



author Michal Pecio <michal.pecio@gmail.com> 2025-03-06 16:49:44 +0200
committer Greg Kroah-Hartman <gregkh@linuxfoundation.org> 2025-05-02 07:59:18 +0200
commit [16a7a8e6c47fea5c847beb696c8c21a7a44c1915](#) (patch)
tree [097e20cfd8e8dcd0285ad8956ad6355d10fa5aa1](#)
parent [cbfa55bda1feb96dac7cf2312d607dddef768089](#) (diff)
download [linux-16a7a8e6c47fea5c847beb696c8c21a7a44c1915.tar.gz](#)

diff options

context:
space:
mode:

usb: xhci: Fix isochronous Ring Underrun/Overrun event handling

[Upstream commit 906dec15b9b321b546fd31a3c99ffc13724c7af4]

The TRB pointer of these events points at enqueue at the time of error occurrence on xHCI 1.1+ HCs or it's NULL on older ones. By the time we are handling the event, a new TD may be queued at this ring position.

I can trigger this race by rising interrupt moderation to increase IRQ handling delay. Similar delay may occur naturally due to system load.

If this ever happens after a Missed Service Error, missed TDs will be skipped and the new TD processed as if it matched the event. It could be given back prematurely, risking data loss or buffer UAF by the xHC.

Don't complete TDs on xrun events and don't warn if queued TDs don't match the event's TRB pointer, which can be NULL or a link/no-op TRB. Don't warn if there are no queued TDs at all.

Now that it's safe, also handle xrun events if the skip flag is clear. This ensures completion of any TD stuck in 'error mid TD' state right before the xrun event, which could happen if a driver submits a finite number of URBs to a buggy HC and then an error occurs on the last TD.

Signed-off-by: Michal Pecio <michal.pecio@gmail.com>

Signed-off-by: Mathias Nyman <mathias.nyman@linux.intel.com>

Link: <https://lore.kernel.org/r/20250306144954.3507700-6-mathias.nyman@linux.intel.com>

Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>

Signed-off-by: Sasha Levin <sasha@kernel.org>

Diffstat

```
-rw-r--r-- drivers/usb/host/xhci-ring.c 20
```

1 files changed, 14 insertions, 6 deletions

diff --git a/drivers/usb/host/xhci-ring.c b/drivers/usb/host/xhci-ring.c
index 7001f9725cf7c7..6eb297b168fa44 100644

--- a/drivers/usb/host/xhci-ring.c

+++ b/drivers/usb/host/xhci-ring.c

```
@@ -2662,6 +2662,7 @@ static int handle_tx_event(struct xhci_hcd *xhci,  
    int status = -EINPROGRESS;  
    struct xhci_ep_ctx *ep_ctx;  
    u32 trb_comp_code;  
+    bool ring_xrun_event = false;
```

```

slot_id = TRB_TO_SLOT_ID(le32_to_cpu(event->flags));
ep_index = TRB_TO_EP_ID(le32_to_cpu(event->flags)) - 1;
@@ -2768,14 +2769,12 @@ static int handle_tx_event(struct xhci_hcd *xhci,
    * Underrun Event for OUT Isoch endpoint.
    */
    xhci_dbg(xhci, "Underrun event on slot %u ep %u\n", slot_id, ep_index);
-   if (ep->skip)
-       break;
-   return 0;
+   ring_xrun_event = true;
+   break;
    case COMP_RING_OVERRUN:
        xhci_dbg(xhci, "Overrun event on slot %u ep %u\n", slot_id, ep_index);
-       if (ep->skip)
-           break;
-       return 0;
+       ring_xrun_event = true;
+       break;
    case COMP_MISSED_SERVICE_ERROR:
        /*
         * When encounter missed service error, one or more isoc tds
@@ -2851,6 +2850,7 @@ static int handle_tx_event(struct xhci_hcd *xhci,
        */
        if (trb_comp_code != COMP_STOPPED &&
            trb_comp_code != COMP_STOPPED_LENGTH_INVALID &&
+           !ring_xrun_event &&
            !ep_ring->last_td_was_short) {
            xhci_warn(xhci, "Event TRB for slot %u ep %u with no TDs queued\n",
                      slot_id, ep_index);
@@ -2885,6 +2885,10 @@ static int handle_tx_event(struct xhci_hcd *xhci,
            goto check_endpoint_halted;
        }

+       /* TD was queued after xrun, maybe xrun was on a link, don't panic yet */
+       if (ring_xrun_event)
+           return 0;
+
        /*
         * Skip the Force Stopped Event. The 'ep_trb' of FSE is not in the current
         * TD pointed by 'ep_ring->dequeue' because that the hardware dequeue
@@ -2931,6 +2935,10 @@ static int handle_tx_event(struct xhci_hcd *xhci,
        */
        } while (ep->skip);

+       /* Get out if a TD was queued at enqueue after the xrun occurred */
+       if (ring_xrun_event)
+           return 0;
+
        if (trb_comp_code == COMP_SHORT_PACKET)
            ep_ring->last_td_was_short = true;
        else

```