comando imediato de gerenciamento de tarefas e um não-imediato por conexão a qualquer momento [3].

A camada alvo deve entregar os comandos para execução na ordem especificada pelo CmdSN. Os comandos imediatos podem ser entregues para execução o quão cedo são detectados. A entrega de alguns comandos pode ser evitada caso algum outro comando SCSI ou iSCSI assim o requerir. Em qualquer conexão, o iniciador iSCSI deve enviar os comandos em ordem crescente de CmdSN, com exceção de comandos que são retransmitidos devido a recuperação de erros de condensação e de conexão.

Para o mecanismo de numeração o iniciador e o alvo mantêm três variáveis para cada sessão:

I. CmdSN – é o número da seqüência de comando atual. Ë incrementado em 1 para cada comando enviado, exceto para comandos marcados para entrega imediata. O CmdSN sempre contém o número a ser designado para o próximo comando UDP;
II. ExpCmdSN – o próximo comando esperando pelo alvo. O alvo reconhece todos os comandos até, mas não incluso, este número;
III. MaxCmdSN – o número máximo a ser enviado. A capacidade de fila da camada iSCSI que está recebendo é MaxCmdSN – ExpCmdSN + 1.

O ExpCmdSN e MaxCmdSN do iniciador são derivados dos campos UDPs alvo-para-iniciador. O alvo deve silenciosamente ignorar qualquer comando não-imediato que não tenha valor entre ExpCmdSN e MaxCmdSn (incluído) ou comandos não imediatos não duplicados fora destes valores. O CmdSN carregado por comandos imediatos não precisa estar dentre estes valores.

Os campos MaxCmdSN e ExpCmdSN são processados pelo iniciador como mostrado a seguir:

a) Se a UDP MaxCmdSN é menor que o UDP ExpCmdSN-1, ambos são ignorados;
b) Se a UDP MaxCmdSN é maior que o MaxCmdSN local, o MaxCmdSN é atualizado. Caso contrário é ignorado;
c) Se a UDP ExpCmdSN é maior que o ExpCmdSN local, o ExpCmdSN é atualizado. Caso contrário é ignorado.

Esta seqüência é requerida por que as atualizações podem chegar fora de ordem (por exemplo: as atualizações são enviadas em conexões TCP diferentes).

Um pedido iSCSI numerado não irá mudar seu CmdSN, independente das circunstâncias e do número de vezes que ele é reenviado. No alvo, o CmdSN é apenas relevante quando o comando ainda não tenha criado nenhum estado relacionado a sua execução (estado de execução). Depois que esta relação acontece o CmdSN torna-se irrelevante. O teste para identificar o estado de execução (através da Etiqueta de Tarefa do Iniciador) deve preceder qualquer outra ação que ocorre no alvo. Se não é encontrado um estado de execução, então o comando segue para ordenação e entrega. Se um estado de execução é encontrado então o comando segue para entrega. [3]

Um alvo não deve emitir uma resposta de comando ou UDP Dado-Entrada com seu status antes de reconhecer o comando. Entretanto, o reconhecimento pode ser incluído na resposta ou na UDP Dado-Entrada.

Respostas transitando entre um alvo e um iniciador são numeradas. O StatSN (*Status Sequence Number*) é usado para este propósito. Ele é um contador mantido em cada conexão. O ExpStatSN é usado pelo iniciador para reconhecer o *status*. A numeração do *status* começa com a resposta de *login* para o primeiro pedido de *login* de uma conexão. A sua resposta inclui um valor inicial (qualquer valor é válido) para a numeração de *status*.

Para permitir a recuperação de comandos, o alvo pode manter informações de estado suficientes para a recuperação de *status* e dados após uma falha de conexão. Esta informação pode ser seguramente descartada após o reconhecimento da entrega do *status* para o comando, através de um ExpStatSN.

Uma grande diferença absoluta entre os valores de StatSN e ExpStatSN pode indicar uma falha de conexão. Iniciadores e alvos devem suportar o esquema de numeração de resposta.

3.2.3 *Protocol Data Units* (Unidades de Dados do Protocolo)

O protocolo iSCSI define seus próprios tipos de pacotes, que são as Unidades de Dados do Protocolo. As UDPs iSCSI consistem de um cabeçalho e dados, onde o comprimento do dado é especificado dentro do cabeçalho da UDP iSCSI. Uma UDP iSCSI é enviada como conteúdo de um ou mais pacotes TCP.

O tipos de UDPs iSCSI mais comumente usados são:

I. Comando/Resposta SCSI;
II. *Data In/Out* (Dado Dentro/Fora);
III. Pronto para transferir (R2T);
IV. Pedido/Resposta de *Login*.

A UDP do comando SCSI é usada para transferi-lo do iniciador para o alvo. Se o comando SCSI faz uma requisição para ler dados do alvo, este vai enviar os dados ao iniciador em uma ou mais UDPs *Data In*. Se o comando SCSI faz uma requisição para escrever dados no alvo, o iniciador os enviará a ele em uma ou mais UDPs *Data Out* [3]. O alvo pode enviar ao iniciador uma UDP R2T, que especifica ao iniciador qual parte dos dados ele deve enviar. Quando toda a transferência de dados estiver completa, o alvo envia um comando SCSI UDP Resposta para o iniciador, indicando que o comando teve êxito ou mostrando que alguma condição de erro foi detectada. Para cada comando SCSI UDP há uma única UDP Resposta, mas possivelmente múltiplas UDPs Dado ou nenhuma. Dados SCSI e UDPs Resposta devem ser enviados pela mesma conexão TCP em que seus comandos SCSI UDP foram originados.

3.2.4 *Login*

O propósito ou finalidade de um *login* iSCSI é permitir o uso de uma conexão TCP para fazer a autenticação de ambos os lados e a negociação dos parâmetros da sessão e marcar a conexão como pertencente à uma sessão iSCSI [3].

Os alvos ficam esperando conexões em uma porta TCP específica. Quando um iniciador conecta em uma dessas portas, através de uma UDP de Requisição de *Login*, é iniciado o processo de *login*. Como parte deste processo os dois devem tentar autenticar-se e podem colocar um protocolo de associação de segurança para a sessão. Ipsec [RFC3723] que irá proteger a conexão TCP. Nesta etapa também são feitas negociações de parâmetros operacionais.

Um procedimento de *login* deve ser efetuado imediatamente ao estabelecimento de uma conexão TCP entre o iniciador e o alvo iSCSI. O iniciador

envia um UDP de Requisição de *Login* para o alvo. Os dois podem tentar autenticar-se um com o outro e podem negociar parâmetros operacionais. Todas as implementações condizentes ao iSCSI devem ter suporte ao método de autenticação padrão, o *Challange-Handshake Authentication Protocol* (CHAP). Alguns dos parâmetros operacionais que podem ser negociados são o tamanho máximo das UDPs Dados, o número máximo de conexões a serem usadas na sessão, a quantidade de dados não solicitados que podem ser enviados ao iniciador (sem o uso do R2T), o nível suportado de recuperação de erro e se *digests* serão usados ou não para a detecção de erros [3]. Depois que os dois lados estão satisfeitos com a autenticação e com os ajustes operacionais, o alvo envia uma UDP *Login Response* com a indicação que os procedimentos de *login* estão completos. Só então é que a conexão pode ser usada para passar comandos SCSI e dados.

Antes do estabelecimento da fase completa, apenas UDPs *Login Request* e *Login Response* são permitidas. Pedidos e respostas de *login* devem ser usados exclusivamente durante esta fase. Em qualquer conexão, a fase de *login* deve ser seguida imediatamente pelo estabelecimento de uma conexão UDP e não deve ocorrer antes de finalizar uma conexão.

No caso de um alvo receber qualquer outra UDP antes da fase de *login*, a não ser uma UDP *Login Request*, ele deve imediatamente terminar a conexão aonde esta UDP foi recebida. Se o alvo receber um comando UDP qualquer (exceto uma UDP *Login Request*) depois do início da fase de *login*, ele deve enviar um *login reject* (com o *status* "inválido durante o *login*") e então desconectar. A conexão deve ser terminada imediatamente quando um iniciador recebe uma UDP qualquer, exceto uma UDP *Login Response*.

Quando uma autenticação ocorre e todos os seus parâmetros operacionais foram ajustados, a sessão faz a transição para a fase completa (*Full Feature Phase*) e o iniciador pode então começar a enviar comandos SCSI.

Durante o estabelecimento da sessão, o alvo identifica a porta do iniciador SCSI (o "I" do "*I_T nexus*") através do par de valores (*InitiatorName*, ISID). Qualquer estado persistente no alvo (ex.: reservas especiais), que está associado com a porta do iniciador SCSI, é identificado com base no valor do par. Qualquer estado associado com a porta do alvo SCSI (o "T" no "*I_T nexus*") é externamente identificado pelo *TargetName* e pela etiqueta de grupo do portal. A parte do

identificador da sessão do iniciador está sujeita a restrições de reuso pois é usada para identificar um estado persistente.

Uma vez que o iniciador é autorizado a fazê-lo, a sessão iSCSI passa para a Fase iSCSI Completa. Uma sessão está na fase completa quando ela consegue terminar com sucesso a fase de *login* na primeira conexão da sessão. Uma conexão está na fase completa se a sessão está na fase completa e a conexão de *login* foi completada com sucesso. Uma sessão não está na fase completa quanto ela não tem estabelecida uma conexão de transporte ou, quando tem uma conexão de transporte, mas não fez um *login* ou a conexão foi desconectada [3].

Em uma fase completa o iniciador pode enviar comandos SCSI e dados para várias unidades lógicas no alvo, isso ocorre através do encapsulamento dessas informações em UDPs iSCSI que trafegam por sessões iSCSI estabelecidas.

Dados SCSI de saída (dados do usuário ou parâmetros de comando do iniciador para o alvo) são enviados tanto como dados solicitados e não solicitados. Dados solicitados são enviados em resposta a UDPs R2T. Dados não solicitados podem ser enviados como parte de um comando UDP iSCSI (“dado imediato”) ou em UDPs Dado iSCSI em separado.

Dados imediatos se originam no *offset* 0 no buffer de escrita SCSI do iniciador (buffer de saída de dados). Todas as outras UDPs Data tem o seu *offset* do buffer ajustados no cabeçalho UDP.

Um iniciador pode enviar dados não solicitados como imediatos até o FirstBurstLength (maior tamanho UDP previamente negociado), em uma seqüência UDP em separado ou ambos. Todos os dados subseqüentes devem ser solicitados. O maior tamanho de uma UDP Data ou, a parte imediata do primeiro disparo não solicitado, podem ser negociados na fase de *login*.

O tamanho máximo de dados não solicitados que podem ser enviados junto a um comando são negociados no *login* através da chave FirstBurstLength. Um alvo pode separadamente permitir dados imediatos (através da chave ImmediateData) sem permitir a forma mais geral de dados não solicitados (através da chave InitialR2T).

Dados não solicitados na escrita servem para reduzir o efeito da latência no *throughput* (nenhum R2T é necessário para começar o envio de dados). E ainda mais, dados imediatos servem para reduzir o excesso (*overhead*) do protocolo, tanto em largura de banda como em tempo de execução.

Um iniciador iSCSI pode escolher não enviar dados não solicitados. Caso ele envie estes dados, os mesmos podem ser apenas comandos com dados imediatos ou *bytes* FirstBurstLength dos dados não solicitados. Se algum dado não solicitado não imediato é enviado, o total de dados não solicitados deve ser FirstBurstLength, ou todos os dados, se a quantidade total é menor que o FirstBurstLength.

Para o iniciador é considerado um erro enviar UDPs *Data* não solicitadas para um alvo que opera no modo R2T. Apenas dados solicitados são permitidos neste modo. Também é um erro quando o iniciador envia mais dados não solicitados, sejam imediatos ou UDPs em separado, do que FirstBurstLength. Um iniciador deve honrar uma requisição de dados R2T para um comando válido e deve entregar todos os dados requeridos, supondo que o comando dado deve entregar dados de saída e que o R2T especifique os dados dentro dos limites do comando. A ação do iniciador não é específica para o recebimento de requisições R2T que especificam todos ou parte dos dados, fora dos limites do comando. Um alvo não deve descartar dados silenciosamente e então pedir sua retransmissão através de um R2T. Os iniciadores não devem rastrear os dados transferidos de/para um alvo. Os alvos SCSI fazem um cálculo de contagem residual para verificar a quantidade de dados que foram realmente transferidos por um comando transferido para um dispositivo. Estes valores podem diferir da quantidade de dados enviados e/ou recebidos pelo iniciador por causa de erros e retransmissões. Leitura ou comandos bidirecionais solicitam implicitamente a transmissão de toda a quantidade de dados cobertos por um comando. Os pacotes de dados SCSI são combinados com seus comandos SCSI correspondentes através de etiquetas especificados no protocolo [3].

Os iniciadores e os alvos iSCSI ainda devem usar algumas regras de ordenação. Quando dados não solicitados são usados, a ordem deles em cada conexão deve combinar com a ordem na qual os comandos naquela conexão são enviados. Comandos e UDPs *Data* não solicitados podem ser inseridos em uma única conexão enquanto que os requerimentos de ordenação de cada um são mantidos. Um alvo que recebe dados fora de ordem pode terminar a sessão.

3.2.5 *Naming* (Nomeação)

O iSCSI utiliza um esquema baseado em URL para nomear seus alvos, como outros protocolos de internet o fazem. Nomes iSCSI devem ser globais, similares aos *World Wide Names* usados pelo *Fibre Channel*. Uma entidade iSCSI mantém seu nome mesmo tendo seu endereço IP modificado. Então ela é identificada por seu nome e não seu(s) endereço(s). Isto permite um manuseio mais fácil dos nomes iSCSI por parte dos *proxies*, *gateways*, caixas de tradução de endereços de rede, *firewalls*, e assim por diante. Nomes iSCSI devem ser únicos no mundo, um nome iSCSI típico se parece com este: iqn.2001-04.com.cco992:storage.disk2.sys1.xyz .

O prefixo *iqn* siginifica *iSCSI qualified name*, ou seja, nome qualificado iSCSI. O dispositivo nomeado no exemplo acima foi produzido por uma empresa que possuía um domínio com nome *cco992.com* durante 2001-04 (mês de abril do ano de 2001). Pode ser seguido por um conjunto de caracteres, que o dono julgue apropriado para o nome do domínio, para melhor qualificar o nome deste dispositivo e assim o tornar único.

3.2.6 Estado Persistente

O iSCSI não requer qualquer manutenção de estado persistente através das sessões. Entretanto, em alguns casos, o SCSI requer identificação persistente do nome da porta do iniciador SCSI.

As sessões iSCSI não persistem após operações de boot e através de ciclos de energia. Todas as sessões e parâmetros de conexão são reinicializados no momento da criação da conexão e da sessão.

Os comandos podem persistir mesmo com a terminação de uma conexão, apenas se a sessão não tiver sido desfeita e se a recuperação de comandos dentro de uma sessão estiver habilitada. Entretanto, quando uma conexão cai, a execução de comandos é suspensa até que uma nova “aliança” seja estabelecida pela função “redesignador de tarefas” do gerenciador de tarefas.[3]

3.2.7 Sincronização de Mensagens e Manobrabilidade

O iSCSI apresenta um mapeamento do protocolo SCSI sobre o TCP. Esta encapsulação é conseguida através do envio de UDPs iSCSI de tamanhos variados. Como o TCP não tem um mecanismo nativo para sinalizar os limites da camada TCP, o iSCSI, para superar esta limitação, coloca o comprimento da mensagem no cabeçalho da sua mensagem. Isto serve para delinear o final da mensagem atual bem como o começo da próxima mensagem.

Quando os pacotes de IP são entregues em ordem, os quadros de mensagem iSCSI não são um problema e as mensagens são processadas seqüencialmente. Se há perdas de quadros é desejável manobrar (manejar) o dado SCSI, dentro dos segmentos TCP desordenados, para *buffers* SCSI previamente alocados, ao invés de guardá-los em *buffers* temporários. Isto faz diminuir a necessidade de *buffers* dedicados de reorganização e diminui o atraso e a largura de banda necessária para fazer cópias extras dos dados.

Mas só depender da informação do tamanho da mensagem de um cabeçalho de uma mensagem iSCSI pode tornar impossível de se encontrar os limites de uma mensagem iSCSI em segmentos TCP subseqüentes. Isso porque pode ocorrer a perda de um segmento TCP que contém a informação do tamanho da mensagem iSCSI. Este segmento perdido deve ser recebido antes que qualquer segmento seguinte seja manobrado para o *buffer* SCSI correto, devido à inabilidade em determinar as fronteiras da mensagem iSCSI. Já que esses segmentos não podem ser manobrados para seu local correto, eles devem ser salvos em *buffers* temporários e então serem copiados para os *buffers* SCSI [3].

Diferentes tipos de esquemas podem ser usados para recuperar a sincronização. Para fazê-los funcionar, as implementações iSCSI devem assegurar que as camadas de protocolo apropriadas são providas com informação suficiente para implementar um mecanismo de sincronização ou manejo de dados.

O esquema de Marcadores de Intervalo Fixo (*Fixed Interval Markers* – FIM) funciona com a inserção de marcadores no fluxo de dados em intervalos fixos que contenham a saída para o começo da próxima UDP iSCSI.

Sob circunstâncias normais, sem perda de UDPs ou recepção de dados fora de ordem, o manejo dos dados é obtido através das etiquetas de identificação e os campos de saída no cabeçalho iSCSI além da seqüência de numero do cabeçalho TCP. A etiqueta ajuda a associar a UDP com um endereço de *buffer* SCSI enquanto que a

saída de dados e a seqüência de número TCP são usados para determinar a saída do *buffer*.

Os marcadores são utilizados para minimizar os danos de uma perda de cabeçalho UDP iSCSI da seguinte forma:

I. Marcadores indicam aonde a próxima UDP iSCSI começa e permitem a continuação do processamento mesmo quando cabeçalhos iSCSI são perdidos devido a erros de dados descobertos no nível iSCSI;
II. Marcadores ajudam a minimizar a quantidade de dados que devem ser mantidos pela camada iSCSI/TCP enquanto esperam a chegada de pacotes TCP ou sua recuperação. Eles podem ajudar a encontrar cabeçalhos UDP iSCSI e então usar a informação contida neles para manejar os dados para *buffers* SCSI.

Para minimizar a quantidade de *buffering*, é recomendado que o comprimento da UDP iSCSI seja restringido a um pequeno valor (talvez 2 segmentos TCP em comprimento). Durante o *login*, cada extremidade da sessão iSCSI especifica o comprimento máximo de uma UDP iSCSI que irá aceitar.

3.2.8 Descoberta (*Discovery*)

Quando se utiliza dispositivos de armazenamento sobre uma rede é necessário lidar com a capacidade de o iniciador conseguir descobrir dispositivos que ele pode usar. Um administrador pode tentar resolver este problema configurando estaticamente o iniciador. Basta fazer uma lista dos nomes e endereços com que os dispositivos iSCSI podem se conectar. Se fosse adicionado um novo dispositivo iSCSI a esta rede, o iniciador configurado estaticamente não seria capaz de acessá-lo sem ter que ser reconfigurado. Um método alternativo mais dinâmico seria usar SLP, que já existe nos protocolos da família IP. Os alvos iSCSI podem se registrar usando SLD, e os iniciadores podem acessar agentes SLP para obter informação sobre alvos registrados. Desta maneira, alvos iSCSI podem ser adicionados à rede, e a topologia pode mudar com o tempo, mas os iniciadores podem encontrar facilmente novos alvos

sem ter que ser reconfigurados. Um mecanismo similar é fornecido pelo recentemente definido protocolo iSNS.

O *Sendtargets*, um mecanismo adicional de descoberta, é fornecido com o próprio protocolo iSCSI, que é especialmente útil para dispositivos como *gateways*. Neste método, um iniciador é configurado estaticamente para conectar a um dispositivo de *gateway* iSCSI específico [3]. O iniciador estabelece uma sessão de descoberta com o *gateway* iSCSI e então manda requisições *Send-Targets* para o *gateway* iSCSI. Este responde com uma lista de alvos iSCSI conectados à rede e que estão disponíveis ao iniciador, que então pode proceder para conectar com os dispositivos iSCSI alvo.

3.2.9 Integridade dos Dados (*Data Integrity*)

O TCP possui um recurso, o *checksum*, que ajuda a detectar erros que acontecem durante a transmissão. Apesar da probabilidade do *checksum* do TCP ser pequena em falhar na tentativa de achar um erro, ele não é bom o bastante para alguns ambientes de armazenamento. O *checksum* do TCP também não fornece proteção contra corrupções que ocorrem enquanto a mensagem está na memória de algum roteador, que é quando a informação de cabeçalho pode ser recalculada e o dado não é mais protegido por um *checksum*. Então o iSCSI define seu próprio *checksum* de checagem de redundância cíclica (*cyclic redundancy check* – CRC) para assegurar integridade fim-a-fim dos seus dados e cabeçalhos de pacotes. Iniciadores e alvos podem negociar se querem ou não utilizar *checksum* CRC.

3.2.10 Segurança (*Security*)

Quando os dispositivos de armazenamento eram diretamente conectados às máquinas hospedeiras, os dados nos dispositivos de armazenamento eram considerados seguros por serem inacessíveis para o mundo externo. Este não é mais o caso com o uso de dispositivos de armazenamento conectados através de iSCSI. Um grande problema de segurança pode surgir se dados sensíveis armazenados são acessados sobre uma rede de dados geral. Uma solução possível é usar uma rede

fisicamente separada dos dados armazenados, similar ao que é feito com *Fibre Channel*. Esta solução requer uma segunda rede física de IP, que ainda é mais barata do que ter uma segunda rede física de *Fibre Channel*. Outra alternativa possível seria utilizar encriptação dos dados armazenados em uma rede física de IP. Encriptação esta que pode ser fornecida pelo IPSec. O iSCSI usa simplesmente o protocolo de segurança da família-IP já existente para proteger dados armazenados sensíveis de possíveis ataques como *sniffing* e *spoofing*.

3.2.11 Colocação Direta de Dados (*Direct Data Placement*)

Em implementações típicas de TCP, os dados que chegam a uma conexão TCP são primeiro copiados em *buffers* temporários. O *driver* TCP então examina a informação de identificação da conexão, número das portas e os endereços fonte e de destino, para poder determinar o receptor dos dados. Então os dados são copiados para o *buffer* deste receptor. Para dados SCSI, em qualquer momento, pode haver muitos comandos SCSI pendentes, e os dados recebidos devem ser copiados em um buffer específico fornecido pela camada SCSI para o comando particular. Todo este procedimento pode requerer que o *host* receptor tenha que copiar os dados um número de vezes antes que ele acabe no buffer de destino final. Tais cópias requerem uma quantidade significativa de uso do processador e memória, o que afeta prejudicialmente a performance do sistema. É desejável ser capaz de colocar os dados no seu destino final com um número mínimo de cópias.

Cabeçalhos UDP *Data* iSCSI contêm informação suficiente para permitir que um adaptador iSCSI (*Host Bus Adapter*) possa realizar colocação direta de dados. Esta informação do cabeçalho inclui uma etiqueta de transferência, que serve para identificar o comando SCSI e seu buffer correspondente, um *byte* de compensação relativo ao começo do *buffer* correspondente e um parâmetro de comprimento de dados indicando o número de *bytes* que estão sendo transferidos no pacote de dados. Esta informação é suficiente para permitir a colocação direta dos dados que chegam em *buffers* SCSI pré-registrados [3]. Um adaptador iSCSI que executa o processamento TCP e iSCSI obterá informação suficiente dos seus cabeçalhos para colocar os dados iSCSI que estão chegando diretamente nos *buffers* SCSI apropriados, sem ter precisado copiar os dados em *buffers* temporários na máquina *host*.

3.2.12 Recuperação (*Recovery*)

O protocolo iSCSI define vários níveis de recuperação para fornecer flexibilidade em face a uma grande quantidade de falhas e erros. É esperado que a recuperação e controle de erro no iSCSI seja uma ocorrência rara e pode envolver uma quantidade significativa de *overhead*. Sabe-se de antemão que a maioria dos ambientes computacionais não necessitarão de todos os níveis de recuperação definidos na especificação iSCSI.

A recuperação da falha de sessão é a classe mais básica de recuperação. Todas as implementações de acordo com a especificação iSCSI devem implementar a recuperação de falha de sessão. A recuperação de sessão envolve fechar todas as sessões de conexões TCP, cancelar todos os comandos SCSI daquela sessão, terminar todos os comandos SCSI cancelados no iniciador com a resposta de serviço SCSI apropriada e reinicializar a sessão com um novo conjunto de conexões TCP. Diferentes implementações podem fazer recuperação de falha de sessão para qualquer erro iSCSI detectado.

Um tipo menos drástico que uma implementação de recuperação pode usar é a recuperação chamada de *digest failure*. Se um erro de CRC *checksum* é detectado num dado iSCSI, o pacote do dado deve ser descartado. Ao invés de fazer a recuperação de sessão, pode ser utilizado o mecanismo de recuperação *digest failure* que pergunta se quem conectou quer reenviar apenas o dado que está faltando. Também pode ser utilizado de maneira similar quando ocorre um estouro de tempo de resposta, comandos e/ou respostas incompletas, perguntando se quem conectou quer reenviar esta informação faltante.

Se um erro de CRC *checksum* é detectado num cabeçalho de pacote iSCSI, ele deve ser descartado já que ele está corrompido. Isso causa uma perda de sincronização entre o iniciador e o alvo. Porém o protocolo iSCSI permite que uma nova conexão TCP seja estabelecida dentro da mesma sessão, e define mecanismos que permitem a sincronização do iniciador e do alvo para que eles continuem interagindo. Uma conexão TCP que aparenta estar com defeito pode ser substituída por uma nova conexão TCP. O processamento de comandos que foram iniciados na

conexão TCP com defeito podem ser continuados nesta nova conexão. Este nível de recuperação é chamado de recuperação de conexão. [3]

3.3 Nomes iSCSI

Tanto alvos quanto iniciadores precisam ter nomes para poderem ser identificados numa SAN iSCSI. Os nomes permitem que os recursos de armazenamento iSCSI possam ser administrados independentemente do seu endereço físico. O nome de um nodo iSCSI é também o nome de um dispositivo SCSI. O nome iSCSI de um dispositivo SCSI é o objeto principal utilizado para a autenticação de alvos e iniciadores e vice-versa. Este nome também é usado para identificar e administrar recursos de armazenamento iSCSI.

Estes nomes têm que ser únicos dentro do domínio operacional do usuário final. Porém o domínio operacional de uma rede IP será potencialmente de âmbito global, e por isso o formato do nome iSCSI foi arquitetado para ser único globalmente. Para ajudar as autoridades de nomeação na construção de nomes únicos globais o iSCSI fornece dois formatos de nomes, que serão explicados posteriormente neste trabalho.

Os nomes são associados a nodos iSCSI, e não à placas adaptadores de rede iSCSI. Isto assegura que a substituição de uma placa adaptadora de rede não irá obrigar uma reconfiguração de todas as informações de alocação de recursos iSCSI e SCSI.

Um iniciador pode descobrir nomes de alvos iSCSI que ele pode ter acesso, juntamente com seus respectivos endereços, através de uma requisição *SendTargets text request*, ou através de outras técnicas que pode ser encontradas no [RFC3721] e que não serão discutidas neste trabalho.

3.3.1 Propriedades do Nome iSCSI

Iniciadores e alvos devem suportar o recebimento de nomes iSCSI com o tamanho máximo de 223 *bytes*.

Na primeira requisição de *login* de uma nova seção ou conexão o iniciador deve apresentar tanto o nome do iniciador iSCSI quanto o nome do alvo ao qual ele deseja se conectar. A única exceção ocorre quando irá se estabelecer uma *discovery session* (sessão de descoberta). Neste caso o nome do iniciador ainda é requerido, porém o nome do alvo pode ser omitido.

Nomes iSCSI tem as seguintes propriedades:

a) São únicos globalmente, não há dois iniciadores ou alvos com o mesmo nome;
b) São permanentes, um nodo iniciador ou alvo mantêm seu nome por toda sua existência;
c) Um nome iSCSI não implica um endereço ou localização específica. Um alvo ou iniciador pode mudar de lugar ou ter múltiplos endereços. Uma mudança de endereço não implica em uma mudança de nome;
d) Não dependem de uma única agência de nomeação;
e) Suportam a integração com esquemas de nomeação únicos já existentes;
f) O nomes iSCSI tem suporte em autoridades de nomeação já criadas, não criou uma autoridade nova de nomeação.

A codificação de um nome iSCSI tem as seguintes propriedades:

a) Os nomes iSCSI possuem o mesmo método de codificação independentemente dos protocolos de outras camadas;
b) São simples de serem comparados. O algoritmo para comparar a equivalência de dois nomes iSCSI não depende de um servidor externo;
c) São formados por apenas caracteres apresentáveis. É permitido o uso de caracteres internacionais e maiúsculas não são diferenciadas de minúsculas. Caracteres em branco não são usados;
d) Podem ser transportados usando tanto protocolos binários quanto baseados em ASCII.

Um nome iSCSI nomeia na verdade uma entidade de *software* lógica, e não é amarrada a uma porta ou a outro *hardware* que pode vir a ser mudado. Por exemplo, um nome iniciador deve nomear o nodo iniciador iSCSI e não uma NIC ou HBA.

Quando múltiplas NICs estão sendo usadas, todas devem geralmente apresentar o mesmo nome do iniciador iSCSI para seus alvos, porque eles são apenas caminhos para a mesma camada SCSI. Na maioria dos sistemas operacionais a entidade nomeada é a imagem do sistema operacional.

De forma semelhante, um nome de um alvo não deve estar amarrado às interfaces de *hardware* que podem ser mudadas [3]. O nome de um alvo deve identificar o alvo lógico e deve ser o mesmo para o alvo independente da porção física sendo endereçada. Isto ajuda os iniciadores a determinar que os dois alvos que ele descobriu são na verdade dois caminhos para o mesmo alvo.

3.3.2 Codificação de Nomes iSCSI

Um nome iSCSI deve ser um codificação UTF-8 de um conjunto de palavras com caracteres Unicode com as seguintes propriedades:

a) Estar na forma C de normalização [UNICODE].
b) Só conter caracteres permitidos pelo resultado do modelo *stringprep* iSCSI [RFC3722].
c) Usar os seguintes caracteres para a formatação de nomes iSCSI:
    a. hífen ('-'=U+002d)
    b. ponto ('.'=U+002e)
    c. dois-pontos (':'=U+003a)
d) A codificação UTF-8 do nome não pode ser maior que 223 *bytes*.

O processo *stringprep* está descrito em [RFC3454] e seu uso por parte do iSCSI está explicado em [RFC3722]. O *stringprep* não precisa ser implementado se os nomes são gerados apenas usando caracteres do alfabeto numéricos e minúsculos. Uma vez normalizados, os nomes iSCSI codificados em UTF-8 podem ser comparados *byte-a-byte* com segurança.

3.3.3 Estrutura do Nome iSCSI

Um nome iSCSI consiste de duas partes, um designador de tipo seguido de um *string* de nome único. As autoridades de nomeação suportadas são: um nome Qualificado-iSCSI, que usa nomes de domínio para identificar a autoridade de nomeação; e o formato EUI, na qual a autoridade de registro IEEE ajuda na construção de nomes únicos globalmente (formato EUI-64).

As *strings* do tipo de designador atualmente definidos são:

i) iqn. – nome Qualificado-iSCSI;
ii) eui. – o resto da *string* é um identificador IEEE EUI-64, codificado em ASCII hexadecimal.

O tipo “iqn.” (*iSCSI Qualified Name*) pode ser usado por qualquer organização que é proprietária de um domínio. Este formato de nomeação é útil quando um usuário final ou um provedor de serviço deseja atribuir nomes iSCSI para alvos e/ou iniciadores.

Para gerar nomes deste tipo, a pessoa ou organização que está gerando o nome deve ser dona de um domínio registrado. Este domínio não precisa estar ativo e não precisa resolver um endereço, ele só precisa estar reservado/registrado para prevenir que outros gerem nomes iSCSI usando o mesmo domínio.

Como o registro de um domínio pode expirar e ser adquirido por outra entidade, ou pode ser usado para gerar nomes pelos seus dois donos, o nome do domínio deve também ser qualificado pela data na qual a entidade era dona do domínio. Assim então, um código com a data faz parte do formato “iqn.”.

Um nome qualificado completo consiste de:

a) A palavra “iqn.”, utilizada para distinguir dos nomes “eui.”.
b) Um código com a data no formato aaaa-mm. O mês deve ser o primeiro mês na qual a entidade era dona do domínio.
c) Um ponto “.”.
d) O domínio reverso da entidade que está criando o nome iSCSI
e) Um conjunto de palavras opcionais, prefixadas por um sinal de dois-pontos “:”, contendo tipos de produtos, números seriais, identificadores de máquinas, ou qualquer outra palavra que o dono do domínio quiser.

É de responsabilidade da entidade, que é a autoridade de nomeação, assegurar que os nomes que ela aloca são únicos globalmente. Por exemplo, “Cco992, A Pior Turma de Todos os Tempos”, pode ser dona do domínio “cco992.com”. A seguir estão alguns exemplos de nomes Qualificados-iSCSI que poderiam ser gerados pela “Cco992, A Pior Turma de Todos os Tempos”.

Tipo	*Data*	***Autoridade de Nomeação***	Conjunto de palavras definidas por “cco992.com”
iqn.*2004-10.***com.cco992**:armazenamento:diskarrays-sn-f8102918			
iqn.*2004-10.***com.cco992**			
iqn.*2004-10.***com.cco992**:armazenamento.disco1.sistema1.truta			
iqn.*2004-10.***com.cco992**:armazenamento.disco3.sistema1.truta			

Tabela 3.1: Exemplo de nome “iqn.”

O tipo “eui.” (formato IEEE EUI-64) é um serviço para designar identificadores globais únicos [EUI]. O formato EUI-64 também é usado para construir identificadores globais em outros protocolos de rede.

O formato deste tipo de endereço é o “eui.” Seguido de um identificador EUI-64 (16 dígitos hexadecimal codificados em ASCII).

Um exemplo de nome iSCSI seria: eui.02004567A425678D

Este formato de nome pode ser usado quando um fabricante já seja registrado na Autoridade de Registro IEEE e quando ele usa nomes globais únicos formatados em EUI-64 em seus produtos. Mais exemplos de construção de nomes pode ser encontrado em [RFC3721].

3.4. Negociação de Login e Fase Completa

Os parâmetros iSCSI são negociados no estabelecimento da sessão ou conexão, através do uso de Pedidos e Respostas de *Login*, e durante a fase completa através de Pedidos e Respostas de Texto. Em ambos os casos o mecanismo usado na negociação é a troca de pares de iSCSI-*text-key=value*. Para maior facilidade iSCSI-*text-keys* serão chamados apenas de chaves.

Chaves podem ser declarativas ou podem necessitar de negociação e a descrição da chave indica se ela é declarativa ou se requer negociação.

Para chaves declarativas, o lado que está declarando especifica um valor para a chave. Esta especificação indica se uma chave pode ser declarada pelo iniciador, alvo ou ambos.

Para as chaves que necessitam de negociação um dos lados (o que está propondo a negociação) propõe um valor ou conjunto de valores através da inclusão do *key=value* na parte destinada ao dado de um *Login* ou de uma UDP Resposta ou uma UDP de Requisição de Texto. O outro lado (que está aceitando a negociação) faz uma seleção baseado no valor ou na lista de valores propostos e o inclue em um *key=value* na parte destinada ao dado de um *Login* ou de uma UDP Resposta ou uma UDP de Requisição de Texto. Para a maioria das chaves tanto o alvo quanto o iniciador pode ser o lado que está fazendo a proposta.

O processo de *login* procede em dois estágios – o estágio de negociação de segurança e o estágio de negociação do parâmetro operacional. Ambos os estágios são opcionais, mas pelo menos um deles tem que estar presente para permitir a configuração de parâmetros mandatários. Se estiver presente, o estágio de negociação da segurança precede o estágio de negociação de parâmetros operacionais.

O avanço de estágios é controlado pelo *bit* T (Transição) no cabeçalho UDP de Requisição/Resposta de *Login*. Quando o *bit* T está setado em 1, o iniciador está indicando que gostaria de fazer uma transição. O alvo concorda com a transição (e seleciona o próximo estágio) quando estiver preparado/pronto. Um campo presente no cabeçalho UDP de *Login* indica o estado atual. Durante a transição outro campo indica o próximo passo proposto (iniciador) e selecionado (alvo).

O processo de negociação de texto é usado para negociar ou declarar parâmetros operacionais. O processo de negociação é controlado pelo *bit* F (final) no cabeçalho UDP. Durante negociações de texto, o *bit* F é usado pelo iniciador para indicar que está pronto para terminar a negociação e o alvo usa este *bit* F para concordar/aceitar o fim da negociação.

Como alguns pares *key=value* podem não caber inteiramente em uma única UDP, o *bit* C (continuação) é utilizado (tanto no *Login* quanto no Texto) para indicar que “virão mais”.

A negociação de texto usa um mecanismo adicional no qual um alvo pode entregar grandes quantidades de dados para um iniciador que estava perguntando. O

alvo prepara uma Etiqueta de Serviço do Alvo para ser usada como uma marcação para que, quando retornada pelo iniciador, signifique “continue”. Se esta etiqueta for mudada para “valor neutro”, significa “esqueça de todo o resto”. [3]

3.4.1. Negociação em Modo Texto

Durante o login e em diante, alguns parâmetros de sessão e conexão são declarados ou negociados através da troca de informação textual.

O formato de uma declaração é:

```
Declarador-> <key>=<valuex>
```

O formato da negociação de texto é:

```
Proposer-> <key>=<valuex>
Acceptor-> <key>={<valuey>|NotUnderstood|Irrelevant|Reject}
```

Então uma declaração é uma troca de texto de apenas uma direção enquanto que uma negociação é uma traca de duas direções.

Quem faz a proposta ou a declaração pode tanto ser o iniciador como o alvo, e quem aceita pode ser tanto o alvo como o iniciador, respectivamente. Alvos não são limitados responder à pares *key=value* como são propostos pelo iniciador. O alvo pode propor seus próprios pares *key=value*.

Todas as negociações são explícitas e não há propostas implícitas. Se uma proposta não é feita então uma resposta não deve ser esperada. Uma modelagem conservativa não deve depender de valores padrão quando o uso de outros valores terá sérias conseqüências.

Os valores propostos ou declarados podem ser um valor numérico, um espaço de valores, um valor binário, texto, nome iSCSI, nome iSCSI-local, valor booleano ou uma lista de textos separados por vírgual. Um espaço de valores, um valor numérico muito grande, um nome iSCSI e um nome iSCSI-local podem ser usados apenas se explicitamente permitidos. Um valor aceitável pode ser um número, um texto ou um valor booleano. Se uma chave não for relevante para a negociação, quem a está aceitando pode responder com uma constante “Irrelevante”, porém a negociação não é considerada que falhou. A resposta “Irrelevante” serve para aqueles casos em que

várias chaves são apresentadas por quem está fazendo a proposta e quem está aceitando escolhe uma delas, tornando as outras irrelevantes.

Uma chave não entendida por quem está aceitando pode ser ignorada sem afetar a função básica dela. Porém a resposta para uma chave não entendida deve ser *key=NotUnderstood*. As constantes "*None*", "*Reject*", "*Irrelevant*", e "*NotUnderstood*" são reservadas e só devem ser usadas como descrito aqui. A violação desta regra causa um erro de protocolo.

Uma negociação é considerada completa mesmo quando quem está aceitando envia um par de chaves valor, mesmo que este valor seja um "*Reject*", "*Irrelevant*" ou "*NotUnderstood*". O reenvio da chave seria considerado uma re-negociação e é proibido para várias chaves.

Se quem está aceitando a chave envia um "*Reject*" como resposta, a chave negociada permanece com seu valor atual (ou valor padrão se nenhum valor foi atribuído). Se este valor não é aceitável na conexão ou sessão na qual foi enviado por quem o propôs, o mesmo pode decidir terminar a conexão ou a sessão.

Novas chaves podem ser introduzidas por implementadores, bastando prefixá-las com um "X-", seguida por seu domínio reverso. Ou podem ser introduzidas novas chaves registradas com a IANA prefixando-as com "X#". Um exemplo de chave nova para uma entidade com o domínio "cco992.com" seria:

```
X-com.example.bar.foo.do_something=3
```

Ou uma nova chave registrada pela IANA poderia ser:

```
X#SuperCalyPhraGilistic=Yes
```

Negociações devem ser consideradas como operações atômicas. Alguns parâmetros podem estar sujeitos à regras de integridade. Regras de integridade são especificadas com as chaves sempre quando requeridas. A checagem para concordância com a regra de identidade deve ser feita depois que todos os parâmetros se tornam disponíveis [3]. Um alvo iSCSI deve checar a integridade de um novo parâmetro antes que ele se efetive. Um iniciador pode fazer a checagem de integridade também.

Um iniciador ou alvo podem terminar uma negociação que não termina dentro de uma quantidade de tempo ou de trocas razoáveis.

3.4.1.1 Negociações de Lista

Na negociação de lista o originador envia uma lista de valores (que pode incluir "*None*") em uma ordem de sua preferência. O outro lado deve responder com a mesma chave e o primeiro valor que ele suporta da lista enviada pelo originador.

A constante "*None*" deve sempre ser usada para indicar uma função que está faltando. Entretanto "*None*" é uma seleção válida apenas se ele é proposto explicitamente.

Se algum valor da lista não é entendido por quem o está aceitando, ele deve ser ignorado. A constante "*Reject*" ou a terminação da negociação pode ocorrer se quem está aceitando a chave não suportá-la, entendê-la ou se não tiver permissão para usar qualquer uma das opções propostas pelo originador. A seleção de um valor não proposto deve ser lidada como um erro de protocolo.

3.4.1.2 Negociações de Valor Simples

Para negociações de valor simples o lado que está aceitando deve responder com a mesma chave. O valor que ele seleciona se torna o resultado da negociação. A proposta de um valor inadmissível pode ser respondida com a constante "*Reject*" ou o aceitante pode selecionar um valor admissível.

A seleção, por parte do aceitante, de um valor não admissível sob as regras de seleção é considerado como um erro de protocolo. As regras de seleção são específicas para cada chave. Para um espaço numérico o valor selecionado deve ser um inteiro dentro deste espaço ou um "*Reject*" (se o inteiro não estiver no espaço definido).

Em negociações booleanas (que a resposta das chaves deve ser Sim ou Não), o lado que está aceitando deve responder com a mesma chave e com o resultado da negociação quando o valor recebido não consegue determinar o resultado por si só. O último valor transmitido se torna o resultado da negociação. As regras para selecionar

o valor da resposta são expressos como funções booleanas dos valores recebidos, e o provável valor que o aceitante teria selecionado se tivesse escolha.

3.4.2 Fase de *Login*

A Fase de *Login* estabelece uma conexão iSCSI entre um iniciador e um alvo; também cria uma nova sessão ou associa esta conexão com uma sessão já existente. Na Fase de *Login* é ajustado os parâmetros do protocolo iSCSI, parâmetros de segurança e é feita a autenticação entre o alvo e o iniciador.

A Fase de *Login* só é implementada através de Requisições e Respostas de *Login*. Toda essa fase é considerada como uma única tarefa e tem apenas uma Etiqueta de Iniciador de Tarefa.

O valor padrão de *MaxRecvDataSegmentLength* é usado durante o *Login*.

A seqüência de requisições e respostas da Fase de *Login* se procede como a seguir:

i) Requisição inicial de *Login*.
ii) Resposta parcial de *Login* (opcional).
iii) Mais requisições e respostas de *Login* (opcional).
iv) Resposta final do *Login* (obrigatório).

A requisição inicial de *Login* de qualquer conexão deve incluir o par *key=value InitiatorName*. Na requisiçao inicial de *Login* da primeira conexão de uma sessão pode também conter o par *key=value SessionType*. Para qualquer conexão dentro de uma sessão que o seu tipo não é de “Descoberta”, a primeira requisição de *Login* também deve incluir o par *key=value TargetName*. A resposta final do *Login* aceita ou rejeita a requisição de *Login*.

Esta fase pode ter um estágio de negociação de segurança, um estágio de negociação operacional ou ambos, mas não pode ter nenhum dos dois. Os estágios incluídos podem ser vazios, exceto para nomes obrigatórios.

Alguns parâmetros operacionais podem ser negociados fora do *Login* através de Requisições e Respostas de Texto. Em alguns contextos o alvo ou iniciador pode

não estar interessado em fazer a sua autenticação com sua contrapartida, isto é possível através da Requisição e Resposta de *Login.*

O iniciador e o alvo podem querer negociar parâmetros de autenticação iSCSI. Uma vez que esta negociação está completa o canal é considerado seguro.

A transição de estágio é feita através da troca de comando (requisição/resposta) que carrega o *bit* T e o código do estágio atual. Durante esta troca, o próximo estágio é selecionado pelo alvo através do código do próximo estágio. O código do próximo estágio não deve exceder o valor definido pelo iniciador. O iniciador pode requerer uma transição sempre que o mesmo estiver pronto, mas o alvo só pode responder com uma transição após uma ter sido proposta pelo iniciador [3].

Quando uma transição é requerida pelo iniciador e confirmada pelo alvo tanto um quanto o outro mudam para o estágio selecionado.

As transições de estágio durante o *Login* (incluindo entrada e saída) são possíveis como mostrado na tabela seguinte:

Para (estágio) ➔ De (estágio) 🡻	Segurança	Operacional	Fase Completa
(início)	sim	sim	não
Segurança	não	sim	sim
Operacional	não	não	sim

Tabela 3.2: Transições de estágio durante o *Login*.

Nem o iniciador nem o alvo devem tentar declarar ou negociar um parâmetro mais de uma vez duranto o *login*, exceto em caso de respostas a chaves específicas que permitem explicitamente declarações de chaves repetidas. Uma tentativa de redeclarar/renegociar parâmetros não especificamente permitidos deve ser detectado pelo iniciador e pelo alvo. Se tal tentativa é detectada pelo alvo, ele deve responder com um *Login reject* (erro do iniciador); se detectado pelo iniciador, ele deve derrubar a conexão.

3.4.2.1 Começo da Fase de *Login*

A requisição inicial de *Login* inclui:

a) A versão do protocolo suportado pelo iniciador.
b) O nome do alvo e do iniciador iSCSI.
c) Os identificadores de conexão, do iniciador e do alvo.
d) O estágio de negociação em que o iniciador está pronto à entrar.

Entre um nodo iniciador iSCSI e um alvo iSCSI, definido por um *TargetName* e um *Target Portal Group Tag* iSCSI, os resultados do *Login* são definidos como mostrado na tabela 3.3:

ISID	TSIH	CID	Target action
new	non-zero	any	fail the login ("session does not exist")
new	zero	any	instantiate a new session
existing	zero	any	do session reinstatement
existing	non-zero existing	new	add a new connection to the session
existing	non-zero existing	existing	do connection reinstatement
existing	non-zero new	any	fail the login ("session does not exist")

Tabela 3.3: Resultados do *Login.*

A determinação de “existente” ou “novo” é feita pelo alvo.

3.4.2.2 Negociação de Segurança iSCSI

A negociação de segurança iSCSI é feita através de trocas de mensagens que configura um mecanismo de segurança e autenticam o iniciador com o alvo. A troca procede de acordo com o método de autenticação escolhido na fase de negociação e é conduzido através de parâmetros *key=value* de Requisições e Respostas de *Login*.

3.4.2.3 Negociação de Parâmetros Operacionais Durante a Fase de *Login*

A Negociação de Parâmetros Operacionais Durante a Fase de *Login* pode ser feita:

I. Começando com a primeira Requisição de *Login*, caso o iniciador não proponha nenhuma opção de integridade/segurança.
II. Começando imediatamente após a negociação da segurança, caso o iniciador e o alvo tenham feito tal negociação.

Uma sessão está operacional quando ela tem pelo menos uma conexão na *FullFeaturePhase*. Conexão novas ou sobressalentes só podem ser adicionadas à uma sessão após ela estar operacional.

3.4.2.4 Restabelecimento de Conexão

O restabelecimento de uma conexão é, em termos práticos, equivalente à uma nova conexão de *login* (em adição ao logout implícito da conexão velha).

3.4.2.5 Restabelecimento da Sessão, Fechamento e Tempo de Expiração

O restabelecimento da sessão é o processo de o iniciador relogar com seu identificador que está possivelmente ativo na perspectiva do alvo. Ou seja, o iniciador faria o *log out* da sessão que corresponde ao seu identificador e restabeleceria uma nova sessão iSCSI no seu lugar (com o mesmo identificador). Para sinalizar o restabelecimento da sessão a etiqueta do alvo, com o nome do iniciador na UDP *Login*, deve ter seu valor igual a zero. O restabelecimento da sessão causa a todas as tarefas que estavam ativas na sessão antiga a serem terminadas imediatamente pelo alvo, sem dar qualquer aviso ao iniciador [3].

O estado da sessão de um iniciador deve ser *FAILED* quando o mesmo tenta fazer o restabelecimento de uma sessão.

O fechamento de uma sessão é um evento definido como um dos seguintes:

a) Um *logout* “fecha sessão” feito com sucesso.

b) Um *logout* “fecha conexão” feito com sucesso para a última fase de conexão da Fase Completa e quando não há nenhuma outra conexão na sessão esperando por limpeza e não há tarefas na sessão a espera de re-atribuição.

O tempo para uma sessão expirar é um evento definido para ocorrer quando o tempo para expirar do último estado de conexão expira e não há mais tarefas à espera de re-atribuição. Isto leva a sessão a ter seu estado LIVRE.

A camada iSCSI provê à camada SCSI com a notificação “*I_T nexus loss*” quando um dos seguintes eventos acontece:

a) Restabelecimento da sessão completada com sucesso.
b) Evento de fechamento de sessão.
c) Evento de tempo expirado de uma sessão.

3.4.2.6 Continuação de uma Sessão e Falha

A continuação de uma sessão é o processo no qual o estado de uma sessão já existente continuar a ser usado por um restabelecimento de conexão, ou pela adição de uma conexão com um novo identificador de conexão. Essas duas conexões associam a nova conexão de transporte com o estado da sessão.

A falha da sessão é um evento aonde a conexão de Fase Completa alcança um estado de *CLEANUP_WAIT*, ou completa com sucesso um *logout* de recuperação, assim causando à todas as tarefas ativas que comecem a esperar pela re-atribuição da tarefa.

3.4.3 Negociação de Parâmetro Operacional Fora da Fase de *Login*

A negociação de parâmetro na Fase Completa é feita através de requisições e respostas de Texto. Essas negociações podem envolver várias trocas de Texto, que são sempre inicializados pelo iniciador através do uso da mesma etiqueta de tarefa do

iniciador. Setando o *bit* F em 1, o iniciador está indicando sua intenção em terminar a negociação e, na última resposta, o alvo seta o *bit* F em 1.

Se o alvo responde à uma requisição de Texto com o *bit* F setado em 1 e uma resposta de Texto com o *bit* F setado em 0, o iniciador deve continuar enviando requisições de Texto (mesmo que vazias) com o *bit* F setado em 1, enquanto ele ainda quiser terminar a negociação, até que receba uma resposta de Text com o *bit* F setado em 1. É desencorajado responder uma requisição de Texto com o *bit* F setado em 1 com uma resposta vazia com o *bit* F setado em 0.

Em uma resposta de Texto com o *bit* F setado em 1 os alvos não devem submeter parâmetros que requerem um requisição de Texto adicional do iniciador.

Em uma seqüência de negociação, as configurações do *bit* F em um par de requisições-respostas de Texto não têm relação com as configurações do próximo par.

Sempre que um alvo responder com o *bit* F setado em 0, a Etiqueta de Transferência do Alvo deve ser configurada com um valor diferente do seu valor padrão, 0xffffffff.

Um iniciador pode recomeçar uma negociação de parâmetro operacial através da emissão de requisições de Texto junto da Etiqueta de Transferência do Alvo, setada com o valor 0xffffffff, após receber uma resposta com a Etiqueta de Transferência do Alvo com um valor diferente de 0xffffffff. Um alvo pode recomeçar uma negociação de parâmetro operacional respondendo uma requisição de Texto com uma UDP *Reject*.

Parâmetros negociados por uma seqüência de negociação de texto só se tornam efetivos após esta negociação ter sido completada.

3.5 Recuperação e Manipulação de Erros iSCSI

Os principais objetivos do esquema de recuperação de erro iSCSI são:

a) Permitir que implementações iSCSI se adequem a diferentes requisições através da definição de uma coleção de mecanismos de recuperação de erro que as implementações podem escolher.
b) Assegurar a interoperabilidade entre quaisquer duas implementações que possuem conjuntos diferentes de capacidades de recuperação de erro.

c) Definir os mecanismos de recuperação de erro que assegurem a ordenação de comandos, mesmo que possam ocorrer erros, para iniciadores que exigem ordenação.
d) Quem não sejam feitas adições no caminho rápido, mas que permida complexidade moderada no caminho de recuperação de erro.
e) Prevenir que o iniciador e o alvo tentem recuperar o mesmo conjunto de UDPs ao mesmo tempo.

3.5.1 Classes de Recuperação

O iSCSI permite as seguintes classes de recuperação (em ordem crescente de escopo de tarefas iSCSI afetadas):

a) Dentro de um comando (sem que um comando *restart* seja requerido).
b) Dentro de uma conexão (sem que a conexão tenha que ser reconstruída, talvez só necessitando de um comando *restart*).
c) Recuperação de uma conexão (talvez requerendo que conexões sejam reconstruídas e que os comandos sejam emitidos denovo).
d) Recuperação de sessão.

Em todos os casos, o implementador tem a escolha de deferir os erros para o iniciador SCSI (com o código de resposta apropriado), neste caso a tarefa, se existir alguma, tem que ser removida do alvo e todos os seus efeitos colaterais devem ser considerados.

O uso de classes de recuperação dentro da conexão e dentro do comando não devem ser tentados antes de a conexão estar em sua Fase Completa.

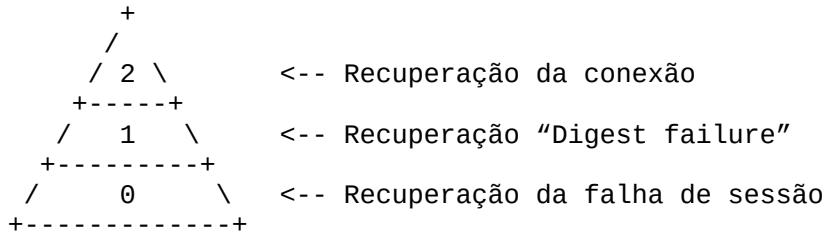
3.5.2 Hierarquia de Recuperação de Erro

As classes de recuperação de erro descritas até agora estão organizadas em uma hierarquia para facilitar seu entendimento e para limitar a complexidade da

implementação. É mais fácil alcançar a interoperabilidade com alguns níveis de recuperação bem definidos. Os atributos desta hierarquia são os seguintes:

a) Cada nível é um super conjunto das capacidade do nível anterior.
b) Como um corolário, suportando um maior nível de recuperação de erro significa uma sofisticação aumentada e possívelmente um aumento nas necessidades de recursos.
c) O suporte ao nível de recuperação de erro “n” é mostrado e negociado por cada entidade iSCSI através da troca de chaves de texto “*ErrorRecoveryLevel=n*”. O menor dos dois valores trocados é o nível de recuperação de erro operacional para a sessão.

O diagrama a seguir representa a hierarquia de recuperação de erro.



A tabela abaixo lista as capacidades de recuperação de erros esperadas das implementações que suportam cada nível de recuperação de erro.

ErrorRecoveryLevel	Capacidades associadas de recuperação de erro
0	Classe de recuperação de sessão
1	Recuperação de falha de consolidação (*digest*)* mais as capacidades do nível 0
2	Classe de recuperação de conexão mais as capacidades do nível 1.

Tabela 3.4: Capacidades de recuperação de erros que as implementações suportam.

(* A recuperação de falha *digest* é composta de duas classes de recuperação: classe de recuperação dentro da conexão e dentro do comando).

Quando um valor definido de "*ErrorRecoveryLevel*" é proposto por um originador em uma negociação de texto, ele deve suportar a funcionalidade definida para o valor proposto e, adicionalmente, suportar a funcionalidade correspondente a qualquer valor numérico definido que seja menor que o proposto. Quando um valor definido de "*ErrorRecoveryLevel*" é retornado por um respondedor em uma negociação de texto, ele deve suportar a funcionalidade correspondente ao "*ErrorRecoveryLevel*" que ele está aceitando.

Quando ambos os lados tentam usar a funcionalidade de recuperação de erro além do que foi negociado, esta tentativa pode falhar, a menos que tenha ocorrido um acordo prévio entre os dois lados que dê suporte a esta tentativa.

As implementações iSCSI devem suportar nível de recuperação de erro "0", enquanto que o resto é opcional para se implementar. Em termos de implementação a estriação acima significa que seguinte sofisticação incremental é requerida com cada nível.

Nível de transição	Requerimento de incremento
0 → 1	Retransmissões de UDP na mesma conexão
1 → 2	Retransmissão através das conexões e nova atribuição de aliança

Tabela 3.5: requerimento de incremento das transições de nível.

3.5.3 Uso do "*Retry*" (nova tentativa)

Um iniciador tenta "plugar" (o que ele acredita ser) as descontinuidades na ordenação *CmdSN* no final do alvo enviando o mesmo comando UDP iSCSI ("*retry*") na ausência de um reconhecimento de comando (através de uma atualização *ExpCmdSN*) ou resposta. Comandos UDPs descartados, devido a erros *digest*, podem ter criado estas descontinuidade.

Um *Retry* não deve ser usado senão para plugar espaços de seqüências de comandos, e em particular, não pode ser usado para requerer retransmissões de UDP do alvo.

Quando uma nova tentativa de um comando iSCSI é feita, o comando UDP deve carregar a Etiqueta original de Tarefa do Iniciador e os atributos operacionais originais, bem como o *CmdSN* original. O comando que está sofrendo a nova tentativa deve ser enviado na mesma conexão que o comando original, a não ser que esta conexão já tenha sido desconectada (logged out) com sucesso.

3.5.4 Nova Atribuição de Lealdade

Através da emissão de uma requisição de gerência de tarefa “nova atribuição de tarefa” o iniciador sinaliza a sua intenção para continuar um comando já ativo (mas sem lealdade de conexão) como parte de uma recuperação de conexão. Isto significa que uma nova lealdade de conexão é requerida para o comando, e este procura se associar à conexão na qual a requisição de gerenciamento de tarefa está sendo emitida. Antes que a tentativa de uma nova atribuição de lealdade seja feita para a tarefa, um *Logout* implícito ou explícito com o código “remova a conexão para a recuperação” tenha sido completado para a conexão anterior que a tarefa era leal.

O suporte de uma nova atribuição de lealdade para os alvos é opcional. Ela é negociada através de uma chave de texto *ErrorRecoveryLevel* durante o período de *login*. Quando um alvo não suporta a nova atribuição de lealdade ele deve responder com um código de resposta do gerenciador de tarefas “Nova atribuição de lealdade não suportada”. Se a nova atribuição de lealdade não é suportada pelo alvo, mas a tarefa permanece leal a uma conexão diferente ou não foi feita uma recuperação de *Logout* de uma conexão leal anterior, o alvo deve responder com o código de resposta do gerenciador de tarefas “Tarefa ainda leal”.

Se uma nova atribuição de lealdade é suportada pelo alvo, a resposta do gerenciador de tarefas para a requisição de nova atribuição deve ser emitida antes que a nova atribuição se torne efetiva.

Se uma nova atribuição em um comando SCSI que envolve entrada de dados é feita, qualquer etiqueta *SNACK* que ele carrega consigo para uma resposta final da conexão original é deletada e o valor padrão “0” deve ser usado. [3]

3.5.5 Uso da UDP *Reject* na Recuperação

Alvos não devem terminar uma tarefa ativa com o envio de uma UDP *Reject* para qualquer troca de UDP durante a vida desta tarefa. Se um alvo decide terminar uma tarefa, uma UDP resposta (SCSI, Texto, Tarefa, etc.) deve ser retornada pelo alvo para que a tarefa seja concluída. Se a tarefa nunca tivesse estado ativa anteriormente a um *Reject*, os alvos não deveriam enviar respostas pois o próprio comando estaria sendo descartado.

A regra acima significa que o iniciador pode eventualmente esperar uma resposta em *Rejects* recebidos, se o *Reject* recebido for para uma UDP exceto o próprio comando UDP. Os *Rejects* não-comando só tem valor de diagnóstico em fazer o registro de erros, e podem ser usados pelos iniciadores para decisões de retransmissão.

3.5.6 Gerenciamento do Intervalo de Parada de uma Conexão

O iSCSI define dois valores de Intervalo de Parada globais na sessão (em segundos) – *Time2Wait* e *Time2Retain* – que são aplicáveis quando uma conexão iSCSI de Fase Completa é tirada de serviço intencionalmente ou por uma exceção. *Time2Wait* é a “pausa” antes de tentar fazer o *Logout* explícito/implícito para o identificador da conexão em questão ou a nova atribuição de tarefa para as tarefas afetadas (se existir alguma). *Time2Retain* é o tempo máximo após a pausa inicial que a tarefa e/ou estado(s) de conexão é/são garantidos em ser mantidos no alvo para servir para uma possível tentativa de recuperação. Tentativas de recuperação para a conexão ou tarefa não devem ser feitas antes dos segundos *Time2Wait*, mas devem ser completadas dentro de *Time2Retain* segundos. [3]

3.5.7 Terminação Implícita de Tarefas

Um alvo pode terminar implicitamente as tarefas ativas devido as dinâmicas do protocolo iSCSI como nos seguintes casos:

a) Quando uma conexão é implícita ou explicitamente deslogada, com o código de razão “Fechar a conexão” e quando há tarefas ativas leais a aquela conexão.
b) Quando uma conexão falha e o estado da conexão expira e quando há tarefas ativas leais a aquela conexão.
c) Quando um *Logout* é feito com sucesso com o código de razão “remover a conexão para recuperação” enquanto existem tarefas ativas leais a aquela conexão, e para aquelas tarefas que eventualmente tem seu tempo expirado após períodos *Time2Wait* e *Time2Retain* sem nova atribuição de lealdade.
d) Quando uma conexão é deslogada implícita ou explicitamente com o código de razão “Fechar a sessão” e existem tarefas ativas nesta sessão.

Se as tarefas terminadas nos quatro casos acima são tarefas SCSI, elas devem ser terminadas internamente como *CHECK CONDITION status.* Este *status* só tem significado para a manipulação apropriada do estado interno SCSI e seus efeitos colaterais em relação à ordenação por que este *status* nunca é comunicado devolta como um *status* de terminação para o iniciador. Porém ações adicionais podem vir a terem que ser tomadas no nível SCSI dependendo do contexto SCSI definido nos seus padrões, ver [SAM2] e [SPC3].

3.5.8 Erros de Formatação

As duas violações explícitas das regras de *layout* UDP à seguir são erros de formatação:

a) Conteúdos ilegais em qualquer campo do cabeçalho UDP exceto o *Opcode*.
b) Conteúdos inconsistentes nos campos.

Erros de formatação indicam uma grande falha de implementação em um dos lados. Quando um alvo ou um iniciador recebe uma UDP iSCSI com um erro de formatação ele deve terminar imediatamente todas as conexões de transporte de uma sessão, com um fechamento de conexão ou com um reajuste (*reset*) da conexão, e deve escalar (mandar) o erro de formatação para a recuperação da sessão.

3.5.9 Erros de Condensação (*Digest*)

Quando um alvo ou iniciador recebe qualquer UDP iSCSI com um erro de condensação de cabeçalho ele deve tanto descartar o cabeçalho como todos os dados recebidos até o começo da última UDP ou então fechar a conexão. Por o erro de condensação indicar que o tamanho do campo do cabeçalho possa estar corrompido, a localização do começo da última UDP precisa ser determinado com segurança por outros meios, como as operações da camada de sincronia e manobrabilidade.

Quando um alvo recebe qualquer UDP iSCSI com um erro de condensação no seu conteúdo ele deve responder com uma UDP *Reject*, com código de razão *Data-Digest-Error*, e descartar esta UDP.

Se esta UDP descartada é uma UDP *data* iSCSI solicitada ou não solicitada o alvo deve fazer o seguinte:

a) Pedir a retransmissão com uma recuperação *R2T*.

b) Terminar com um *CHECK CONDITION Status* e uma condição de “erro de CRC do serviço do protocolo”.

Se a UDP for *non-data* é descartada, não é necessário nenhuma ação a mais para os alvos. Em caso da presença de dados imediatos em um comando descartado, o dado imediato é recuperado implicitamente quando a tarefa é tentada denovo, seguida de uma transferência de dados inteira para a tarefa.

O iniciador deve descartar a UDP quando ele recebe qualquer UDP iSCSI com um erro de condensação no seu conteúdo.

3.5.10 Erros de Seqüência

Quando um iniciador recebe UDP *R2T/data* iSCSI fora de ordem ou uma resposta SCSI com uma UDP *data* que está faltando, significa que o iniciador detectou um erro de condensação em uma ou mais UDPs anteriores. Estes erros são tradados pelo iniciador como descrito na seção de erros de condensação.

3.5.11 Outros Erros

A seguir estão listados outros erros que podem ocorrer e que não serão aprofundados:

a) Tempo Expirado SCSI.
b) Falhas de Negociação.
c) Erros de Protocolo.
d) Falhas de Conexão.
e) Erros de Sessão.

CAPÍTULO IV

4. Ambiente de Testes

O projetista que pretende trabalhar com o armazenamento de dados distribuído não deve se preocupar apenas em integrar SAN com NAS, mas deve também montar uma estratégia de gerenciamento no trâmite de dados e em toda a infra-estrutura. O objetivo é oferecer hardware, software e serviços.

O hardware não é a principal dificuldade em uma rede de armazenamento distribuído. Graças a algumas importantes inovações, redução gradual dos preços e a iniciativa de fabricantes, a tecnologia SAN tem estado mais acessível às pequenas e médias empresas.

A consultoria, serviços de implementação e o software correspondem de 70% a 90% da demanda no desenvolvimento de uma SAN. Com o hardware disponível, inicia-se o serviço de mapeamento, definição da política e processos de armazenamento até a recuperação da informação.

A tecnologia iSCSI traz para a SAN um novo perfil de cliente. Utilizando placas HBA dentro do servidor, fibra óptica e switch, seus componentes tecnológicos chegam a custar quase US$ 3mil. Esta configuração é recomendada para serviços críticos. Uma opção mais barata seria usar um gateway conectado à rede Ethernet com SAN. O servidor comunica-se via rede e a caixa converte para fibra óptica. Neste caso, o cliente só precisa investir em placas, tornando mais fácil o acesso à tecnologia de armazenamento distribuído.

O tráfego de dados dentro de uma rede IP pode utilizar o protocolo Fibre Channel IP, que utiliza espelhamento remoto de dados, substituindo assim o uso da fibra óptica, cujo valor é mais alto, pelo espelhamento de link a baixo custo.

O iSCSI vai proporcionar mais volume, pois a Microsoft homologou essa tecnologia para rodar no seu Exchange, suportando armazenamento de e-mails no ambiente, o que vai viabilizar aumento de armazenamento on line desse tipo de arquivo.

A HP lançou uma linha de serviços de armazenamento distribuídos com base em ATA, tecnologia melhorada oriunda dos antigos discos IDE, para máquinas pequenas. A vantagem da ATA é o armazenamento de informações com acesso não muito freqüente.

O projetista deve analisar quando e como usar determinada tecnologia. Há mais de 20 anos, o SCSI foi criado como protocolo padrão de comunicação entre desktops e periféricos. Com o passar do tempo surgiram os protocolos Fibre Channel – que não é recente nas SANs – e o SATA (Serial ATA), é mais veloz que o ATA paralelo. É importante salientar que o ATA entra em backup de disco para disco e, hoje, o processo é o disco para fita. O período do restore de um volume de dados é no mínimo o dobro do tempo para fazer o backup. Para agilizar, em um segundo estágio, pode-se colocar outro disco mais acessível, baseado em ATA, elemento interno de backup, diminuindo assim o custo do disco array.

SANs tradicionais têm sido configuradas usando-se tecnologia Fibre Channel, que é um protocolo de rede serial onde a maior parte da manipulação do protocolo é feita em hardware. O Fibre Channel supera as limitações do SCSI-3 em temos de máximo comprimento dos cabos, e pode ser estendida para mais de 10 km usando cabos ópticos padrões single-mode. A maioria dos Fibre Channel utilizados hoje no mercado operam entre 1 Gbps e 2 Gbps.

Recentemente, alternativas para SANs Fibre Channel baseadas em tecnologia IP começaram a aparecer. Uma das mais interessantes desta alternativas é o iSCSI, como já citado anteriormente, que possibilitam transferência SCSI (block-level) através de redes TCP/IP. O servidor em uma rede iSCSI é chamado iniciador iSCSI e guarda dispositivos que agem como alvos iSCSI. Os alvos iSCSI e o iniciador podem ser conectados usando adaptadores Ethernet comuns, possibilitando soluções a baixo custo, visto que o custo por porta Ethernet em switch é atualmente muito menor que o custo por porta de cabos de fibra óptica. Se NICs comuns são usados em conjunto com um software iniciado iSCSI, a conectividade do servidor é oferecida a um custo muito menor usando iSCSI ao invés de Fibre Channel.

O problema em se utilizar uma rede TCP/IP para tráfego no armazenamento de dados é que TCP/IP foi desenvolvido para redes inseguras, com grande perda de pacotes, necessitando, assim, de muito processamento para dar suporte ao monitoramento dos dados. Já o Fibre Channel foi desenvolvido como uma rede com baixa taxa de erros, resultando em uma penalidade mínima por processamento no monitoramento dos dados.

Com o mercado todo voltado nestas tecnologias concorrentes de Storage Area Network, Fibre Channel e iSCSI, muitas organizações se sentirão forçadas a tomar uma decisão sobre qual tecnologia usar, a menos que combinem as duas em

uma única rede. A possibilidade de escolha das SANs hoje em dia, confunde principalmente nos dispositivos de rede, como placas e switches, pois muitos dispositivos de armazenamento ainda falham em algumas funções nativas do iSCSI, fazendo necessária uma bridge iSCSI-FibreChannel chamada de Roteador de Armazenamento. Este trabalho investiga a performance relativa ao Servidor da SAN, mencionando as diferenças ao se comparar Ethernet NICs com HBA Fibre Channel em diferentes tipos de tráfegos de dados. No teste estão inclusos três diferentes placas de rede, uma Fast Ethernet, uma Gigabit Ethernet e uma Gigabit Ethernet com TCP offloading feito em hardware, conhecido como TNIC ou NIC com TCPOE. Na parte de Fibre Channel foi usado um HBA de 2Gbps.

O teste de performance deste trabalho envolve análises dos adaptadores de rede e do HBA para um número de diferentes tráfegos de dados emulando aplicações de um servidor web, um servidor de arquivos e um banco de dados. Um número de características de performance foram consideradas incluindo largura de banda, latência e utilização da CPU. [19]

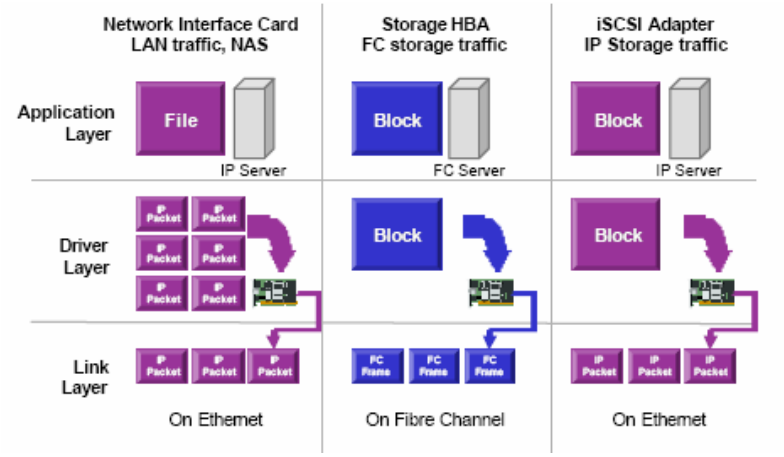


Figura 4.1: Camadas para os diferentes tipos de placas adaptadoras e HBA.

4.1. Configurações para o teste

4.1.1 Hardware

A seguinte máquina foi usada no teste:

- Servidor Dell PowerEdge 1600SC com um processador Pentium Xeon 2.4GHz e 512MB de memória RAM

- Emulex LP9002 2Gbps Fibre Channel HBA
- Alacritech 1000x1 Gigabit Ethernet Base-T TNIC com TCP Offloading
- Intel Pro/1000 MT Gigabit Ethernet Base-T NIC
- Intel Pro/100 B Fast Ethernet Base-T NIC
- Cisco 5428 Storage Router com oito 2Gbps portas Fibre Channel e duas Gigabit Ethernet Ports
- Cisco 3550-12T Gigabit Ethernet Switch com doze Gigabit Ethernet Ports
- Hitachi Data Systems 9570V com dezesseis Hitachi DK32DJ-72FC Fibre Channel Disks (72GB, 10000rpm) em uma configuração RAID-0 usando 10GB LUNs

O Emulex Fibre Channel HBA, o Alacritech NIC, e o Intel Pro/100 NIC são todas placas PCI colocadas no servidor Dell durante todo o teste. A placa de rede Intel Pro/1000 era integrada na placa mãe do servidor. A configuração do Fibre Channel era uma conxão ponto-a-ponto direto do servidor para o dispositivo de armazenamento, como mostra a Figura 1. Tanto o Fibre Channel HBA e o Hitachi Data Systems 9570V são do tipo 2GBps, isto é, são capazes de transferir taxas maiores que 200MB/segundo. Alguns teste com o Fibre Channel foram no entanto executados limitando-se a velocidade para 1Gbps para assegurar uma velocidade referente a iSCSI através da Gigabit Ethernet, que tem uma taxa máxima de 1,25 gigabauds e transferência máxima perto de 125MB/segundo.

Figura 4.2: Configuração ponto-a-ponto usados nos testes com FC.

Como o dispositivo de armazenamento Hitachi Data Systems 9570V não contém uma interface iSCSI nativa, um roteador de armazenamento Cisco 5428 foi

usado para ligar o iSCSI e o Fibre Channel. Junto do roteador de armazenamento, um *switch* Cisco 3550-12T foi usado entre os dispositivos de rede do servidor e o roteador de armazenamento, devida a requisição do roteador de uma interface Base-F, enquanto todos os dispositivos de rede são Base-T. A disposição usada para o teste do iSCSI é mostrada na figura 2. Tanto a placa Fast Ethernet quanto a Gigabit Ethernet são dispositivos de rede padrões com processamento TCP feito por software no servidor. E finalmente, a placa Alacritech emula o processamento TCP usando ASIC no dispositivo.

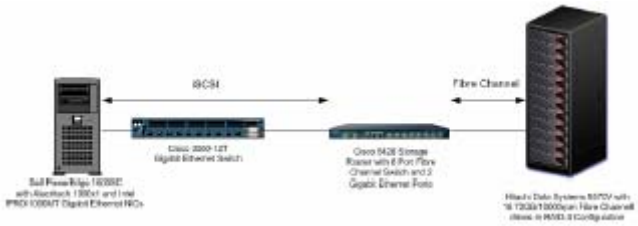


Figura 4.3: Configuração do teste iSCSI usando um roteador de armazenamento Cisco para fazer uma ponte entre iSCSI e FC.

Em todos os testes, para assegurar que a largura de banda na parte de armazenamento não afetaria os resultados, o HDS 9570V foi internamente configurado com uma instalação RAID-0 usando 16 discos Fibre Channel 10000rpm dividindo os dados em 16 drives para assegurar uma performance excelente. Desde que o switch Cisco Gigabit Ethernet é totalmente non-blocking, e o roteador de armazenamento pode assegurar um troughput de 120MB/s no roteamento iSCSI-para-FC, a rede não possuirá um gargalo. Entretanto, se um grande número de operações de entrada e saída por segundo forem requeridas, a comunicação iSCSI-para-Fibre Channel no roteador de armazenamento pode limitar a performance como o número de transações crescer rapidamente. [19]

4.1.2. Software

O servidor rodou o Microsoft Windows Server 2003 Enterprise Edition, atualizado com todos os patches aplicáveis de 25 de Outubro de 2003. O iniciador iSCSI padrão da Microsoft incluído no Windows Server 2003 foi usado para ligar o

tráfico iSCSI do servidor para o dispositivo de armazenamento. O software iSCSI iniciador suporta o uso de iSNS (Internet Storage Name Server) para descoberta de dispositivos e servidor de nomes, desde que o teste envolva apenas um único disco de sistema. O cache de escrita dos arquivos de sistema foi desligado, mas a escrita em cachê do dispositivo de armazenamento continuou com tamanho de 1GB por controlador. Em testes anteriores, a escrita em cachê dos arquivos de sistema se mostrou performance positiva em certos tipos de arquivos, mas como nosso interesse era testar a performance do NIC por si só foi decidido desligar bem como usar um tamanho de arquivo muito grande.

Para a geração do tráfego, foi usada a versão 2003.05.10 do IOMeter para plataforma Win32. IOMeter é uma ferramenta sintética de benchmark originalmente desenvolvida pela Intel para medir a performance dos subsistemas de entrada e saída. IOMeter assegura tanto a geração do workload quanto a medida de performance. O workload pode ser modificado para satisfazer certos tipos de aplicações ajustando através de outros tipos de tamanhos para transferência, seqüências de leitura e escrita, porcentagem de acessos seqüenciais e aleatórios, números de entradas e saídas concorrentes. Existe um grande número de diferentes significados, mas o foco deste trabalho é a performance dos HBAs e dos dispositivos de rede, o objetivo é a análise de toda a largura de banda, as operações de entrada e saída por segundo, a média na latência nas transações, a utilização da CPU tanto quanto muitos outros valores considerados para uma boa performance. É recomendado um worker por processador no cliente IOMeter, como o servidor usado contém um único CPU, um único worker foi usado em todos os teste.

O número médio de operações de entrada e saída por segundo é uma medida do número de transações que são executadas por segundo. Não foi usado nenhum reconhecimento de leitura/escrita ou resposta nos testes, no entanto as transações consistem apenas em requisições de transação e operações de leitura/escrita. A média de largura de banda agregada indica a taxa de transferência da aplicação, e é incrementada com grande quantidade de transferências. A média do tempo de resposta demonstra a média de latência de cada operação. Quando há um grande número de operação de entrada e saída, o valor é acrescido devido a espera na fila antes da sua execução. Operações seqüenciais levam a uma largura de banda muito grande e latência menor que operações aleatórias, operações como tempo de procura no acesso aos dados quando um drive é adicionado. Finalmente, a utilização do CPU demonstra

o quanto intensivo foi o uso da CPU durante a transferência de dados. Se uma grande quantidade de processamento é feito em hardware na placa onde o protocolo é direcionado, os valores da utilização do CPU são baixos.

Um acréscimo de tempo pode ser usado para demonstrar que o sistema possui uma carga característica no início da medição de desempenho usada na computação estatística. Todos os testes usaram um tempo de 15 segundos para assegurar que a medida dos valores não foi afetada pelo início das transferências, exceto para os testes de leitura/escrita de 32MB onde um tempo de inicialização de 30 segundos foi usado devido a grande quantidade de dados requeridos enquanto um grande número de operações de entrada e saída eram usadas. Cada teste foi rodado por um minuto, e vários deles foram rodados várias vezes para assegurar que o resultado obtido foi consistente. Uma pequena variação pode ser vista nos testes, desde que alguns *workloads* contêm certa quantidade de variações, mas as diferenças encontradas mostraram-se menor que 1%. [19]

4.1.3 Definição de Workloads

Oito diferentes *workloads* forma usados para o IOMeter ao analisar a performance relativa do iSCSI e Fibre Channel. Para todos os *workloads*, o atraso na transferência foi colocado em 0 e tamanho do disparo em 1. Todas as entradas e saídas foram alinhadas na fronteiras do setor, e o tamanho do arquivo para teste foi de 1GB para assegurar que a memória e o cache do servidor não afetariam a performance. Os *workloads* usados podem ser divididos em *workloads* gerais representando aplicações típicas e *file-centric workloads* para pequenas (512B-2kB) e grandes (32MB) transferências de arquivos. Os vários *workloads* são detalhados na Tabela 1 onde os fatores de leitura/escrita, fatores aleatórios/seqüenciais, e os tamanhos de transferência dos *workloads* são indicados.

Type of Workload	Transfer Size (B)	% of size	% reads	% random
OLTP	8192	100	67	100
Workstation	8192	100	80	80
File Server	512	10	80	100
	1024	5	80	100
	2048	5	80	100
	4096	60	80	100
	8192	2	80	100
	16384	4	80	100
	32768	4	80	100
	65536	10	80	100
Web Server	512	22	100	100
	1024	15	100	100
	2048	8	100	100
	4096	23	100	100
	8192	15	100	100
	16384	2	100	100
	32768	6	100	100
	65536	7	100	100
	131072	1	100	100
	524288	1	100	100
512B Random Read	512	100	100	100
2kB Random Read	2048	100	100	100
32 MB Seq. Read	33554432	100	100	0
32 MB Seq. Write	33554432	100	0	0

Figura 4.4: Cargas de trabalho usados na caracterização da performance do *IOMeter*.

4.1.4 Processamento de Transações *Online* (OLTP)

OLTP envolve uma gama de leituras e escritas de igual tamanho (geralmente 4kB ou 8kB). Como o acesso de leitura e escrita é em um banco de dados no disco, o acesso aos registros é completamente aleatório e não seqüencial. Bancos de Dados são

geralmente muito escritas, pois grande maioria da informação é modificada no servidor e escrita em disco. Um grande número de maquias clientes compartilham o banco de dados através do servidor de dados, gerando um número cada vez maior de operações de entrada e saída no servidor.

4.1.5. Estações de Trabalho (*Workstation*)

As estações de trabalho do workload são baseadas em um cluster de 8kB de tamanho, onde a grande maioria das operações são de acesso do tipo aleatório. As estações de trabalho assumirão ocasionalmente quantidade do swap conduzindo a uma pequena quantidade de acesso seqüencial, visto que a probabilidade de se transferir grande quantidade de dados é pequena.

4.1.6. Servidor de Arquivo (*File Server*)

A principal função do servidor é disponibilizar os arquivos para os clientes da rede local. Arquivos que variam em tipo e tamanho, e os testes padrões do tamanho de transferência são completamente amplos em sua definição.

4.1.7. Servidor *Web*

Um servidor web disponibiliza arquivos para clientes localizados na Intranet ou na Internet. A maioria da informação armazenada consiste em pequenos arquivos de texto ou pequenas imagens, mas ocasionalmente, grandes arquivos ou multimídia também são hospedados no servidor.

4.2 Resultados dos Testes

4.2.1. *Workloads* Gerais

O primeiro teste envolveu processamento dos *workloads* emulando aplicativos comuns. Os *workloads* testados foram OLTP (banco de dados), um workstation workload, um workload servidor de arquivos e um workload servidor web. Cada um dos *workloads* foi testado em todos os HBAs e NICs com valores de entrada e saída de 1, 4 d 16. Cada workload foi analizado para cada NIC/HBA em termos de média de número de operações entrada/saída por segundo, média da taxa de transferência, média da latência nas transações, e média da utilização da CPU como também outras métricas de performance do IOMeter foram todas analisadas durante o processo.

O resultado para o número de operações de entrada/saída por segundo pode ser visto na figura 3. O HBA demonstrou a melhor performance (tanto para 1Gbps e 2Gbps) para todos os *workloads* nos três valores referentes à operações de entrada/saída. Para determinadas operações de entrada/saída, todos os valores ficaram dentro de 25% demonstrando que eles são relativamente comparáveis. Entretanto em quatro e dezesseis operações de entrada e saída, o HBA Fibre Channel supera em quantidade substancial os NICs. Em dezesseis operações de entrada e saída por segundo, o HBA supera a melhor perfomance do NIC em 100% e a pior performance NIC em 400%, ambos a 1Gbps e 2Gbps. Entre os NICs, o Alacritech ofereceu a melhor performance enquanto o NIC Intel Fast Ethernet foi o mais lento.

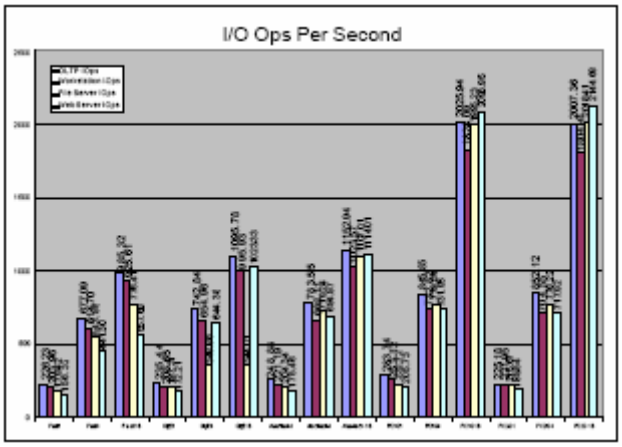


Figura 4.5: Número médio de operações de E/S por segundo para um *workload* geral.

A observação mais surpreendente foi que a NIC Intel *Gigabit Ethernet* perdeu considerável performance no workload do servidor de arquivos quando as requisições de quatro e dezesseis operações entrada/saída são usadas. Quando

analisada a performance através do tempo, os valores crescem inicialmente até estabilizar a um nível muito menor que o esperado. A razão para isso deve ser que o driver e a placa Intel são pouco otimizados para manipular a freqüente grande quantidade de transferência que ocorre no workload do servidor de arquivos.

Enquanto os quatro *workloads* foram comparados nas operações de entrada e saída por segundo em cada um dos NICs e no HBA, a diferença é maior em termos de largura de banda. O workload do servidor web, particularmente, teve taxa de transferência maior que outros *workloads* graças à sua ênfase no tamanho razoavelmente grande de transferência. Os resultados dos testes de largura de banda podem ser vistos na Figura 4.

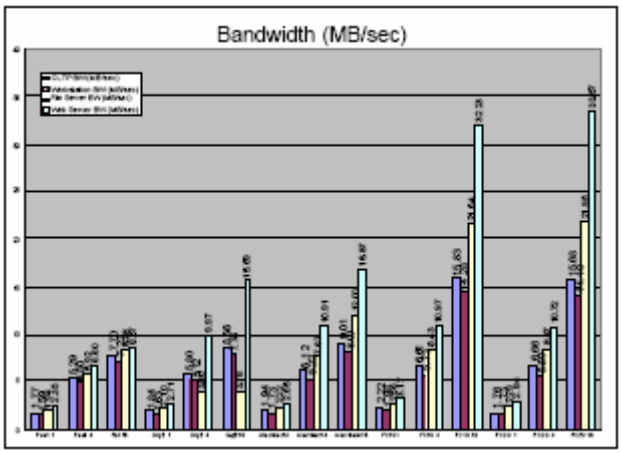


Figura 4.6: Taxa de transferência para *workloads* gerais.

Novamente o HBA teve a melhor performance rodando tanto a 1Gbps e 2Gbps. A placa Alacritech teve o melhor dos NICs, embora as diferenças só são significantes quando operados em 16 operações de entrada/saída.

Enquanto a carga era pequena, o tempo de resposta de todas as placas eram razoavelmente iguais aos mostrados na Figura 5. Entretanto, quando o número de operações de entrada e saída era incrementado a dezesseis, havia um acréscimo notável na média do tempo de resposta. Com exceção do workload do servidor de arquivos, a placa Intel Gigabit Ethernet foi a grande perdedora quando se movia grandes cargas, enquanto o HBA mais uma vez perdeu menos em tempos de performance.

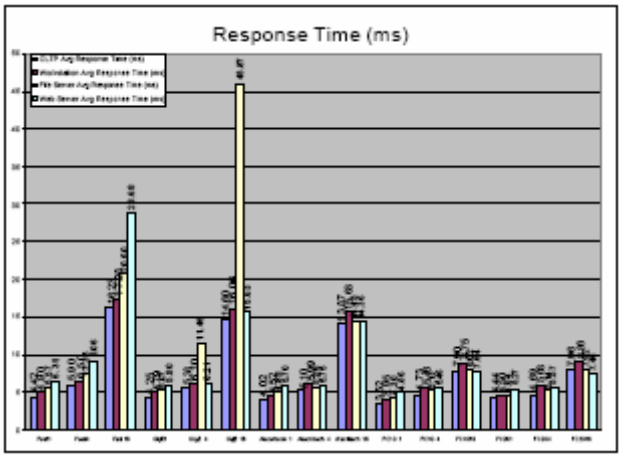


Figura 4.7: Tempo de resposta médio para *workloads* gerais.

Não surpreendentemente, o HBA Fibre Channel teve uma vantagem sobre os NICs, pois grande parte do processamento do protocolo é feito em hardware. A placa Alacritech teve também uma performance concreta provendo pequenos níveis de utilização da CPU graças aos suas capacidades TCP offloading integradas na placa. O adaptador Fast Ethernet realmente sofreu a dezesseis operações de entrada e saída por segundo com médias de utilização da CPU de até 13% para o workload do servidor web comparado a do HBA Fibre Channel de 4%. Os resultados nos testes de utilização da CPU podem ser encontrados na figura 6.

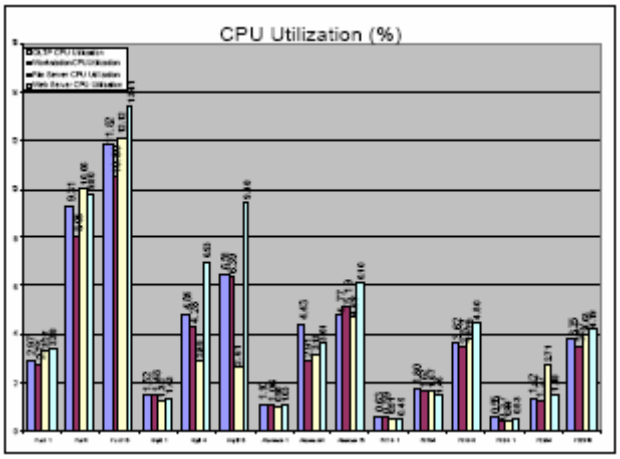


Figura 4.8: Utilização média do processador para *workloads* gerais.

4.2.2. Small Transfer Size Random Workloads

Depois de completado o primeiro conjunto de testes para *workloads* gerais, um segundo grupo de testes usará pequenos tamanhos de transferência com acessos aleatórios a blocos. Os testes de transferência de tamanhos pequenos são bons indicadores do máximo de operações de entrada e saída por segundo. O workload usado pode ser encontrado na Tabela 1. Os testes foram executados com alguns valores diferentes no número de operações de entrada e saída para promover o aumento da computabilidade.

Analisando a performance do workload para os diferentes adaptadores, uma ênfase foi dada nas operações de entrada e saída por segundo, como pode ser visto na Figura 7.

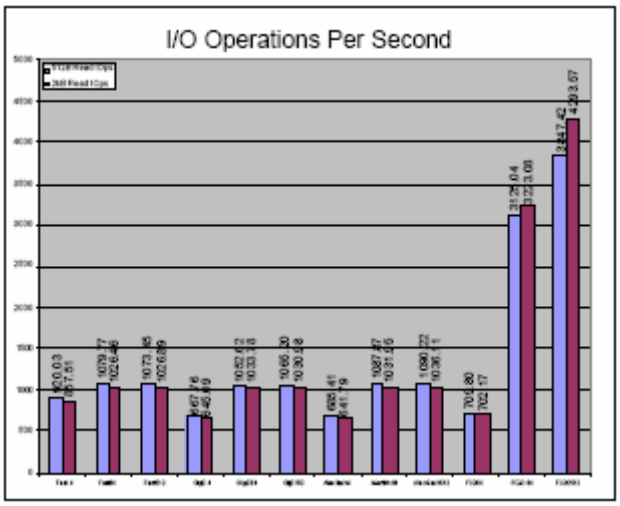


Figura 4.9: Média das operações de E/S por segundo para *workloads* aleatórios de tamanho de transferência pequeno.

A utilização da CPU foi focada em detalhes para os tipos de *workloads* que podem ter processamento intensivo. Os resultados da utilização de CPU podem ser vistos na Figura 8.

Para pequenos tamanhos de transferência com acessos aleatórios, o HBA Fibre Channe teve de longe a melhor performance em termos tanto de operações de entrada e saída por segundo quanto em largura de banda, oferecendo uma melhora de quase três vezes na performance e nas NICs quando o número de operações de entrada e saída crescia. É provável que esta diferença no parâmetro não é apenas entre dois adaptadores entre si, mas pode ser causada por limitações na ligação entre o iSCSI-para-FC e o roteador de armazenamento.

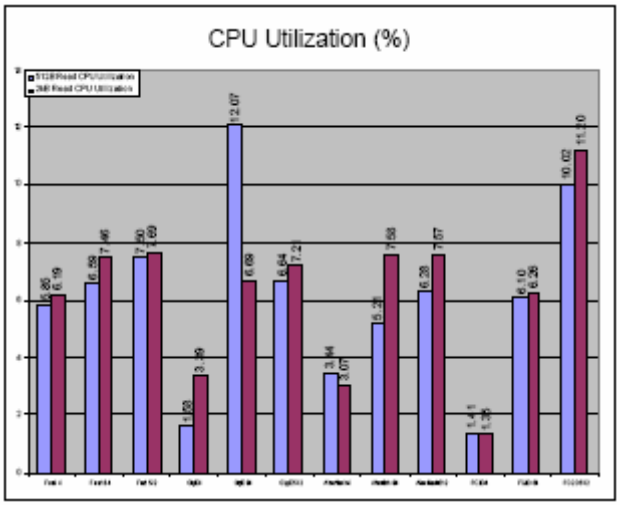


Figura 4.10: Média da utilização do processador para *workloads* aleatórios de tamanho de transferência pequeno.

O HBA foi melhor em termos de latência e utilização do CPU também, mas não com grande margem, que é compreensível devido que o número de operações de entrada e saída por segundo era muito grande para os NICs. Os Ethernet NICs eram aproximadamente igual em termos de largura de banda e número de operações de entrada e saída por segundo.

4.2.3. Large Transfer Size Sequential Workloads

Os testes com *workloads* contendo grandes transferências e acessos seqüenciais foram feitos para comparar a máxima largura de banda que diferentes adaptadores podem apresentar. Os testes foram executados tanto para leituras quanto para escritas usando vários números de operações de entrada e saída. As definições podem ser encontradas na parte inferior da Tabela 1.

Não surpreendentemente, os testes foram mais reveladores em termos de capacidade das diferentes placas adaptadoras. Os resultados das médias das taxas de transferência podem ser encontradas na Figura 9. Todos os adaptadores exceto o adaptador Intel Gigabit Ethernet ficaram dentro de 20% do seu valor teórico de pico da largura de banda quando o workload incidia para operações de leitura. As características na performance de escrita foram comparáveis às características da performance de leitura mesmo com os níveis total de largura de banda um tanto

menores. Uma vez que a placa Fibre Channel teve o maior pico no valor de largura de banda tanto na leitura quanto na escrita, ele teve a melhor performance no teste. A Intel Gigabit Ethernet não foi capaz de gerar larguras de banda perto do seu pico durante o processamento associado com a manipulação do protocolo TCP.

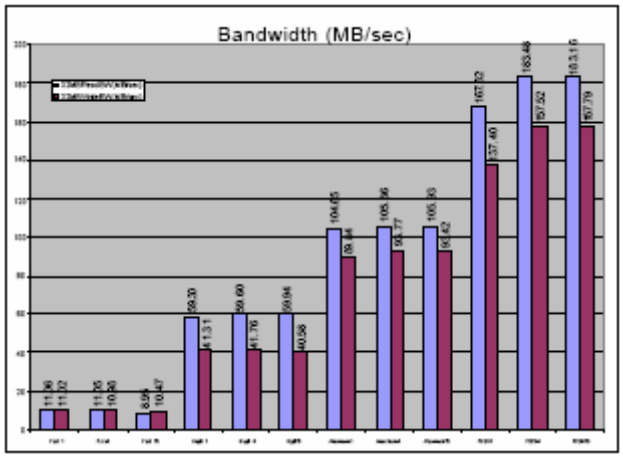


Figura 4.11: Taxa de transferência média para *workloads* seqüenciais de tamanho de transferência grande.

A utilização do CPU (que pode ser visto na figura 10) foi muito alta para o adaptador Intel Gigabit Ethernet e foi não maior que 20% para qualquer número de operações de entrada e saída. A utilização do CPU para a placa Fast Ethernet também foi substancial, enquanto o Alacritech TNIC e o HBA Fibre Channel tiveram utilização da CPU muito limitada dada pela largura de banda.

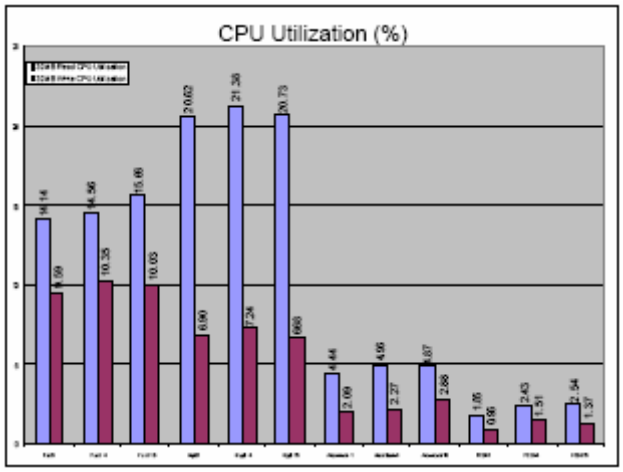


Figura 4.12: Média da utilização do processador para *workloads* seqüenciais de tamanho de transferência grande.

4.3. Projeto Linux iSCSI

O *driver* iSCSI para Linux atua como um iniciador do protocolo iSCSI para as requisições e respostas do transporte SCSI sobre um rede IP entre o cliente e um alvo iSCSI como um roteador de armazenamento Cisco SN 5428-2.

Arquiteturamente falando, o driver iSCSI se combina com a pilha TCP/IP do cliente, *drivers* de rede e placas de rede para prover as mesmas funções de um driver adaptador SCSI com um HBA (*Host Bus Adapter*) [17].

Este *driver* requer um kernel 2.4.x do Linux com versão 2.4.20 ou superior. A compilação irá requerer que os cabeçalhos dos arquivos do kernel sejam compatíveis com a versão do kernel que se deseja utilizar com o driver.

Deve-se ter um dispositivo compatível ao iSCSI conectado à rede para poder acoplar ao armazenamento. O dispositivo pode estar na mesma rede que o Linux hospedeiro ou o tráfego iSCSI pode ser roteado usando métodos normais de roteamento IP. O *daemon* e o driver do kernel estão disponíveis sob os termos GNU de Licença Pública. [17]

O driver iSCSI provê um transporte para as requisições SCSI e respostas à dispositivos de armazenamento através uma rede de IP ao invés do uso de um canal de barramento SCSI acoplado diretamente (direct attached SCSI bus) ou à uma conexão Fibre Channel. O roteador SN Série 5400 transporta essas requisições e respostas SCSI, recebidas através da rede IP, entre ele e os dispositivos de armazenamento acoplados a ele [17].

Uma vez que o driver iSCSI está instalado, a máquina hospedeira irá proceder com um processo de descoberta para dispositivos de armazenamento como a seguir:

1. O driver iSCSI faz a requisição de alvos através do mecanismo de descoberta *SendTargets* que estão configurados no arquivo de configuração /etc/iscsi.conf;
2. Cada alvo iSCSI envia nomes de alvos iSCSI disponíveis para o driver iSCSI;
3. O processo *daemon* de descoberta do driver iSCSI procura cada alvo descoberto no arquivo /etc/iscsi.bindings. Se existe uma entrada de um alvo no arquivo, o identificador correspondente do alvo SCSI é atribuído

ao alvo iSCSI. Se não existir uma entrada para o alvo, o menor identificador do alvo SCSI disponível é atribuído, e uma entrada é escrita no arquivo /etc/iscsi.bindings. O driver então envia uma requisição de login para o alvo iSCSI;

4. O alvo iSCSI aceita o login e envia os identificadores do alvo;
5. O driver iSCSI indaga os alvos por informações do dispositivo;
6. Os alvos respondem com a informação do dispositivo;
7. O driver iSCSI cria uma tabela de dispositivos alvo disponíveis.

Uma vez que a tabela está completa, os alvos iSCSI estão disponíveis para o uso por parte do hospedeiro (*host*) utilizando todos os mesmos comandos e utilidades como se fosse um dispositivo diretamente acoplado (*direct attached storage device*).[17]

CAPÍTULO V

5. CONCLUSÕES E TRABALHOS FUTUROS

O teste feito no capítulo IV mostra que os HBAs do *Fibre Channel* se adequaram melhor para tráfego de armazenamento do que para as placas Adaptadoras *Ethernet*, porém as placas adaptadoras *Ethernet* podem ser usadas para implementar uma rede iSCSI com uma penalidade mínima na performance para certos tipos de aplicações com cargas de trabalho gerais e servidores de arquivos de carga moderada, como em estações de trabalho.

Para algumas cargas de trabalho gerais não críticas, onde a carga de entrada e saída é pequena durante grande parte do tempo, a diferença entre os adaptadores, tanto na largura de banda quanto no número de operações de entrada e saída por segundo foi relativamente pequena. Notou-se que a grande penalidade na performance da implementação iSCSI foi devido o processamento do protocolo TCP/IP. Isto significa que a utilização do processador subiu muito rapidamente para as placas adapatadoras *Fast Ethernet* e *Gigabit Ethernet* quando a carga aumentava. A placa Alacritech *Gigabit Ethernet* resolveu este problema manipulando o descarregamento do TCP no hardware da placa, levando a uma melhor utilização do processador e a uma melhor performance total se comparada às outras duas placas Ethernet.

O *Fibre Channel* é o único modo para implementar uma SAN que necessite de grande largura de banda e de uma grande quantidade de operações de entrada e saída. Este fato é comprovado nos testes de transferência de pequenas quantidades de dados quanto nos de grandes quantidades, aonde o HBA *Fibre Channel* obteve os melhores resultados.

Conseqüentemente, para conectar um grande número de servidores de baixa largura de banda à SAN ou para integrar computadores de fora da SAN, o iSCSI é recomendado devido ao seu custo de implementação mais baixo. Para servidores que possuem uma grande capacidade de processamento uma placa adaptadora *Gigabit Ethernet* ou *Fast Ethernet* provê uma boa solução para a conectividade da SAN iSCSI. Entretanto, se os servidores possuem baixo desempenho de processamento, são indicadas placas adaptadoras de rede com descarregamento (offloading) próprio, retirando o processamento do servidor.

Para servidores de aplicações de performance crítica, o *Fibre Channel* pode ser usado devido aos seus elevados níveis de performance e necessidade de baixos processamentos. Entretanto, para arquitetar a melhor SAN possível em termos de custo e desempenho, ambas as tecnologias podem ser combinadas em uma única SAN utilizando um roteador de armazenamento para manejar a comunicação entre iSCSI e *Fibre Channel*.

Sugestão para trabalhos futuros:

a) Implementar o Linux iSCSI em uma rede *Gigabit Ethernet*;
b) Fazer funcionar um servidor *web* com banco de dados utilizando armazenamento iSCSI.
c) Testar a qualidade dos dispositivos de segurança disponíveis na implementação de uma SAN iSCSI.
d) Criar dentro da rede inf uma SAN iSCSI, para manter *backups* dos servidores ou para disponibilizar um maior espaço para os diretórios *home* dos usuários.

Bibliografia

[1] http://whatis.techtarget.com/definition/0,,sid9_gci750136,00.html

[2] Internet Protocol (IP); http://ietf.org/rfc/rfc0791.txt?number=791

[3] http://www.ietf.org/rfc/rfc3720.txt

[4] Meth, K. Z; Satran, J.; Features of the iSCSI Protocol. IEEE Communications Magazine, Agosto de 2003, páginas 72-75.

[5] Padrão SCSI; http://www.danbbs.dk/~dino/SCSI/

[6] Cain, T.; Técnicas de utilização de discos rígidos – tradução de Ivo Korytowski. – Rio de Janeiro: Campus, 1988.

[CORD] Chadalapaka, M. and R. Elliott, "SCSI Command Ordering Considerations with iSCSI", Work in Progress.

[7] http://www.ufpa.br/dicas/arq-uni.htm, acessado em 16/10/2004.

[8] http://www.axis.com/es/seguridad/storage_pt.htm, acesado em 16/10/2004.

[9] http://www.baboohardware.com.br/absolutenm/anmviewer.asp?a=9047&z=203, acessado em 16/10/2004.

[10] http://www.resellerweb.com.br/resellerschool/artigo.asp?id=23724, acessado em 16/10/2004.

[11] http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=819, acessado em 16/10/2004.

[12] http://www.cesce.pt/pt/not_analise.php3?id=87, acessado em 16/10/2004.

[13] http://www.magnet.com.br/classic/raiox/conect4.html, acessado em 16/10/2004.

[14] http://www.guiadohardware.net/artigos/194/, acessado em 16/10/2004.

[15] http://www.infowester.com/cluster.php, acessado em 16/10/2004.

[16] http://netra01.lncc.br/Principal_arquivos/O_que_e_Grid.htm, acessado em 16/10/2004.

[17] http://linux-iscsi.sourceforge.net/, acessado em 04/11/2004.

[18] http://www.crn.com.br/noticias/artigo.asp?id=50225&p=2&pct=5 acessado em terça-feira, 19 de outubro de 2004.

[19] Strand, S.; Ahlstrand, M.; Paper I/O Performance Characteristics of iSCSI NICs and Fibre Channels HBAs. Novembro de 2003, Stockholm.

- ANEXO -

INTERNET SMALL COMPUTER SYSTEM INTERFACE: UM ESTUDO DE CASO PARA STORAGE AREA NETWORKS

João Pedro de C. R. D. Costa
Ricardo Delcastanher

Ciências da Computação
Universidade Federal de Santa Catarina

Resumo

Este trabalho apresenta um estudo sobre o iSCSI, mostra o resultados de testes comparativos entre Fibre Channel e iSCSI e faz uma consideração sobre onde o iSCSI pode ser aplicado.

Palavras-chave: Rede de Armazenamento de Dados (Storage Area Network), Internet Small Computer Systems Interface (iSCSI), Fibre Channel (FC).

Introdução

Para resolver a demanda cada vez mais crescente de armazenamento de grandes quantidades de dados, bem como um rápido acesso a estes dados, *Storage Area Network* ou SANs têm se tornado comum nas empresas. *Storage Area Network* representa um modelo de armazenamento distribuído onde vários *switches* conectam os servidores da SAN aos seus dispositivos de armazenamento. Elas permitem virtualização e compartilhamento de dados, bem como facilidade de backup e recuperação.

A Internet já é conhecida como um dos principais meios para troca de informação, e vem se tornando também a principal solução para troca de arquivos. Não bastasse a "troca" de arquivos, existe também a necessidade de se deixarem expostos milhares de arquivos para serem acessados de todas as partes do mundo. Dependendo do tipo de serviço, estes arquivos podem facilmente ultrapassar a barreira do 1Terabyte.

A partir deste problema surge um mercado para as empresas prestadoras de serviço de hospedagem para a Internet, que podem optar por máquinas com armazenamento local utilizando discos rígidos muito caros, ou, como mostra este documento, utilizar de armazenamento distribuído.

A opção de se utilizar armazenamento distribuído não só baixa os custos, como também tende a aumentar a segurança e a performance. Os dados não precisam estar geograficamente no mesmo lugar e poderão ser acessados de qualquer computador conectado à Internet. Caso todo o espaço disponível esteja sendo usado, para aumentar sua capacidade basta acrescentar uma máquina com tecnologia de armazenamento local barata a esta rede.

Armazenamento de Dados

1. *Direct Attached Storage* (DAS) - Trata-se provavelmente da solução mais comum de armazenamento em disco rígido utilizado em quase todos os tipos de computadores conhecidos. A limitação depende da capacidade do computador e do número máximo de discos rígidos que o mesmo pode suportar.[8]

***Advanced Technology Attachment* (ATA) -** A *Advanced Technology*

Attachment é uma implementação no disco que integra o controlador no próprio disco. Existem várias versões, desenvolvidas pelo Comitê SFF (*Small Form Factor Committee*).[9]

***Small Computer System Interface* (SCSI) -** O SCSI é uma interface paralela para ligar periféricos, usada inicialmente pela Apple Macintosh e Unix. Atualmente, é estendida a qualquer computador, mas devido ao seu valor mais alto em relação a outras implementações, este disco rígido é usado em locais onde os dados precisam de mais segurança, velocidade de busca e funcionamento 24 horas por dia e 7 dias por semana, como em servidores, por exemplo[5].

***Redundant Array of Inexpensive Disks* (RAID) -** Traduzido como disposição redundante de discos baratos, o RAID conta com vários discos rígidos combinados para aumentar a performance. Num nível mais complexo, o RAID pode ser usado também para melhorar a confiabilidade do equipamento através de espelhamento ou paridade [8].

2. *Network Attached Storage* (NAS) - No NAS, o sistema de arquivos organiza blocos de armazenamento em objetos que são convenientes para os aplicativos que lidam com eles. Este é responsável por alocar espaço e evitar complicações nas solicitações de acesso a arquivos. No lado do servidor, um cliente traduz as solicitações de Entrada e Saída de arquivos dos aplicativos em mensagens de rede, e as envia para o dispositivo NAS, para que sejam executadas.

***Network File System* (NFS) -** O NFS é o sistema de arquivos em rede padrão do Linux, e foi desenvolvido pela Sun. Qualquer operação com arquivos executada por um programa em uma máquina, é enviada pela rede para outro computador. Esse procedimento emula um ambiente em que todos os arquivos encontram-se no mesmo equipamento onde ele está sendo executado. Isso torna o compartilhamento de informações muito simples, já que não requer nenhuma modificação nos programas utilizados [11].

3. *Storage Area Network* (SAN) - SAN trata-se de uma rede especificamente concebida não só para movimentar grandes volumes de dados, devido sua largura de banda, mas também para melhorar e tornar mais eficiente a distribuição dos recursos de armazenamento pelos vários sistemas clientes [12].

***Fibre Channel* (FC) -** *Fibre Channel* é o padrão de conexão mais rápido até o momento disponível comercialmente. Sua velocidade chega a 400 MB/s, podendo atingir até 1 GB/s, e suportando até cento e vinte e seis dispositivos conectados.[13]

***Internet Small Computer Systems Interface* (iSCSI) -** O iSCSI é um padrão de armazenamento de dados em redes baseadas em IP [2], desenvolvido pela IETF (Força Tarefa de Engenharia para Internet), servindo para ligar instalações de armazenamento de dados [1]. O iSCSI é utilizado para facilitar transferências de dados pelas *intranets* e para gerenciar armazenamento à grandes distâncias. Isto é possível porque ele se utiliza de comandos SCSI sobre redes IP.

Infiniband - O principal objetivo do Infiniband é interligar servidores e dispositivos de armazenamento localizados a curtas distâncias. Funcionando como uma opção mais rápida às redes *Ethernet* ou a servidores de bancos de dados, ele possibilita acessar um dispositivo de armazenamento externo, sem nenhum gargalo, como se fosse um dispositivo local. Facilidade esta que abre muitas possibilidades nos servidores de alto desempenho, *clusters* e fazendas de servidores. [14]

4. *Cluster* - É o nome dado a um sistema montado com mais de um computador de forma que pareça que eles sejam um computador só, fazendo com que todo o processamento da aplicação seja distribuído. Com isso, é possível realizar processamentos que até então somente computadores de alta performance eram capazes de fazer [15].

5. *Grid* - Um *grid* pode ser definido como uma forma de gerenciar racionalmente recursos computacionais distribuídos (tanto de hardware como de software). Através do *grid* podemos agregar os recursos da computação tradicional e da computação de alto desempenho, além de dar subsídios para a realização de projetos de aplicações temáticos baseados em ambientes de Computação Distribuída [16].

iSCSI

O iSCSI é um padrão de armazenamento de dados em redes baseadas em IP com a finalidade de interligar instalações de armazenamento de dados. O iSCSI é utilizado para facilitar transferências de dados pelas *intranets* e para gerenciar armazenamento à grandes distâncias. Isto é possível porque ele se utiliza de comandos SCSI sobre redes IP.

O protocolo iSCSI é uma das tecnologias que vêm sendo cada vez mais utilizadas nas *Storage Area Networks*, pois ele aumenta a capacidade e performance de transmissão de dados armazenados, utilizando uma rede IP já existente. Por isso pode ser utilizado para transmitir dados sobre Redes Locais (LANs), Redes Distribuídas Geograficamente (WANs) ou através da Internet, permitindo armazenamento e acesso aos dados independente da sua localização.

Conceitos SCSI

No nível mais alto, o SCSI é uma família de interfaces que requisitam serviços de dispositivos de entrada/saída, incluindo discos rígidos, *drives* de fita, CD e DVD, impressoras e scanners. Na terminologia SCSI um dispositivo entrada/saída é chamado de “unidade lógica” (UL).

SCSI é uma arquitetura cliente-servidor. Clientes são chamados de iniciadores. Eles emitem comandos SCSI que requisitam serviços de componentes e unidades lógicas de um servidor, que é chamado de alvo. O dispositivo servidor na unidade lógica aceita comandos SCSI e os processa [5].

Conceitos e Características do iSCSI

O protocolo iSCSI é um mapeamento do modelo de invocação remoto SCSI [5] sobre o protocolo TCP. Os comandos SCSI são carregados por requisições iSCSI bem como as respostas e *status* SCSI são carregados por respostas iSCSI. Por estar sendo usado por algumas décadas, o protocolo TCP tem a vantagem de ser o mais utilizado e o mais compreendido, coisa que outros protocolos como o SCTP não tem, apesar de possuírem muitas das características do TCP [4]. Algumas das características do protocolo TCP que são utilizadas pelo iSCSI estão listadas à seguir:

V. Os dados são entregues em ordem e de forma confiável;
VI. Efetua retransmissão automática dos dados que não tiveram confirmação de recebimento;
VII. Em uma rede congestionada, aplica controle de fluxo e de congestionamento, necessários para evitar a sobrecarga da mesma;
VIII. Funciona sobre uma grande variedade de topologias e mídias físicas.

Da mesma maneira que protocolos similares fazem, o iniciador e o alvo dividem suas comunicações em mensagens. Essas mensagens são chamadas de “unidades de dados do protocolo iSCSI” (UDP iSCSI). Um dado associado a um comando pode ser enviado como uma mensagem única, do iniciador para o alvo, e a resposta, juntamente com algum dado, pode ser enviada a partir dos alvos.

A direção da transferência iSCSI é definida respeitando o iniciador. Transferências de saída são transferências de um iniciador para um alvo, enquanto que transferências de entrada são transferências de um alvo para um iniciador. Uma

tarefa iSCSI é uma requisição iSCSI na qual uma resposta é esperada [3].

Ambiente de Testes

***Workloads* Gerais** - O primeiro teste envolveu processamento dos *workloads* emulando aplicativos comuns. Cada workload foi analizado para cada NIC/HBA em termos de média de número de operações entrada/saída por segundo, média da taxa de transferência, média da latência nas transações, e média da utilização da CPU como também outras métricas de performance do IOMeter foram todas analisadas durante o processo.

***Small Transfer Size Random Workloads* -** Depois de completado o primeiro conjunto de testes para *workloads* gerais, um segundo grupo de testes usará pequenos tamanhos de transferência com acessos aleatórios a blocos. Os testes de transferência de tamanhos pequenos são bons indicadores do máximo de operações de entrada e saída por segundo

Large Transfer Size Sequential Workloads - Os testes com *workloads* contendo grandes transferências e acessos seqüenciais foram feitos para comparar a máxima largura de banda que diferentes adaptadores podem apresentar. Os testes foram executados tanto para leituras quanto para escritas usando vários números de operações de entrada e saída.

Projeto Linux iSCSI

O *driver* iSCSI para Linux atua como um iniciador do protocolo iSCSI para as requisições e respostas do transporte SCSI sobre um rede IP entre o cliente e um alvo iSCSI como um roteador de armazenamento Cisco SN 5428-2.

Deve-se ter um dispositivo compatível ao iSCSI conectado à rede para poder acoplar ao armazenamento. O dispositivo pode estar na mesma rede que o Linux hospedeiro ou o tráfego iSCSI pode ser roteado usando métodos normais de roteamento IP. O *daemon* e o driver do kernel estão disponíveis sob os termos GNU de Licença Pública. [17]

Conclusões e Trabalhos Futuros

O teste feito no capítulo IV mostra que os HBAs do *Fibre Channel* se adequaram melhor para tráfego de armazenamento do que para as placas Adaptadoras *Ethernet*, porém as placas adaptadoras *Ethernet* podem ser usadas para implementar uma rede iSCSI com uma penalidade mínima na performance para certos tipos de aplicações com cargas de trabalho gerais e servidores de arquivos de carga moderada, como em estações de trabalho.

Para algumas cargas de trabalho gerais não críticas, onde a carga de entrada e saída é pequena durante grande parte do tempo, a diferença entre os adaptadores, tanto na largura de banda quanto no número de operações de entrada e saída por segundo foi relativamente pequena. Notou-se que a grande penalidade na performance da implementação iSCSI foi devido o processamento do protocolo TCP/IP. Isto significa que a utilização do processador subiu muito rapidamente para as placas adapatadoras *Fast Ethernet* e *Gigabit Ethernet* quando a carga aumentava. A placa Alacritech *Gigabit Ethernet* resolveu este problema manipulando o descarregamento do TCP no hardware da placa, levando a uma melhor utilização do processador e a uma melhor performance total se comparada às outras duas placas Ethernet.

O *Fibre Channel* é o único modo para implementar uma SAN que necessite de grande largura de banda e de uma grande quantidade de operações de entrada e saída. Este fato é comprovado nos testes de transferência de pequenas quantidades de dados quanto nos de grandes quantidades, aonde o HBA *Fibre Channel* obteve os melhores resultados.

Conseqüentemente, para conectar um grande número de servidores de baixa largura de banda à SAN ou para integrar computadores de fora da SAN, o iSCSI é

recomendado devido ao seu custo de implementação mais baixo. Para servidores que possuem uma grande capacidade de processamento uma placa adaptadora *Gigabit Ethernet* ou *Fast Ethernet* provê uma boa solução para a conectividade da SAN iSCSI. Entretanto, se os servidores possuem baixo desempenho de processamento, são indicadas placas adaptadoras de rede com descarregamento (offloading) próprio, retirando o processamento do servidor.

Para servidores de aplicações de performance crítica, o *Fibre Channel* pode ser usado devido aos seus elevados níveis de performance e necessidade de baixos processamentos. Entretanto, para arquitetar a melhor SAN possível em termos de custo e desempenho, ambas as tecnologias podem ser combinadas em uma única SAN utilizando um roteador de armazenamento para manejar a comunicação entre iSCSI e *Fibre Channel*.

Sugestões para trabalhos futuros:

e) Implementar o Linux iSCSI em uma rede *Gigabit Ethernet*;
f) Fazer funcionar um servidor *web* com banco de dados utilizando armazenamento iSCSI.
g) Testar a qualidade dos dispositivos de segurança disponíveis na implementação de uma SAN iSCSI.
h) Criar dentro da rede inf uma SAN iSCSI, para manter *backups* dos servidores ou para disponibilizar um maior espaço para os diretórios *home* dos usuários.

Bibliografia

[1]http://whatis.techtarget.com/definition/0,,sid9_gci750136,00.html

[2]Internet Protocol (IP); http://ietf.org/rfc/rfc0791.txt?number=791

[3]http://www.ietf.org/rfc/rfc3720.txt

[4]Meth, K. Z; Satran, J.; Features of the iSCSI Protocol. IEEE Communications Magazine, Agosto de 2003, páginas 72-75.

[5] Padrão SCSI; http://www.danbbs.dk/~dino/SCSI/

[6]Cain, T.; Técnicas de utilização de discos rígidos – tradução de Ivo Korytowski. – Rio de Janeiro: Campus, 1988.

[CORD]Chadalapaka, M. and R. Elliott, "SCSI Command Ordering Considerations with iSCSI", Work in Progress.

[7]http://www.ufpa.br/dicas/arq-uni.htm, acessado em 16/10/2004.

[8]http://www.axis.com/es/seguridad/storage_pt.htm, acesado em 16/10/2004.

[9]http://www.baboohardware.com.br/absolutenm/anmviewer.asp?a=9047&z=203, acessado em 16/10/2004.

[10]http://www.resellerweb.com.br/resellerschool/artigo.asp?id=23724, acessado em 16/10/2004.

[11]http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=819, acessado em 16/10/2004.

[12]http://www.cesce.pt/pt/not_analise.php3?id=87, acessado em 16/10/2004.

[13]http://www.magnet.com.br/classic/raiox/conect4.html, acessado em 16/10/2004.

[14]http://www.guiadohardware.net/artigos/194/, acessado em 16/10/2004.

[15]http://www.infowester.com/cluster.php, acessado em 16/10/2004.

[16]http://netra01.lncc.br/Principal_arquivos/O_que_e_Grid.htm, acessado em 16/10/2004.

[17]http://linux-iscsi.sourceforge.net/, acessado em 04/11/2004.

[18]http://www.crn.com.br/noticias/artigo.asp?id=50225&p=2&pct=5 acessado em terça-feira, 19 de outubro de 2004.

[19]Strand, S.; Ahlstrand, M.; Paper I/O Performance Characteristics of iSCSI NICs and Fibre Channels HBAs. Novembro de 2003, Stockholm.