USB Attached SCSI Protocol (UASP)

Curtis E. Stevens
Sr. Staff Engineer Standards & Specifications
Western Digital Technologies, Inc.
Agenda

• Justification for New Protocol Development
• UAS Overview
• Device Discovery
• SCSI Support
New Protocol Goals & Justification
Why A New Protocol

- USB 1.1 completed in 1998
- BOT (Bulk-Only Transport) completed development in 1999, over 10 years ago
  - Designed to support USB 1.1
  - USB throughput was approximately the same as a parallel port
  - Mass storage devices were mainly Flash dongles, Floppy drives, and CD-ROM drives
  - Speed/capacity ratios remained low
    - Devices only needed to receive and process one command at a time
Why A New Protocol (continued)

• USB 2.0 completed in 2000
  • BOT not updated
  • Speed/capacity ratio allowed larger devices including hard disk to become acceptable
    • Bus speed was still slower than many devices making command queuing not necessary

• Attached devices continue to speed up and the USB bus remains a limitation

• USB 3.0 completed in 2008
  • Speed/capacity ratio now allows much larger devices including RAID
    • The USB bus no longer is slower than the attached device
  • BOT protocol has system challenges which limit full bus utilization

New protocol is needed
UAS Development Goals

• Address system limitations due to protocol overhead
  • Enable maximum asynchronous processing

• Increase the efficiency of devices
  • Allow queuing
  • Enable Task Management to support management of many commands

• Enable device command and control capabilities using today’s standards
  • SPC-4, SBC-3, and SAM-5
  • Existing performance driver stacks
• Round trip times in non-queuing drivers
  • Time from command completion to receiving the next command can be longer than the actual command

• Round trips between BOT driver and USB driver
  • There are normally 3 round trips between the BOT driver and the USB driver for each command
  • Each round trip is one or more USB bus transactions
  • Each round trip has to completed before the next one begins
  • Bus utilization is lost when the USB driver completes a round trip and is waiting for the next
UAS Capabilities – USB 2.0

• Provides a modest increase in bus utilization
• Provides a mechanism for the device to store several commands and optimize the use of its media
• Complies with current command and transport standards
• Eliminates much of the round trip time between command phases
  • Hidden behind queuing
• Switches the storage driver to a queuing model
UAS Capabilities – USB 3.0

- Maximizes bus utilization through the use of streams
- Provides a mechanism for the device to store several commands and optimize the use of its media
- Complies with current command and transport standards
- Eliminates much of the round trip time between command phases
  - Hidden behind queuing
- Switches the storage driver to a queuing model
USB Attached SCSI Overview
Resources

• 4 pipe model
  • Command, Status, Data-in, Data-out
  • Protocol does not mix commands, data, or status
  • Eliminates the need to wait or for the next command phase

• Pipe Usage Descriptor
  • BOT relied on pipe type for usage
  • UAS has several pipes with an explicit declaration about the usage of each pipe
  • Allows for vendor specific pipes to be present
    • This case not well handled by BOT
Typical Command Sequence – USB 2.0

• Setup Status pipe to receive status
• Issue commands on the Command pipe
  • Several commands can be issued
• When status is received
  • Re-setup the Status pipe with a new buffer
  • If the packet is a SENSE IU, then complete the command with the associated tag
  • If the packet is a READ READY IU, then setup the DATA-IN pipe to transfer data from the device to the host
  • If the packet is a WRITE READY IU, then setup the DATA-OUT pipe to transfer data from the host to the device
Typical Command Sequence – USB 3.0 -- For each command --

- Assign a stream ID
- Setup a status buffer on the Status pipe with the associated stream ID
- If the command reads data, then setup a read buffer on the DATA-IN pipe with the associated stream ID
- If the command writes data, then setup a write buffer on the DATA-OUT pipe with the associated stream ID
- Send the command with the stream ID as the command tag
  - The Command pipe is NOT a streams pipe
Typical Command Sequence – USB 3.0
-- When status is received --

- The stream ID indicates the command that has completed

- If the command required data transfer, then ensure that the transfer has completed
  - If the transfer is not completed, then either the data is still in flight (e.g., the USB driver has received the data but not completed the outstanding transaction to the UAS driver)
  - Or, there is a data underrun
    - In this case, the UAS driver reports a service delivery subsystem error to the layer above
  
- Report command completion to the level above the UAS driver
• Commands are sent to the device without waiting for data transfers or individual statuses
  • The device will return a queue full on the status pipe when it is no longer able to accept commands

• The device selects commands to process
  • The host can set command priority

• The device drives the usage of the Status, Data-in, and Data-out pipes

• The host may abort individual commands in the device command queue
• The host needs to be sure a clean up loose transactions
  • Although UAS requires the data transfer to be acknowledged as complete on the data pipe before status is reported, the asynchrony of the UAS protocol may cause the UAS driver to see status reported before the data transfer is reported as complete
  • This happens and needs to be accounted for in the UAS driver
• If the status reports an error, then the host needs to clean up loose transactions
  • Data transfer may still be outstanding
Figure Notation

ITr (Pipe[_stream] [,IU]) – Input Transaction request
OTr (Pipe[_stream] [,IU]) – Output Transaction request
ITc (Pipe[_stream] [,IU]) – Input Transaction complete
OTr (Pipe[_stream] [,IU]) – Output Transaction complete

Pipes
- Cmd – Command pipe
- Stat – Status pipe
- Data-in – Data-in pipe
- Data-out – Data-out pipe

IUs
- CIU – COMMAND IU
- SIU – SENSE IU
- RIU – RESPONSE IU
Example Command Sequence (Non-data)

- **Command**: Device may receive these in any order
- **Status**: UAS Command Sequence

**Key:**
- Status Pipe
- Command Pipe
- UAS Command Sequence
Non-data Transfer

- Status reporting uses a stream on the status pipe
- Command may be completed while other commands are transferring data
Example Command Sequence (Write-data)

**UAS Driver**
- ITr (Stat)
- OTr (Data-out)
- OTr (Cmd, CIU)
- OTc (Cmd)
- OTc (Data-out)
- ITc (Stat, SIU)

**xHCI Driver**
- ITr (Stat)
- OTr (Data-out)
- OTr (Cmd, CIU)
- OTc (Cmd)
- OTc (Data-out)
- ITc (Stat, SIU)

**xHCI Controller**
- IN ACK (Stat, Prime)
- NRDY (Stat, Prime)
- DP (Data-out, Prime)
- NRDY (Data-out, Prime)
- DP (Cmd, CIU)
- ACK (Cmd)
- ERDY (Data-out)
- DP (Data-out)
- ACK (Data-out)
- DP (Data-out)
- ACK (Data-out)
- ERDY (Stat)
- IN ACK (Stat)
- DP (Stat, SIU)
- ACK (Stat)

**USB-3 Device**

Device may receive these in any order

**Key:**
- Status Pipe
- Command Pipe
- Data-out Pipe
- UAS Command Sequence
Write-data Transfer

- Uses a stream on the status pipe
- Uses a stream on the data-out pipe
- Command may be completed while other commands are transferring data or reporting status
- Notice that there is no round trip wait for the write data transfer to begin and the status to be received
Example Command Sequence (Read-data)

Key:
Status Pipe, Command Pipe, Data-in Pipe, UAS Command Sequence
Write-data Transfer

- Uses a stream on the status pipe
- Uses a stream on the data-in pipe
- Command may be completed while other commands are transferring data or reporting status
- Notice that there is no round trip wait for the read data transfer to begin and the status to be received
Example Command Sequence (Abort Command)

**Command**
- Device may receive these in any order

**Abort**
- Device may receive these in any order

**Abort Complete**

**Status**

Key:
- Status Pipe, Command Pipe, UAS Command Sequence
Abort Command

- Uses a stream on the status pipe
- Abort command may be completed while other commands are transferring data or reporting status
- Abort task should be followed by a status on the status pipe for the command being aborted
Example Command Sequence
(Multiple Command)

Key:
- Status Pipe, Command Pipe, Data-in Pipe, Data-out Pipe, UAS Command Sequence
Multiple Command

- Both commands are processed at the same time
- Data for the second command starts transfer before the data for the first command
- There is a period of time where data is transferring in both directions at the same time
- High level of USB bus utilization
  - Once the first command is received, the only time delays are waiting for the device
  - Notice the high level on asynchrony on the USB bus
- The device chooses to complete the first command received before the second command received
  - The device could have chosen to complete the second command first
• Configuration Descriptor – Follows USB specifications

• Device Descriptor – Follows USB specifications

• Interface Descriptor – Follows USB specifications
  • For backward compatibility – BOT interface should be 1\textsuperscript{st} and UAS interface should be 2\textsuperscript{nd}
  • If backward compatibility is not important, then only UAS interface is necessary
The following may be in any order:

- Bulk-out descriptor followed by a USB 3.0 companion descriptor and a pipe usage descriptor indicating Command pipe
- Bulk-out descriptor followed by a USB 3.0 companion descriptor and a pipe usage descriptor indicating Data-out pipe
- Bulk-in descriptor followed by a USB 3.0 companion descriptor and a pipe usage descriptor indicating Data-in pipe
- Bulk-in descriptor followed by a USB 3.0 companion descriptor and a pipe usage descriptor indicating Status pipe

The pipe usage descriptor shall appear after the bulk pipe it is describing and before the next bulk pipe.

The Status, Data-in, and Data-out pipes should all indicate that they support the same number of streams.

The pipe with the least number of streams limits the total number of streams the host may use.
SCSI Support
• TEST UNIT READY
  • Non-data transfer command
  • Returns the current status of the device
  • Should be the first command after device discovery

• REPORT LUNS
  • If the device has multiple Logical Units (LU), then this command returns a list of the Logical Unit Numbers (LUN)
  • If the device has only 1 LU, then it only has LUN 0
  • Should be the second command issued to a device
• INQUIRY
  • Reports the device type and other configuration information
    • Most popular types are MultiMedia (DVD/BluRay), Block (HDD/SSD), and RBC (used by USB BOT devices)
    • USB 3.0 is bringing new devices…
  • Should include a device descriptor indicating UAS, SBC-3 and SPC-4 support

• REQUEST SENSE
  • Legacy compatibility, returns the status of the last command
Typical Initialization Sequence

• Device attached to USB
• Assign address and read descriptors
• Find UAS interface
• Configure UAS pipes and begin operation
• Issue TEST UNIT READY command and receive status
• Issue REPORT LUNS command and receive list of LUNS
• Issue INQUIRY to each LUN and discover device type
  • Load associated class driver for each device type
• Class drivers configure each LU based on its device type using MODE SENSE and MODE SELECT commands
• Start using each LU
Demo
Current work in progress

- UAS Compliance Test Specification
  - Provides test assertions and test procedures for validating UAS
Contact Information

- Curtis E. Stevens
  - Curtis.Stevens@wdc.com
- UAS, SPC-4, and SBC-3
  - www.t10.org
- UASP and Mass Storage Class Overview Rev 1.4
  - www.USB.org