

# *Linux TX Multiqueue Implementation*

David S. Miller

Red Hat Inc.

Seattle, 2008

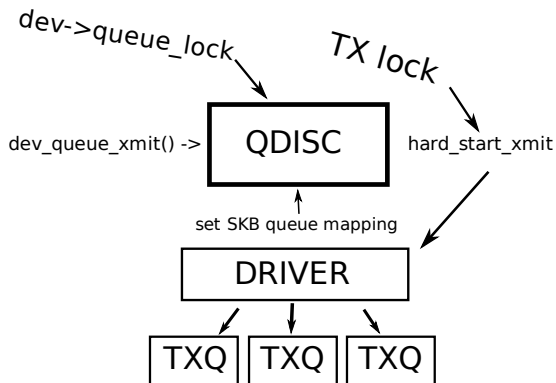
# IMPETUS

- 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)

# PRESENTATIONS

- Tokyo 2008
- Berlin 2008
- Implementation plan changing constantly
- End result was different than all of these designs

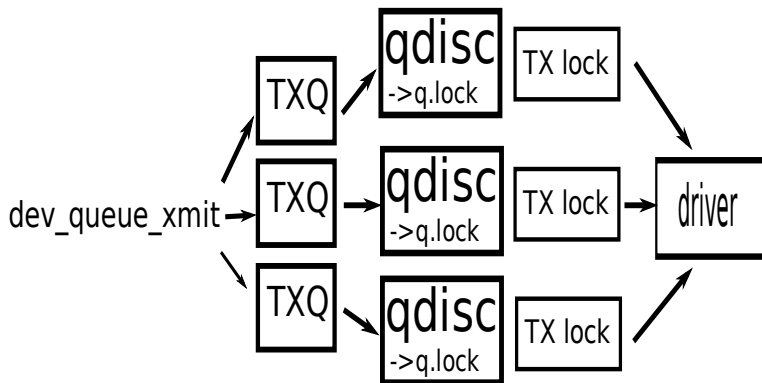
# PICTURE OF TX ENGINE



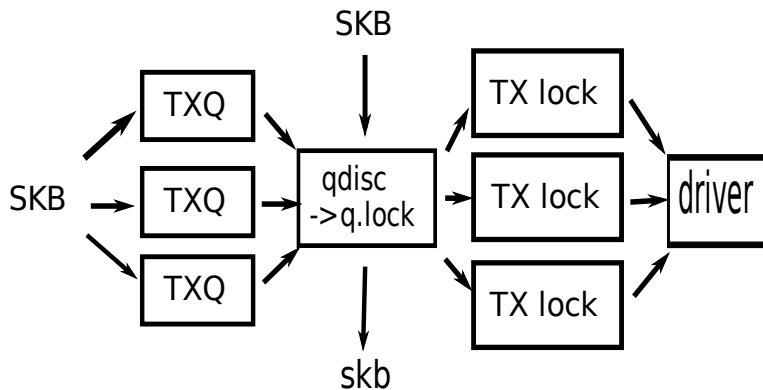
## DETAILS TO NOTE

- 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 OF DEFAULT CONFIGURATION



# PICTURE WITH NON-TRIVIAL QDISC



# 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`

# 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`