Use of Class 2 for Fibre Channel Tapes T10/97-184R1.TXT

970527

This paper describes the use of the Fibre Channel Class 2 protocol when communicating to a tape device that implements the SCSI-3 Streaming Commands (SSC) device model. Some preliminary discussion on this topic is in document T10/97-155R3.TXT.

1. Scope

1.1 The basic proposal here is to "use Class 2 for tapes".

1.2. The protocol is intended to work using the FC-PH Class 2 behavior as it applies to switched Fibre Channel, FC-AL, and FCL environments. This has numerous implications such as, for example, the possibility of out-of-order delivery of frames within a sequence.

1.3. While this discussion is narrowly targeted at SSC devices, it is believed that the protocol described here would work for all SCSI-3 device models.

2. Other possibilities

Refer to 97-189Rx.TXT (the Crossroads proposal) for discussion of a protocol for the use of Class 3 for tapes. This Class 3 proposal is to poll the target device at certain times to determine the status of a transfer.

3. FC/FCP/SCSI Reminder

According to the FCP mapping of SCSI to Fibre Channel, each SCSI command is completely processed within a single Fibre Channel exchange.

Throughout this discussion it is held as a fundamental assumption that when a SCSI command, contained within the bounds of an exchange, is issued by the initiator every possible attempt shall be made to successfully transfer that command, and its associated data and return status, between the initiator and the target.

Proposals that require commands to be retried at the ULP level, except in the most severe cases, are not under discussion.

4. Proposal

4.1. Class 2 Concepts and Rules

- Use Class 2 ACK 0 model. This is one ACK per sequence.
- Use E_D_TOV to detect most errors.
- Use a specified number "n" of retry sequences, n is currently TBD.
- Rely on ULP timeout for detection of certain remaining errors (mostly target device failures).

- Use existing FCP Information Units (IU) and FC-PH features.
- If E_D_TOV expires before a given sequence expires, retransmit the entire IU in a new sequence using new sequence ID and counts.
- Follow all existing Class 2 rules regarding the use of ABTS and RRQ.

The basic rules are as follows. It is intended that these not conflict with the normal use of Class 2 as described in FC-PH.

a.) For each exchange, the exchange initiator starts a ULP timer using a value defined by the SCSI command timeout for the given command. If the timer expires before the SCSI status is successfully returned in the FCP_RSP IU, then the exchange and SCSI command have failed and this is reported to the user's program.

The ULP timer is primarily intended for detection of a failed target device, and is not used in the error recovery process (this is new since the last discussion).

b.) For each sequence within the exchange, the sender starts a sequence timer with a value of E_D_TOV (as defined on FC-PH page 261) upon the transmission of the last frame of the sequence. If the sequence ACK is not received by the time the timer expires, then perform the ABTS process. After the ABTS process completes, send the same IU in a new sequence.

The process (send-wait-resend) is repeated n times, where n is a number that has yet to be determined but will be a fixed feature of the protocol. The value of n is the same for both the initiator and the target.

Previously it was suggested to use the ULP timer for this purpose, but there are two difficulties with this:

- the target doesn't know the value of the ULP timer
- resending the same sequence for 10 minutes seems a bit excessive

(Note also the significant difference here between this proposal and the Class 3 proposal in T10/97-189R0. There, the assumption is that only two sequential ABTS failures cause the system to return to the ULP driver--an assumption that I think is too weak.)

The same rule is followed by the initiator and the target. Whoever starts a sequence applies the E_D_TOV timeout test. Context for each sequence is held until the timer expires or the ACK is received.

c.) If the target is acting as the Sequence Initiator and it is unable to successfully send the sequence and get the associated ACK, then the sequence timeout (also E_D_TOV) will cause the sequence to be aborted.

4.2. Notes on Rules

4.2.1. Discard Policy

We have not talked about this, but I think the policy of interest

is "discard a single sequence" (see FC-PH 29.6.1) since that allows full out-of-order delivery at both the frame and sequence level.

4.2.2. Sequence initiative (SI)

The holder of SI is the sequence initiator. If a node holds SI and receives a frame for that exchange, it transmits a P_RJT.

SI is considered accepted by the sequence recipient when the ACK for the sequence is sent. Recovery of SI is the primary reason for the ABTS requirement. See FC-PH 24.6.4.

This becomes an issue if a sequence that transfers SI is lost and as a result neither side thinks it has SI.

4.2.3. Sequence Error Detection

Sequence timeout (FC-PH 29.2.4) is the basic Class 2 error detection method. The sequence timer is started with a value of E_D_TOV, and if no ACK is received for the sequence within that time, a sequence timeout has occured.

If a sequence timeout is detected by the sequence initiator, then it performs the ABTS protocol. The first step in this protocol is to send an ABTS Basic Link Services frame.

The sequence recipient also runs an E_D_TOV timer, and if all frames for a sequence have not been received within E_D_TOV (if they arrive out of order it's ok), a sequence timeout has occured.

If a sequence timeout is detected by the sequence recipient, then it performs the abnormal sequence termination protocol (FC-PH 29.7.1). The first step in this process is to return an ACK with the Abort Sequence Condition bits set.

4.2.4. ABTS process

This is used if the failure is detected by the initiator. See below for the case where the failure is detected by the recipient.

ABTS may be sent without holding SI (FC-PH 21.2.2).

The sequence initiator sends ABTS, then starts an E_D_TOV timer. The ABTS frame is considered part of the aborted sequence, but runs under its own timer, since the sequence timer has already expired (that's what got us here). The ABTS condition is indicated by a bit in the F_CTL field of a frame.

After the ABTS is transmitted, the sequence is in an indeterminate state.

At this point the recipient returns the Basic Accept (BA_ACC, FC-PH 21.2.2). The BA_ACC payload includes the Recovery Qualifier, a data structure that allows the initiator and recipient to synchronize their understanding of the status of the sequence. This is done by providing a list

of SEQ_IDs that are non-deliverable, so that the initiator can be sure to not send that sequence ID again.

In perhaps the most typical case, a single frame in a multi-frame sequence will have vanished for some reason. After the timeout, the initiator sends ABTS in a frame appended to the end of the sequence. [Even though it has already sent the last frame of the sequence, which is so marked.] Since at the recipient's end the frame never arrived, the ABTS indicates a sequence to be aborted. [Note that the recipient might time out the sequence before it receives the ABTS.] The SEQ_ID is known. The high SEQ_CNT in the Recovery Qualifier range is the SEQ_ID of the ABTS frame, while the low SEQ_CNT of the Recovery Qualifier is that of the first frame in the sequence (probably 1).

Thus by FC-PH 21.2.2.1 "a sequence is in error" (page 135--"ABTS Recipient"), so a recovery range "shall be established for both N_Ports".

[I don't see why, since the proposal is to use a new SEQ_ID next time, the recovery range is needed. If it could be abandoned then the complexity of the RRQ could be avoided, and the loss of ABTS or BA_ACC frames would be just nested versions of the normal error recovery process.]

Assuming that the BA_ACC makes it back to the initiator, it then knows that the sequence has been successfully aborted, and it may proceed to send the same IU in a new sequence. After E_D_TOV expires at the recipient end, the recipient may discard its context for that aborted sequence and proceed.

[Clarify how SI is coordinated using non-RRQ method.]

4.2.5. What if the recipient has never heard of this sequence?

If the first frame of the sequence failed, the recipient will have no information about the sequence. In this case, when it gets the ABTS it returns a BA_RJT instead of a BA_ACC.

Regardless of whether it's a BA_RJT or a BA_ACC, when the response to the ABTS arrives then the initiator knows it's ok to send a new sequence.

4.2.6. What if the ABTS fails?

If, after sending ABTS, the E_D_TOV timer expires at the initiator before a BA_ACC or BA_RJT is received, then the initiator sends another ABTS.

4.2.7. What if the BA_ACC fails?

From the initiator's viewpoint it's the same as above.

4.2.8. What if the BA_RJT fails?

From the initiator's viewpoint it's the same as above.

4.2.6. RRQ process

The recovery qualifier is a data structure provided by the recipient to the initiator in the BA_ACC to the ABTS. (FC-PH 29.7.1.1)

I am not convinced that we need this.

It indicates the range of sequence count values that shall not be reused by the initiator during the next E_D_TOV timeout period. This is to flush these count values out of the system.

After E_D_TOV expires then the initiator reinstates the recovery qualifier using RRQ.

4.2.6. Abnormal Sequence Termination

This is used if the failure is detected by the recipient. See below for the case where the failure is detected by the initiator.

[more needed here]

4.3. Examples

The following examples show the case of processing an error frame that occurs during the transfer of an FCP_DATA sequence. Errors that occur during the many other frames and sequences are discussed in the notes following the examples.

==========

Regardless of whether the command is a READ or a WRITE, there are really only about three cases that need to be handled:

- first sequence in an exchange, including exchange initiation
- mid-exchange sequences going in either direction
- last sequence in an exchange, including exchange completion

The next version of this document will cover these three cases without specifying whether a given sequence is for READ data or WRITE data, since they are really the same when considered at the Fibre Channel level.

4.3.1. First Sequence in an Exchange

In this case, in addition to the normal sequence processing there is some overhead related to starting up the exchange. However, the

procedure for trying to get the sequence to the recipient is the same as for the mid-exchange case.

4.3.2. Mid-Exchange Sequences

This is the "normal" case.

4.3.3. Last Sequence in an Exchange

In this case the primary issue is running down the exchange and releasing various resources.

The specific difficulty is at the target end, because the target must at some point decide to discard the context for the current exchange. This is done when the ACK for the last sequence is received at the target. The last sequence in every exchange must be from the target to the initiator so that the last event in the exchange is the return of the ACK to the target. This is currently the case with FCP.

If the ACK does not make it back to the target, then the target will (after performing the ABTS protocol) attempt to send the data again in a new sequence. This process is to be repeated for a number of times--but that number of times is not currently defined. The amount of time spent on an exchange before discarding it is related to the characteristics of the errors on the interconnect.

==========

4.4. Mappings of FCP to Class 2

This section shows how FCP is used with Class 2.

4.4.1. WRITE

Example: Normal case (no error). Transfer of 2 data sequences, each containing 4 frames of data. Target indicates its ability to accept data by use of FCP_XFR_RDY IUs.

```
Initiator          Target
-------------------------------------------

FCP_CMD ---------->

        <----------  ACK
                     Receipt of ACK by initiator indicates FCP_CMD is ok.
                     (FCP_CMD is a single-frame sequence.)

                     A long period of time may be required here
                     for the target to find space for the data or to
                     do some preliminary media positioning.

        <----------  FCP_XFR_RDY
                     Target indicates readiness to accept one
                     sequence of data
```

```
ACK     ---------->

        ---------->  DATA sequence ID = 1, CNT = 1 (frame 1)
        ----->X      DATA sequence ID = 1, CNT = 2 (frame 2)
                     Error occurs on interconnect at "X". The frame is lost.

        ---------->  DATA sequence ID = 1, CNT = 3 (frame 3)
        ---------->  DATA sequence ID = 1, CNT = 4 (frame 4)

Initiator has sent all the data frames,
so it starts an E_D_TOV timer for this sequence.

                     Target does not get all the frames, so it doesn't
                     send an ACK. [Does it experience a sequence
                     timeout? Probably E_D_TOV.]

Initiator's timer expires. ACK not received,
so the sequence has failed.
Initiator sends ABTS (with LS bit set) to make sure that this
sequence is aborted. [Required by FC-PH but of no apparent value.]
Initiator retransmits the data in a new sequence.

[Note requirement for support of non-ascending tranfers.]

        ---------->  DATA sequence ID = 2, CNT = 1 (frame 1)
        ---------->  DATA sequence ID = 2, CNT = 2 (frame 2)
        ---------->  DATA sequence ID = 2, CNT = 3 (frame 3)
        ---------->  DATA sequence ID = 2, CNT = 4 (frame 4)

        <----------  ACK

Receipt of ACK by initiator indicates that the sequence is ok.
Initiator may discard context for this sequence.

                     If no additional data for this [mumble] is
                     received by E_D_TOV after sending the ACK,
                     the target may discard context for this sequence.

                     When the tape is ready to receive additional
                     data, it sends another FCP_XFR_RDY.

[The wording of this example is not meant to imply that streaming is not
allowed,
but we must verify that streaming works properly.]

        <----------  FCP_XFR_RDY
                     Target indicates readiness to accept one
                     sequence of data
ACK     ---------->

        ---------->  DATA sequence ID = 3, CNT = 1 (frame 1)
        ---------->  DATA sequence ID = 3, CNT = 2 (frame 2)
        ---------->  DATA sequence ID = 3, CNT = 3 (frame 3)
        ---------->  DATA sequence ID = 3, CNT = 4 (frame 4)
```

```
        <----------  ACK

                     Receipt of ACK indicates that the sequence is ok.
                     Proceed to next sequence.

                     Target observes that it has enough data to
                     satisfy the requirements of the SCSI WRITE command,
                     so it sends the SCSI status back.

        <----------  FCP_RSP
                     With SCSI Status.
ACK     ---------->
                     Target closes exchange and deletes command context.
```

Initiator waits for E_D_TOV after sending ACK to make sure no more sequences are coming. (This would be a resend of the FCP_RSP if the last ACK had been lost on its way to the target.) After timout expires, Initiator discards context for this exchange.

==========

4.4.2. READ

Example: Transfer of 2 data sequences, each containing 4 frames of data. Assume the host can accept all the data specified in the command.

```
Initiator          Target
-------------------------------------------

FCP_CMD ---------->
        <----------  ACK
                     Receipt of ACK by initiator indicates FCP_CMD is ok.
                     (FCP_CMD is a single-frame sequence.)

                     A long period of time may be required here
                     for the target to get the data from the media.

        <----------  DATA sequence ID = 1, CNT = 1 (frame 1)

             X<----  DATA sequence ID = 1, CNT = 2 (frame 2)
                     Error occurs on interconnect at "X". The frame is lost.

        <----------  DATA sequence ID = 1, CNT = 3 (frame 3)
        <----------  DATA sequence ID = 1, CNT = 4 (frame 4)

                     Target has sent all the data frames,
                     so it starts an E_D_TOV timer for this sequence.
```

Initiator does not get all the frames, so it doesn't send an ACK.
[Does it experience a sequence timeout? Probably E_D_TOV.]

```
                     Target's timer expires. ACK not received,
                     so the sequence has failed. Retransmit sequence.
```

[Does frame header give enough info to allow Initiator to put

```
the re-sent data in the right place?]
[Note requirement for support of non-ascending tranfers.]

        <----------  DATA sequence ID = 2, CNT = 1 (frame 1)
        <----------  DATA sequence ID = 2, CNT = 2 (frame 2)
        <----------  DATA sequence ID = 2, CNT = 3 (frame 3)
        <----------  DATA sequence ID = 2, CNT = 4 (frame 4)

ACK     ---------->
                     Receipt of ACK indicates that the sequence is ok.
                     Proceed to next sequence. [Not meant to imply
                     that streaming is not allowed.]

        <----------  DATA sequence ID = 3, CNT = 1 (frame 1)
        <----------  DATA sequence ID = 3, CNT = 2 (frame 2)
        <----------  DATA sequence ID = 3, CNT = 3 (frame 3)
        <----------  DATA sequence ID = 3, CNT = 4 (frame 4)

ACK     ---------->
                     Receipt of ACK indicates that the sequence is ok.
                     Proceed to next sequence.

        <----------  FCP_RSP
                     With SCSI Status.
ACK     ---------->
                     Target closes exchange and deletes command context.

Initiator waits for E_D_TOV after sending ACK to make sure no more
sequences are coming. (This would be a resend of the FCP_RSP if the
last ACK had been lost on its way to the target.) After timout expires,
Initiator discards context for this exchange.

==========================================

4.5. Notes on Examples

Any frame in the exchange may fail. The following lists the
handling of each possible case. This is intended to follow
the FC-PH protocol exactly.

4.5.1. Loss of FCP_CMD single-frame sequence. In this case the
target never sees the command. When initiator's E_D_TOV timer
expires it first sends an ABTS with the LS bit set. This has
no effect on the target (which simply ignores this attempt to
abort a sequence in an exchange it has never heard of).
Then the initiator resends the FCP_CMD IU using the same information
and the same exchange identifier in a new sequence.

[See FC-PH 29.7.1.1. for discussion of ABTS at start of exchange.
[Note need for BA_ACC frame to acknowledge the ABTS. What's the point?]

4.5.2. Loss of ACK after FCP_CMD. In this case the target saw the command
and has constructed exchange context. When initiator's E_D_TOV timer
expires it first sends an ABTS with the LS bit set; this aborts
the sequence but maintains the exchange. Then it resends the FCP_CMD IU
```

using the same information and the same exchange identifier in a new sequence.

4.5.3 Loss of ABTS after loss of FCP_CMD sequence. In this case the target has not seen the command anyway. [Check for ACCEPT needed for handshake on ABTS.]

4.5.4. Loss of ABTS after loss of ACK after FCP_CMD sequence. In this case the target has constructed exchange context. When the FCP_CMD is resent by the initiator, the target observes that the exchange identifier is for an existing exchange and thus ignores the redundant information.

5. Advantages of Class 2

- Uses existing FC-PH features.
- Uses existing FCP Information Units.
- Good opportunity to automate the processing of each exchange.
- Sequence Initiative management is already defined in FC-PH.
- Sequence streaming is already defined in FC-PH.