IPS Mallikarjun Chadalapaka
Internet Draft Rob Elliott
draft-chadalapaka-commandordering-00.txt Hewlett-Packard Co.
Category: Informational-track

SCSI Command Ordering Considerations with iSCSI

Status of this Memo

Abstract

iSCSI is a SCSI transport protocol designed to run on top of TCP. The iSCSI session abstraction is equivalent to the SCSI I_T nexus, and the iSCSI session provides an ordered command delivery from the SCSI initiator to the SCSI target. This document goes into the design considerations that led to the iSCSI session model as it is defined today, relates the SCSI command ordering features defined in T10 specifications to the iSCSI concepts, and finally provides guidance to system designers on how true command ordering solutions can be built based on iSCSI.

Acknowledgements

We are grateful to the IPS working group whose work defined the iSCSI protocol. Thanks also to David Black (EMC) who encouraged the publication of this document. Special thanks are also in order for Randy Haagens (HP) for his insightful review comments.

1. Definitions and Acronyms

1.1 Definitions

- I_T nexus: As per [SAM2], the I_T nexus is a relationship between a SCSI Initiator Port and a SCSI Target Port. For iSCSI, this relationship is an iSCSI session, defined as a relationship between an iSCSI Initiator's end of the session (SCSI Initiator Port) and the iSCSI Target's Portal Group (SCSI Target Port). The I_T nexus can be identified by the conjunction of the SCSI port names; that is, the I_T nexus identifier for iSCSI is the tuple (iSCSI Initiator Port Name, iSCSI Target Port Name).

- PDU (Protocol Data Unit): The initiator and target divide their communications into messages. The term "iSCSI protocol data unit" (iSCSI PDU) is used for these messages.

- SCSI Device: This is the SAM-2 term for an entity that contains one or more SCSI ports that are connected to a service delivery subsystem and supports a SCSI application protocol. For iSCSI, the SCSI Device is the component within an iSCSI Node that provides the SCSI functionality. The SCSI Device Name is defined to be the iSCSI Name of the node.

- Session: The group of TCP connections that link an initiator with a target form a session (equivalent to a SCSI I-T nexus). A session may consist of multiple connections, and TCP connections can be added and removed dynamically from a session. The multiplicity of connections at the iSCSI level is completely hidden for the initiator SCSI layer. Across all connections within a session, a SCSI initiator port sees one and the same SCSI target port.

1.2 Acronyms

Acronym	Definition
ACA	Auto Contingent Allegiance
ASC	Additional Sense Code
ASCQ	Additional Sense Code Qualifier
CRN	Command Reference Number
IETF	Internet Engineering Task Force
ITT	Initiator Task Tag
LU	Logical Unit
LUN	Logical Unit Number
NIC	Network Interface Card
PDU	Protocol Data Unit
TMF	Task Management Function
SAM-2	SCSI Architecture Model - 2
SAN	Storage Area Network
SCSI	Small Computer Systems Interface
TCP	Transmission Control Protocol
UA	Unit Attention
WG	Working Group

2. Introduction

iSCSI is a SCSI transport protocol designed to enable running SCSI application protocols on the Internet. Given the size and scope of Internet, iSCSI thus enables some exciting new SCSI applications. Potential application areas for exploiting iSCSI’s value include -

a) Larger (diameter) Storage Area Networks (SANs) than had been possible until now.
b) Asynchronous remote mirroring
c) Remote tape vaulting

Each of these applications takes advantage of the practically unlimited distance possible between a SCSI initiator and a SCSI target that iSCSI allows. In each of these cases, because of the long delays involved, there is a very high incentive for the initiator to stream SCSI commands back-to-back without waiting for the SCSI status of previous commands. Command streaming may be employed primarily by two classes of applications - while one class may not particularly care about ordered command execution, the other class does rely on ordered command execution (i.e. there is an application-level dependency on the ordering among SCSI commands). As an example, cases b) and c) listed earlier clearly require ordered command execution - a mirroring application may not want the writes to be committed out of order on the remote SCSI target, so as to preserve the transactional integrity of the data on that target. To summarize, SCSI command streaming is extremely valuable for a critical class of applications in long-latency networks when coupled with the guarantee of ordered command execution on the SCSI target.

This document reviews the various protocol considerations in designing storage solutions that employ SCSI command ordering. This document also analyzes and explains the design intent of [iSCSI] with respect to command ordering.

3. Overview of the iSCSI Protocol

3.1 Protocol mapping description

The iSCSI protocol is a mapping of the SCSI remote procedure invocation model (see [SAM2]) over the TCP protocol.

SCSI's notion of a task maps to an iSCSI task. Each iSCSI task is uniquely identified within that I_T nexus by a 32-bit unique identifier called Initiator Task Tag (ITT). The ITT is both an iSCSI identifier of the task and a classic SCSI task tag.

SCSI commands from the initiator to the target are carried in iSCSI requests called SCSI Command PDUs. SCSI status back to the initiator is carried in iSCSI responses called SCSI Response PDUs. SCSI Data-out from the initiator to the target is carried in SCSI Data-Out PDUs, and the SCSI Data-in back to the initiator is carried in SCSI Data-in PDUs.

3.2 The I_T nexus model

In iSCSI, the SCSI I_T nexus model is a virtual abstraction, spanning one or more TCP connections. The iSCSI protocol defines the semantics in order to realize one logical flow of bidirectional communication across multiple TCP connections (as many as 2^16). The iSCSI connection multiplicity is thus completely contained at the iSCSI layer, while the SCSI layer is presented with a single I_T nexus in a multi-connection session. A session between a pair of given iSCSI nodes is identified by the session identifier (SSID) and each connection within a given session is uniquely identified by a connection identifier (CID) in iSCSI.

There are four crucial functional facets of iSCSI that together present this single logical flow abstraction to the SCSI layer across multiple iSCSI connections.

a) Ordered command delivery: SCSI commands that are striped across all the connections in the session get "reassembled" by the target iSCSI layer based on a Command Sequence Number (CmdSN) that is unique across the session, so as to make it appear as if all the commands had travelled in one flow.
b) Connection allegiance: All the PDU exchanges for a SCSI Command are required to flow on the same iSCSI connection, up to and including the SCSI Response PDU for the command. This will again hide the multi-connection nature of a session because the initiator SCSI layer will never see the PDU contents out of order (for ex., status cannot bypass data).
c) Task set management function handling: When all active tasks in a session are aborted (ABORT TASK SET) or cleared (CLEAR TASK SET) using SCSI task management functions (TMF),

[iSCSI] defines an ordered sequence of steps for the target handling the TMF which guarantees that the TMF Response arrives after the SCSI Response PDUs of all unaffected tasks are received on all the connections of the iSCSI session. This is again intended to preserve the single flow abstraction to the SCSI layer.

d) Immediate task management function handling: When a task management function is marked as "immediate" (i.e. only has a position in the command stream, but did not consume a CmdSN), [iSCSI] still defines semantics that require the target iSCSI layer to ensure that the TMF request is executed as if the commands and the TMF request were all flowing on a single logical channel. This ensures that the TMF request will act on tasks that it meant to manage.

The following sections will analyze the "Ordered command delivery" aspect in more detail, since command ordering is the focus of this document.

3.3 Ordered command delivery

3.3.1 Issues

There has been a lot of debate on this particular aspect in the IPS WG. Most of the debate was centered on two specific questions -

a) What should be the required command ordering behavior required of iSCSI implementations when there are transport errors (such as TCP checksum failures)?
b) Should [iSCSI] require initiators and targets to enforce command ordering?

3.3.2 The session guarantee

The final disposition of question a) in section 3.3.1 was reflected in [RFC3347], "iSCSI MUST specify strictly ordered delivery of SCSI commands over an iSCSI session between an initiator/target pair, even in the presence of transport errors.". Stated differently, an iSCSI digest failure, or an iSCSI connection termination must not cause the iSCSI layer on a target to allow executing the commands in an order different from that intended (as indicated by the CmdSN order) by the initiator. This design choice is enormously helpful in building storage systems and solutions that can now always assume command ordering to be a service characteristic of an iSCSI substrate.

Note that by taking the position that an iSCSI session always guarantees command ordering, [iSCSI] was indirectly implying that the principal reason for the multi-connection iSCSI session abstraction was to allow ordered bandwidth aggregation for an I_T nexus. In deployment models where this cross-connection ordering mandated by [iSCSI] is deemed expensive, a serious consideration should be given to deploying multiple single-connection sessions in stead.

3.3.3 Ordering onus

The final resolution of b) in section 3.3.1 by the iSCSI protocol designers was in favor of not requiring the initiators to use command ordering always. This resolution is reflected in dropping the ACA requirement on the initiators, and allowing ABORT TASK TMF to plug command holes etc. The net result can be discerned by a careful reader of [iSCSI] - the onus of command ordering is on the iSCSI targets, while the initiators may or may not use command ordering. iSCSI targets being the servers in the client-server model, do not really have a way to establish whether or not the client intends to take advantage of command ordering service - so the iSCSI targets simply always provide the guaranteed service. Besides this rationale, there are inherent SCSI dependencies as we shall see in building a command ordered solution that are beyond the scope of [iSCSI], to mandate the usage or otherwise.

3.3.4 Final intent

To summarize the design intent of [iSCSI] -

> The service delivery subsystem (see [SAM2]) abstraction provided by an iSCSI session can be assumed to have the intrinsic property of ordered delivery of commands under all conditions. This command ordering is across the entire I_T nexus spanning all the LUs that the nexus is authorized to access. It is the initiator’s discretion to make use of this property.

4. The Command Ordering Scenario

A storage systems designer working with SCSI and iSCSI has to consider the following protocol features in SCSI and iSCSI layers, each of which has a role to play in realizing the command ordering goal.

4.1 SCSI layer

The SCSI application layer has several tools to enforce ordering.

4.1.1 Command Reference Number (CRN)

CRN is an ordered sequence number which when enabled for a device server, increments by one for each I_T_L nexus (see [SAM2]). The one notable drawback with CRN is that there is no SCSI-generic way (such as through mode pages) to enable or disable the CRN feature. [SAM2] also leaves the usage semantics of CRN for the SCSI transport protocol, such as iSCSI, to specify. [iSCSI] chose not to support the CRN feature for various reasons.

4.1.2 Task Attributes

SAM-2 defines the following four task attributes - SIMPLE, ORDERED, HEAD OF QUEUE, and ACA. Each task to an LU may be assigned an attribute. [SAM2] defines the ordering constraints that each of these attributes conveys to the device server that is servicing the task. In particular, judicious use of ORDERED and SIMPLE attributes applied to a stream of pipelined commands could convey the precise execution schema for the commands that the initiator issues, provided the commands are received in the same order on the target.

4.1.3 Auto Contingent Allegiance (ACA)

ACA is an LU-level condition that is triggered when a command (with the NACA bit set to 1) completes with CHECK CONDITION and that prevents any commands other than those with the ACA attribute from executing until the CLEAR ACA task management function is executed, while blocking all the other tasks in the task set. See [SAM2] for the detailed semantics of ACA. Since ACA is closely tied to the notion of a task set, one would ideally have to select (by setting the TST bit to 1 in the control mode page of the LU) the scope of the task set to be per-initiator in order to prevent command failures in one I_T_L nexus from impacting other I_T_L nexuses through ACA.

4.1.4 UA interlock

When UA interlock is enabled, the logical unit does not clear any standard unit attention condition reported with autosense and in addition, establishes a unit attention condition when a task is terminated with one of BUSY, TASK SET FULL, or RESERVATION CONFLICT statuses. This so-called "interlocked UA" is cleared only when the device server executes an explicit REQUEST SENSE ([SPC3]) command from the same initiator. From a functionality perspective, the scope of UA interlock today is slightly different from ACA's because it enforces ordering behavior for completion statuses other than CHECK CONDITION, but otherwise conceptually has the same design intent as ACA. On the other hand, ACA is somewhat more sophisticated because it allows special "cleanup" tasks (ones with ACA attribute) to execute when ACA is active. One of the principal reasons UA interlock came into being was that SCSI designers wanted a command ordering feature without the side effects of using the aforementioned TST bit in the control mode page.

4.2 iSCSI layer

As noted in section 3.2 and section 3.3, the command ordering that iSCSI enforces per iSCSI session using the CmdSN is an attribute of the SCSI transport layer. Note that any command ordering solution that seeks to realize ordering from the initiator SCSI layer to the target SCSI layer would be of practical value only when the command ordering is guaranteed by the SCSI transport layer. In other words, the related SCSI application layer protocol features such as ACA etc. are based on the premise of an ordered SCSI transport. Thus iSCSI's command ordering is the last piece in completing the puzzle of building solutions that rely on ordered command execution, by providing the crucial guarantee that all the commands handed to the initiator iSCSI layer will be transported and handed to the target SCSI layer in the same order.

5. Connection failure considerations

[iSCSI] mandates that when an iSCSI connection fails, the active tasks on that connection must be terminated if not recovered within a certain negotiated time limit. When an iSCSI target does terminate some subset of tasks, there is a danger that the SCSI layer would simply move on to the next tasks waiting to be processed and execute them out-of-order unbeknownst to iSCSI. To preclude this danger, [iSCSI] further mandates the following -

a) The tasks terminated due to the connection failure must be internally terminated by the iSCSI target "as if" due to a CHECK CONDITION. The "as if" is meaningful because this particular completion status is never communicated back to the initiator, but is required because if the initiator were using ACA as the command ordering mechanism of choice, a SCSI-level ACA will be triggered due to this mandatory CHECK CONDITION. This addresses the aforementioned danger.

b) After the tasks are terminated due to the connection failure, the iSCSI target must report a unit attention condition on the next command processed on any connection for each affected I_T_L nexus of that session. This is required because if the initiator were using UA interlock as the command ordering mechanism of choice, a SCSI-level UA will trigger a UA-interlock. This again addresses the aforementioned danger. iSCSI targets must report this UA with the status of CHECK CONDITION, and the ASC/ASCQ value of 47h/7Fh ("SOME COMMANDS CLEARED BY ISCSI PROTOCOL EVENT").

6. Implementation considerations

In general, command ordering is automatically enforced if targets and initiators comply with the iSCSI specification. However, here are certain things for the iSCSI initiators and targets to take note of.

a) iSCSI initiators may proactively seek to preclude scenarios that would normally lead to out-of-order command execution even when they have designed their systems never to execute commands out of intended order. This is simply because the SCSI command ordering features such as UA interlock are likely to be costlier in performance when they are allowed to be triggered. [iSCSI] provides enough guidance on how to implement this proactive detection of transport errors.

b) The whole notion of command streaming does of course assume that the target in question supports command queueing. An iSCSI target desirous of supporting command ordering solutions should ensure that the SCSI layer on the target supports command queuing. Especially the remote backup (tape vaulting) applications that iSCSI enables make a compelling case that tape devices must also start supporting command queuing.

c) An iSCSI target desirous of supporting high-performance command ordering solutions that involve specifying a description of execution schema should ensure that the SCSI layer on the target in fact does support the ORDERED and SIMPLE task attributes.

d) There is some consideration of expanding the scope of UA interlock to encompass CHECK CONDITION status and thus make it the only required command ordering functionality of implementations to build command ordering solutions. Until this is resolved in T10, the currently defined semantics of UA interlock and ACA warrant implementing both features by iSCSI targets desirous of supporting command ordering solutions.

7. IANA Considerations

This document does not have any IANA considerations.

8. Security Considerations

This document does not have any security considerations.

9. References and Bibliography

9.1 Normative References

[iSCSI] J. Satran et. al. draft-ietf-ips-iscsi-20.txt (work in progress)

[RFC790] J. Postel, ASSIGNED NUMBERS, September 1981.

[RFC793] TRANSMISSION CONTROL PROTOCOL, DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION, September 1981.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", RFC 2026, October 1996.

[RFC2119] Bradner, S. "Key Words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2434] T. Narten, and H. Avestrand, "Guidelines for Writing an IANA Considerations Section in RFCs.", RFC2434, October 1998.

[SAM] ANSI X3.270-1998, SCSI-3 Architecture Model (SAM).

[SAM2] T10/1157D, SCSI Architecture Model - 2 (SAM-2).

[SBC] NCITS.306-1998, SCSI-3 Block Commands (SBC).

[SPC3]T10/1416-D, SCSI Primary Commands-3.

9.2 Informative References:

[RFC3347] M. Krueger et. al., "iSCSI Requirements and Design Considerations"

10. Authors' Addresses

Mallikarjun Chadalapaka
Hewlett-Packard Company
8000 Foothills Blvd.
Roseville, CA 95747-5668, USA
Phone: +1.916.785.5621
E-mail: cbm@rose.hp.com

Rob Elliott
Hewlett-Packard Company
MC 150801
PO Box 692000
Houston, TX 77269-2000 USA
Phone: +1.281.518.5037
E-mail: elliott@hp.com

Comments may be sent to Mallikarjun Chadalapaka.

Full Copyright Statement

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.