

# HEADS UP: network stack and socket hackery over the next few weeks

---

*Source:* <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/current/2006-03/msg00504.html>

---

- *From:* Robert Watson <[rwatson@xxxxxxxxxxxxx](mailto:rwatson@xxxxxxxxxxxxx)>
  - *Date:* Fri, 17 Mar 2006 14:25:13 +0000 (GMT)
- 

Over the next few weeks, I'll be doing a fairly serious workworking of the socket and protocol reference models, in order to clean up a number of long-standing race conditions and provide infrastructure for significant locking optimizations for several protocols (including TCP). This is high risk work, in that this part of the socket code is very complex and there are a great many subtleties. Part of the goal of the work is to eliminate some of this complexity, and make the subtle a bit more obvious (and documented), so I think it's all for the good in the long term. However, it will likely introduce significant instability in the short term, especially in the TCP code where there will be substantial changes in the memory management model.

I've started merging minor parts of the patch over the last few days, but things will get serious around April 1 when the deadline for maintenance on the netatm stack expires (see arch@ and net@ posts about this), allowing me to bring in changes that are not known to work with netatm. As such, be warned that things may get a bit messy!

Just as a high level outline of changes in the pipeline:

- Increase the strength of invariants relating to so\_pcb pointers in protocols, including generally guaranteeing that as long as the socket hasn't been detached or aborted, those pointers will be non-NULL.
- Eliminate sotryfree(), leaving just sofree().
- Eliminate use of so\_pcb checking in the socket layer as an implicit reference model for protocols that need to hold onto sockets. For now, replace it with SS\_PROTOREF, but in the future, possibly just so\_count. Generally normalize the socket reference count model.
- Eliminate the need to hold locks in order to follow so\_pcb pointers throughout the protocol code, allowing us to avoid acquiring the pcbinfo lock in many important protocol entry paths from the socket code in TCP. Especially, the send and receive paths.
- This requires significant reworking of the memory management model of TCP so that it doesn't spontaneously discard PCB state all over the place. This is a high risk change, but with significant payoffs.
- pru\_abort and pru\_detach will no longer be allowed to fail.

## HEADS UP: network stack and socket hackery over the next few weeks

Things facilitated by these changes:

- Eliminate a number of known race conditions.
- Edge towards eliminating ACCEPT\_LOCK().
- Avoid acquiring global TCP locks in common protocol paths from the socket layer, reducing tcbinfo lock contention.
- Eliminate many excessive and sometimes gratuitous checks relating to so\_pcb, avoiding a lot of complicated and now unnecessary error-handling in protocols.
- Pave the way for adding true references to TCP PCB's in the in-bound TCP path, further reducing contention on tcbinfo.

All good stuff, but requires the ground-work to be laid first. For those with p4 access, you can track the current work branch in `rwatson_sockref`.

Oh, and despite my best efforts, and testing by a number of developers, it's likely TCP will become somewhat broken during this work. Be warned!

Robert N M Watson

---

freebsd-current@xxxxxxxxxxx mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-current>

To unsubscribe, send any mail to "freebsd-current-unsubscribe@xxxxxxxxxxx"