To: T10 Technical Committee
From: Rob Elliott, Compaq Computer Corporation (Robert.Elliott@compaq.com)
Date: 5 May 2000
Subject: Target-controlled congestion relief

One basic problem revealed by the BUSY/TASK SET FULL debate is that the target is the only device that knows when it has a queue slot free, not the initiator. Any attempts by the initiator to guess when the target has a queue slot free will be imperfect, either consuming excess bus traffic with retries or not fully utilizing the target's queues. Letting the target indicate when to continue the connection solves this problem.

Lacking that capability, the target can communicate information about its queues so the initiators can throttle themselves in a more intelligent manner.

Changes from Revision 0:
Totally revamped. Revision 0 had the target reselect the initiator when it was ready to accept commands. Most initiators cannot maintain state for outstanding rejected commands, which is needed to accept a reselection quickly. This revision has the target send some new state information to the initiators so they can throttle themselves.

1.1 Description
During each status response (e.g. FCP_RSP IU), have the target return this information:
1. QUEUE_SLOTS: Total number of queue slots in the target
2. ACTIVE_INITIATORS: Number of initiators that are actively using the target

QUEUE_SLOTS would be relatively stable. ACTIVE_INITIATORS would be more volatile.

Each initiator may assume it gets an equal percentage of target resources. If only one initiator is active, it gets to issue the full amount of commands supported by the target. If two initiators are active, each initiator drops down to issuing only half the commands supported by the target.
64 slots 1 initiator => use 64 slots
64 slots 2 initiators => use 32 slots each
64 slots 3 initiators => use 21 slots each (1 lost due to rounding)
64 slots 4 initiators => use 16 slots each

Initiators recalculate their throttle limits every time the values are received.

This lets the initiators both decrement and increment their outstanding command limits in an intelligent fashion, at run time. It serves as a form of feedback from the target to the initiator, which is what is fundamentally missing today.

It would be up to the target implementation to determine how to count the number of active initiators. For Fibre Channel, it might be how many initiators are logged in. For packetized SCSI or Fibre Channel, it might be how many initiators have commands currently outstanding, how many have issued commands within the last <n> commands, or how many have issued commands within some time period.

This is still compatible with Sun's suggestion of reserving one queue slot per initiator. Those reserved slots can be excluded from the number of slots returned in QUEUE_SLOTS.
63 slots 1 initiator => use 63 slots
62 slots 2 initiators => use 31 slots each
61 slots 3 initiators => use 20 slots each (1 lost due to rounding)
60 slots 4 initiators => use 16 slots each

1.2 Conveying the information

Each protocol would have a different way to carry this information.

1.2.1 Fibre Channel

Bytes 8 and 9 are currently reserved in the FCP_RSP IU and could be used to carry this information in Fibre Channel.

1.2.2 Packetized SCSI

Packetized SCSI's L_Q bytes 16 and 17 are available and could be used to carry this information during reselections. Alternatively, the status packet could be expanded.

1.2.3 Parallel SCSI

Parallel SCSI has no convenient place to convey this extra information. The status byte could be expanded into a multi-byte data phase.

1.3 Variations

The target could calculate the recommended number of slots for each initiator itself and return one value rather than two. This would remove control of the algorithm from the operating system, however.

The target could communicate CURRENT_SLOTS_IN_USE in addition to ACTIVE_INITIATORS. This would let each initiator ramp up its number of requests until CURRENT_SLOTS_IN_USE matches QUEUE_SLOTS. Initiators can divide QUEUE_SLOTS by ACTIVE_INITIATORS to determine if they are getting more or less than a “fair” share. If an initiator is not getting its share, it can continue making requests and cause the other initiators to throttle down. If an initiator has more than its share and the queue is full, it can throttle down its own requests.