Protocol Question: What happens when a Data Packet Header (DPH) is determined as invalid by a receiver?

USB 3.0 retransmission mechanisms are quite different from USB 2.0 because of the new routing infrastructure. USB 3.0 has separate retransmission mechanisms in the Link and the Protocol Layers for optimizing data buffering in devices and especially hubs. We will examine in this document the traffic exchanged between two link partners (such as a host and a device) when a DPH is received as invalid by one end.

Firstly, we should characterize what invalidates a header packet. Section 7.2.4.1.4 of the USB 3.0 specification states that a header packet is "received properly" when the following conditions are met:

- the CRC5 and CRC16 are correct;
- the receiving device has a header buffer available;
- the Header Sequence Number (HSN) matches the expected Next Rx Header Sequence Number (NRS) on the receiving device.

In our example, a host will send a Data Packet (DP) which consists of a Data Packet Header (DPH), followed by its associated Data Packet Payload (DPP), to a device. Note that the DPH contains two CRC fields (CRC5 and CRC16). The CRC16 protects the preceding 12 bytes and the CRC5 is used to protect the LCW located within the DPH.

The DPH and DPP are transmitted contiguously by the host. The DPH received by the device is checked for the criteria mentioned above, and in this case, logic at the device's Link Layer determines the CRC16 received is not correct. The DPH is discarded and an LBAD link command is queued for transmission back to the host. The LBAD instructs the host to resend not just this header packet, but also any header packet that was sent but not acknowledged by the device via the LGOOD_n link command.

The device sends the LBAD to the host, and will also ignore all subsequent packets until it receives a LRTY link command (or until the link enters the recovery state). The LRTY is now sent by the host followed by the replayed DPH. The replayed DPH will contain the same HSN that was previously rejected (HSN = 0), and it will also set the Delayed (DL) bit in its LCW to indicate a replayed header (the LCW's CRC5 is also recalculated).

So whatever happened to the DPP associated with the corrupted DPH? According to Section 7.2.4.1.6 of the specification, the DPP must be dropped by the receiver if the DPH is bad. But the link layer is only responsible to control the resending of the packet header, not the DPP, as we saw via the LBAD/LRTY exchange above.

The replay sequence for the DPP is managed by the protocol layer. The device responds to the host transmission of the second DPH with an LGOOD_0, indicating good reception of the replayed DPH. This replayed DPH did not include an associated DPP.

At this point, the device releases a credit to the host (LCREDIT_A) and the protocol layer on the device now "knows" the DPP is invalid, missing actually, since only the DPH was sent (no DPP).

The device now replies to the host with an ACK Transaction Packet (TP) with the Retry bit set. The host replies with an LGOOD_0 and LCREDIT_A pairing, followed by the DP (a third DPH, now with the DPP attached). The device's Link Layer acknowledge the DPH with LGOOD_1 and LCREDIT_B, and acknowledges the DPP with an ACK TP.

Dir	Traffic
→	DPH (HSN=0, SeqNum=0, CRC16 Invalid)
→	DPP
←	LBAD
→	LRTY
→	DPH (HSN=0, SeqNum=0, DL=1)
←	LGOOD_0
←	LCREDIT_A
←	ACK TP (HSN=0, SeqNum=0, Retry=1)
→	LGOOD_0
→	LCREDIT_A
→	DPH (HSN=1, SeqNum=0)
→	DPP
←	LGOOD_1
←	LCREDIT_B
←	ACK TP (HSN=1, SeqNum=1, Retry=0)
→	LGOOD_1
→	LCREDIT_B

Note: → From Host ← From Device

Summary

To summarize this sequence, the DPH is sent three times. The first was corrupt, the second was a replay, but without the DPP attached, and the third precedes the replayed DPP. The link layer managed the corrupt DPH and replay of the DPH, while the protocol layer managed the replay of the DPP. Annex B below represents the same retransmission as above, but involving Host, a Hub and a Device.

Note that there are other possibilities to the sequences described here, for example a recovery sequence will have certain effects not described here as will sequences involving multiple DPs and certain issues resulting in timeout conditions, corrupt link commands, etc. This example simplifies and streamlines things somewhat in order to focus on the effects of a bad CRC in the DPH. As always, refer to the specification for the final word.

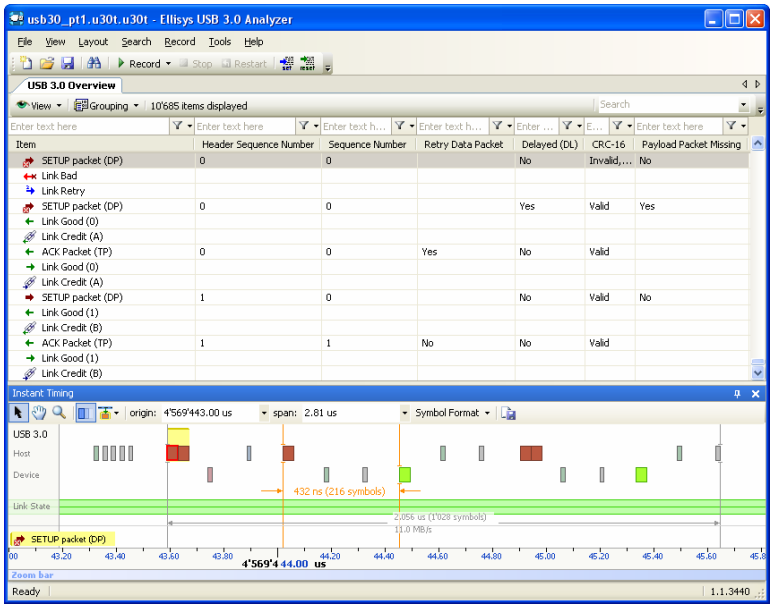
Resources

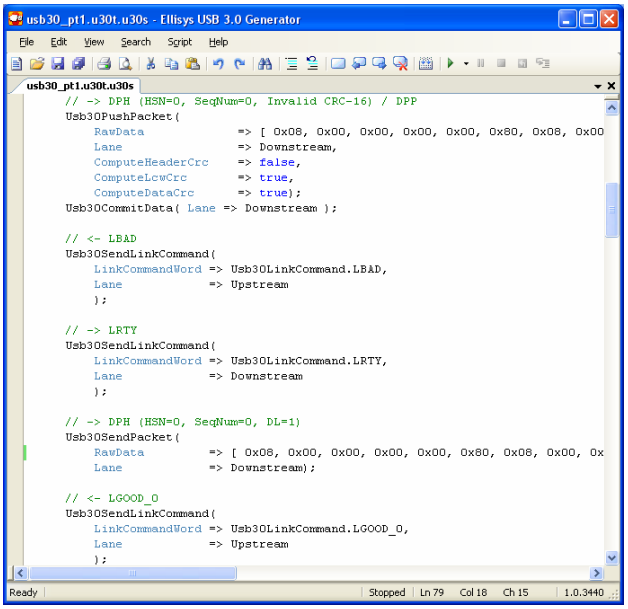
To see this sequence in the Ellisys EX280A SuperSpeed USB 3.0 protocol analyzer viewer application, please visit:
www.ellisys.com/technology/ellisys_usb30_note1.u30t

To understand how to manage this sequence using the Ellisys EX280G SuperSpeed USB 3.0 protocol generator, please see:
www.ellisys.com/technology/ellisys_usb30_note1.u30s

The latest version of this document is available online at:
www.ellisys.com/technology/ellisys_usb30_note1.pdf

Annex A: Using Ellisys EX280 SuperSpeed USB 3.0 Analyzer and Generator to Characterize this Example





Analyzer: The EX280 Analyzer captures the sequence, and illustrates it in the USB 3.0 Overview as well as the Instant Timing view. Note that the first replayed DP Header is flagged by the analyzer with a "Yes" in the "Payload Packet Missing" field (a fly-over in this DPH also indicates the missing DPP). Two sets of cursors in the Instant Timing view show the time consumed for the entire sequence (2.056 us) as well as the time between the first replayed DPH and the following ACK TP from the device (432 ns). The CRC-16 error is visible in the Details view, the USB 3.0 Overview, and in the Instant Timing view.

Generator: The EX280 Generator is able to create this test case. In this case, the Generator emulates the host, sending the invalid DPH CRC-16 to a device (note that the Generator can also work in a "hub" mode to handle both host and device activities, which is helpful in testing scripts). In the screenshot above, the DPH is corrupted with the invlaid CRC-16 by temporarily turning off the Generator's CRC Auto-Compute function to enable the transmission of an invalid CRC-16.

Annex B: Same retransmission as above, but involving a Host, a Hub and a Device

Host	Hub	Dev	Traffic
→			DPH (HSN=0, SeqNum=0, CRC16 OK)
→			DPP
←			LGOOD_0
	→		DPH (HSN=0, SeqNum=0, CRC16 Invalid)
	→		DPP
	←		LBAD
	→		LRTY
	→		DPH (HSN=0, SeqNum=0, DL=1)
	←		LGOOD_0
←			LCREDIT_A
	←		LCREDIT_A
	←		ACK TP (HSN=0, SeqNum=0, Retry=1)
	→		LGOOD_0
←			ACK TP (HSN=0, SeqNum=0, Retry=1)
→			LGOOD_0
	→		LCREDIT_A
→			LCREDIT_A
→			DPH (HSN=1, SeqNum=0)
→			DPP
←			LGOOD_1
	→		DPH (HSN=1, SeqNum=0)
	→		DPP
	←		LGOOD_1
	←		LCREDIT_B
←			LCREDIT_B
	←		ACK TP (HSN=1, SeqNum=1, Retry=0)
	→		LGOOD_1
←			ACK TP (HSN=1, SeqNum=1, Retry=0)
→			LGOOD_1
	→		LCREDIT_B
→			LCREDIT_B

Note 1 – Link Layer and Protocol Layer retransmission mechanisms

Rev. A. June 1st, 2009