970819

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". There are no newly
invented primitives or FC-PH concepts. This proposal fits in exactly
with the error handling philosophy described in sections 4.15 (page 31)
and 29.7 (page 265) of FC-PH. Most of this paper is a tutorial on the
Fibre Channel error handling mechanism for Class 2 as described in
FC-PH.

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. It may be particularly useful in cases where the
physical environment is less than ideal, or where command sequentiality
is of particular concern (e.g. systems using XOR functions in disks).

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. If command sequentiality is required, it is to be
managed entirely by the initiator.

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 RES (Read Exchange Status) to inquire about the state of the other device.
- Use a specified large number "n" of retry sequences; n is currently TBD.
- Allow ULP timeout if n retries fail (gross target device failures).
- Use existing FCP Information Units (IU) and FC-PH features.
- If E_D_TOV expires before a given sequence completes, 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.
- Use the same rules whether on a loop or a fabric.
- Release sequence and exchange data structures as early as possible.

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 completely failed target device or communication path, and is not used in the error recovery process.

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.

For the first sequence in an exchange, if after sending the sequence the sequence ACK is not received by the time the timer expires, then the initiator must attempt to determine whether

        i.) the FCP_CMD was lost, or
        ii.) the FCP_CMD was delivered but the ACK and FCP_RSP were lost,
        and the target thinks that the command completed successfully.

To do this, the initiator sends an RES (Read Exchange Status) Extended Link Service in a new sequence in the same exchange (passing SI to the recipient). The reply indicates which case it is.

If the RES sequence fails (detected by E_D_TOV timeout), the initiator resends the RES up to n times before terminating the exchange. The RES process is describe in more detail below.

After the RES is successfully sent and the state of the other sequence recipient known, the initiator perform the ABTS process. This process cleans up any outstanding frames that might otherwise pop out of the fabric later. See below for details of the ABTS process. (The ABTS process also includes the use of RRQ, as described below.)

After the ABTS process completes, if the initiator had determined that
the exchange was unknown (case i.) then it sends the same IU in a new
sequence. If the initiator had determined that the exchange was known
(case ii.) then it expects that the target should be attempting to
resend the FCP_RSP up to n times. In this case the initiator waits
until either it successfully receives the FCP_RSP or until the ULP
timer expires.

The process described above (send-wait-timeout-RES-ABTS-RRQ-resend) is
repeated n times, where n is a (large) 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. Presumably n
failures of this type would be an extremely rare event.

c.) The same basic rule is followed by the initiator and the target for
each sequence in the exchange. Context for each sequence is held until
the ACK is received or until the sequence retry count n is equalled. In
the case of a target doing the retries, after n retries the context for
the exchange is discarded and the ULP timer in the initiator is allowed
to expire.

d.) 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

The following notes are to clarify the above overview.

4.2.1. The transmission of a single SCSI command (in one exchange)
is performed using the following four steps:

a.) Normal exchange, sequence, and frame delivery, including correct
placement of data into the reciever's buffer.
b.) Detection of errors by missing ACKs or other methods.
c.) Run-down and clean-up of the remains of any failed exchanges,
sequences, and/or frames. This includes the use of the RES, ABTS and
RRQ processes.
c.) Retransmission of the SCSI command or data in a new sequence.

4.2.2. Timers

Review FC-PH 29.2. The E_D_TOV timer is for error detection, currently
said to be 2 seconds in various profiles.

The R_A_TOV timer is for "how long a frame may be held in a fabric".
Currently 120 seconds.

4.2.3. Discard Policy

The policy under discussion 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.

4.2.4. Sequence initiative (SI)

The holder of Sequence Initiative (SI) is the sequence initiator. If a
node holds SI and receives a frame for that exchange, it transmits a
P_RJT because it's not supposed to be getting any incoming frames.

The sequence initiator may transfer sequence initiative to the sequence
recipient by setting a control bit in the frame header. SI is
considered accepted by the sequence recipient when the ACK for the
sequence is *sent* by the old recipient back to the old initiator. See
FC-PH 24.6.4.

4.2.5. 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.

In normal operation the ACK will be returned well before the timer
expires, and the context for the sequence may be discarded at that
point. Thus in normal operation the only sequence context that is
pending is that related to sequences that have not yet had ACKs
returned.

A sequence timeout may be detected either by the sequence initiator or
the sequence recipient.

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.

4.2.6. RES Process

The Read Exchange Status Block (RES) link service requests an N_Port to
return the contents of the Exchange Status Block for the RX_ID or OX_ID
specified in the request (FC-PH 21.4.10). The response to the RES may
be either a BA_ACC or a BA_RJT, depending on the following rules.

At the start of a new Exchange, the originator assigns an OX_ID and
embeds it in all frames of the Exchange (FC-PH 4.13.4.1). The responder
assigns an RX_ID and communicates it to the originator before the end
of the first sequence in the exchange. From then on, all frames have
both the OX_ID and the RX_ID in them for the duration of the Exchange.

The responder to the RES uses only the RX_ID, ignoring the OX_ID. If
the sequence under discussion is the first sequence in the exchange,
then the originator will not have found out the RX_ID yet, so it will
use FFFF as the RX_ID (FC-PH 24.3.2, 29.7.1.1 "Special Case – new
Exchange).

Thus the possible responses to the RES are:

If RX_ID == FFFF, return BA_ACC (in order to allow the following RRQ to
be set up)
If RX_ID != FFFF, then
        If OX_ID/RX_ID pair describing an exchange is found, then BA_ACC
        Else BA_RJT

Thus the return of a BA_ACC does not indicate whether or not the target
has knowledge yet of the command. However, if the Special Case rules
are being followed then the BA_ACC will contain an indication of
Invalid Sequence ID in its payload. On the other hand, if the BA_ACC is
from the case where there is a known OX_ID/RX_ID pair, then the BA_ACC
will contain a valid indication along with the Exchange Status Block.

Thus the algorithm to determine whether the target knows about the
Exchange yet is:

Send RES
If response is BA_ACC with invalid Sequence ID indication, then target
doesn't know about the command yet
If response is BA_ACC with valid Sequence ID indication and expected
contents of exchange status block, then target does know about it

Depending on the result of this test, the initiator knows whether to
start the exchange again. The same exchange ID could be used as
previously, but the sequence ID would need to be different in that
case.

[At this point I think that the only time that RES is needed is on the
first sequence of a new exchange, but perhaps it should be used for
every sequence?]

4.2.7. 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 LS bit in the ABTS is cleared in all cases. (LS = 1, as used in
current protocols, causes the entire exchange to be terminated when an
error occurs. This is perhaps the primary change associated with this
proposal.)

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. [Question here of possible race
condition if the recipient times 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".

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.

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

This is detected by the RES process. The ABTS process with RRQ insures
that the old sequence does not accidently get reused, and the sequence
initiator sends the same data in a new sequence.

4.2.9. 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.

4.2.10. What if the BA_ACC fails?

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

4.2.11. What if the BA_RJT fails?

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

4.2.12. 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) It describes
a completely qualified exchange and sequence and a range of frames.

The RRQ process is required because the fabric is allowed to hold
frames for up to R_A_TOV time (120 seconds). Thus if an ACK is lost
(i.e. not delivered within E_D_TOV) the goal is to construct a range of
frames that are declared invalid for R_A_TOV time.

For example, if the RRQ process were not used, if a frame were to
suddenly pop out of the fabric with a sequence ID unknown to the
recipient, should the recipient think that this is a mid-sequence frame
for a sequence whose first frame has not yet arrived? If so it would
hold on to it.

Using RRQ, the Recover Qualifier allows invalid frames to be made known
to the recipient in advance.

After R_A_TOV expires, the initiator issues an RRQ Link Service
(FC-PH 21.4.14) in a separate exchange. If the RRQ exchange fails,
a nested ABTS and new subsequent RRQ may be used, up to the nesting
capability of the hardware.

4.2.13. Abnormal Sequence Termination

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

If a sequence error is detected by the exchange recipient, this fact is
conveyed to the exchange initiator by the Abort Sequence Condition bits
in an ACK frame (FC-PH 29.6.3). The ABTS process is then started by the
exchange initiator.

4.2.14. R_RDY to Precede Every Expected ACK

It is implied in FC-PH, but not clearly stated (FC-PH 16.3.2, 20.2.1,
20.3.1, table 56) that an R_RDY must be transmitted before each
sequence for which an ACK is expected. This is to insure that a buffer
is available for the ACK frame. This requirement implies that, since
the time until the ACK is returned is unpredictable, the sequence
initiator must have an available ACK buffer before sending
a sequence for which an ACK is expected.

This also applies to switches. Consider the following case, suggested
by Bill Martin.

Suppose there is a loop attached to a fabric, with an NL_Port device on
the loop. Suppose some other devices on the fabric want to send some
frames (say, 10) to the device. The other devices send R_RDYs, then
frames, to the fabric. The R_RDYs indicate that the devices have buffer
space for the expected ACKs for those frames (as required by the above
rule).

Then the fabric opens the loop, sends the R_RDYs to indicate that it
has buffer space for the ACKs it expects to receive, then sends the
frames to the device, then closes the loop (because none of the devices
are ready to send data back yet—say it's all READs). The target NL_Port
gets the R_RDYs and frames, then opens the loop to send the ACKs back
to the switch which will then send ACKs back to the original devices
that sent the frames in the first place.

The original devices that send the frames have control over how many
frames they send and can thus maintain the required ACK buffer space.
However, the switch doesn't throttle incoming frames so it can't
reserve the required ACK buffer space. Clearly this won't work; the
question is whether it is possible for switches to throttle the
incoming frames down to the limit required by their available ACK
buffer space. Recall that the switch sent the R_RDYs indicating that it
had the ACK buffer space in the first place; perhaps the question is
whether switches are to maintain this buffer space on a general basis
considering that the need for it is likely to be fairly low,
statistically.

This issue requires more clarification, particularly with comment from
switch implementors.

4.2.15 Use of Relative Offset

This is intended to be a straightforward reading of FC-PH.

Sequences and frames are normally transmitted in sequential order
by the sequence initiator. The fabric may reorder both. Thus
the recipient must reassemble frames into sequences and must deliver
complete sequences of data to the upper layer in the proper order.
This is achieved by the use of Relative Offset (FC-PH 18.11, 27, 24).

(Sequence count (i.e. "frame id within sequence", FC-PH 18.8) could
also be used to define the storage location, but is not in this
proposal.

(Sequence ID (i.e. "location of sequence within this exchange",
FC-PH 18.6) cannot be used to define the storage location because if
a sequence fails, the same data will be sent in a new sequence
with a new sequence ID.)

(Sequence Count: 2 Bytes
 Relative Offset: 4 Bytes
 Sequence ID: 1 Byte)

Relative Offset is defined with respect to the beginning of the
data to be transmitted in the SCSI command (i.e. Exchange), in
terms of Bytes. This rule defines the Relative Offset Space
(FC-PH 27.2). The use of this rule is indicated by word 2, bit 3 = 1
in the F_CTL field (FC-PH 18.5, table 37). Continuously Increasing
relative offset shall be used (FC-PH 27.6).

The sequence recipient shall present data to the ULP only
after a sequence has been completely received. If frames arrive
out of order, the contents of the frames is stored in the recipient's
buffer until the sequence is complete. Before data is presented
to the ULP, all previous data (as indicated by completeness of
the relative offset values) shall have been presented to the ULP.

4.3. Introduction to 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 three general 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

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.

The difference is that the RES procedure described above is used to
determine whether or not the target successfully received the first
sequence of the exchange.

4.3.2. Mid-Exchange Sequences

This is the "normal" case, and is handled with the ABTS and RRQ
processes.

4.3.3. Last Sequence in an Exchange

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

In the case of the target, the context for the current exchange is
discarded when the ACK for the last sequence is received.

In the case of the initiator, the context for the current exchange is
discarded when the ACK for the last sequence is sent.

If the ACK does not make it back to the target, then the target
will perform the ABTS protocol. However, the initiator will return a
BA_RJT to the ABTS because it no longer has exchange context. At this
point the target must give up because it has no way to communicate to
the initiator. Thus it discards exchange context and allows the
initiator to allow the ULP timer to expire.

If the last sequence from the target does not make it back to the
initiator (this will be the one containing the FCP_RSP and SCSI
status), then the target will detect a sequence error and perform the
ABTS protocol. When the ABTS is received at the initiator the initiator
returns a BA_ACC describing the exchange context. The target then
performs a normal sequence recovery to return the FCP_RSP to the
intiator.

The recovery process is tried n times by the target.

4.4. Mappings of FCP to Class 2

This section shows how FCP is used with Class 2.

```
4.4.1. WRITE

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

Initiator's timer expires. ACK not received,
so the sequence has failed. Initiator sends ABTS to
make sure that this sequence is aborted.

ATBS     ---------->

         <----------  BA_ACC with Recovery Qualifier

                      Target returns BA_ACC indicating acceptance
                      of the sequence abort. Also included is the
                      Resource Recovery Qualifier.

Initiator starts an R_A_TOV timer (120 seconds)
for use with RRQ.

120 seconds later the timer expires and the resources described in the
RRQ are released. [This need to keep stuff around "forever" in the RRQ
process is also something of a problem...]

Initiator retransmits the same data as before in a new sequence.
```

```
[Charles asked whether it would make sense to only resend part of the
data, if some got through ok. I think that would be too fancy. The
sending of data doesn't take very long...]

          ----------> 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

                      Target discards context for the sequence.

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

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

          <---------- 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.
                      Discard sequence context and 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       ---------->
Initiator discards exchange status since
as far as it's concerned, the command is complete.

                      Target closes exchange and deletes command
                      context.


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

```
4.4.2. READ

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.

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

        <----------  ABTS
BA_ACC  ---------->
with recovery qualifier

                     Target starts R_A_TOV timer for RRQ process.

                     Target retransmits sequence.

        <----------  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.

        <----------  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.

120 seconds later the timer expires and the resources described in the
RRQ are released.

==

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 RES, then an ABTS. Depending on the result of the RES it can
determine whether the command needs to be sent again. If so, it resends
the FCP_CMD IU using the same information and the same exchange
identifier in a new sequence.

[I'm not sure about the same exchange? - Charles]

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 RES to determine what needs to be done.
This is followed by an ABTS to clean up the sequence but maintain the
exchange.

4.5.3 Loss of ABTS after loss of FCP_CMD sequence. In this case
the ABTS protocol times out. The ABTS protocol is tried n times.

4.5.4. Loss of ABTS after loss of ACK after FCP_CMD sequence. In this
case the target has constructed exchange context, which is detected by
the RES. The ABTS is retried n times.

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.
- Good resource management, event driven in all cases:
    - Sequence recipient releases context as soon as ACK is sent.
    - Sequence initiator releases context as soon as ACK is received.
    - Target releases command (exchange) context as soon as final ACK is received for the command.
    - Initiator releases command (exchange) context as soon as final ACK is sent for the command.