More Discussion of Tapes in PLDA T10/97-155r3

970512

This paper is a discussion of the question "What should be done about tapes on Fibre Channel?" This version reflects modifications based on discussion that took place during the FCW and SSC meetings during the May 1997 T10 week, plus some additional proposals.

This is expected to be the final version of this document. There are only a couple of minor changes from 155r2. Another document will continue the discussion of the Class 2 proposal only.

The paper now has two sections:

I. General discussion of the issues.

II. Proposal for solutions using both Class 2 and Class 3.

I. Discussion of Issues

1. Scope

1.1. Note that there are two configurations where this topic arises:

i. Native Fibre Channel tape drives.

ii. Native Fibre Channel subsystem controllers (e.g. RAID controllers) that have the capability of having a tape drive behind them. This drive could be a regular SCSI tape drive. A tape-oriented protocol on the FC connection to the host is needed even if no native Fibre Channel tape drive is ever built.

1.2. Although this discussion originated in the context of PLDA and thus FC-AL, the point in 1.1.ii. above clearly implies that any solution must work in both loop and switched Fibre Channel environments. This has numerous implications such as, for example, the possibility of out-of-order delivery of frames within a sequence or the loss of frames due to congestion in a switch.

1.3. Definitions

Throughout this discussion the term "media" refers to the magnetic medium upon which the data bits are eventually stored, not the interconnect medium used to transfer the bits from device to device.

An "error on the interconnect" is the failure of the interconnect to deliver all the frames and sequences of an exchange with a correct set of sequence ID and sequence count fields and correct EDC values and meeting all the other rules for frame delivery. This is different from a failure resulting from microcode errors, media errors, or the complete failure of a device or system.

A "transient error" is an error on the interconnect that repairs itself without specific action by the protocol. For example, a bypass circuit activation causes one or more transient errors on the loop, but later the loop resumes normal operation. Similarly a burst of random errors will later come to an end without action by the protocol. However,

a failure caused by the permanent physical disconnection of the cable from a cabinet is not a transient error. This paper deals exclusively with transient errors.

Note the distinction between "the application" (user's FORTRAN program) and "the driver" (operating system device driver).

Also note that the driver has two parts: "the class driver" (knows about tapes, not interconnects) and "the port driver" (knows about interconnects, not tapes). A goal is to keep these two 100% distinct.

2. Special Characteristics of Tape Devices

Tape devices have protocol requirements that are different from those of disks. The special characteristics of tapes are summarized below.

2.1.

2.2. If a SCSI command sent to a tape device fails because of an error on the interconnect, the recovery requires more than simply the reissuance of the command as is currently done for disks using the FCP and PLDA profile operating rules.

The operating system device driver software or the user's application must manage the position of the media by issuing a sequence of repositioning commands in addition to (i.e. before) reissuing the failed I/O command. The mechanical process required to complete the recovery is time consuming. For this reason it is very desireable to avoid SCSI command failures.

2.3. A properly constructed parallel SCSI bus experiences errors at an extremely low rate--weeks or months may pass between parity errors. The SCSI tape model is based on an implicit assumption that errors on the interconnect occur extremely rarely.

2.4. Any solution must handle the case of very long transfers done by a single SCSI command. Applications that issue multi-megaByte SCSI commands have been identified, and even larger transfers must be supported.

2.5. There is no connection between "blocks" in the tape sense and sequences on the interconnect.

2.6. The SSC tape device model describes a method of operation. Correct operation is defined by externally visible characteristics: There is no requirement that devices be implemented in any particular way, just that they meet the requirements of the model. Thus, for example, from the viewpoint of the model, the sequence size requested in the FCP_XFR_RDY is the size that the tape drive can buffer without requiring media movement. However, actual implementations may actually require media movement to access the data defined in the FCP_XFR_RDY, but this movement must be performed within the time constraints of the protocol. (This is not a special characteristic of the tape model--any SCSI device may "lie" about how it works.)

3. Fibre Channel Reminders

The following are reminders of some features of Fiber Channel.

3.1. Each SCSI command is completely processed within a single Fibre Channel exchange.

3.2. A SCSI command may fail on the interconnect because of a

random system event. For example, a frame may be lost due to congestion at a switch. Or, a random bit error may occur on the interconnect due to electrical noise. Also, the statistics of the underlying physical components will cause random errors.

3.3. A SCSI command may fail on the interconnect because of a system event triggered by an operator action. For example, when devices are swapped on an FC-AL loop the loop signal is disrupted, causing an error.

3.4. Because of the above two possibilities, a properly constructed Fibre Channel configuration may experience fairly frequent errors even while fully complying with all the requirements of the standards. Worst-case calculations indicate that hardware complying with the standards may deliver an error bit as frequently as every 10 seconds on the average.

Perhaps many systems may experience error rates much lower than this, and one may argue what "typical" delivered error rates will be. In any case, in order to minimize risk at the system level the various profiles (PLDA and FLA) must protect against the worst case environment. This is mandatory to insure that Fibre Channel is suitable for use in enterprize-class environments.

3.5. Note that attempting to mix Class 2 and Class 3 sequences in a single exchange is forbidden according to the current FC-PH rules. This could be changed, but the following proposals use only one class each.

3.6. Note that there is no concept of a Class 2 NAK (non-acknowledge, i.e. "report that the recipient detected an error in the frame and is reporting it back to the sender"). There is only a Class 2 ACK. The NAK function is performed by using an E_D_TOV timer; if the ACK doesn't arrive withing E_D_TOV then a NAK is assumed.

3.7. The best place to fix the tape problem is at the FCP level as described in PLDA. FC-PH and SCSI are long-established, and changes to SCSI driver software or FC-PH hardware are not desireable. Furthermore, it has already been agreed by the owner of FCP that FCP could be changed if a need can be demonstrated. Small changes to FCP and PLDA cause the minimum amount of disturbance to the status quo. It is generally held that FC-PH has all the tools needed to solve this problem and that the limitations of the current approach are restricted to FCP and/or PLDA.

3.8. Methods of Detecting Errors

There are in general three ways to detect errors on a Fibre Channel system:

i. Upper level protocol (ULP) timeouts (e.g. SCSI command timeout)
ii. Fibre Channel protocol timeouts (e.g. E_D_TOV)
iii. Methods supported by the protocol (e.g. invalid EDC in frame)

In cases where the protocol methods for detecting errors fail to detect an error, then the Fibre Channel protocol timeouts will frequently detect the error. If the Fibre Channel protocol timeouts fail to detect an error, it will be detected by the ULP timeout.

The currently agreed upon value for E_D_TOV is 2 seconds. Typical tape driver software uses a ULP value of 10 minutes to enable normal tape device operation. Timeout values vary widely depending on the environment.

The use of the ULP timeout for routine error detection is not acceptable because it is too long. The use of the E_D_TOV timeout is acceptable assuming that errors occur at no more than once very four seconds (or some similar small integer multiple of E_D_TOV).

3.9. The proposals discussed below do not attempt to do any special handling for errors that are detected by the protocol. There are only two cases:

    i. Frames delivered (perhaps out of order) with valid EDCs, a proper set of sequence IDs and sequence counts within a given sequence, and all other frame parameters valid. These are "valid frames".

    ii. Frames delivered with invalid EDC, or incorrectly addressed, or not delivered within the applicable timeout (usually E_D_TOV), or with some other invalid frame parameter. These frames are eventually discarded, and are considered "invalid frames", "undelivered frames", or "lost frames".

3.10. In order to send a sequence, a device must own "sequence initiative" (SI). Sequence initiative is transferred from one device to the other by a bit in the last frame of the sequence. See section 24.6.3 ff of FC-PH for a description of this feature. Note particularly the difference between the use of SI in Class 2 compared to Class 3.

3.11. Sequence streaming is when the originator sends sequences before receiving the ACKs for all previous sequences. Note that if out-of-order delivery is allowed, sequence streaming may impose substantial buffer management difficulties on the sequence recipient. [Also it's not so obvious that the Class 3 proposal below can handle sequence streaming.]

Sequence streaming doesn't have anything to do with "streaming" in the tape drive sense.

II. Possible Solutions

This section describes a possible Class 2 solution, a possible Class 3 solution, and a list of some pros and cons for each.

Both examples show single SCSI commands, each represented by two sequences of four frames each, for a total of 16 kBytes of data to be transferred. These values are chosen to make the examples concise, and probably don't reflect the values that will be used in real implementations.

1. Class 2 Concepts

The Class 2 proposal is listed first because it is conceptually easier to understand. I think that this describes approximately what Fiber Channel pros mean when they say "Why not just use Class 2?"

Concepts are:

- Use Class 2 ACK 0 model. This is one ACK per sequence.
- Use E_D_TOV to detect most errors.
- Rely on ULP timeout for all of the few remaining errors.

- 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.
- If E_D_TOV is set to two seconds, and bit errors occur less than once every four seconds, then this scheme will "work" in the sense of having enough time to recover between errors. (Based on raw averages.)

2. Class 2 Proposal

The rules are as follows:

a.) For each exchange, the sender starts a ULP timer (SCSI command timeout). 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.

b.) For each sequence within the exchange, the sender starts an E_D_TOV timer upon the transmission of the last frame of the sequence. If the sequence ACK is not received by the time the timer expires, then send the same IU in a new sequence. Repeat the wait-resend procedure until until ULP timer expires.

The same rule is followed by the initiator and the target. Whoever starts a sequence applies the E_D_TOV timeout test. Context for the 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.

2.1. Class 2 WRITE

==========

[tbd]

==========

2.2. Class 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 the
                     same information in a new sequence.

[Does frame header give enough info to allow Initiator to put
the re-sent data in the right place? Yes.]
[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.
```

---

3. Class 3 Concepts

The use of a Class 3 protocol allows both disks and tapes to use the same class. This requires additions to FCP and a more complicated set of rules. The goal is to construct an analog of the Class 2 ACK 0 protocol using Class 3 sequences and the FCP IUs.

Concepts:

- Add new FCP_CONF IU to do confirmation of sequence delivery, essentially doing the same thing as the Class 2 ACK 0.
- Use E_D_TOV to detect most errors.
- Rely on ULP timeout for all of the few remaining errors.

The basic rule is that each sequence is acknowledged by another

sequence going in the opposite direction containing an FCP_CONF IU.

4. Class 3 Proposal

The proposal is to use a new FCP Information Unit (IU)
to allow confirmation of each sequence. This allows the initiator
to request retransmission of a sequence, and does not involve
the user's application program in the retransmission.

The rules are as follows:

a.) For each exchange, the sender starts a ULP timer (SCSI command
timeout). 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.

b.) The goal is to handle some of the errors using
the E_D_TOV timeout value of 2 seconds. This is possible at the two
points in the protocol that are indicated by * on the examples below.
After sending the last frame of an FCP_DATA sequence or an FCP_RSP
sequence the device shall start an E_D_TOV timer. If the associated
FCP_CONF sequence (which is one frame in length) is not received
correctly before the timer expires, then the device shall send the
FCP_DATA or FCP_RSP sequence again.

Not defined yet is how to recover if the target ends up holding
SI and is still chattering frames after the initiator has given up
based on ULP timout expiration.

Also not clear is whether this will handle streamed sequences
because of the need to pass SI back and forth. (Not a problem
in Class 2 because there's a method defined in FC-PH for how to
handle SI if the ACK is lost.)

I've temporarily dropped the concept of allowing FCP_CONF to act as
a sequence NAK because it makes it easier to compare the Class 2 and
Class 3 example cases.

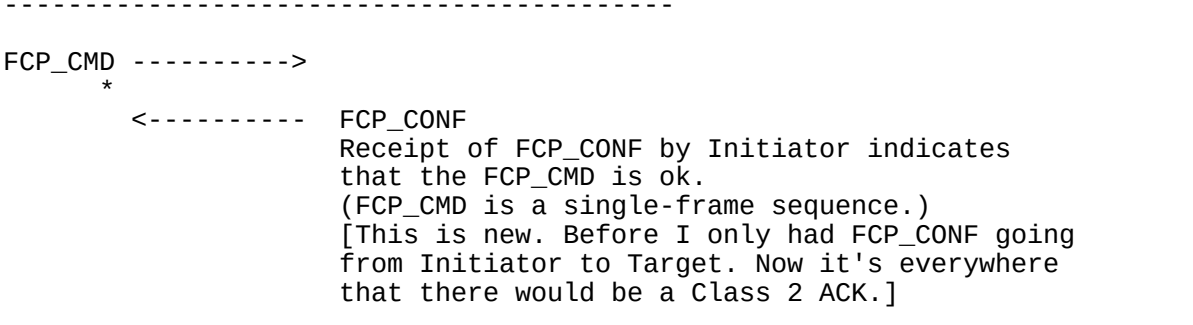
==========

4.1 Class 3 WRITE

==========

4.2. Class 3 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.



```
                    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 FCP_CONF.
[Does it experience a sequence timeout?]

                    Note that last frame of the sequence transfers SI.
                    If the last frame gets lost, then
                    SI doesn't get over to the Initiator, and the
                    transfer stalls until E_D_TOV timeout expiration.
                    [Verify this against Class 3 SI rules, if it's
                    wrong then the stall is for ULP time--much worse.]

                    Target's timer expires. ACK not received,
                    so the sequence has failed. Retransmit the
                    same information in a new sequence.

[Does frame header give enough info to allow Initiator to put
the re-sent data in the right place? Yes.]

        <----------  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)
                    *
FCP_CONF---------->
                    Receipt of FCP_CONF indicates that the sequence is ok.
                    Proceed to next sequence. [Not obvious that
                    sequence streaming is 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)
                    *
FCP_CONF---------->
                    Receipt of FCP_CONF indicates that the sequence is ok.
                    Proceed to next sequence.

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

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

================================
```

5. Class Comparisons

5.1. Originally I used FCP_CONF to give a NAK function in case the sequence recipient detected an error in a frame. This only works if in-order delivery of frames is used, which is not the general case. Also I only had it going one way; now FCP_CONF is in exactly the same places as the Class 2 ACK 0 is used.

The tradeoff was that Class 3 was chosen in part to maximize the loop performance by eliminating the additional ACK delays. Thus the Class 3 proposal was to try to maximize loop performance by trying to report detected errors as early as possible. However it now looks like the difficulties with SI handling in Class 3 make it the lower performance case (forces use of ULP timeout in some cases), plus it's not clear that sequence streaming works in Class 3.

5.2. There are several pros and cons related to the question of whether to use Class 2 or Class 3.

One point is that if different classes are used, the port driver must be told by the class driver what kind of device each command is to be sent to, and the port driver must handle the two cases differently. I will leave judgement about the acceptability of this requirement to the operating system experts. My opinion is that this approach does not reflect proper layering of the software.

The question of how to make mixed Class 2 and Class 3 commands work on an SCC device was a red herring. There's a command (REPORT LUNS) that can be used to find out what device type each LUN is.

5.3. The advantages of the use of Class 3 for all Fibre Channel SCSI transfers include:

- All devices work the same.
- Compatible with existing device implementations.
- Avoids "Class 2 ghetto" of special host adapters, etc. that support Class 2 in addition to Class 3. (Compared to mainstream devices that might only support class 3.)
- Class 3 is mainstream, and is more mature. Note FC-AL bugs still being discovered.
- Use of mixed classes exhibits poor layering between device and class drivers.
- Avoids possible unexpected interactions between Class 2 and Class 3 frames in the same system.
- Possibly higher performance on loop due to fewer ACKs. [Probably not an advantage now with all the extra FCP IUs?]

5.4. The advantages of Class 2 include:

- Protocol is simpler with fewer wierd cases.
- Does not require new FCP IUs
- Processing of each exchange is automatic: one driver interrupt per command
- Sequence Initiative management is already defined in FC-PH.
- Sequence streaming is already defined in FC-PH.
- Question of whether the generation of the FCP_CONF sequences requires a processor interrupt which would make them a lot slower than the ACK generation which is presumably entirely done in hardware.