To:	T10 Technical Committee
From:	Rob Elliott, HP (elliott@hp.com)
Date:	6 May 2003
Subject:	T10/03-186r0 SAS-1.1 Transport layer retries

Revision History
Revision 0 (6 May 2003) first revision

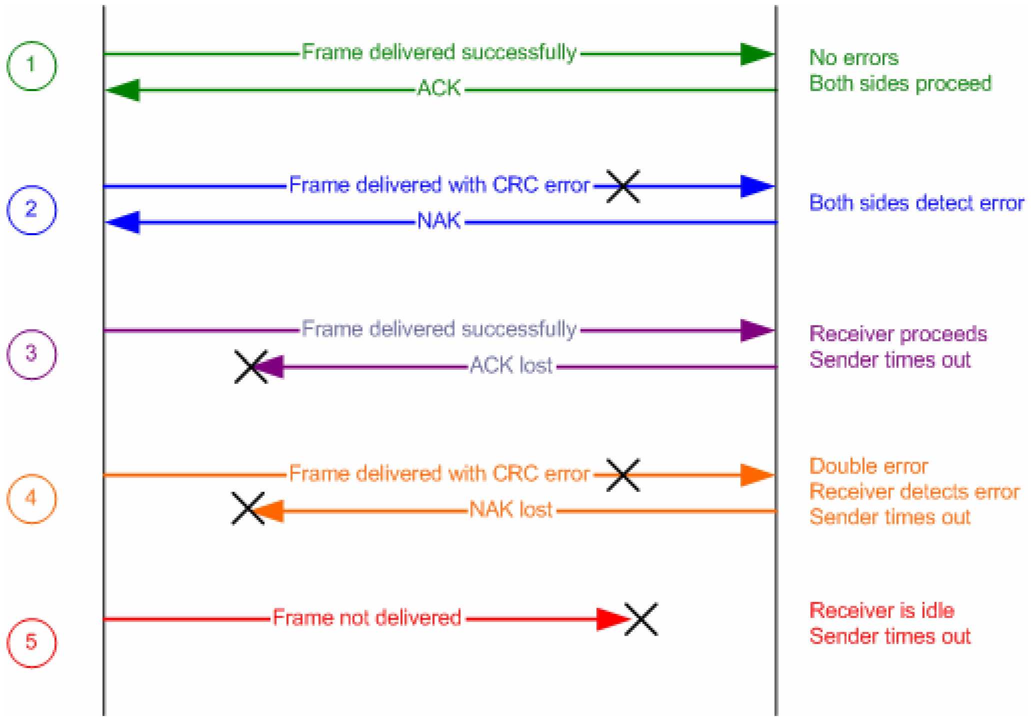
Related Documents
sas-r03g – Serial Attached SCSI revision 3g

Overview
In SAS-1, errors transmitting frames result in either the logical unit terminating the command with CHECK CONDITION status, or the application client aborting the command.

Possible responses to sending a frame:

- frame arrives; ACK arrives
- frame arrives with CRC error; NAK arrives
- frame arrives; ACK lost
- frame arrives; NAK lost
- frame lost

Possible errors:



Depending on the type of frame, different things happen in SAS-1.1:

	COMMAND or TASK (I to T)	**XFER_RDY (T to I)**	**RESPONSE (T to I)**	**read DATA (T to I)**	**write DATA (I to T)**
1. Frame arrives OK; ACK arrives OK	Target runs the command after sending ACK.	Initiator replies with write DATA frame(s).	Both finish the command. Initiator can reuse tag after evidence of target progression.	Move on to more data or RESPONSE (or XFER_RDY for bidi commands)	Move on to more data, XFER_RDY, or RESPONSE (or read DATA for bidi commands)
2. Frame arrives with CRC error; NAK arrives OK	Target idle after sending NAK. Initiator sees the NAK and can resend the command.	Target terminates command.	Target can resend with retransmit=1.	Target terminates command	Initiator aborts command
3. Frame arrives OK; ACK lost	Target runs the command after sending ACK. I to T direction hung interlocked. Initiator detects ACK/NAK timeout in 1 ms. T to I direction could deliver DATA, XFER_RDY, or RESPONSE frames in that time. *Initiator cannot tell between ACK lost, NAK lost, and frame lost w/o using QUERY TASK.*	T to I hung interlocked. Target detects ACK/NAK timeout in 1 ms. I to T direction could deliver write DATA frames in that time. *Target cannot tell between ACK lost, NAK lost, and frame lost.*	T to I hung interlocked. Target detects ACK/NAK timeout in 1 ms. I to T direction not supposed to deliver new command with the same tag yet. Target resends with retransmit=1 in a new connection.	Subsequent ACKs/NAKs for data in flight are misassigned by the target. Target ACK/NAK timeout. Target terminates cmd.	Subsequent ACK/NAKs for data in flight are misassigned by the initiator. Initiator ACK/NAK timeout. Initiator aborts cmd.
4. Frame arrives with CRC error; NAK lost (double error)	Target idle after sending NAK. I to T direction hung interlocked. Initiator detects ACK/NAK timeout in 1 ms. Initiator can resend cmd later. *Initiator cannot tell between ACK lost, NAK lost, and frame lost w/o using QUERY TASK.*	T to I hung interlocked. Target detects ACK/NAK timeout in 1 ms. *Target cannot tell between ACK lost, NAK lost, and frame lost.*	T to I hung interlocked. Target detects ACK/NAK timeout in 1 ms. Target resends with retransmit=1 in new connection.	Subsequent ACKs/NAKs for data in flight are misassigned by the target. Initiator sees data offset gap. Target ACK/NAK timeout. Target terminates command. Initiator aborts cmd because of gap (if subseq. DATA frames)	Subsequent ACK/NAKs for data in flight are misassigned by the initiator. Initiator ACK/NAK timeout. Initiator aborts cmd. Target terminates command because of gap (if subseq. DATA frames). Initiator Response Timeout may occur causing target to terminate cmd.
5. Frame lost	Target idle. I to T direction hung interlocked. Initiator detects ACK/NAK timeout in 1 ms. Initiator can resend cmd later. *Initiator cannot tell between ACK lost, NAK lost, and frame lost w/o using QUERY TASK.*	T to I hung interlocked. Target detects ACK/NAK timeout in 1 ms. Target terminates cmd. *Target cannot tell between ACK lost, NAK lost, and frame lost.*	T to I hung interlocked. Target detects ACK/NAK timeout in 1 ms. Target resends with retransmit=1 in new connection.	Subsequent ACK/NAKs for data in flight are misassigned by the target. Initiator sees data offset gap. Target ACK/NAK timeout. Target terminates command. Initiator aborts cmd because of gap.	Subsequent ACK/NAKs for data in flight are misassigned by the initiator. Initiator ACK/NAK timeout. Initiator aborts cmd. Target terminates command because of gap (if subseq. DATA frames). Initiator Response Timeout may occur causing target to terminate cmd.

<u>Discussion in 5/6 SAS Protocol WG</u>

Some backup applications will fail the whole backup if any of their commands are terminated with CHECK CONDITION.

Write commands – rewinding to the last XFER_RDY would be good
Read commands – rewinding to start of command may not be needed. Restore software does better job of retrying.

Special recovery features not always wanted – per LUN controls needed
Mode page per LUN to enable/disable special recovery mode (not EMDP bit)
For write data problems

- Bit in XFER_RDY to tell the initiator that it should not get upset if its DATA frames encounter errors.
- Don't want to depend on one XFER_RDY per write data frame
- Initiator sends a RECOVERY frame to target if it has trouble sending a DATA frame. Target can then send a new XFER_RDY when it wants.

or probably simpler:

- Bit in XFER_RDY to tell the initiator to retry write DATA frames that encounter errors
- initiator resends DATA frames that are or may be bad (backing up to last XFER_RDY point if unsure about any frame in that data region)
- each frame has correct Data Offset and retransmit=1 (all resent frames have retransmit=1).
- If resent frames cross a RESPONSE frame, target discards subsequent frames (with a now-unknown tag)
- if resent frames cross an XFER_RDY, initiator has to accept it. Target should use different Target Port Transfer Tag to help differentiate them.
- crossing only occurs if an ACK/NAK timeout occurs because an ACK was lost.
- receiving a RESPONSE frame or XFER_RDY does NOT serve as a link layer ACK for the outbound direction. That frame must be honored, though.

For XFER_RDY problems

- if target sees write DATA frames without seeing an ACK for the XFER_RDY, target discards them
- target sends XFER_RDY with retransmit=1 if XFER_RDY has problems
- write DATA frames in response to an XFER_RDY with retransmit do not themselves have retransmit=1
- target shall change Target Port Transfer Tag between the XFER_RDYs
- initiator might stop transferring DATA for the first one, but doesn't have to

For read data problems

- let target retry any read DATA frames with retransmit=1
- also add a "changing data pointer" bit sent when it knows it is rewinding
- target frames are monotonically increasing between those bits
- target must get ACK/NAK balance before going throwing away data that might have to be retransmitted

For command problems

- initiator could receive a RESPONSE frame before it gets an ACK for a command (if ACK is lost). Command might be REWIND or ERASE. Need to treat the command as successful somehow.
- Must accept RESPONSE at any time during this
- If ACK/NAK timeout occurs, send QUERY TASK (in new connection) to see if the command is there.
- If there, then assume it's running fine.
- If not, then free to resend the command.

- If RESPONSE slipped in, don't try to abort the command and don't report it as aborted to the app client.
- if QUERY TASK is not supported, initiator will have to try ABORT TASK instead. Tape drives should support QUERY TASK.

Suggested Changes
TBD.