Accredited Standards Committee
X3, Information Processing Systems

Doc: X3T10.1 / 96a139r0
Date: May 16, 1996
Project:
Ref Doc.:
Reply to: John Scheible

To: X3T10.1 Membership
From: John Scheible

Subject: SSA Pitch to tape people at X3T10 SCC meeting

BACKGROUND

For your information...

On 5/09/96 I presented the following presentation to the SCC study group during the X3T10 week in Ft. Lauderdale immediately following the Fiber Channel presentation. See the SCC Working group minutes for more details (X3T10/96-174r0).

Fiber Channel presented (Doug Hagerman) FC-AL running the Private Loop Direct Attach (FC-DA) with the purpose of insuring Tape Device support. The goal was live within the restrictions of FC-AL running over the FC-PLDA (Fiber Channel Private Loop Device Attach profile).

The key concerns raised in the FC-PLDA presentation (X3T10/96-156r0) was the following:

1) Since neither frame nor sequence level re-transmission is to be used, an error will cause the loss of a exchange (command) which must be resent. With a potential error rate of once every 17 minutes, a command could fail every 17 minutes. The profile does not allow for re-request of data, therefore a bit error in the data transfer causes the data transfer to fail. In either case, a fully utilized bus will cause a command to fail once every 17 minutes.
2) There was some question as to whether a bit error caused a sequence to disappear (without notifying the upper level) or to create an error notification to the upper level. This is especially critical, since if the last sequence is lost, and no notification occurs, then the command hangs.
3) Since any bit error causes the command to terminate (without knowing where the error was), the command must be retried and the tape device must reposition to the start of the command and rewrite good data. This is not optimum.

The tape applications were broken into two camps, small (limited to buffer size) and large (such as geophysical) transfer sizes. A poll of the companies determined that 64K bytes was the current size of most transfers with geophysical going to maybe 4 M bytes. This argues that the loss of a command every 17 minutes might not be so bad (although clearing the queue and re-issuing all commands could make it worse). The 64K and 4M limits were today without any guarantee for the future direction.

I then presented SSA with an emphasis on SSA features and how to minimize conversion. The SSA features of frame level retry was an advantage as was 85% target code reuse a hit. The group was particularly impressed with the confirmed status feature and its ability to restart commands and redirect commands over an alternate path when a break occurred (such as hot plug). The SSA direction of hiding many serial features in the protocol layer and making the device driver layer as close to parallel SCSI as possible was also appreciated.

Sincerely,

John Scheible
Voice: (512) 823-8208
FAX: (512) 823-0758
Email: Scheible@vnet.ibm.com

PURPOSE

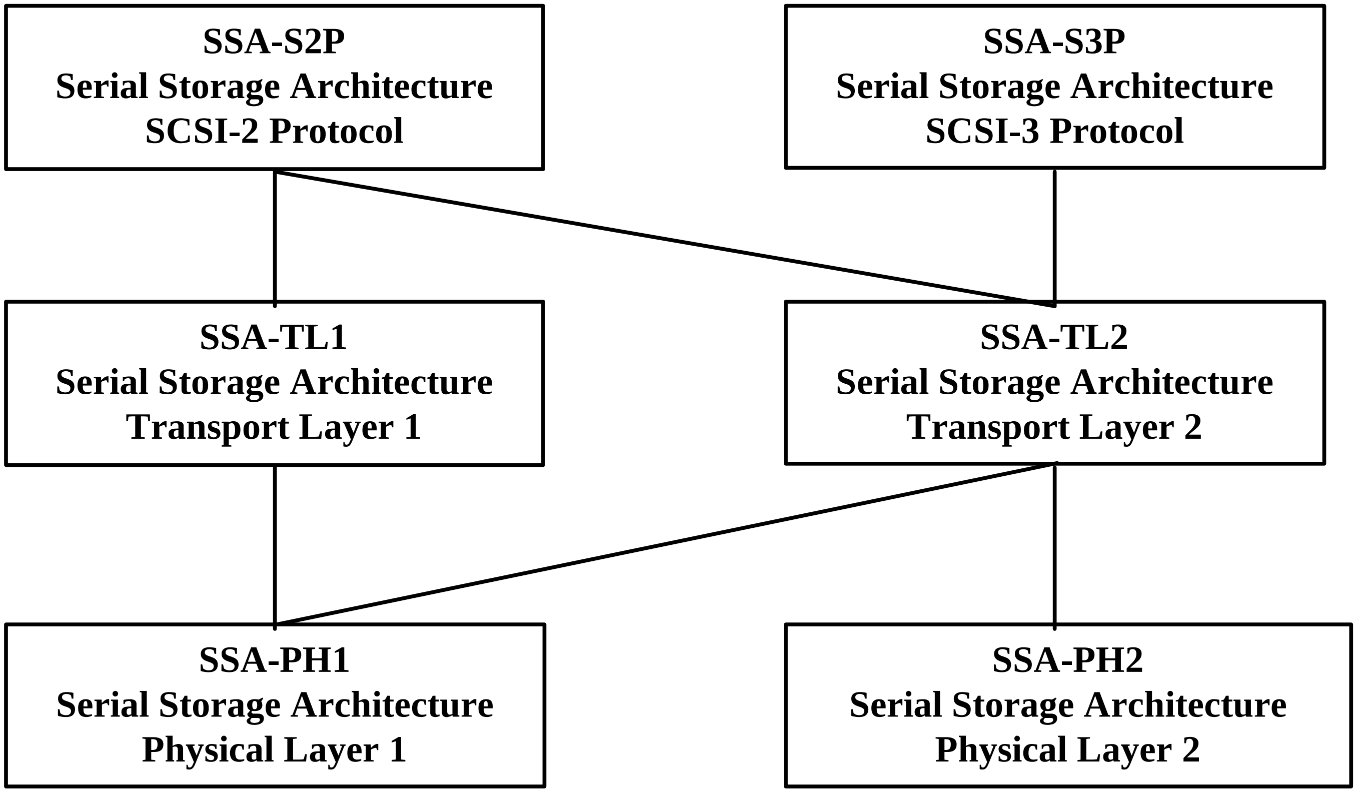
- Discussion conversion between parallel SCSI and serial SSA

- New features of SSA

AGENDA

- What is SSA?
- Improvements
- Minimize conversion
- Optional enhancements

WHAT IS SSA?


SSA-S2P
Serial Storage Architecture
SCSI-2 Protocol
SSA-S3P
Serial Storage Architecture
SCSI-3 Protocol
SSA-TL1
Serial Storage Architecture
Transport Layer 1
SSA-TL2
Serial Storage Architecture
Transport Layer 2
SSA-PH1
Serial Storage Architecture
Physical Layer 1
SSA-PH2
Serial Storage Architecture
Physical Layer 2

PH1 is 20 MB/s per direction per port

PH2 is 40 MB/s per direction per port

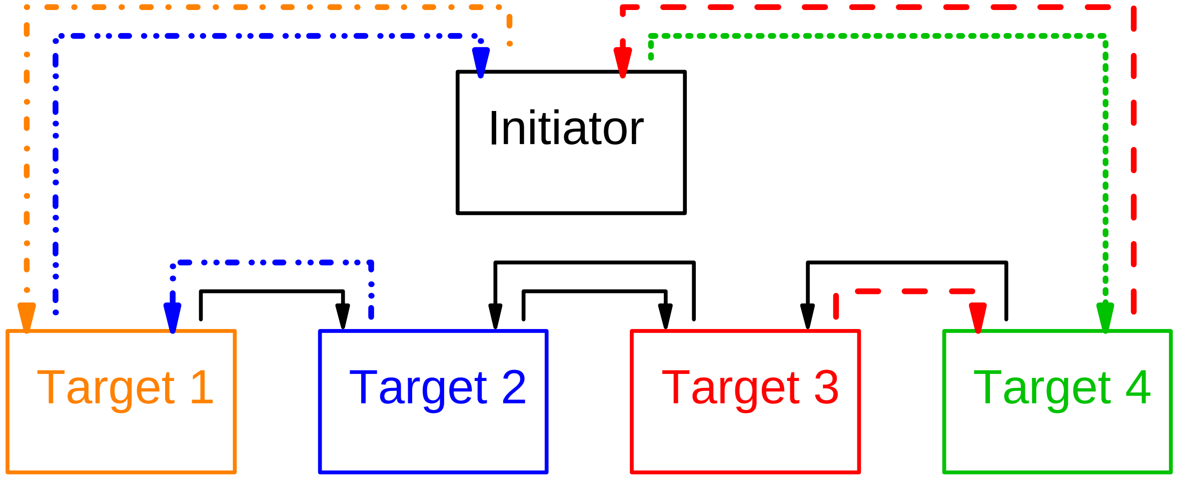
WHAT IS SSA?

- Packetized serial interface
- Series of point to point connections in loop, string, switch
- Redriven, terminated at each hop
- Originally a storage interface
- Hundreds of terrabytes of production level in field today
- Main goal to minimize conversion from parallel SCSI while gaining performance

IMPROVEMENTS

- Peak rate: 160 MB/s (simple loop)
- No arbitration
- Combine messages
- No terminator blocks
- Better error rates

PEAK RATE = 160 MB/s



Initiator

Target 1

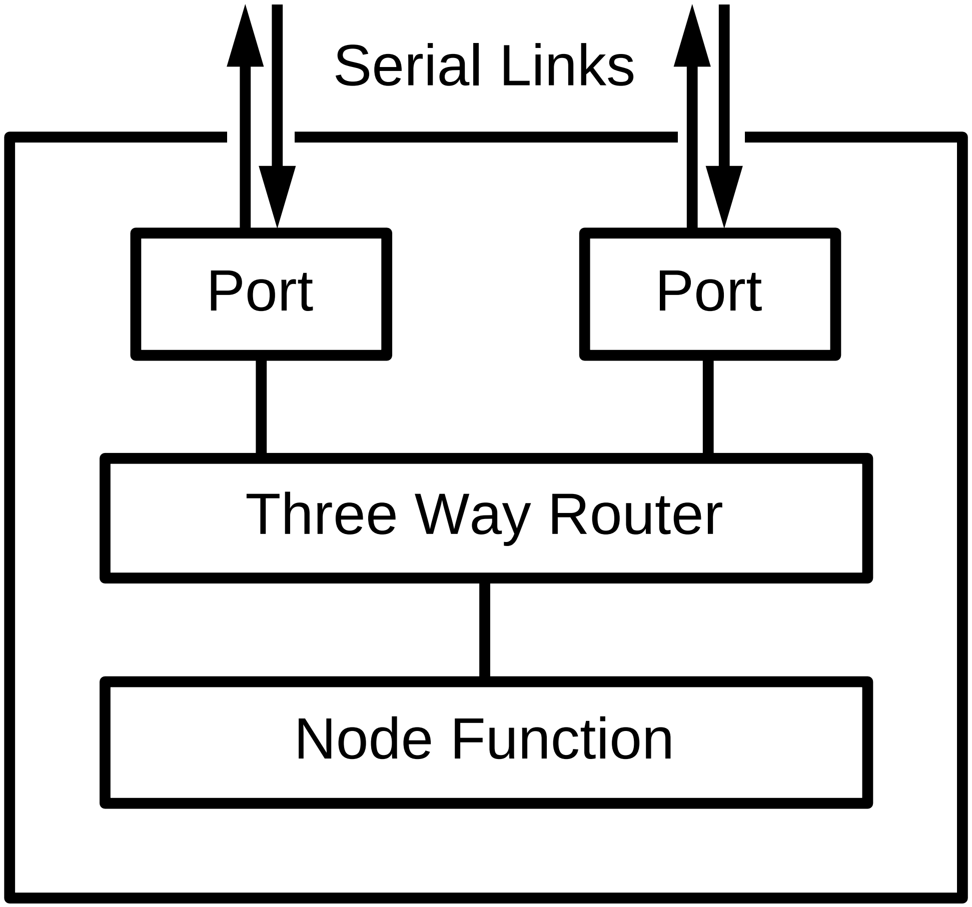
Target 2

Target 3

Target 4

- Write to Target 1 (40 MB/s) -·-·-·-
- Read from Target 2 (40 MB/s) -··-··-··-
- Read from Target 3 (40 MB/s) – – – –
- Write to Target 4 (40 MB/s) ···············

NO ARBITRATION



COMBINED MESSAGES

Parallel SCSI	SSA
Arbitration phase	SCSI COMMAND frame
Selection phase	
Message out phase Identify Queue Tag	
Command phase	
(transfer CDB (async))	
Message In Disconnect	
Arbitrate	DATA REQUEST frame Data frames
Reselection	
Message out phase Identify Queue Tag	
Data Transfer	
(data, Sync)	
Status	SCSI STATUS frame
Message In Command Complete	

NO TERMINATOR BLOCKS

- Each hop is terminated on the device
- Each hop is re-driven (more noise immune)

BETTER ERROR RATES

- Error rate of less than 1 error in 10^{12} bits
- Frame level retry

MINIMIZE CONVERSION

- 85% code reuse from parallel SCSI (target)

- New code for
  - Configuration process
  - Link Error recovery
  - Reformatted messages

- Same as parallel SCSI

Queueing, Command Descriptor Blocks, Status bytes, Sense Data

Similarities (Continued)

- Ordered delivery (same path for all cmds)
- Confirmed Status
- Disconnect if insufficient buffer

Data transfer messages can break up a larger transfer

Differences

- Pipelined interface, not interlocked
  - Auto Contingent Allegiance
  - Busy, Reservation Conflict, Queue Full, Etc.

- Support of Contingent Allegiance

- Multiple paths
  - Backup in case of error (no problem)
  - Hot plug (dynamic repath of commands)
  - Multiple active paths (not ordered across paths)

- Unique ID

Serial and networks use 8 byte globally unique Ids
rather than 3 or 4 bits of jumper block (or SCAM)

OPTIONAL ENHANCEMENTS

- DDRM (Disable Data Ready Message)

  Improve performance at expense of DMA resources

- OOT (Out of order transfers)

  Split R/W supported easily

- Auto Sense

  Easy to concatenate sense data to status

- Asynchronous Event Notification

Easy to send sense data at will

- Confirmed Status

Command not complete until status receipt confirmed

- Restart interrupted command

Rather than confirm status, initiator requests command restart at a specified point over a specified path (active or queued)

- Hot plug

Use restart feature to redirect/restart commands over a different path