1394 PWG Technical Report

Date: January 23, 1998
reported by *Fumio Nagasaka*
EPSON Software Development Laboratory Inc.,
Sumiyoshi 54-1,Ueda Nagano-ken Japan 386
VOICE: +81 268 25 4111 FAX: +81 268 25 4627

Subject: IEEE 1284.4 supportive Data Link protocol for SBP-2

1. Purpose

This document summarizes 1394 PWG's problems with unordered execution. This document describes the in-ORB / out-ORB request method for the 1394 PWG profile. This document attempts to clarify why we at the 1394 PWG require 1284.4 CBT as the transport layer.

2. Why 1284.4

2.1. Objections for unordered execution

2.1.1. Out of order completion

The current SBP-2 specification allows for the unordered execution of ORBs (as quoted below).

7.4.8 Logical_Unit_Characteristics entry

….

The ordered bit (abbreviated as o in the figure above) specifies the manner in which the target executes tasks signaled to the normal command block agent. If the target executes and reports completion status without any ordering constraints, the ordered bit shall be zero. Otherwise, if the target both executes all tasks in order and reports their completion status in the same order, the ordered bit shall be one.

….

However once an initiator allows a target to execute ORBs out of order, the initiator may have difficulties appending a new Task T´. These difficulties are described below.

10.2 Basic task management model

….

The unordered model is characterized by unrestricted reordering of the active tasks. The target may reorder the actual execution sequence of any tasks in a task set in any manner. Unrestricted reordering places the responsibility for the assurance of data integrity on the initiator. If the integrity of data on the device medium could be compromised by unrestricted reordering involving a set of active tasks, {T_0, T_1, T_2, … T_N} and a new task T´, the initiator shall wait until {T_0, T_1, T_2, … T_N} have completed before appending T´ to an active request list.

….

If the 1394 PWG profile defines a data link layer based on SBP-2 which allows unordered execution, the overlaying transport layer must have task management functionality. As described above, this is because the transport layer may not be allowed to send new request to the data link layer until the transport confirms that the previous task set has been completed.

2.1.2. ORB_POINTER register

In SBP-2, the ORB_POINTER register keeps a pointer to the ORB which the target is currently executing. Thus, in unordered execution, the initiator might monitor the ORB_POINTER register while the target is executing its linked list of ORBs. However, that is quite awkward for the initiator.

2.2. In order execution

As the result, the 1394 PWG profile shall specify in-order execution for printing transaction.

2.3. Target Agent Stalled problem

However, if the 1394 PWG profile chooses the in-order execution model, it must avoid target agent stalls. For instance, a printer could have some fatal problem. It would not consume any printing data while it has this problem. In this situation, data from an initiator occupies the whole FIFO space of the target device (printer). The target agent will not fetch any more ORBs. If a printer had multiple logical channels, this problem could be avoided. But still, multiple logical channels have a problem, as shown below.

	SBP-2 data link *without* 1284.4 Credit Based Transport layer
	1) If one data channel is stalled, the fetch agent will not pass data toward the upper layer (in this case the upper layer is a transport layer). 2) Thus, the fetch agent stops consuming data from the linked list of ORBs. 3) However, an application might append a new ORB to request a status reply, since the application must solve the problem. (The new ORB would use a different logical channel, but the logical channel number is enclosed in the data buffer associated with this ORB) 4) The fetch agent does not read this new data buffer because it is busy. And it does not have any internal FIFO space to read the new data. 5) Finally the fetch agent does not pass the new request to the upper layer until user solves the printer problem manually.

2.4. Temporary solution raised in telephone-conference

The scheme illustrated above suggests that some logical channel information should be stored in the CDB. If only the data buffer has logical channel information, the fetch agent will be forced to examine data stored in the ORB's data buffer. But if the CDB has channel information, the fetch agent need examine only the ORB data structure.

Logical channel information in the CDB helps a fetch agent deliver requests to specific logical

channels. Thus, even if one logical channel was stalled, the fetch agent could still deliver requests through other logical channels. In this case, unordered execution might be necessary to skip stalled logical channels.
However a better solution described below was suggested through the telephone conference. In other words, the solution was existing from the beginning of 1394 PWG's discussion. And it was re-discovered again, through the telephone conference.

2.5. How 1284.4 CBT solves this problem

The 1394 PWG profile shall use the 1284.4 Credit Based Transport layer to avoid the agent stalled problem, as illustrated below.

	SBP-2 data link *with* 1284.4 Credit Based Transport layer
	1) When an initiator starts the transaction, the target peer gives credits for each logical channel. 2) If the initiator's out-ORBs do not exceed the given credits, all of these out-ORBs will be stored in the target peer's FIFO without over-flow. 3) Thus logical channels will not be stalled.

2.6. Reverse directional transaction

2.6.1. A target agent does not consume in-ORBs in some case

On the other hand, when the target peer starts a transaction, the initiator peer gives credits for each logical channel. In this case the initiator needs to append one in-ORB for each credit. Because if the initiator appends N+k in-ORBs in a list, and the target has only N requests, then k in-ORBs will not be consumed.

2.6.2. An initiator appends one in-ORB at a time

From this point of view, an initiator shall place one in-ORB at a time. Each in-ORB shall have a notify bit set to one. The initiator will be notified when the provided in-ORB is consumed by the target peer. Then the initiator may append the next in-ORB onto the linked list.

End of file