author	Jeremy Kerr <jk@codeconstruct.com.au>	2022-11-10 13:31:35 +0800
committer	Greg Kroah-Hartman <gregkh@linuxfoundation.org>	2022-11-26 09:27:34 +0100
commit	a5915a9a3ab4067ef8996a57738d156eabeb3a12 (patch)	
tree	7f15352bb31003683e39823fd83475c872c67591	
parent	9a23917697f9f8ddeb5a4424eb45df73fdc7e036 (diff)	
download	linux-a5915a9a3ab4067ef8996a57738d156eabeb3a12.tar.gz	

diff options

context: 3

space: include

mode: unified

mctp i2c: don't count unused / invalid keys for flow release

```
[ Upstream commit 9cbd48d5fa14e4c65f8580de16686077f7cea02b ]

We're currently hitting the WARN_ON in mctp_i2c_flow_release:

    if (midev->release_count > midev->i2c_lock_count) {
        WARN_ONCE(1, "release count overflow");

This may be hit if we expire a flow before sending the first packet it
contains - as we will not be pairing the increment of release_count
(performed on flow release) with the i2c lock operation (only
performed on actual TX).

To fix this, only release a flow if we've encountered it previously (ie,
dev_flow_state does not indicate NEW), as we will mark the flow as
ACTIVE at the same time as accounting for the i2c lock operation. We
also need to add an INVALID flow state, to indicate when we've done the
release.

Fixes: f5b8abf9fc3d ("mctp i2c: MCTP I2C binding driver")
Reported-by: Jian Zhang <zhangjian.3032@bytedance.com>
Tested-by: Jian Zhang <zhangjian.3032@bytedance.com>
Signed-off-by: Jeremy Kerr <jk@codeconstruct.com.au>
Link: https://lore.kernel.org/r/20221110053135.329071-1-jk@codeconstruct.com.au
Signed-off-by: Jakub Kicinski <kuba@kernel.org>
Signed-off-by: Sasha Levin <sashal@kernel.org>
```

Diffstat

-rw-r--r--	drivers/net/mctp/mctp-i2c.c	47

1 files changed, 32 insertions, 15 deletions

```
diff --git a/drivers/net/mctp/mctp-i2c.c b/drivers/net/mctp/mctp-i2c.c
index 53846c6b56ca2f..aca3697b09620a 100644
--- a/drivers/net/mctp/mctp-i2c.c
+++ b/drivers/net/mctp/mctp-i2c.c
@@ -43,6 +43,7 @@
 enum {
         MCTP_I2C_FLOW_STATE_NEW = 0,
         MCTP_I2C_FLOW_STATE_ACTIVE,
+        MCTP_I2C_FLOW_STATE_INVALID,
 };
```

```
 /* List of all struct mctp_i2c_client
@@ -374,12 +375,18 @@ mctp_i2c_get_tx_flow_state(struct mctp_i2c_dev *midev, struct sk_buff *skb)
          */
         if (!key->valid) {
                 state = MCTP_I2C_TX_FLOW_INVALID;
-
-        } else if (key->dev_flow_state == MCTP_I2C_FLOW_STATE_NEW) {
-                key->dev_flow_state = MCTP_I2C_FLOW_STATE_ACTIVE;
-                state = MCTP_I2C_TX_FLOW_NEW;
         } else {
-                state = MCTP_I2C_TX_FLOW_EXISTING;
+                switch (key->dev_flow_state) {
+                case MCTP_I2C_FLOW_STATE_NEW:
+                        key->dev_flow_state = MCTP_I2C_FLOW_STATE_ACTIVE;
+                        state = MCTP_I2C_TX_FLOW_NEW;
+                        break;
+                case MCTP_I2C_FLOW_STATE_ACTIVE:
+                        state = MCTP_I2C_TX_FLOW_EXISTING;
+                        break;
+                default:
+                        state = MCTP_I2C_TX_FLOW_INVALID;
+                }
         }

         spin_unlock_irqrestore(&key->lock, flags);
@@ -617,21 +624,31 @@ static void mctp_i2c_release_flow(struct mctp_dev *mdev,

 {
         struct mctp_i2c_dev *midev = netdev_priv(mdev->dev);
+        bool queue_release = false;
         unsigned long flags;

         spin_lock_irqsave(&midev->lock, flags);
-        midev->release_count++;
-        spin_unlock_irqrestore(&midev->lock, flags);
-
-        /* Ensure we have a release operation queued, through the fake
-         * marker skb
+        /* if we have seen the flow/key previously, we need to pair the
+         * original lock with a release
          */
-        spin_lock(&midev->tx_queue.lock);
-        if (!midev->unlock_marker.next)
-                __skb_queue_tail(&midev->tx_queue, &midev->unlock_marker);
-        spin_unlock(&midev->tx_queue.lock);
+        if (key->dev_flow_state == MCTP_I2C_FLOW_STATE_ACTIVE) {
+                midev->release_count++;
+                queue_release = true;
+        }
+        key->dev_flow_state = MCTP_I2C_FLOW_STATE_INVALID;
+        spin_unlock_irqrestore(&midev->lock, flags);

-        wake_up(&midev->tx_wq);
+        if (queue_release) {
+                /* Ensure we have a release operation queued, through the fake
+                 * marker skb
+                 */
+                spin_lock(&midev->tx_queue.lock);
+                if (!midev->unlock_marker.next)
+                        __skb_queue_tail(&midev->tx_queue,
+                                         &midev->unlock_marker);
```

```
+                spin_unlock(&midev->tx_queue.lock);
+                wake_up(&midev->tx_wq);
+        }
 }

 static const struct net_device_ops mctp_i2c_ops = {
```