uSCSI 设计文档

v0.01

余上

2012-03-31

概述

iSCSI 协议由 RFC3720 定义，简单的说，iSCSI 就是 SCSI over TCP，即在 TCP 链路上传输 SCSI 命令，iSCSI 协议扩展了存储可以覆盖的地理范围，主机和存储设备之间只要有 TCP 网络连接就可以访问，这大大增加了部署的灵活性，实际上，iSCSI 已经是诸多 SAN 互联解决方案之一。

iSCSI 有两种实现方式，一种是硬件实现，直接实现在网卡中，好处是速度较快，吞吐量也很高，缺点是需要专用的网卡，而且成本也较高；另外一种是软件实现，配合千/万兆以太网卡，以及多 PDU 链路，能够提供性价比不错的传输性能，目前有多种软件实现，例如，微软公司的 iSCSI Initiator，或者 Linux 下的 Open-iSCSI 项目等，根据 Open-iSCSI 提供的数据，该实现能够提供最大 550MB/s 的读和 810MB/s 的写，详细情况可以参考这里。

uSCSI 是我实现的一个 Windows 平台 iSCSI 协议驱动（启动器），这是一个基本的实现，仅包括了 iSCSI 运转所需要的基本功能，一些高级功能，例如发现会话、错误恢复等等还有待扩充。本文是 uSCSI 的设计文档，包括两部分，前半部介绍 iSCSI 协议要点，后半部介绍 uSCSI 的架构，附录部分是 uSCSI 重要 API 的列表。

协议

iSCSI 协议的基本运行机制是请求/响应模式，本章简要介绍协议的主要部分，包括
1，协议数据单元
2，会话及登录
3，文本命令
4，SCSI 命令
等，详细的协议内容请参考原文档

启动器和目标

可以简单的理解启动器为主机，而目标是存储设备

协议数据单元

iSCSI 协议的基本传输单位是 PDU，即，**协议数据单元**，PDU 主要由两部分组成，协议头和数据，协议头包括 BHS（Basic Header Segment）**基本协议头**和 AHS（Additional Header Segment）**附加协议头**，基本协议头长度为固定的 48 字节，基本协议头包含了许

多重要的控制字段，参见下表：

字段	说明
Opcode	操作码，表明了 PDU 的意义，例如，是 SCSI 命令还是数据写等，完整列表参见命令小节
TotalAHSLength	附加协议头大小，单位为 4 字节
DataSegmentLength	数据部分大小，单位为字节
CmdSN DataSN StatSN ExpStatSN ExpCmdSN MaxCmdSN	参见编号节、状态同步及控制小节
Final	标记 PDU 序列，参见数据 PDU 小节
Continue	标记 PDU 片段，参见文本交换小节
Transit	阶段转移标志，参见阶段及转移节

附加协议头是一个可选结构，并且长度是任意的（4 字节对齐），结构的第一个字段表明了附加协议头的大小，第二个字段表明了附加协议头的类型；协议头之后是数据，数据也是可选结构，例如，对于不涉及数据传输的命令就没有数据部分。

PDU 分类

PDU 封装的是所谓的 iSCSI 命令，分为如下几类

分类	操作码	说明
登录	LOGIN	登录，包括省份验证，参数协商等
注销	LOGOUT	注销
文本	TEXT	文本交换，实现协议扩展，运行时参数协商
SCSI	CMD	传输 SCSI 命令
数据	DATA-OUT DATA-IN	传输数据（写和读）
传输请求	R2T	数据传输请求，用于目标向启动器请求数据传输
心跳	NOP-OUT NOP-IN	在没有 PDU 时传输控制信息
异步消息	ASYNC-MSG	
杂项	SNACK REJECT	

以上这些命令大部分都具有请求/相应的模式，例如，登陆命令使用登录请求和登陆响应来交换登录信息，注销、文本、SCSI 都有相应的请求/响应。

编号、状态同步及流控制

iSCSI 会话支持**多 PDU 链路**，SCSI 命令/数据可以同时从多个 PDU 链路发出，先发出的命令/数据有可能比后发出的命令/数据更晚到达目标，因此 iSCSI 使用命令/数据编号的方式来区分命令/数据的逻辑顺序，目标严格按照命令的编号顺序来执行命令，按照数据编号顺序来重构数据，下表列示了 PDU 中和编号相关的控制字段及含义：

方向	编号		同步/流控制
	命令/响应	数据	
启动器->目标	CmdSN	DataSN	ExpStatSN
目标->启动器	StatSN	DataSN	ExpCmdSN，MaxCmdSN

上表中，CmdSN 字段用于 SCSI 命令编号，命令编号在整个会话范围内唯一；StatSN 用于 SCSI 命令响应编号，响应编号也是会话范围内唯一；DataSN 用于数据写/数据读编号，数据写/数据读编号在对应命令范围内唯一；ExpStatSN 用于响应的告知，ExpStatSN=N 表示启动器已经收到 ExpStatSN-1 为止的所有响应，N 是启动器期望接收的下一个响应号；

ExpCmdSN 用于命令告知，ExpCmdSN=N 表示目标已经收到 ExpCmdSN-1 为止的所有命令 ,N 是目标期望收到的下一个命令号 ;MaxCmdSN 是目标可以接收的最大命令号，MaxCmdSN=N 表示启动器发送的命令编号不能超过 N，启动器使用下面公式计算**命令窗口**：

MaxCmdSN – ExpCmdSN + 1

命令窗口就是启动器当前可以发送的最大命令条数，命令窗口实现了 iSCSI 命令的流控制

文本交换

文本交换，是指启动器和目标之间交换文本类型的数据，称为**文本数据**，文本数据具有“**键=值**”的形式，多个键值对之间用 ‘\0’ 字符分割。**文本 PDU** 是指那些数据部分是文本数据的 PDU，包括：登录请求\响应、文本请求\响应。

封装文本数据时，单个文本 PDU 的文本数据大小不能超过接收方最大数据段长度（MaxRecvDataSegmentLength）参数设定的值，如果文本数据大于这个值，需要将其拆分到多个文本 PDU 中 ,这样的 PDU 称之为**文本 PDU 片段**(简称 **PDU 片段**),多个 PDU 片段组成一个**逻辑文本 PDU（LTDS）**。

iSCSI 使用 PDU 的 C 标志位来标记 PDU 片段 ,当 C=0 时 ,表示这是一个逻辑文本 PDU 或者逻辑文本 PDU 的最后一个 PDU 片断，当 C=1 时，表示这是中间的一个 PDU 片断。启动器和目标以逻辑文本 PDU 为单位来处理文本 PDU。

文本交换协议是指启动器和目标之间如何传输逻辑文本 PDU，iSCSI 规定，发送逻辑文本 PDU 时，发送方完全占有发送通道，直到一方发送完毕。

逻辑文本 PDU 是以一系列 PDU 片段的形式发送的，接收方在接收到一个片段后，必须响应一个数据段长度为 0 的 PDU，这个 PDU 叫做**同步 PDU**，其作用是告诉发送方，前

面一个片段已经接收完毕，可以继续发送下一个片段。

接收方通过检测片段的 C 位来判断逻辑文本 PDU 的发送是否完成。

文本交换协议适用于登录阶段的**参数协商**，以及全功能阶段的**文本命令**。

数据 PDU

数据 PDU 是指那些用来传送数据的 PDU ,包括数据写(DATA-OUT)和数据读(DATA-IN ）两种。有时候 ,需要传输的数据太大 ,无法用一个 PDU 来封装 ,需要把数据分组到多个 PDU 中，这样的一组 PDU 称之为 **PDU 序列**，PDU 序列组成一个**逻辑数据 PDU**。iSCSI 使用 F 位来支持逻辑数据 PDU，当 F==0 时，表示这个 PDU 是一个序列的开始或者序列的一部分，当 F==1 时，表示这是唯一的一个 PDU 或者序列中的最后一个 PDU：

PDU0(F==0),PDU1(F==0),PDU2(F==1)

PDU0 到 PDU2 是一个 PDU 序列，下面是影响数据分组的一些参数：

参数名	说明
MaxRecvDataSegmentLength	数据段大小 ,数据接收方能够接受的单个数据 PDU 数据部分最大值
MaxBurstLength	突传数据大小，数据接收方能够接受的逻辑数据 PDU 数据部分最大值
FirstBurstLength	初始数据大小，同上，仅适用于非请求数据

数据接收方如果检测到超出突传数据大小或者初始数据大小的逻辑数据 PDU，可以丢弃数据并关闭链路。

错误及恢复

PDU 在传输的过程中可能遇到各种各样的**错误**，例如，传输层错误、校验和错误，数据未被确认等等，出现错误后需要进行**恢复**，消除错误带来的影响，例如，PDU 出现校验和错误时，启动器可以要求目标重新传输出现错误的 PDU。根据进行恢复所影响范围的大小，iSCSI 定义了 4 种类型的恢复，包括：

命令内恢复 ,指不需要重新提交 SCSI 命令便可以进行的恢复 ,数据 PDU 校验和错误、数据序列错误等都需要命令内恢复，例如，启动器检测到数据读 PDU（ DATA-IN,DI ）校验和错误时，可以要求目标重新传输 DI，这个例子中，SCSI 命令仍然继续执行，仅仅部分 DI 需要恢复。

链路内恢复，指不需要重建链路，需要 SCSI 命令重建的恢复。在启动器方，SCSI 命令未被确认、SCSI 命令响应校验和错误、SCSI 命令响应编号错误、S 标志位置位的数据读 PDU 校验和错误等需要链路内恢复 ;在目标方包括响应未被确认等。例如 ,启动器如果认为 SCSI 命令未被目标接收，需要重新传输该命令，这个例子中，原命令所在的链路仍然可用，但是 SCSI 命令重提交了。

链路恢复，是指在 PDU 链路出现故障后需要进行的操作，启动器在检测到传输层错误

时，必须注销该链路（通过别的可用链路，如果没有，需要新建，注销原因为“从会话中移除链路”），并且重新分配（使用任务管理命令）该链路上的待定命令（到别的链路上）；另一种情况是启动器收到目标发送的异步消息表明一个或多个链路已经失败。链路恢复由启动器来起动。

会话恢复，会话恢复要求重启会话，重启所有的链路。

会话及登录

iSCSI 将启动器和目标之间的一个关联叫做**会话**，每个会话都有一个唯一的会话标识，两者之间建立第一个 **PDU 链路**（简称**链路**）之后便开始了一个会话。

iSCSI 定义了 2 种类型的会话，**普通会话**（normal）和**发现会话**（discovery），后者用于查询可用的目标，即，如果启动器不知道目标的名字，可以向目标门户查询，这时使用的会话叫做发现会话，目标门户在这个会话中返回可用目标。在确定目标明后，启动器便可以向目标发起连接，这时使用的会话叫做普通会话。

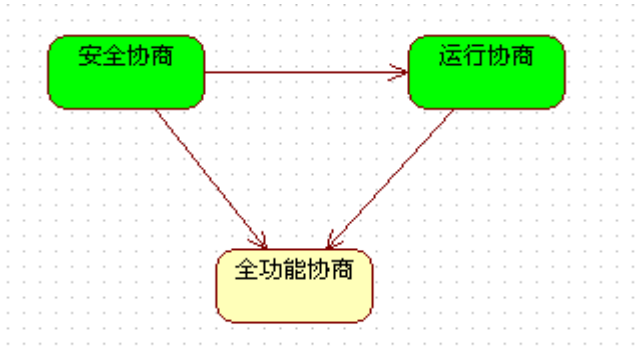
普通会话可以建立多条链路，每条链路的建立都要经历登录过程，登录主要包括启动器和目标间的身份证明以及参数协商。登录过程结束后，启动器和目标进入工作状态。

阶段及转移

iSCSI 将整个登录过程划分为三个阶段，参见下表

编号	阶段名	必需	说明
0	安全协商	否	安全相关参数的协商
1	运行协商	否	运行相关参数的协商
3	全功能协商	是	全功能运行

启动器和目标在上述一个或多个阶段中进行相关参数协商，一个阶段协商完毕后，再转移到另一个阶段，整个登录过程包括多个阶段间的转移，当全功能协商完毕后，启动器和目标进入正常工作状态。下图表明了阶段的转移关系：



启动器可以选择从**安全协商**或**运行协商**开始登录过程，如上图中的绿色方框所示，如果选择从安全协商开始，那么下一个阶段可以是运行协商或者全功能协商，如果选择从运行协商开始那么下一个阶段只能是全功能协商，启动器不能直接进入全功能协商。

阶段转移由**阶段转移协议**来控制，协议使用 PDU 的 T 标志位实现阶段转移的申请和答复。阶段转移由启动器发起，当启动器需要进行阶段转移时，将 T 标志位置位（T=1），表明启动器已经准备好阶段转移，并在 PDU 的**当前阶段**（CSG）字段中填写当前阶段（编号），在**下一阶段**（NSG）字段中填写申请转移的阶段；

目标在检测到 T 标志位置位后，会检查自己的内部状态，以决定是否响应这个请求，如果同意阶段转移，目标将响应 PDU 的 T 位置位，T=1，并且在 NSG 中填写同意转移的阶段，目标不一定会选择启动器提交的阶段，但是目标选择的转移阶段（编号）不能超过启动器申请转移的阶段（编号），比如，启动器可以请求从安全协商直接进入全功能协商，但是目标可以选择进入运行协商；如果目标不同意转移，将 T 标志位置 0，T=0。

启动器在接收到这个响应 PDU 后，检测 T 标志位，如果置位表示目标同意了阶段转移，启动器将当前阶段标记为目标选择转移的阶段（NSG），从而开始新阶段的**参数协商**，否则，启动器需要继续申请。下面是一个阶段转移的例子：

序号	操作	说明
1	I->LOGIN,T=1CSG=0,NSG=3	启动器提出阶段转移申请
2	T->LOGIN,T=0	目标拒绝转移
3	I->LOGIN,T=1,CSG=0,NSG=3	保持申请
…		
N	I->LOGIN,T=1,CSG=0,NSG=3	保持申请
N+1	T->LOGIN,T=1,CSG=0,NSG=1	目标同意转移，但是同意转移到阶段 1，而不是启动器申请的阶段 3

下面是基本协议头中用于控制阶段协商及转移的字段列表

字段	说明
T	阶段转移申请/回复
CSG	当前协商阶段
NSG	下一协商阶段

参数协商

iSCSI 定义了一些控制参数，这些参数控制着协议的运行。参数分为两大类，**声明**类和**协商**类。声明类参数仅起到告知接收方某个信息的作用，而不需要接收方发表意见，例如，启动器名字（InitiatorName）就是一个声明类的参数，再如，最大数据大小（MaxRecvDataSegmentLength）也是一个声明类的参数；

对于协商类参数，启动器和目标需要进行谈判，为参数确定一个合理的值，如果谈判失败，启动器和目标可以选择退出登录。大部分参数在**安全协商**和**运行协商**阶段中中协商完毕。参数协商使用前面小节中说明的**文本交换协议**，下面是一个登录过程中参数协商的例子（下表中操作列语法取自 iSCSI 协议文本，含义请参考原文本，下同）：

序号	操作	说明
1	I->LOGIN,C=1	

2	T->LOGIN	
3	I->LOGIN,C=0	
4	T->LOGIN,C=1	
5	I->LOGIN	
6	T->LOGIN,C=1	
7	I->LOGIN	
8	T->LOGIN,C=0	

这个表展示了登录过程某个阶段中的参数协商过程。1-3 步是启动器参数传输：参数集封装在一个逻辑文本 PDU 中，这个 PDU 由 1、3 两步中的 PDU 片段组成，分两次传输完毕。第 2 步为目标发送的同步 PDU，启动器接收到这个同步 PDU 后再发送后续的 PDU 片段。目标在第 3 步中检测到逻辑文本 PDU 发送完毕，并在第 4 步开始自己的参数传输。

4-6 步展示的是目标的参数传输，和启动器参数传输的例子类似，逻辑文本 PDU 由三个 PDU 片段（4，6，8）组成，启动器进行了两次同步（5，7）。

注：每个阶段可以协商的参数请参考 iSCSI 协议文本。

参数列表

下面是在各个阶段协商的参数列表

阶段	参数	说明
安全协商	SessionType	会话类型
	InitiatorName	启动器名字
	TargetName	目标名字
	TargetAddress	目标地址
	InitiatorAlias	启动器别名
	TargetAlias	目标别名
	TargetPortalGroupTag	
	AuthMethod	身份验证方法
运行协商	HeaderDigest	协议头校验和
	DataDigest	数据校验和
	MaxConnections	最大连接数
	SendTargets	
	TargetName	同前
	InitiatorName	同前
	TargetAlias	同前
	InitiatorAlias	同前
	TargetAddress	同前

	TargetPortalGroupTag	同前
	InitialR2T	初始数据传输请求
	ImmediateData	即时数据
	MaxRecvDataSegmentLength	最大 PDU 数据大小
	MaxBurstLength	最大传输数据大小
	FirstBurstLength	最大初始数据大小
	DefaultTime2Wait	
	DefaultTime2Retain	
	MaxOutstandingR2T	待定数据传输请求个数
	DataPDUInOrder	
	DataSequenceInOrder	
	ErrorRecoveryLevel	错误恢复水平
	SessionType	同前

上表中标记为蓝色的参数在两个阶段都可以协商

SCSI 命令

SCSI 命令的执行可以分为两个阶段，任务建立和数据传输，对于启动器发出的 SCSI 命令，目标会为其建立一个任务，这个任务负责执行该命令，任务建立后进入数据传输阶段，根据命令的不同，这个数据传输可能是从启动器到目标（数据写）或者目标到启动器（数据读），或者两者的混合（双向）。

对于数据写，启动器可能在发出命令时传输数据,这些数据可以封装在 SCSI 命令中，叫做**即时数据**，也可以封装在单独的 PDU 中，叫做**初始数据**，这两种形式的数据都是在没有**数据请求**的情况下发送的，统称为**非请求数据**（unsolicited data），iSCSI 规定，启动器最多可以传输**最大非请求数据**（FirstBurstLength）参数规定字节的非请求数据。iSCSI 的两个参数控制着非请求数据的传输，参见下表：

组合	ImmediateData	InitialR2T	可用的数据形式
1	Y	N	即时数据，初始数据
2	Y	Y	即时数据
3	N	N	初始数据
4	N	Y	无数据传输

组合 1 表示先封装为即时数据，如果有剩余还可以封装到初始数据中，组合 2 表示只能封装到即时数据中，其余类推。即时数据和初始数据都必须遵守数据 PDU 小节中描述的数据封装规则。

未能用非请求数据方式发送的剩余数据将在收到目标的**数据请求**后传输，这样的数据叫做**请求数据**（solicited data），目标使用 R2T 类型的 PDU（R2T）来请求数据，目标在 R2T 的**传输标签**（TTT）字段存放一个标签，在其他字段存放要求数据的**偏移**、**长度**等信息，启动器在收到数据请求后，构建数据写 PDU（DATA-OUT,DO）传输数据，每个 DO 都会复

制 R2T 中的传输标签，表明该 DO 和该传输标签关联。

对于数据读 ,启动器在发出命令后便等待目标传输数据 ,目标把数据封装到数据读 PDU（DATA-IN，DI）中，并且在 DI 的**启动器任务标签**（ITT）中填写 SCSI 命令中的 ITT 值，表示 DI 和该标签关联。

数据告知，在进行数据读操作的过程中，目标可以要求启动器告知数据的接收情况，需要告知时，目标将 DI 的 A 标志位置位，启动器在检测到这个标志位后，需要检查到这个 DI 为止的数据接收情况，如果没有数据丢失，需要回复一个 SNACK 类型的 PDU，如果启动器检测到了数据丢失，则需要等到这个错误纠正后再回复。（注：本段描述仅对 ErrorRecovryLevel>0 的会话适用）

无论是数据写还是数据读，在数据传输完毕后，目标都应向启动器发送一个**响应**，这个响应包含了 SCSI 命令的执行情况，例如，是否出错，错误代码等等，启动器可以据此来完成 SCSI 命令。对于数据读的情况，如果命令正常完成，目标可以选择不发送响应，而是将相关信息封装在最后一个数据读 PDU 中，并将 PDU 的 S 标志位置位，这个操作和发送响应等价。

所有的数据传输都使用数据编号来实现错误恢复，数据编号在从 0 开始连续编号，因此，当数据接收方检测不连续的数据编号后可以认为出现了数据丢失。对于数据读，数据编号在命令内唯一，对于数据写，数据编号在数据请求范围内唯一。

下面是一些 SCSI 命令执行的例子，数据写的例子：

序号	操作	说明
1	I->CMD,W=1,DATALEN<>0	SCSI 命令，数据写，并且封装了即时数据
2	I->DATA-OUT	初始数据
3	T->R2T,TTT=<ttt>	目标发出数据请求
4	I->DATA-OUT,TTT=<ttt>	启动器传输剩余数据
5	I->DATA-OUT,TTT=<ttt>	启动器传输更多剩余数据
6	T->REP	目标传送响应

这个例子中，启动器传输了即时数据和初始数据以及请求数据。

文本命令

在全功能运行中，启动器可以对一些参数进行调整，这时便需要使用文本请求；另外文本命令也可以用于实现一些扩展操作。文本命令中的文本交换部分使用**文本交换协议**，详细情况参见前面文本交换小节，其他部分使用**文本命令协议**。

命令的结束 :文本命令协议使用 F 标志位来结束文本命令。文本命令总是由启动器发起，当启动器需要结束时，将 F 标志位置位（F=1）,目标在检测到这个标志位后，可以选择同意退出（置 F=1），或者拒绝退出（置 F=0）。

子会话支持，启动器和目标之间使用 TTT（Target Transfer Tag）来交换子会话标识：启动器通过给目标发送 TTT 保留值（0xffffffff）来开始一个子会话，目标响应一个非保留值的 TTT 作为该子会话的标识，如果启动器愿意继续这个子会话，则复制这个标识到下一

个发送到目标的 PDU 中。下面是文本命令的一个例子：

序号	操作	说明
1	I->TEXT,C=1,TTT=保留值	开始一个逻辑文本 PDU
2	T->TEXT,TTT=子任务编号	同步
3	I->TEXT,C=0,F=1	发送完毕，请求结束

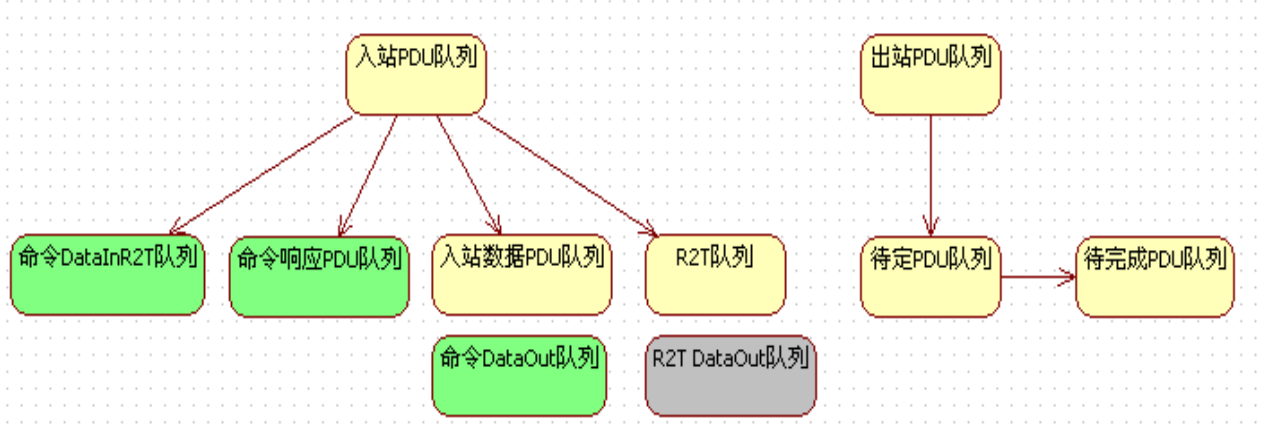
注：和文本交换协议的协调，文本交换协议中的 C 置位时 F 不能置位，也就是说，在逻辑文本 PDU 发送完毕之前，启动器不能结束文本命令。

实现

uSCSI 实现为 Windows 下的内核驱动程序，分为两部分，一部分叫 uSCSI，这是一个小端口磁盘驱动，本文不做详细描述，另一部分叫 uSCSIPort，这部分实现了 iSCSI 协议，包括如下功能：

协议数据传输

实现协议数据单元的接收和发送，uSCSI 中使用 PDU 结构来代表一个协议数据单元，并建立了 10 个 PDU 队列和相关的处理线程，这些设施构成了 uSCSI 协议处理的核心，这些队列间的关系可以参见下图：



如上图所示，传输模块从 TCP 链路接收 PDU 并存放在入站 PDU 队列中，所有这些 PDU 由 uSCSI 内部的一个**派遣线程**来处理，派遣线程根据 PDU 类型的不同将其派发给不同的协议处理函数来处理，例如，命令响应 PDU 派发给 PtProcessResponse 处理，数据读 PDU 派发给 PtProcessDataIn 处理，有些协议处理函数会将 PDU 转移到别的队列以便做进一步的处理。

数据读 PDU（DI）将同时进入命令 DataInR2T 队列和入站数据 PDU 队列，uSCSI 内部有一个**数据复制**线程，这个线程工作在入站数据 PDU 队列上，将数据 PDU 中的数据复制到指定的目的地。每复制完一个 PDU，线程会检查该 DI 关联命令的 DataInR2T 队列中

所有的 DI 是否都复制完成，如果是，触发命令的完成，这个操作唤醒**命令完成**线程，后者负责命令完成相关操作，例如，通知等待该命令的线程等等。

数据请求 PDU（R2T）将同时进入命令 DataInR2T 队列和 R2T 队列，uSCSI 内部有一个**数据请求响应**线程，这个线程工作在 R2T 队列上，负责装配数据写 PDU（DO），所有的 DO 挂入 R2T 的 DataOut 队列，这是一个局部于 R2T 的队列，如上图所示，标记为灰色。装配好的 DO 放入出站 PDU 队列发送至目标。

命令响应 PDU（REP）将进入命令响应 PDU 队列，PtProcessResponse 函数会尝试完成待定命令。

所有出站的 PDU 首先进入出站 PDU 队列，uSCSI 内部有一个**发送线程**来处理这个队列。所有发送成功的 PDU 会被自动释放，但是下列类型的 PDU 例外，他们会被放置到待定 PDU 队列中：

1，SCSI 命令

2，任务请求

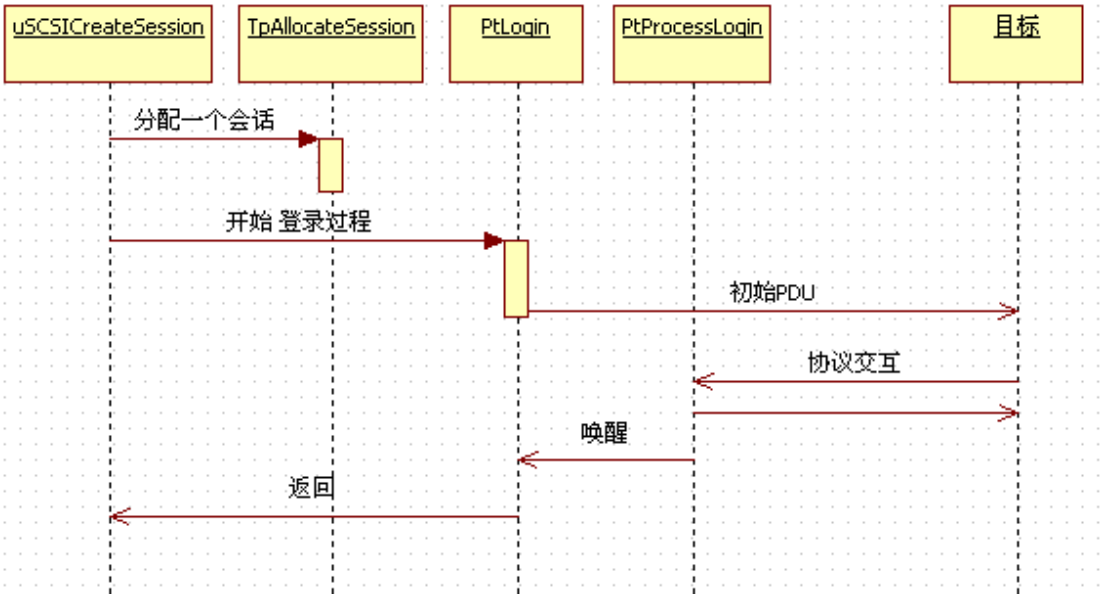
3，空操作（如果要求回显）

这些命令都是需要目标响应的命令，在目标响应之前必须保留命令记录，例如，SCSI 命令，要么向目标写数据要么从目标读数据，因此在数据写或者数据读完成之前，SCSI 命令会暂时放置在待定 PDU 队列中。当命令所需要的处理完成后，例如，需要写的数据成功写入到目标，或者需要读的数据已经全部传输完毕，uSCSI 会把该命令从待定 PDU 列表中移除，并添加到待完成 PDU 队列中。uSCSI 内部有一个**待定命令完成**线程，这个线程负责待定命令的完成工作，例如，拷贝数据等。

注：标记为绿色的队列为局部于每个 SCSI 命令的队列

登录过程

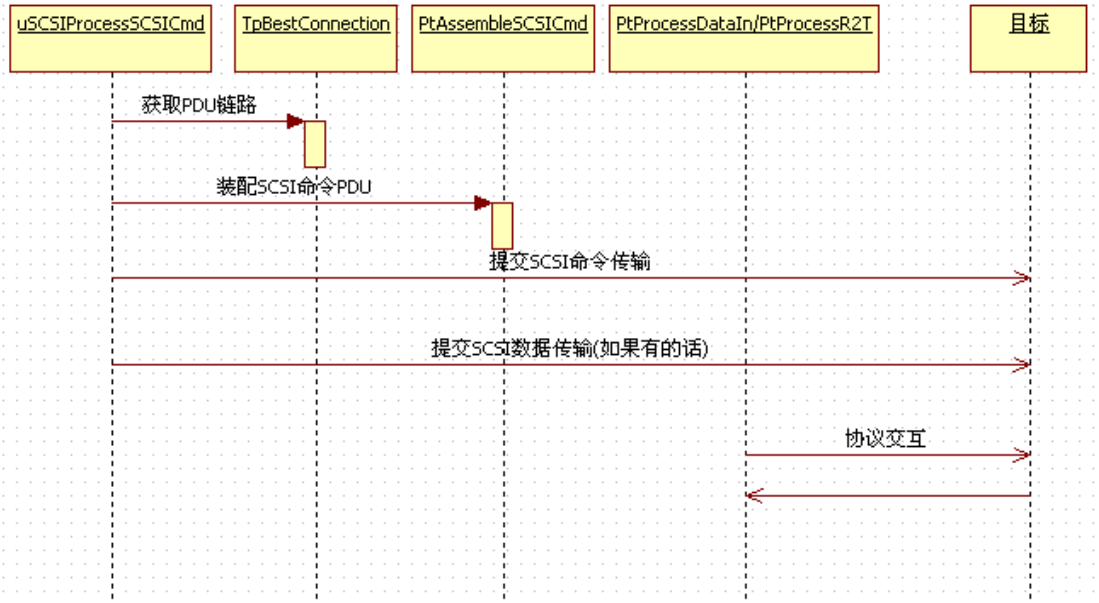
uSCSIPort 的客户端（一般是 uSCSI 驱动）在适当的时候调用 uSCSICreateSession 创建一个 iSCSI 会话，并登录到指定的目标，登录成功后才能使用目标提供的相关服务，下图对登录过程的一个大致描述（不太严谨，仅起示意作用）



参见上图，uSCSI 的登录过程从 uSCSICreateSession 开始，这个函数分配一个会话，然后调用 PtLogin 函数，这个函数会调用 PiCollectStgParms 函数（围在上图中列示），后者根据当前会话状态组装一个登录 PDU，并提交给出站队列，随后 PtLogin 等待在一个登录事件上。目标返回的登录 PDU 交由 PtProcessLogin 函数处理，这个函数实现了 iSCSI 对登录过程的定义。如果登陆成功，PiLoginComplete 函数会被调用，这个函数设置登录事件为有信号，唤醒 PtLogin，后者返回 uSCSICreaeSession，后者再返回调用者。

SCSI 命令执行

uSCSIPort 的客户端（一般是 uSCSI）负责收集主机系统发出的 SCSI 命令并将命令提交到 iSCSI 层，iSCSI 将 SCSI 命令封装到 PDU 中提交到目标，下图为这个流程的一个描述：



如上图所示，客户端调用 uSCSIProcessSCSICmd 函数，这个函数负责将 SCSI 命令传输到目标，PtAssemblSCSICmd 负责装配一个适当的 PDU，例如，封装即时数据、初始数据等

等，装配好的 PDU 提交到目标。PtProcessDataIn 和 PtProcessR2T 是协议的处理函数，前者处理数据读类型的 PDU：包括复制数据到用户缓冲区，完成待定命令等；后者处理数据写类型的 PDU：包括组装数据写 PDU，完成待定命令等。

使用

安装 uSCSI

uSCSI 包含如下组件：

组件	说明
uSCSIPort.sys	iSCSI 协议驱动
uSCSI.sys	磁盘驱动
uscsi.inf	磁盘驱动安装文件
devcon	安装工具
uscsictl	命令行工具，用于添加目标，连接目标

协议驱动

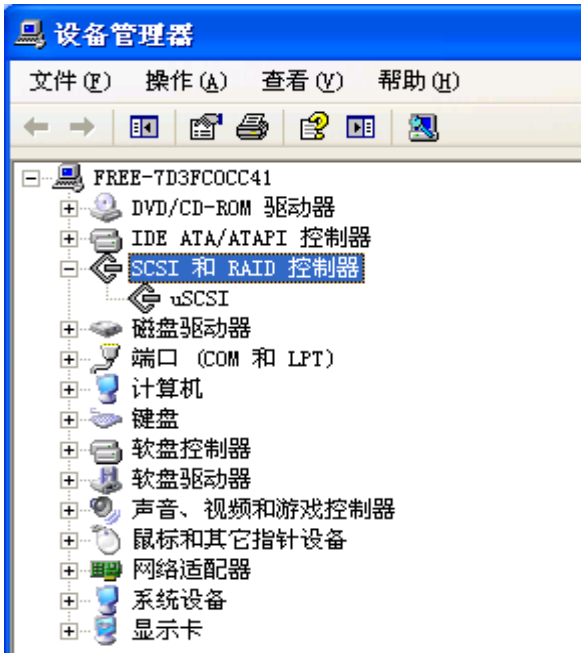
直接把 uSCSIPort.sys 拷贝到系统驱动目录下

安装磁盘驱动

使用下面命令行

devcon install uscsi.inf uSCSI\uSCSI

安装好后在设备管理器的 SCSI 和 RAID 控制器节点下看到一个 uSCSI 节点



配置目标

找一个目标软件，我用的是 StarWind，配置好一个目标后，用 uscsictl 添加目标到 uSCSI 中，例如

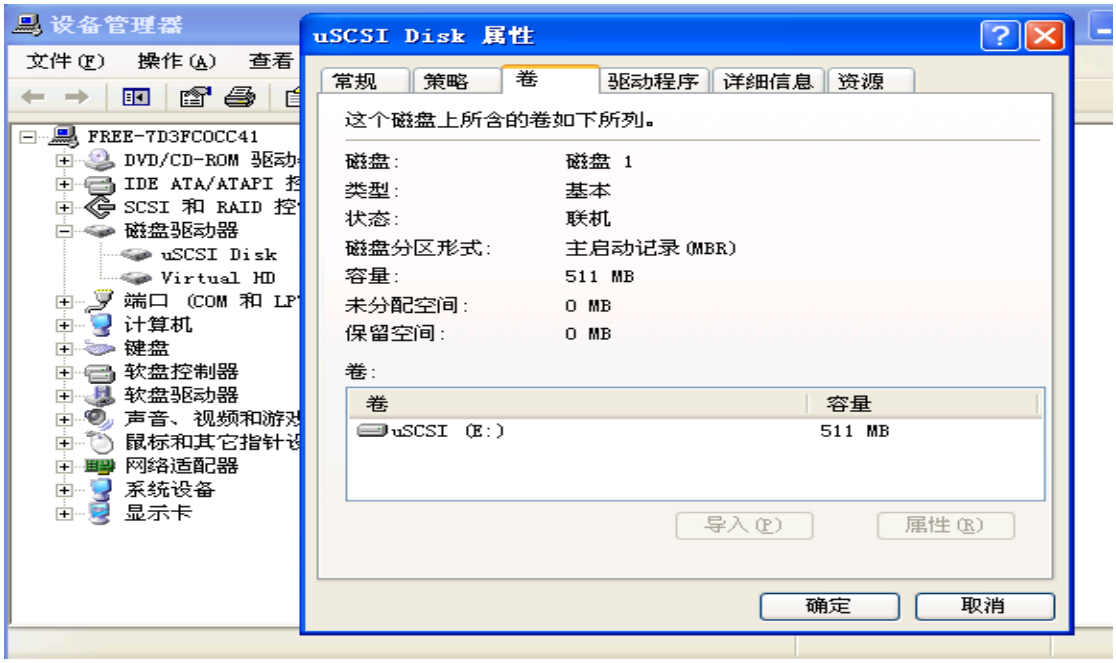
uscsictl -t iqn.com.yushang:vdisk.01 192.168.1.123

其中 iqn.com.yushang:vdisk.01 为目标名称，192.168.1.1.123 为目标地址

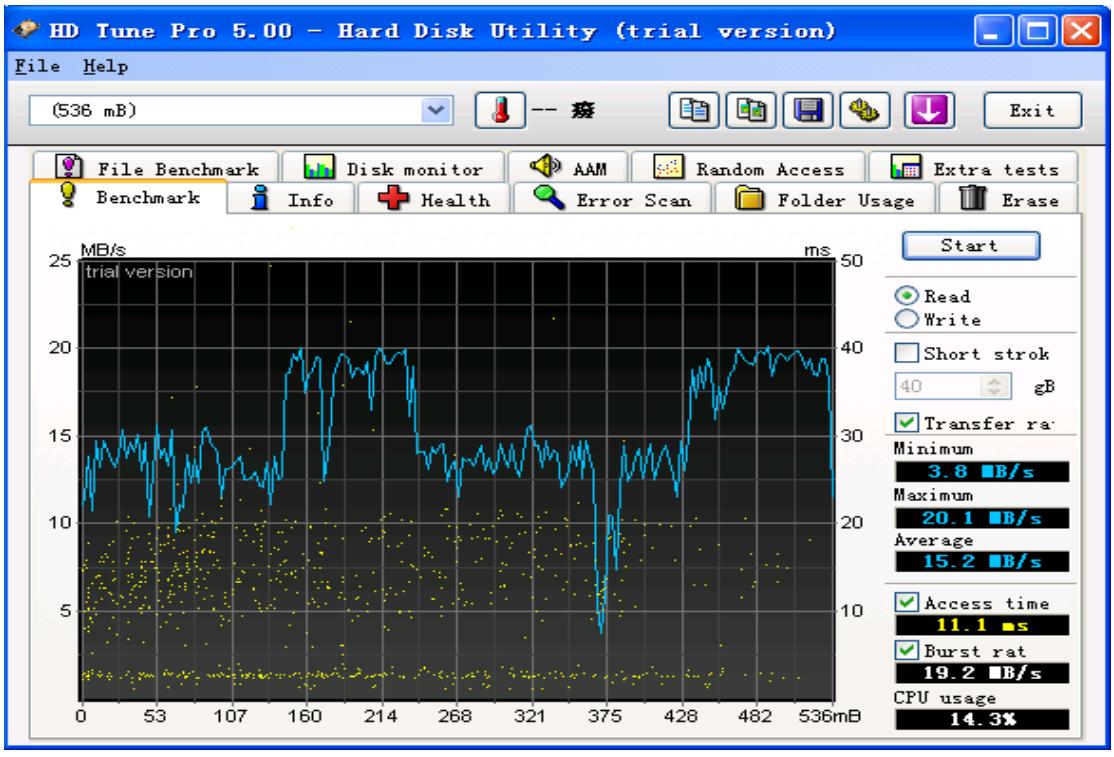
连接目标

uscsictl -c iqn.com.yushang:vdisk.01

这时可以在设备管理器中看到多了一个磁盘，如下图所示：



选择 uSCSI Disk 节点查看属性，磁盘容量为 511M。如果这是一个未格式化的盘，需要用磁盘管理控制台来添加分区并格式化，否则可以在 Windows 资源浏览器里看到多了一个磁盘。下面使用 HD Tune 对磁盘做个简单的测试：



附录

下面是 uSCSI 一些主要函数/结构列表

基本结构

结构名	说明
PDU	协议数据单元
BHS	基本协议头，这是一个 48 字节的结构
AHS	附加协议头，这是一个变长的结构，结构成员 AHSLength 记录了结构的长度，成员 AHSType 则记录了附加协议头的类型
KEY_META	
KEY_VALUE	
SESSION	会话
uCON_CTX	连接上下文
WORKER_THREAD_CTX	
uCONINFO	连接
PDU	负载数据单元
SN	序列号

API

API 是指 uSCSIPort 部分提供给 uSCSI 使用的接口

函数　VOID uSCSIAddTarget(PUCHAR TgtName , ULONG Addr , USHORT Port)
功能　添加一个目标，目前 uSCSIPort 还不支持 Discovery 会话，需要手工添加目标
参数　TgtName，目标名称
　　Addr，目标地址
　　Port，目标端口
返回值　无
函数　VOID uSCSIInitialize()
功能　uSCSI 初始化
参数　无
返回值　无
函数　PVOID uSCSICreateSession(PUCHAR Target , PUSCSI_CALL_BACK CallBack)

功能　创建 iSCSI 会话

参数　Target，目标名称

CallBack，回调函数，uSCSIPort 在适当的时候调用回调函数

返回值　成功，会话句柄，失败，NULL

函数　VOID uSCSIProcessSCSICmd (PVOID Session ,
PUCHAR Cdb ,
ULONG CdbLength ,
PUCHAR DataBuf ,
ULONG WriteSize,
ULONG ReadSize ,
UCHAR TaskAttr,
UCHAR Dir ,
PULONG Handle)

功能　处理 SCSI 命令

参数　Session，会话句柄

Cdb，SCSI 命令描述块

CdbLength，SCSI 命令描述块长度

DataBuf，数据缓冲区

WriteSize，数据写长度

ReadSize，数据读长度

TaskAttr，任务属性

Dir，数据方向，读/写/读写

Handle，可选，返回任务标识

返回值　无

函数　PLIST_ENTRY uSCSIGetTargets()

功能　返回目标列表

参数　无

返回值　目标列表

函数　PLIST_ENTRY uSCSIGetSessions()

功能　返回会话列表

参数　无

返回值　会话列表

函数　PUSCSI_SESSION_INFO uSCSIGetSessionInfo (PVOID Session)

功能　返回指定会话信息

参数　Session，会话句柄

返回值　会话信息

协议处理

函数 VOID
PtProcessResponse (
PPDU Pdu ,
PuCONINFO ConInfo
);
功能 命令响应处理函数
参数 Pdu，响应 PDU
ConInfo，PDU 链路
返回值 无
功能 VOID
PtProcessDataIn (
PPDU Pdu ,
PuCONINFO ConInfo
);
功能 数据读处理函数
参数 Pdu，数据读 PDU
ConInfo，PDU 链路
返回值 无
VOID
PtProcessR2T(
PPDU Pdu ,
PuCONINFO ConInfo
);
功能 数据请求处理函数
参数 Pdu，数据请求 PDU
ConInfo，PDU 链路
返回值 无
VOID
PtProcessTask (
PPDU Pdu ,
PuCONINFO ConInfo
);
功能 任务管理处理函数
参数 Pdu，任务管理 PDU
ConInfo，PDU 链路
返回值 无

VOID
PtProcessText(
PPDU Pdu ,
PuCONINFO ConInfo
);

功能 文本交换处理函数

参数 Pdu，文本交换 PDU
ConInfo，PDU 链路

返回值 无

VOID
PtProcessLogin(
PPDU Pdu ,
PuCONINFO ConInfo
);

功能 登录过程处理函数

参数 Pdu，登录 PDU
ConInfo，PDU 链路

返回值 无

NTSTATUS
PtLogin(
PuCONINFO ConInfo
);

功能 登录初始化

参数 ConInfo，PDU 链路

返回值 登录成功，或者失败

VOID
PtProcessLogout (
PPDU Pdu ,
PuCONINFO ConInfo
);

功能 注销处理函数

参数 Pdu，注销 PDU
ConInfo，PDU 链路

返回值 无

VOID
PtProcessReject (
PPDU Pdu ,
PuCONINFO ConInfo

```
        );
```
功能　拒绝处理函数
参数　Pdu，拒绝 PDU
　　ConInfo，PDU 链路
返回值　无
```
VOID
    PtProcessNopIn (
    PPDU Pdu   ,
    PuCONINFO ConInfo
    );
```
功能　心跳处理函数
参数　Pdu，心跳 PDU
　　ConInfo，PDU 链路
返回值　无
```
VOID
    PtNopOut (
    PuCONINFO ConInfo ,
    BOOLEAN Echo
    );
```
功能　发出心跳
参数　ConInfo，PDU 链路
　　Echo，是否要求回显
返回值　无
```
NTSTATUS
    PtInit();
```
功能　协议初始化
参数　无
返回值　初始化成功，初始化失败
```
NTSTATUS
    PtUnInit();
```
功能　协议去初始化
参数　无
返回值　去初始化成功，去初始化失败
```
VOID
    PtProcessAsyncMsg (
    PPDU Pdu   ,
    PuCONINFO ConInfo
    );
```

功能　异步消息处理函数
参数　Pdu，异步消息 PDU
ConInfo，PDU 链路
返回值　无

```
PPDU
    PtAssembleSCSICmd (
    PuCONINFO     ConInfo ,
    PUCHAR        Cdb ,
    ULONG         CdbLength ,
    PUCHAR        DataBuffer ,
    ULONG         WriteSize ,
    ULONG         ReadSize,
    UCHAR      TaskAttr,
    UCHAR      Dir
    );
```

功能　组装一个 SCSI 命令 PDU
参数　ConInfo，PDU 链路
Cdb，SCSI 命令区
CdbLength，命令区大小
DataBuffer，数据缓冲区
WriteSize，写数据大小
ReadSize，读数据大小
TaskAttr，任务属性
Dir，方向，读，写，读写
返回值　成功，返回 PDU，失败，返回 NULL

传输

```
函数　VOID
        TpRegisterPnPHandlers(
        PSESSION Session
        );
```

功能　注册 TDI 事件处理函数
函数　Session，会话
返回值　无

```
函数　PPDU
        TpAllocatePDU(
```

PuCON_CTX Ctx ,
UCHAR Opcode ,
UCHAR AHSLen ,
ULONG DataLen
);

功能 分配一个 PDU

参数 Ctx，PDU 链路上下文

Opcode，操作码

AHSLen，附加协议头大小

DataLen，数据大小

返回值 成功，返回 PDU，失败，返回 NULL

函数 VOID
TpFreePDU (
PuCON_CTX Ctx ,
PPDU Pdu
);

功能 释放一个 PDU

参数 Ctx，PDU 链路上下文

Pdu，需要释放的 PDU

返回值 无

函数 VOID
TpQueuePDU(
PuCON_CTX Ctx ,
PPDU Pdu ,
UCHAR Which
);

功能 PDU 入列

参数 Ctx，PDU 链路上下文

Pdu，需要入列的 PDU

Which，哪个 PDU 队列 ，下列值之一

WI_PDU_IN，入站 PDU 队列

WI_PDU_OUT，出站 PDU 队列

WI_PENDING_COMPLETE_CMD，待完成命令队列

WI_DATAIN，数据读队列

WI_R2T，数据请求队列

返回值 无

函数 PuCONINFO
TpBestConnection(

PSESSION Session
);

功能　返回线路状况最好的 PDU 链路

参数　Session，会话

返回值　PDU 链路

函数　PuCONINFO TpAddConInfo(PSESSION Session);

功能　添加一个 TCP 链接

参数　Session，iSCSI 会话，新增的 TCP 链接添加到这个会话中

返回值　新增的 TCP 链接

函数　PSESSION TpAllocateSession ();

功能　分配一个会话

参数　无

返回值　成功，新的会话，失败，NULL

函数　VOID TpFreeSession (PSESSION Session);

功能　释放一个会话

参数　Session，会话

返回值　无