To: T10 Technical Committee
From: Rob Elliott, Compaq Computer Corporation (Robert.Elliott@compaq.com)
Date: 31 December 1999
Subject: Target-controlled congestion relief

One basic problem revealed by the BUSY/TASK SET FULL debate is that the target is the only device that knows when it has a queue slot free, not the initiator. Any attempts by the initiator to guess when the target has a queue slot free will be imperfect, either consuming excess bus traffic with retries or not fully utilizing the target's queues. Letting the target indicate when to continue the connection solves this problem.

When the initiator tries to send a command to a target that cannot accept it, the target could return a response that instructs the initiator to stop sending commands to that target. When the target is ready for a command, it reconnects to the initiator and signals that it has queue slots available. The initiator then sends the command as if it had selected the target. This way, the initiator doesn't waste bus bandwidth on retries trying to figure out when the target is free.

Other proposals:

- Interpret BUSY as "retry immediately" and TASK SET FULL as "retry after some delay." The host CPU is kept busy servicing status messages when the device is busy.
- Interpret TASK SET FULL as "go away until one of your commands has completed." If each initiator Is guaranteed one queue slot, it will always be able to run one command; additional commands might receive TASK SET FULL. In the face of other active initiators, an initiator may become single-threaded. If the other initiators stop using the device, the initiator will not get its commands in as early as if it were retrying often.
- After receiving TASK SET FULL, initiator decrements its limit to the number of commands it will allow to be outstanding. Not clear when to raise the limit.

A new status (WAIT) could be created for this, or TASK SET FULL could be defined to have this new meaning when negotiated between the initiator and target. This is proposed as an optional feature for SPI-4, FCP-3 and SAM-2 or SAM-3. Other protocols should be able to adopt this feature.

1.1 Parallel SCSI

In parallel SCSI, target reselection can be used to signal that a target is ready to accept commands.

1.1.1 Data Group transfers

When a target queue is full, it will follow this procedure:

1. The target returns WAIT when its queue is full
2. The target records initiators to which it sends WAIT. No other state is recorded – the nexus and command information are discarded. The amount of storage required depends on the fairness scheme (see below).
3. The initiator treats WAIT as "stop retrying." It stops trying to connect to that target, holding the command pending. An optimized initiator would store the command information locally. An unoptimized initiator would return the WAIT to software.
4. When the target has a free queue slot, it reconnects to one of the initiators to which it sent TASK SET FULL using ARBITRATION and RESELECTION. Then, it enters MESSAGE OUT phase rather than MESSAGE IN.
5. If the initiator is optimized for this protocol, it expects the reconnection and operates as if it had just run ARBITRATION and SELECTION phases itself. It asserts ATN to acknowledge entry into MESSAGE OUT and sends the IDENTIFY message. The target proceeds as if it had been selected, soon moving to COMMAND phase to accept the command.

6. If the initiator is not optimized fort his protocol, it returns a DISCONNECT message to the target and notifies its software that it can resume sending commands.

This algorithm changes the allowed bus phase transitions in SPI-4 - RESELECTION to MESSAGE OUT would now be allowed.

Each initiator and target would negotiate support for this feature using PPR. An INQUIRY bit would indicate that the feature is supported.

Existing expanders should handle this change, since they base their signal direction on CD IO and MSG signals. Therefore, a domain validation test is probably not needed to identify old expanders who don't support the new phase sequences.

Each target should implement a timeout in case it does not receive ATN after entering MESSAGE OUT phase. This will help if the initiator was reset or was replaced by a new initiator that doesn't support this feature.

Initiators' current command timeouts should apply if they do not get reselected. This will help if the target is reset.

1.1.2 Information Unit transfers

1. The target reconnects
2. If the target wants to do a normal reconnection it would go to the DT DATA IN phase and send a SPI L_Q IU just like it does today.
3. If the target wants to allow the initiator a chance to send a command because of a previous WAIT status it would go to the DT DATA OUT phase after reselection. This would be an indication to the initiator to send a SPI L_Q IU followed by a SPI command IU. The initiator could send multiple commands just like an initial connection. If the initiator does not response in some amount of time (same time that would be used in non-packet mode for waiting for ATN) the target would then go bus free.

1.2 Fibre Channel

Fibre Channel is harder to handle, because addresses are larger and there is another protocol layer involved.

Example:
1. Initiator uses FCP_CMD to send command
2 Target uses FCP_RSP to send WAIT
3. Target uses FCP_RECONNECT to request the command again
4. Initiator uses FCP_CMD to send command
5. Resume as if initiator had started things

Create a new packet named FCP_RECONNECT. It uses category 02 (unsolicited control).

Create a new IU for "Target Command Request". Properties include:

- First/Middle/Last = all
- Sequence Initiative = transferred back to the initiator
- Mandatory/Optional = optional

Add a new field to process login (PRLI) to negotiate support for this IU.

1.2.1 Arbitrated Loop issues

In a loop, the target may not want to do a CLS after sending the FCP_RECONNECT. This depends on how long an initiator would take to find the command again, compared to how long a new loop arbitration and open takes.

1.3 Issues affecting all protocols

1.3.1 Initiator resources

Ideally, the initiator should be able to find the command to resend promptly after being reselected. For SCSI, command information for each target might be kept in local memory. With 16 targets, this is not impractical. For Fibre Channel, however, the number of potential targets is huge. The driver is more likely to pass WAIT and RECONNECT information to higher-level software and let it deal with them.

As the initiator takes longer, it opens a longer window for another initiator to slip in.

1.3.2 Target response

The target is free to issue BUSY, TASK SET FULL, or WAIT after the reselection if needed. On parallel SCSI, the target shouldn't have to do this. This might be needed on interconnects where a new command could arrive after the target has sent out its reconnection request.

1.3.3 Target resources: Number of initiators to remember

The target determines how many outstanding initiators to support. Once it reaches its limit, it always returns BUSY instead of TASK SET FULL.

1.3.4 Fairness

When multiple initiators have requests pending, the target gets to choose how to reconnect to them.

Also, this makes transferring a new command subject to the target's ability to win the bus, not the initiator's ability. In a fixed priority SCSI bus after the target queue has opened a slot, this scheme may let more targets win the bus before the initiator can post its new command.

Fixed – the target could always follow a fixed priority. On SPI, it might follow the fixed arbitration priorities 7, 6, .. 0, 15, 14, .. 8. This implies it only needs 16 bits of storage – one per SCSI ID – to remember the pending reconnections for all possible initiators. FCP devices need multiple bits to store longer addresses, so require more storage.

FIFO – the target could remember the order in which it rejected the initiators, and reconnect with the first one that was rejected. On SPI, this means each target needs 16 x 16 bits of storage (to support all devices having a reconnection outstanding)