To: Membership of X3T9.2

Author: Rod DeKoning
NCR Corporation

Date: August 11, 2000

Subject: Dealing with Queue Full

Abstract: The SCSI Queue Full Problem:

When a target device has exhausted its internal resource limitations for receiving queued commands in the SCSI - 2 model, it currently must return Queue Full status to the host. This status provides the host and target no mechanism to indicate when the Queue Full condition is being approached or, once in Queue Full, when the condition has been alleviated. Several schemes have been developed to attempt to handle manage the queue full conditions. In one of the more simple schemes, the host retries continually until the condition is alleviated. In more sophisticated schemes, the host uses attempts to determine a high water mark for the targets queue depth, or the host counts previously issued command status being returned. All of these currently implemented schemes have their inefficiencies in most current configurations, but particularly in the multiple initiator environment.

Other schemes provide a mechanism that re calibrates the queue depth allowed every few seconds or minutes based on the number of commands that are allowed to be issued to the target before queue full is seen. This scheme works well in a single initiator environment. In multiple initiator environments, however, one or more initiators accessing the target at consistently heavy loads may lead to low performance or starvation to an initiator that is starting its IO stream later than the other initiators in the system or has a bursty IO stream characteristic.

As for the target, the memory requirements needed to implement the currently defined queuing model to avoid Queue Full conditions is very large. The standard defines queue depths of up to 256 commands outstanding for every possible ITL nexus. Ignoring the 16 bit SCSI bus option, the target must provide the capability of 1792 commands for every logical unit defined. Assuming 32 logical units, and the target has 57,344 possible outstanding commands. (Double this number if you add the 16 bit SCSI bus option.) Obviously, most systems will have defined a subset of this number possible, but the point should be clear that the target can not be made responsible for providing for a no Queue Full condition guarantee simply by providing the defined maximum queue depth.

Problem Summary:

Problem 1: [The SCSI specification] provides the host and target no mechanism to indicate when the Queue Full condition is being approached or, once in Queue Full, when the condition has been alleviated.

Problem 2: One or more initiators accessing the target at consistently heavy loads may lead to low performance or starvation to an initiator that is starting its IO stream later than the other

initiators in the system or has a bursty IO stream characteristic.

DEFINITIONS:

Command Structure: Target memory associated with one command request from initiator.

Free Pool: Command structures not currently allocated as a result of
1. Preallocation to an initiator queue
2. current command processing.

Maximum Queue Depth for Initiator i is equivalent to:
(Maximum # command structures supported in the target device available for initiator requested command handling
- the sum of all Minimum Queue Depths for all initiators supported
+ the Minimum Queue Depth for initiator i.)

Minimum Queue Depth is defined as:
The minimum number of commands that is guaranteed may be issued to the target device before a QUEUE FULL status may be returned to that initiator for the I_T nexus.

Number of Command Structures in Free Pool is equivalent to:
(# command structures supported in the target device available for initiator requested command handling
- the sum of all Minimum Queue Depths for all initiators supported.)

GOALS:

The following goals should be met in a final solution:

1. Existing targets and initiators implementing SCSI 2 Command queuing should function within this definition without modification. (Performance may be hampered if necessary.)

2. Overhead associated with solution should be minimal for NORMAL data access path.

3. Provide max resource utilization available to initiator and target devices.

PROPOSAL

These problems can be solved and goals met through the implementation of :

1. A message protocol designed to establish a minimum queue depth guarantee without queue full.
2. A change in the defined status byte to allow indication of the current state of the targets queue status.

The following is proposed to be added to the SCSI-3 document.

5.6.24 MINIMUM QUEUE DEPTH REQUEST Message

Table 5-11: Minimum Queue Depth Request Message

Bit / Byte	7	6	5	4	3	2	1	0
0	Extended Message (01h)							
1	Extended Message Length(02h)							
2	Minimum Queue Depth Request Code (04h)							
3	Minimum Queue Depth Request							

A MINIMUM QUEUING DEPTH REQUEST (MQDR) message (Table 5-11) exchange shall be initiated by an SCSI device whenever an initiator wishes to request the minimum command queue depth for the current Initiator-Target nexus be changed from the existing agreement. The agreement becomes invalid after any condition which may leave the queue depth maximum agreement in an indeterminate state such as:

1) after a hard reset condition;
2) after a BUS DEVICE RESET message and;
3) after a power cycle.

In addition, an SCSI device may initiate an MQDR message whenever it is appropriate to request a different queue depth limit. SCSI devices that are incapable of altering the queue depth maximum shall respond to the MQDR message with a MESSAGE REJECT message.

Implementors Note:
Renegotiation instantiated by the initiator immediately after a target instantiated negotiation is not recommended, as the target is responsible for setting the minimum queue depth, and has previously set it to its best case value.

The Minimum Queue Depth Request message exchange establishes the minimum number of commands that is guaranteed may be issued to the target device before a QUEUE FULL status may be returned to that initiator for the I_T nexus. Each Initiator-Target nexus is defined with a default minimum queue depth limit. The default minimum queue depth limit may only be changed after a MINIMUM QUEUE DEPTH REQUEST message is received by the target for each I_T nexus. The default minimum queue depth limit shall be greater than one. The maximum value of the default minimum queue depth limit is not defined by this standard.

In the event the originating device is the initiator, upon receiving a Minimum Queue Depth Request message the target device calculates a new minimum queue depth value for that I_T nexus using the MQDR value received from the host as the maximum minimum queue depth value allowed for this negotiation. The target device then returns the new minimum queue depth value to the initiator in the second half of the Minimum Queue Depth Request negotiation. In the event that the calculation of the new minimum queue depth value changed the minimum queue depth value for another initiator, the target would be required to negotiate with each of the affected initiators for each initiator's the new minimum queue depth requirement.

IMPLEMENTORS NOTE:

It is not required that a target implement a solution that allows the queue depth of other intiators to be affected by the negotiation of another initiator.

In the event the originating device of the MQDR message is the target, upon receiving a MQDR message the initiator device may return a minimum queue depth value equal to or less than the value provided by the target. A valid MQDR value will be the effective minimum queue depth value accepted by the target.

Any command structure slots newly available after any negotiation may be allocated as follows:

a. Allocated to a free pool of queued command buffer space to provide for additional queue depth above the minimum queue depth.
b. Result in other negotiations with other hosts to allocate the available queue command pool subset.

Alter the definition of the status byte to include an Extended Queue bit as defined:

Existing Status Byte Definition:

Bit	7	6	5	4	3	2	1	0
	Reserved	Status Byte Code						Reserved

Table 6-7: Status Byte Code

Bits of Status Byte								
7	6	5	4	3	2	1	0	Status
R	R	0	0	0	0	0	R	GOOD
R	R	0	0	0	0	1	R	CHECK CONDITION
R	R	0	0	0	1	0	R	CONDITION MET
R	R	0	0	1	0	0	R	BUSY
R	R	0	1	0	0	0	R	INTERMEDIATE
R	R	0	1	0	1	0	R	INTERMEDIATE-CONDITION MET
R	R	0	1	1	0	0	R	RESERVATION CONFLICT
R	R	1	0	0	0	1	R	COMMAND TERMINATED
R	R	1	0	1	0	0	R	QUEUE FULL
			All Other Codes					Reserved

Key: R - Reserved bit

Proposed Status Byte Definition:

Bit	7	6	5	4	3	2	1	0
	Reserved	Near Queue Full (NQF)	Status Byte Code					Reserved

Table 6-7: Status Byte Code

Bits of Status Byte

7	6	5	4	3	2	1	0	Status
R	R	0	0	0	0	0	R	GOOD
R	R	0	0	0	0	1	R	CHECK CONDITION
R	R	0	0	0	1	0	R	CONDITION MET
R	R	0	0	1	0	0	R	BUSY
R	R	0	1	0	0	0	R	INTERMEDIATE
R	R	0	1	0	1	0	R	INTERMEDIATE-CONDITION MET
R	R	0	1	1	0	0	R	RESERVATION CONFLICT
R	R	1	0	0	0	1	R	COMMAND TERMINATED
R	R	1	0	1	0	0	R	QUEUE FULL
R	R	1	0	1	0	1	R	NEAR QUEUE FULL
			All Other Codes					Reserved

Key: R - Reserved bit

NEAR QUEUE FULL This status indicates that the target has successfully completed the command. The Near Queue Full Status shall be returned as the status of the command if a command structure could not be preallocated to that I_T nexus. (Preallocation is not required, only check of preallocation capability.) The Near Queue Full status shall be returned only if a MQDR negotiation has previously occurred as initiated by Initiator and the Target

Other Changes Required to the standard, but not yet a part of this proposal:

Add ASC/ASCQ code changes:

Other general statements about error handling during message transfer.

THE FOLLOWING IS AN EXAMPLE OF ONE IMPLEMENTATION THAT COULD BE USED GIVEN THE ABOVE QUEUE FULL PROPOSAL. IT IS NOT INTENDED TO BE THE ONLY METHOD THAT COULD BE DERIVED FROM THE PROPOSAL, AND IS INCLUDED ONLY TO ILLUSTRATE THE USE OF THE NEAR QUEUE FULL MECHANISM PROVIDED IN THE PROPOSAL.

THIS EXAMPLE IS NOT, I REPEAT NOT!, PART OF THE PROPOSED CHANGES TO THE SCSI STANDARD.
EXAMPLE

Assumptions:

Target:

56 command structures available (static).
7 initiators supported by virtue of 8 bit SCSI bus.
Minimum Queue Depth Request value = 2
56 - (7*2) = 42 command structure originally in free pool

Command Structure allocation algorithm requires minimum of 10 structures not allocated to initiators through the negotiation process at all times.

Initiator:

Seeks to issue command queue depth of 60 commands

Example Flow:

The following example attempts to highlight a basic flow using the above approach. The provided algorithm for determining the allocation of slots to individual initiators is a simplistic example only, and is not intended provide for all possible host requests.)

Target	Initiator A	Initiator B
	Assume no Outstanding Commands	Assume no Outstanding Commands
	- Instantiates MQDR negotiation w/ value = 60	
-Calculates new MQDR value for initiator = Total Command Queues Available - (((Min Default Request * # Initiators available) + # reserved for free pool) / # Initiators Requesting >> Minimum Default) + (default reserved for this initiator)) = ((56 - ((7*2) + 10)) / 1) + 2 = 34 (assuming no free pool algorithm)		
- Returns MQDR negotiation w/ value = 34		
	- Complete command in progress	
	- Issues 34 queued commands (none have completed)	
- Initiator A's 34 commands consume its 'guaranteed' command structures		
	- Issues 10 queued commands (no commands from A have completed)	
- Initiator A's 44 commands consume its guaranteed and all of target's free pool command structures.		
	- Issues 1 queued command (none have completed)	
- Returns QUEUE FULL to Initiator A		

- Issues 2 queued commands to the target

- Initiator B's 2 commands fill its 'guaranteed' command structure allocation

- Issues 1 queued command (none have completed)

- Returns QUEUE FULL to Initiator B

- Returns Status for a command to Initiator A w/ Near Full Bit = On. (Init A @ 43 commands)

- Receives status, sees room available for only one command.

- Returns Status for a command to Initiator A w/ Near Full Bit = Off. (Init A @ 42 commands)

- Receives status, and sees room available for at least 2 additional commands in target queue.

- Issues 2 Commands

- Initiator A's 44 commands consume its guaranteed and all of target's free pool command structures.

- Returns Status for a command to Initiator A w/ Near Full Bit = On. (Init A @ 43 commands in queue)

- Returns Status for 3 commands to Initiator A w/ Near Full Bit = Off. (Init A @ 40 commands in queue)

Continuing the example above, if Initiator B were to negotiate at any time after initiator A negotiated, the following would occur if the ability to change other initiator queues based on current negotiations is not implemented by the Target:

Target	Initiator A	Initiator B
		Assume 2 commands outstanding at start of sequence
		- Instantiates MQDR negotiation w/ value = 60
-Calculates new MQDR value for initiator B = 2 since other intiator A has acquired all available command structures under given assumptions.		
- Returns MQDR negotiation w/ value = 2		
		- MQDR negotiated to 2
		- Issue one command to target
- Initiator A's 40 commands consume its 'guaranteed' 34 command structures plus 6 from the free pool, and Initiator B's 3 commands consume its' 2 guaranteed command structures plus 1 from the free pool - 3 in free pool still available to either intiator .		

Continuing from the primary example above (i.e. skipping the short segue above), if Initiator B were to negotiate at any time after initiator A negotiated, the following would occur if the ability to change other initiator queues based on current negotiations is implemented by the target device :

Target	Initiator A	Initiator B

- Instantiates MQDR negotiation w/ value = 60

-Calculates new MQDR value for initiator =
Total Command Queues Available - (((Min Default Request * # Initiators available) + # reserved for free pool) / # Initiators Requesting >> Minimum Default) + (default reserved for this initiator))
= ((56 - ((7*2) + 10)) / 2) + 2
= 18 (assuming no free pool algorithm)

- Returns MQDR negotiation w/ value = 18

- MQDR negotiated to 18

- On NEXT command phase connection with Initiator A, the new MQDR = 18 value is negotiated.

- Attempts to satisfy all commands from Initiator B while Initiator A's outstanding queue depth is being reduced. It is possible that the Near Queue Full bit will be set or that Initiator B will receive Queue Full during this reduction.

At present Initiator A has 40 commands outstanding, 18 in guaranteed command structures, 6 from the free pool, and 16 over run after the latest MQDR negotiation

- New MQDR negotiated

- Allows commands in progress to complete to level in which outstanding commands are below the newly negotiated MQDR value.

- Continues attempting to issue commands as required

Issues 4 commands to target

- Receives 4 commands
- Initiator A still has 40 outstanding as described above, while Initiator B has 6 commands outstanding
- All the free pool is consumed.

- Status returned on 10 commands for Initiator A w/
Near Full Bit = On

- and 1 status returned for Initiator B w/
Near Full Bit = On

- Leaving 30 commands for Initiator A (18 in 'guaranteed pool' and 6 from free pool and 6 over run) and 6 commands for Initiator B outstanding