

## Re: duplicate read/write locks in net/pfil.c and netinet/ip\_fw2.c

**Source:** <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2005-08/0155.html>

---

**From:** Luigi Rizzo ([rizzo\\_at\\_icir.org](mailto:rizzo_at_icir.org))

**Date:** 08/18/05

Date: Thu, 18 Aug 2005 00:57:39 -0700

To: Max Laier <[max@love2party.net](mailto:max@love2party.net)>

On Thu, Aug 18, 2005 at 03:32:19AM +0200, Max Laier wrote:

> *On Thursday 18 August 2005 02:02, Luigi Rizzo wrote:*

...

> > *could you guys look at the following code and see if it makes sense,*

> > *or tell me where i am wrong ?*

> >

> > *It should solve the starvation and blocking trylock problems,*

> > *because the active reader does not hold the mutex in the critical*

^^^^^^

i meant 'writer', sorry... as max said even in the current implementation the reader does not hold the lock.

> > *int*

> > *RLOCK(struct rwlock \*rwl, int try)*

> > *{*

> > *if (!try)*

> > *mtx\_lock(&rwl->m);*

> > *else if (!mtx\_trylock(&rwl->m))*

> > *return EBUSY;*

> > *if (rwl->writers == 0) /\* no writer, pass \*/*

> > *rwl->readers++;*

> > *else {*

> > *rwl->br++;*

> > *cv\_wait(&rwl->qr, &rwl->m);*

> ^^^^^^^

>

> *That we can't do. That's exactly the thing the existing sx(9) implementation*

> *does and where it breaks. The problem is that cv\_wait() is an implicit sleep*

> *which breaks when we try to RLOCK() with other mutex already acquired.*

but that is not a solvable problem given that the \*LOCK may be blocking.

And the cv\_wait is not an unconditioned sleep, it is one where you release the lock right before and wait for an event to wake you up.

In fact i don't understand why you consider spinning and sleeping

freebsd-arch: Re: duplicate read/write locks in net/pfil.c and netinet/ip\_fw2.c

on a mutex two different things.

- > *Moreover will this break for recursive reads e.g.:*
- >
- > *Thread 1: RLOCK() ... RLOCK() -> cv\_wait ...*
- > *Thread 2: WLOCK() -> cv\_wait ...*
- >
- > *This is exactly what pfil\_hooks must be able to do as the packet filter may*
- > *want to call back to the stack in order to send rejects etc.*

that's another story (also in issue in ipfw) and the way it is addressed elsewhere is by releasing and re acquiring the lock. In fact is the topic that started this thread.

- > *change). The problem is, that we still have to invest 4 mutex operations for*
- > *every access. The current implementation has the same basic problem (though*
- > *it only uses 2 mutex operations for the WLOCK/UNLOCK). Ideally the RLOCK/*
- > *UNLOCK should be free unless there is a writer waiting for the lock. In*

"free unless" means not free – you always have to check, be it through an atomic cmpswap or something else. But this is what mtx\_lock does anyways in the fast path.

cheers  
luigi

---

freebsd-arch@freebsd.org mailing list

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

To unsubscribe, send any mail to "freebsd-arch-unsubscribe@freebsd.org"