Linux TX Multiframe Implementation

David S. Miller

Red Hat Inc.

Seattle, 2008
IMPETUS

- Writing NIU driver
- Peter W’s multiqueue hacks
- TX multiqueue will be pervasive.
- Robert Olsson’s pending paper on 10GB routing (found TX locking to be bottleneck in several situations)
<table>
<thead>
<tr>
<th>Presentations</th>
</tr>
</thead>
<tbody>
<tr>
<td>🟢 Tokyo 2008</td>
</tr>
<tr>
<td>🟢 Berlin 2008</td>
</tr>
<tr>
<td>🟢 Implementation plan changing constantly</td>
</tr>
<tr>
<td>🟢 End result was different than all of these designs</td>
</tr>
</tbody>
</table>
dev->queue_lock

dev_queue_xmit() ->

QDISC

TX lock

hard_start_xmit

set SKB queue mapping

Driver

TXQ TXQ TXQ
• Generic networking has no idea about multiple TX queues
• Parallelization impossible because of single queue and TX lock
• Only single root QDISC is possible, sharing is not allowed
Picture with Non-trivial QDISC

- SKB
- TXQ
- TXQ
- TXQ
- qdisc -> q.lock
- skb
- TX lock
- TX lock
- TX lock
- TX lock
- TX lock
- driver
THE NETWORK DEVICE QUEUE

- Direction agnostic, can represent both RX and TX
- Holds:
  - Backpointer to struct net_device
  - Pointer to ROOT qdisc for this queue, and sleeping qdisc
  - TX lock for ->hard_start_xmit() synchronization
  - Flow control state bit (XOFF)
QDISC Changes

- Holds state bits for qdisc_run() scheduling
- Has backpointer to referring dev_queue
- Therefore QDISC root is always qdisc->dev_queue->qdisc
- qdisc->q.lock on root is now used for tree synchronization
- qdisc->list of root qdisc replaces netdev->qdisc_list
RCU

- Bulk of qdisc_destroy() work is now done in RCU handler
- This works because visibility of QDISC tree is gone exactly when RCU of qdisc_destroy() on parent is run
- Massive simplification of QDISC teardown
- Some global state had to be cleaned up, example: u32_list
<table>
<thead>
<tr>
<th>Background</th>
<th>Existing Design</th>
<th>New Design</th>
<th>Implementation Notes</th>
</tr>
</thead>
</table>

**Builtin Qdiscs**

- Must now mostly behave exactly like dynamically allocated ones
- `qdisc_root()` must always work on any qdisc
- Likewise taking `qdisc_lock()` on any qdisc must also work
- Result: `noop_qdisc` and `noqueue_qdisc` given dummy `netdev_queue`