To: T10 Technical Committee
From: Luben Tuikov, Vitesse Semiconductor (ltuikov@vitesse.com)
Date: 18 October 2006
Subject: 06-460r0 SAM-4 Time bounds of task states

Revision history
Revision 0 (18 October 2006) First revision

Related documents
sam4r07 - SCSI Architecture Model - 4, revision 07
spc4r07a - SCSI Primary Commands -4, revision 07a
06-368r0 - SAM-4 QUERY TASK TMF Extended Response, revision 0 (A)
05-284r3 - SPC-4: Self Describing Command Timeouts, revision 3 (B)

Overview
The SCSI Architecture Model document gives detailed description of task states and task set management in section 8.

Recently, we've seen the need for SCSI Targets to be able to convey to Application clients, a command timeout value such that Application clients can better manage error recovery and avoid error recovery "thrashing". See Related documents (A) and (B).

The task states defined in the SCSI Architecture Model are TASK DORMANT, TASK ENABLED, TASK BLOCKED and TASK ENDED. Task set management is defined in terms of the four task states mentioned here, four task attributes and a priority attribute which we will not mention here as it is not relevant to the purpose of this proposal.

That is, SAM-4 compliant task servers implementing task management and a task manager as specified in SAM-4, should at least be able to represent the four task states. In other words, they should be able to map from an internal (implementation and/or proprietary) task state to one of these SAM-4 defined four states.

Error Recognition and Error Recovery
At the application client, error recognition can happen in one of the following ways:

a) Service response,
b) Task status code, or
c) Application client timed out the task after certain timeout period had been allocated for its execution.

Case c) is particularly interesting as it is the topic of 05-284r3 - SPC-4: Self Describing Command Timeouts.

After the task has timed-out its allocated execution time, the application client would normally issue ABORT TASK TMF. This isn't always optimal. Depending on the task state at the device server (target), an application client may abort a perfectly good task, for example in TASK ENABLED or TASK ENDED state. For this reason, it is encouraged that application clients first issue QUERY TASK TMF in order to figure out where the task is and what its state is and then decide whether to issue ABORT TASK TMF or whether to reset their task execution timeout timer and wait again for a certain amount of retries (finite), thus extending the total execution timeout.

This method works fine for some of the cases but not for all. For example, if the task is in TASK DORMANT state after it timed out the first time, an application client may wish to abort the task, and notify the system administrator that external intervention may be required to the logical unit. If, on the other hand, the task is in TASK ENABLED state after it timed out for the first time, it is advised that application clients reset the timer and let the logical unit complete the task.

This is where 06-368r0 - SAM-4 QUERY TASK TMF Extended Response, extends the response of QUERY TASK TMF to optionally to the device server, return the task state of the task being queried, so that application clients can decide whether to abort the task (e.g. TASK DORMANT) or give it another chance (TASK ENABLED).

This is also where 05-284-r3 - SPC-4: Self Describing Command Timeouts, defines how a device server returns the appropriate (suggested) task timeout value, that application clients should wait before aborting a task, or, using the QUERY TASK TMF Extended Response, querying the task's status before deciding whether to abort it or whether to give it another chance.

Problem

The problem is that the timeout value(s) returned by the target as described in 05-284r3 currently does not correspond to any task state. What then does such a timeout value mean ***in the context of SAM-4 and SPC-4?***

Analysis

A task executing at the target device and completing with the application client needs to be represented by *at least two* task states (TASK ENABLED and TASK ENDED) at the target. This implies that the timeout value specifies time spent in at least those two states.

Suggested changes

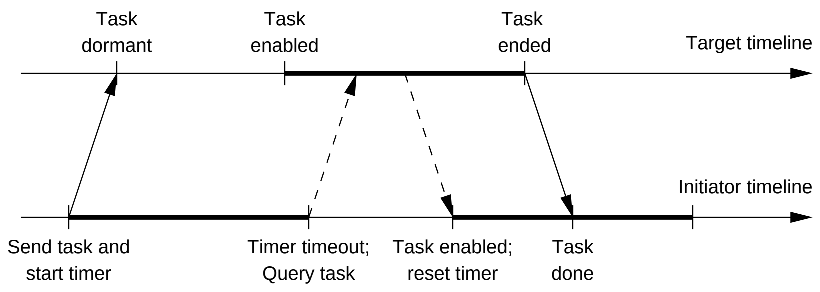
The suggested changes are that the timeout value specify time spent in the TASK ENABLED and TASK ENDED task state. This of course implies that TASK ENABLED and TASK ENDED state is bounded in time. This is not a problem since the time bounds are not specified in the architecture. As far as the architecture is concerned a time bound of 100 million years is just as good as 100 microseconds, for example.

Implications

This would allow application clients to reset their timers if after querying the task, they receive a QUERY TASK TMF Extended response of TASK ENABLED or TASK ENDED, giving the task time to complete and avoiding error recovery "thrashing".

Since application clients do not detect task state transitions, the task timeout timer is already running at the initiator when the task arrives at the target.

When the timer times out, the application client issues QUERY TASK TMF to query the task's location and state. If the task is in the TASK ENABLED or TASK ENDED task state then it would finish in at most the timeout value specified in 05-284r3, plus any transport overhead. Please see the following figure.



If the task is in any other state, it is up to the application client's discretion what to do next.

Summary

This proposal, if accepted, would greatly improve error recovery both at the target and at the initiator. It would give more information to the application client to make better decisions regarding aborting a task, thus reducing transport traffic and unnecessary task abort processing at the target.