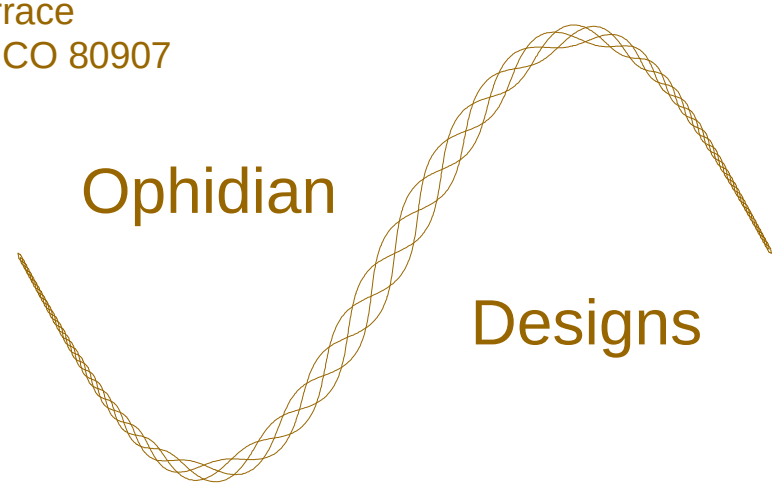
SVP Overview

SCSI VI Protocol Overview

Permission is granted to members of NCITS, its technical committees, and their associated task groups to reproduce this document for the purposes of NCITS standardization activities without further permission, provided this notice is included. All other rights reserved. Any commercial or for-profit duplication is strictly prohibited.

Edward A. Gardner
Ophidian Designs
1262 Hofstead Terrace
Colorado Springs, CO 80907

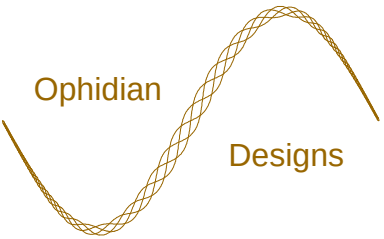
eag@ophidian.com
+1 719 593-8866



Ophidian Designs

Agenda

- **VI Architecture Summary**
- Command Mapping
- Flow Control
- Miscellaneous Issues
- Future Schedule



VI Architecture Summary

- Virtual Interfaces
- VI Connections
- Memory Registration
- Data Transfer Operations
- Reliability Levels
- References

Ophidian Designs

Virtual Interfaces

- A VI (Virtual Interface) consists of a send Work Queue and a receive Work Queue.
- Send Work Queue descriptors identify requests.
- Receive Work Queue descriptors identify receive buffers.
- Completion Queues (optional) allow shared processing of multiple Work Queues.
  - Not relevant to SVP.

Ophidian Designs

VI Connections

- An association between a pair of VIs that allows messages (packets) sent on one VI to be received at the other.
- Connection establishment:
  - Asymmetric: client requests connection to server.
  - Symmetric: defined as an extension (post v1.0).
    - Not relevant to SVP.
- Connection operation:
  - Symmetric: either endpoint may request any data transfer operation supported on that connection.

Ophidian Designs

Memory Registration

- VI Consumer (e.g. SVP Initiator) identifies virtual memory regions to VI Provider (NIC).
  - Locks pages in physical memory.
- Returns memory handle for region.
  - Memory access requires handle, virtual address (equivalent to offset) and length.
- Memory regions may be registered once, then used many times (e.g. by many SCSI commands).

Ophidian Designs

Data Transfer Operation

- Send / Receive
  - Receiving endpoint posts receive buffer descriptor to its receive Work Queue.
  - Sending endpoint posts send buffer descriptor (send request) to its send Work Queue.
  - VI Providers transfer data.
  - Both endpoints notified on completion (both descriptors marked Done).

Ophidian Designs

Data Transfer Operation

- RDMA Write setup:
  - Local endpoint is source, remote is destination.
  - Remote endpoint registers a memory region (data transfer destination), then sends the region's memory handle and virtual address to local endpoint.
  - Local endpoint registers a memory region (data transfer source).

Ophidian Designs

Data Transfer Operation

- RDMA Write operation:
  - Local endpoint posts RDMA Write descriptor (RDMA Write request) to its send Work Queue.
    - Includes memory handles and virtual addresses for local buffer (data transfer source) and remote buffer (data transfer destination) plus transfer length.
  - VI Providers transfer data.
  - Only local endpoint notified on completion (descriptor marked Done).

Ophidian Designs

Data Transfer Operation

- RDMA Write with Immediate Data
  - Similar to RDMA Write without Immediate Data.
  - Transfers 32-bit Immediate Data in addition to buffer-to-buffer data transfer.
  - Consumes receive Work Queue descriptor at remote (destination) endpoint.
  - Both endpoints notified on completion (both descriptors marked Done).

Ophidian Designs

Data Transfer Operation

- RDMA Read setup:
  - RDMA Read is optional.
  - Local endpoint is destination, remote is source.
  - Remote endpoint registers a memory region (data transfer source), then sends the region's memory handle and virtual address to local endpoint.
  - Local endpoint registers a memory region (data transfer destination).

Ophidian Designs

Data Transfer Operation

- RDMA Read operation:
  - Local endpoint posts RDMA Read descriptor (RDMA Read request) to its send Work Queue.
    - Includes memory handles and virtual addresses for local buffer (data transfer destination) and remote buffer (data transfer source) plus transfer length.
  - VI Providers transfer data.
  - Only local endpoint notified on completion (descriptor marked Done).
  - RDMA Read is optional.

Ophidian Designs

Reliability Levels

- Unreliable Delivery
  - Sends and RDMA Write data delivered at most once, but may be lost or delivered out of order.
  - RDMA Reads prohibited.
  - Errors (e.g. lost messages) do not break connections.

Ophidian Designs

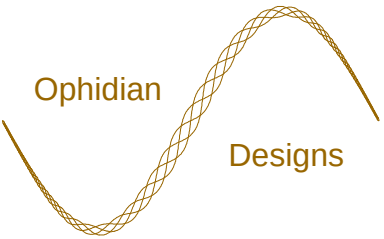
Reliability Levels

- Reliable Delivery
  - Sends and RDMA Write data delivered exactly once and in order (or connection is broken).
  - Send Work Queue descriptor marked Done while data is still in flight.
  - Send / Receive: send Work Queue descriptor marked Done before receive Work Queue descriptor is marked Done.
  - RDMA Reads optional; may complete out of order relative to Sends and RDMA Writes.
  - Any error breaks the connection.

Ophidian Designs

Reliability Levels

- Reliable Reception
  - Sends and RDMA Write data delivered exactly once and in order (or connection is broken).
  - Send Work Queue descriptor marked Done after data delivery has been confirmed.
  - Send / Receive: receive Work Queue descriptor marked Done before send Work Queue descriptor is marked Done.
  - RDMA Reads optional. The specification is not clear on whether there are any ordering guarantees beyond those for Reliable Delivery.
  - Any error breaks the connection.



References

- Intel VI web site:
  - www.intel.com/design/servers/vi/index.htm
- VI Architecture web site:
  - www.viarch.org
  - Appears to be broken.
- VI Developer Forum web site:
  - www.vidf.org
  - Under development.
- Orca web site:
  - www.orca.com/via/docs/

Ophidian Designs

Agenda

- VI Architecture Summary
- **Command Mapping**
- Flow Control
- Miscellaneous Issues
- Future Schedule

Ophidian Designs

Command Mapping

- Basic Concepts
- Reliability Level Use
- SCSI Read command choices
- SCSI Write command choices
- Direct Device Access

Ophidian Designs

Basic Concepts

- VI Connections between SVP (SCSI) Initiators and SVP (SCSI) Targets.
- SCSI Commands, Status, Task Management use Send / Receive operations.
  - Based on FCP formats
- Each VI Connection is independent:
  - Each VI Connection to a SVP (SCSI) Target is a separate SVP (SCSI) Initiator.
  - Commands and Task Management share the same VI Connection.

Ophidian Designs

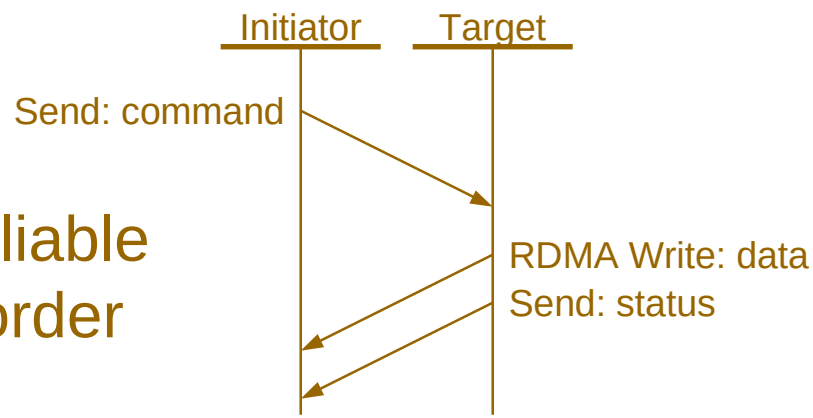
Reliability Level Use

- Assume Reliable Delivery is predominant.
  - Want to use RDMA Read and in-order delivery.
- If special circumstances dictate, may use Reliable Reception.
  - E.g., where FCP uses FCP_CONF.
- Neither prohibit use of Unreliable Delivery nor try to make it reliable.
  - Might be usable for video data or other so-called isochronous data.
  - Only recovery for many errors is to disconnect and start over (equivalent to Abort Task Set).

Ophidian Designs

SCSI Read Command Choice 1

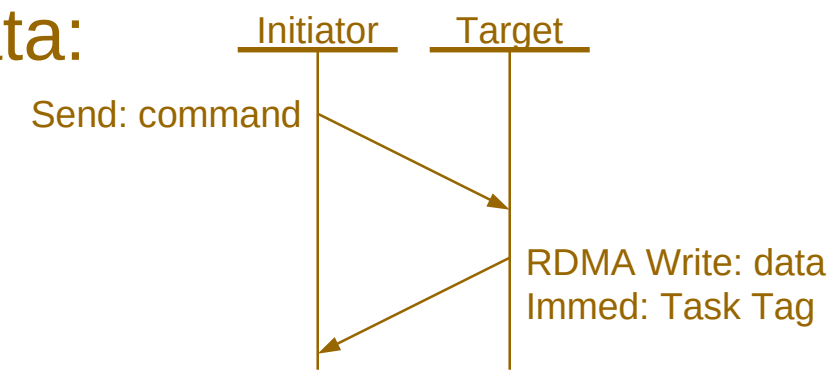
- Initiator sends Read command using Send.
  - Includes Read data buffer's memory handle and virtual address.
- Target returns Read data using RDMA Write.
- Target returns status using separate Send.
  - Target may post Send before RDMA Write completes.
  - Reliable Delivery and Reliable Reception guarantee in order delivery.



Ophidian Designs

SCSI Read Command Choice 2

- Initiator sends Read command using Send.
  - Includes Read data buffer's memory handle and virtual address.
- If success and no autosense data:
  - Target returns Read data using RDMA Write.
  - Immediate data is Task Tag.
- If error or autosense data:
  - Target follows choice 1.



Ophidian Designs

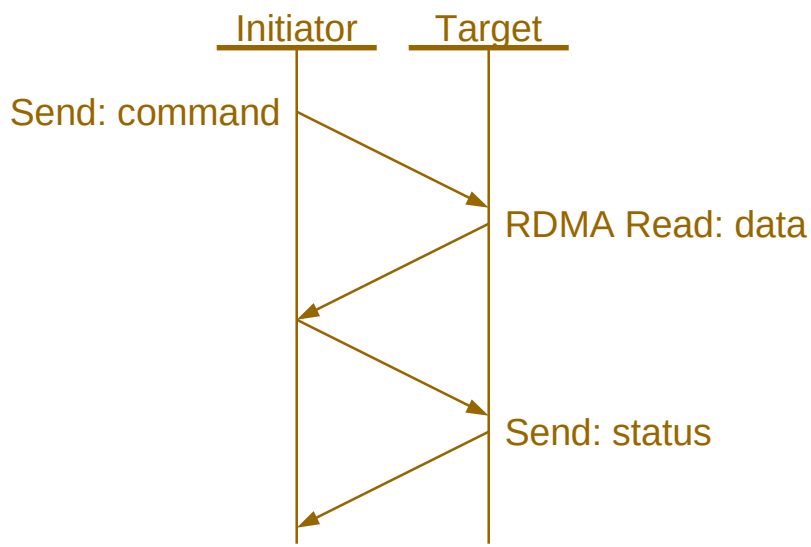
SCSI Read Command Choices

- No difference in round-trip communication with Reliable Delivery.
  - Choice 1 might require an extra round-trip with Reliable Reception (specification ambiguous).
- No difference in receive Work Queue descriptors used.
- Choice 1 required regardless (for errors).
- Choice 2 might be advantageous with Unreliable Delivery connections.
- Recommendation: only allow choice 1.

Ophidian Designs

SCSI Write Command Choice 1

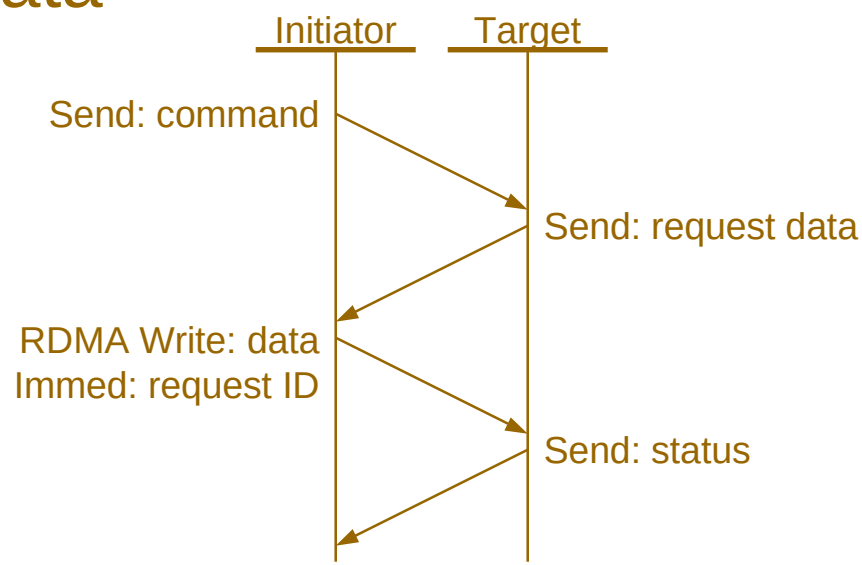
- Initiator sends Write command using Send.
  - Includes Write data buffer's memory handle and virtual address.
- Target fetches Write data using RDMA Read.
- Target returns status using Send.
- Problem: RDMA Read is optional, might not be available with all VI Providers.



Ophidian Designs

SCSI Write Command Choice 2

- Emulate RDMA Read.
- Initiator sends Write command using Send.
- Target requests Write data using Send.
- Initiator returns Write data using RDMA Write.
- Target returns status using Send.
- Requires Initiator software intervention to fetch Write data.



Ophidian Designs

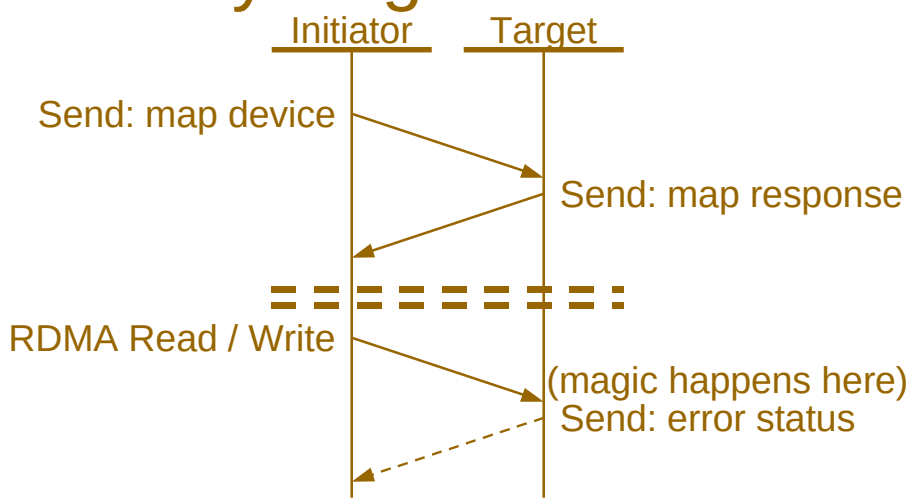
SCSI Write Command Choices

- If RDMA Read is available, Choice 1 is always better than Choice 2.
  - Write data supplied by NIC hardware rather than Initiator software.
- If RDMA Read is not available, Choice 2 is only option.
- Can we mandate RDMA Read?
- Recommendation: assume we can mandate RDMA Read, then see if anyone objects.

Ophidian Designs

Direct Device Access

- General idea is to use RDMA Read to read data from a device, RDMA Write to write data to a device. Goal is lower overhead, etc.
- Would require advance setup to map some portion of device to a “memory” region.
- Requires significant development and enhancement to VI Architecture.



Ophidian Designs

Direct Device Access

- Problems:
  - SCSI device access vs. RDMA memory access.
    - Similarities for disks, but also major differences.
    - Tapes and other non-direct access devices?
  - Error handling and reporting differences.
  - Flow control: RDMA Write sends data at multi-gigabit wire speeds to electromechanical devices.
- When others have solved these problems, Direct Device Access belongs in SVP.
- SVP's schedule is aggressive. This is probably an SVP-2 feature.

Agenda

- VI Architecture Summary
- Command Mapping
- **Flow Control**
- Miscellaneous Issues
- Future Schedule

Flow Control

- Virtual Interface Architecture Specification, Version 1.0, page 15:
  - VI Consumers are responsible for managing flow control on a connection. The VI Consumer on the receiving side must post a Receive Descriptor of sufficient size before the sender's data arrives. If the Receive Descriptor at the head of the queue is not large enough to handle the incoming message, or the Receive Queue is empty, an error will occur. The connection may be broken if it is intended to be reliable. See section 2.5.

Ophidian Designs

Flow Control

- SCSI Initiators may issue an arbitrarily large number of commands, expecting to receive Task Set Full status if the Initiator issues more commands than the SCSI Target or Logical Unit can accept.
- This is common or normal behavior in some system environments.
- This behavior is guaranteed to overrun the SVP (SCSI) Target's receive Work Queue and break the VI Connection between the SVP (SCSI) Initiator and SVP (SCSI) Target.

Ophidian Designs

Flow Control Requirements

- Requirements:
  - An SVP Target must post some minimum number of receive buffers on each connection (i.e., for each initiator).
  - Information about this minimum must be available to the SVP Initiator.
  - The SVP Initiator must not send more commands or task management requests than this minimum.
- The converse is also required: an SVP Target must not overrun an SVP Initiator.

Ophidian Designs

SAM Communication Model

- SAM-2 draft revision 12, page 18:

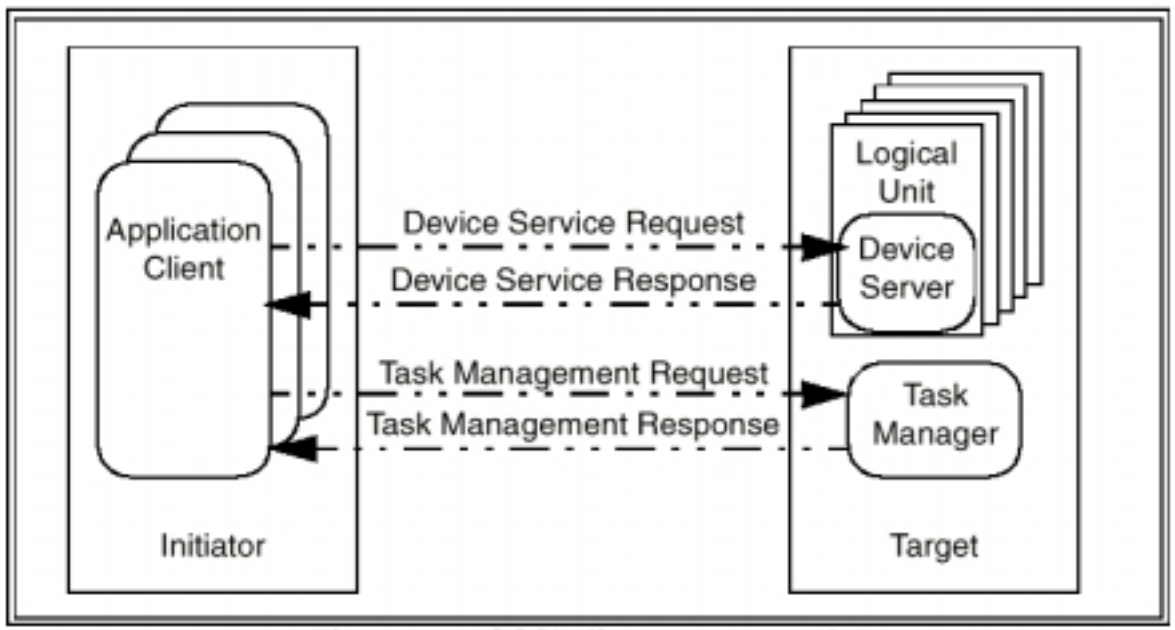
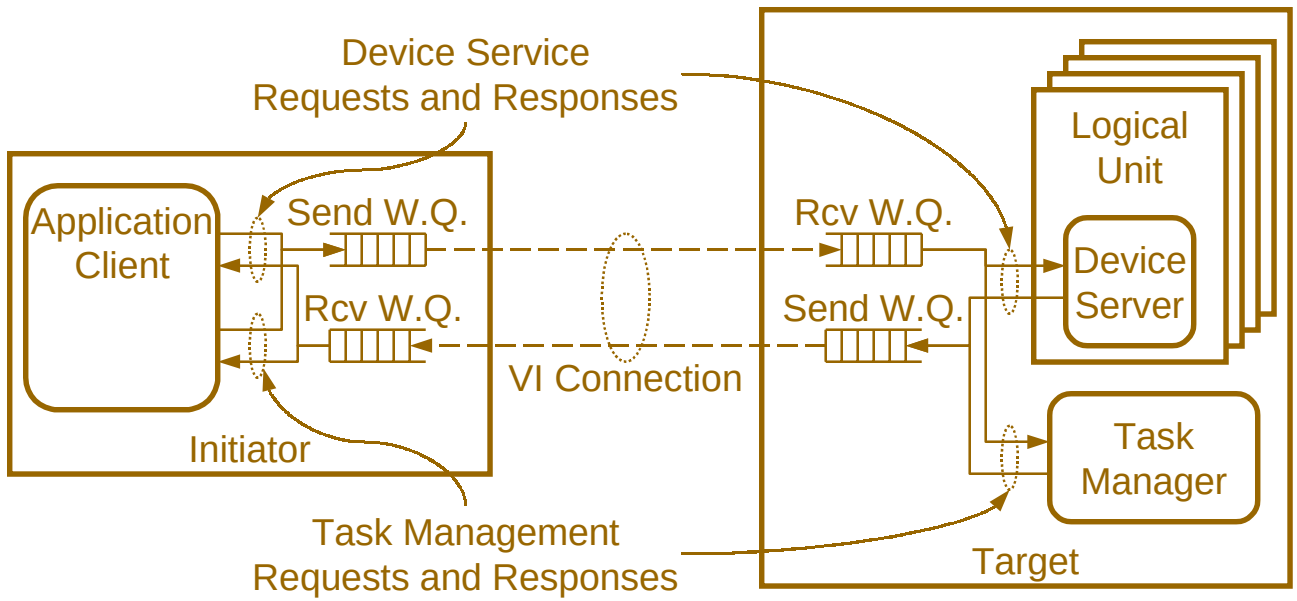


Figure 6 — SCSI client-server model

Ophidian Designs

SVP Communication Model



Ophidian Designs

SVP Communication Model

- SVP Flow Control operates between SVP Initiators and SVP Targets.
  - Analogous to R_RDY flow control in FCP.
  - Independent of Task Set Full mechanism.
- A command is transferred from SVP Initiator to SVP Target using SVP Flow Control.
- The SVP Target then transfers the command to a Logical Unit or Device Server, where the command may incur Task Set Full status.

Ophidian Designs

SVP I-T Flow Control

- SVP Initiators maintain TRD_Count:
  - Target Receive Descriptor Count.
  - Minimum number of descriptors the Initiator knows are present in the Target's receive Work Queue.
  - Initiator decrements TRD_Count whenever it sends (posts a send descriptor containing) a command or task management request to the target.
  - Initiator may not send commands or task management requests when TRD_Count is zero.

Ophidian Designs

SVP I-T Flow Control

- TRD_Count:
  - Initial value determined during VI Connection establishment. (E.g., passed in first message on a new connection).
  - Decremented whenever Initiator sends a command or task management request to Target.
  - Each message from target contains a value that the Initiator uses to adjust TRD_Count (e.g. adds to TRD_Count):
    - Command status.
    - Task management responses.
    - Special purpose “Adjust TRD_Count” message.

Ophidian Designs

SVP I-T Flow Control

- If TRD_Count reaches zero, Target should promptly provide a receive descriptor. Example:
  - If an Initiator sends a command that consumes the Target's last receive Work Queue descriptor (TRD_Count = 0), and the Target is unable to allocate additional descriptors, the Target should (shall?) immediately return Task Set Full status and specify TRD_Count_Increment = 1.
    - Final specification of Task Set Full vs. Busy status deferred until SAM-2 has been completed.

Ophidian Designs

SVP T-I Flow Control

- TRD_Count ensures the Initiator does not overrun the Target's receive Work Queue.
- We also need a mechanism to ensure the Target does not overrun the Initiator's receive Work Queue.
- We could use a similar mechanism (e.g. IRD_Count), with some special rules to avoid deadlock (TRD_Count = IRD_Count = 0).

Ophidian Designs

SVP T-I Flow Control

- Alternative to IRD_Count:
  An SVP Initiator shall post to its receive Work Queue at least as many descriptors as the sum of:
  - One receive descriptor for each of its outstanding commands.
  - One receive descriptor for any outstanding task management request.
  - One additional receive descriptor to allow the target to send an “Adjust TRD_Count” message.
- Recommendation: use this alternative.

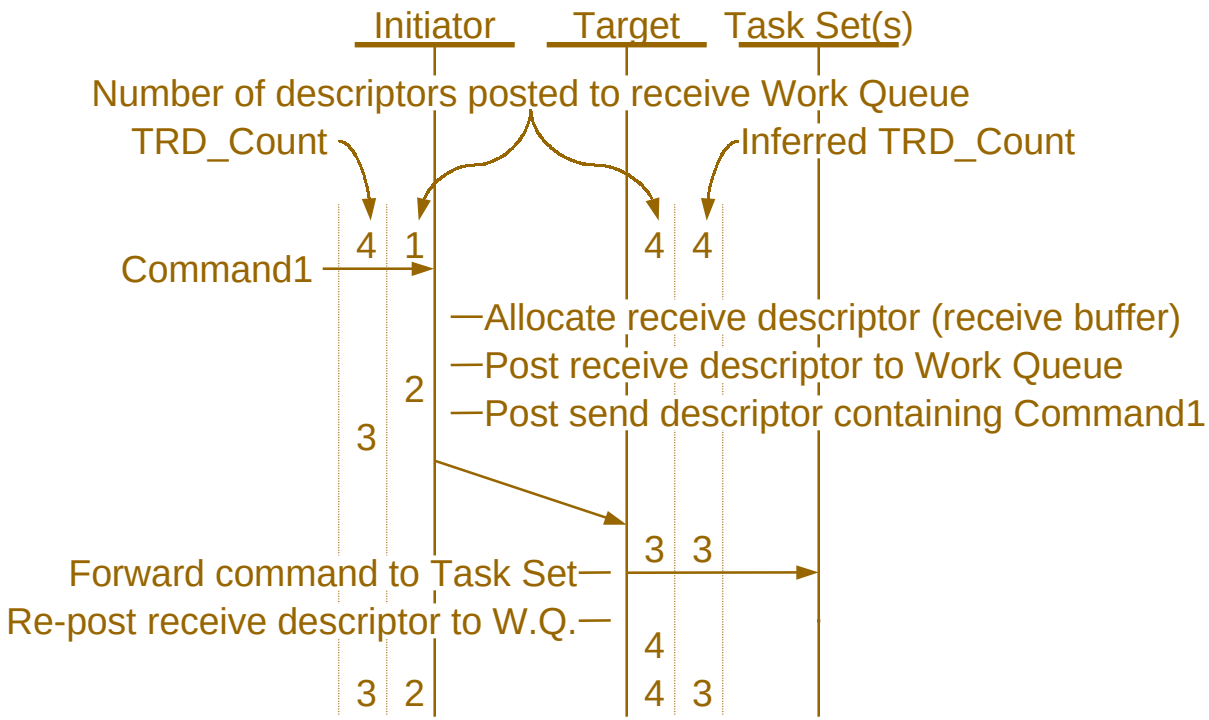
Ophidian Designs

Flow Control Example

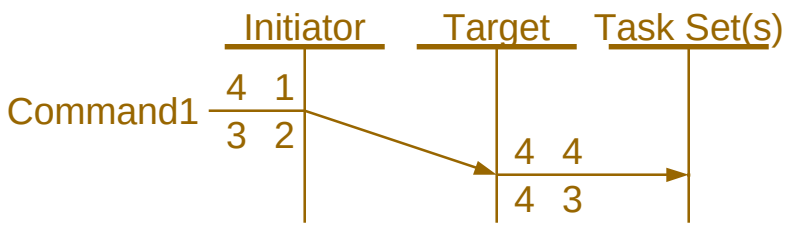
- Assume the Target is a storage array or RAID controller with a large number of LUNs.
  - Total outstanding commands to all LUNs may be very large (hundreds to thousands).
  - Supports a large number of hosts or Initiators.
  - Number of Target receive Work Queue descriptors available for each Initiator or VI Connection is much smaller than the maximum queue depth.
    - Receive Work Queue descriptors and Task Set (command queue) slots are different data structures.
    - This example uses 4 descriptors per connection, values for real devices would probably be larger.

Ophidian Designs

Command Example

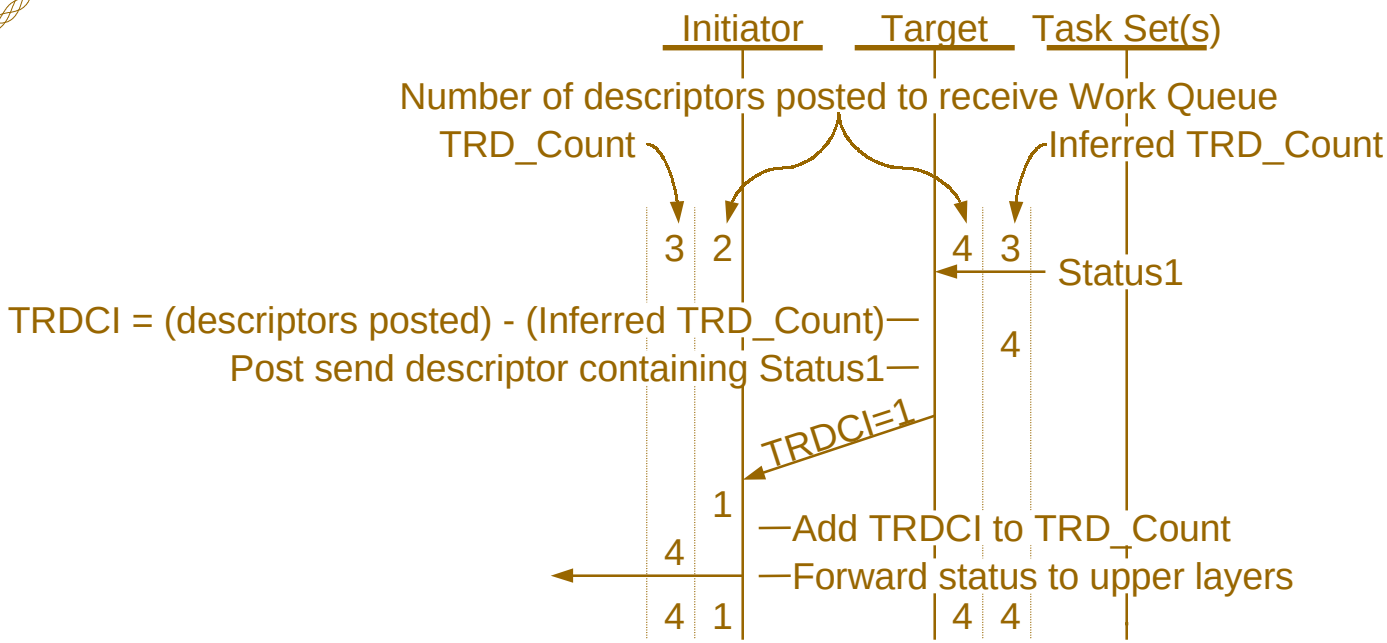


Summarized as:

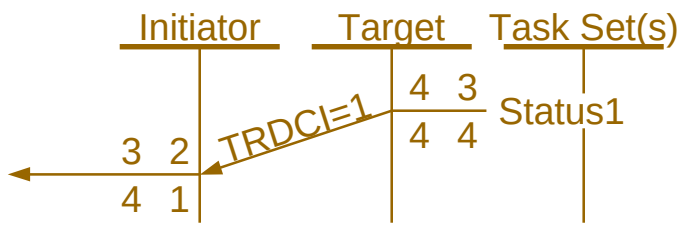


Ophidian Designs

Status Example

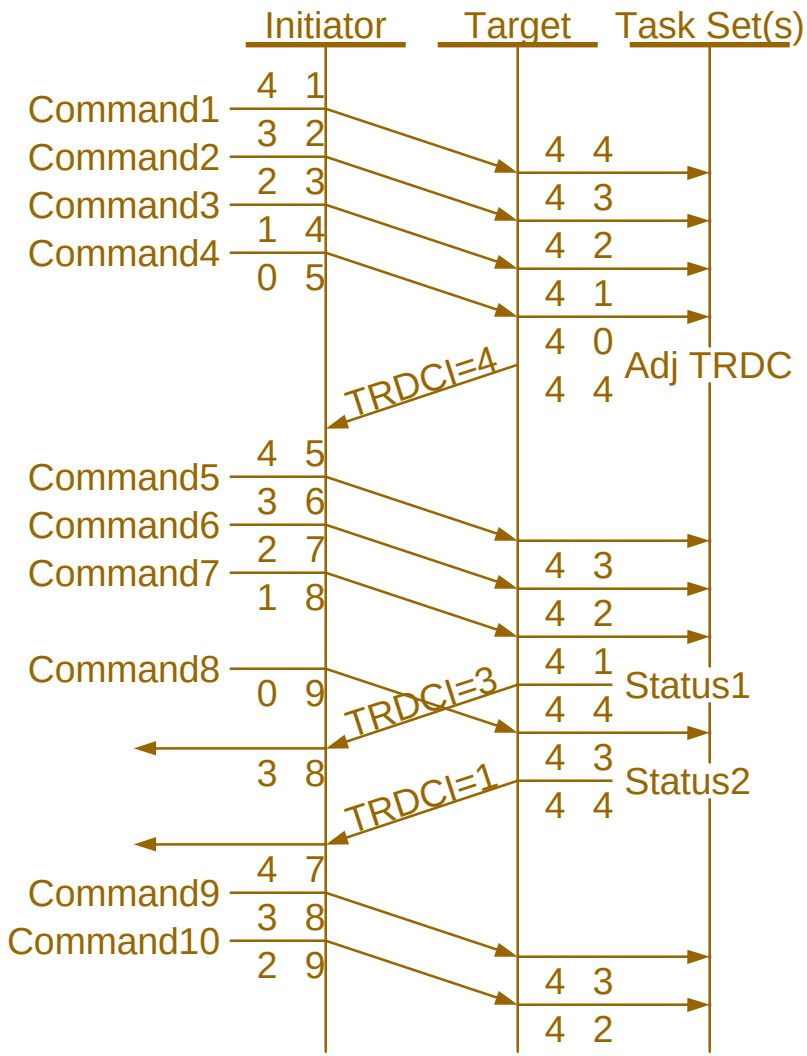


Summarized as:



Ophidian Designs

Flow Control Example



Ophidian Designs

Agenda

- VI Architecture Summary
- Command Mapping
- Flow Control
- **Miscellaneous Issues**
- **Future Schedule**

Ophidian Designs

Miscellaneous Issues

- Task Tag size:
  - Recommend 32 bits.
- Untagged Task support:
  - Untagged Tasks unnecessary, since mandatory Autosense immediately clears any CA condition (same as FCP).
  - Recommend not supported.
  - Alternative: reserve Task Tag value zero to designate an untagged task.
- Linked commands:
  - Recommend not supported.

Ophidian Designs

Deferred Issues

- Recommend deferring the following issues, hoping that VIDF or someone else will address them:
  - Discovery: how does an SVP Initiator determine whether a potential VI Connection supports SVP or some other application protocol.
  - Security: how does an SVP Target determine that a VI Connection is from an authorized SVP Initiator rather than from an arbitrary application program.

Ophidian Designs

Future Schedule

- No plans to discuss SVP during the January, 2000 T10 plenary week (Brisbane, Australia).
- Anticipate first draft of SVP available in late January, 2000.
- Request review via email and during the subsequent T10 plenary week (March 6-10, 2000, Dallas, TX).
- Goal is completion by end of year 2000.