

Re: Locking netatm

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2006-05/msg00096.html>

- *From:* gnn@xxxxxxxxxxx
 - *Date:* Mon, 29 May 2006 13:52:48 +0900
-

At Sun, 28 May 2006 19:00:59 -0400,
Skip Ford wrote:

In my attempt to make netatm MPSAFE, I've run into just a couple situations I'm not sure how to handle. I've more-or-less replaced the splnet() calls with a single subsystem lock. Nearly all of them were protecting structures within the netatm subsystem. But I'm not sure how to handle the existing splimp() calls.

Hi Skip,

I'm not familiar with netatm but I am somewhat familiar with our locking code, and I hope that others will also join in.

There seem to be a few different reasons why splimp() calls were made. First, for timing. Three netatm functions use splimp() to protect access to the list of atm_timeq structures (struct atm_time *). Here is the usage from one of those three functions. I've removed much of the code to simplify what it does:

```
s = splimp();
```

```
FOREACH(atm_timeq);
```

```
...
```

```
... possibly modify structures within atm_timeq ...
```

```
... modify the struct atm_time * passed to this function ...
```

```
(void) splx(s);
```

So my question is, were network interrupts disabled when mucking with the atm_timeq list because a generated interrupt can modify structures within the list? This use is probably very netatm-specific. I'm still studying the timeout code to understand what it's doing.

Re: Locking netatm

The spl() calls haven't disabled real interrupts, as far as I know, for quite a while. They acted as general code locks to prevent simultaneous access to data structures while an update was in progress. In terms of the timeq, the locks were acting as a mutex now would to protect the list during an update.

A second situation where network interrupts were disabled was for netatm memory allocation for devices:

```
in atm_dev_alloc()
```

```
s = splimp();
```

```
FOREACH(atm_mem_head)
```

```
...
```

```
malloc (...)
```

```
(void) splx(s);
```

```
and in atm_dev_free()
```

```
s = splimp();
```

```
FOREACH(atm_mem_head)
```

```
...
```

```
free (...);
```

```
(void) splx(s);
```

I'm not sure how these should be protected. Presumably, we don't want to receive interrupts until the netatm memory for the device is allocated. Would a global subsystem lock protect these calls? I can protect atm_mem_head, so maybe that'd be enough?

I would assume, again, that these are protections of the list, and not the memory allocation routines. A mutex protecting the list, like the one that protects, say, the UDP protocol list is probably what you want.

Another use is to protect calls to other subsystems. For example:

```
within atm_nif_attach(struct atm_nif *nif)
```

```
ifp = nif->nif_ifp;
```

```
s = splimp();
```

Re: Locking netatm

```
if_attach(ifp);
bpfattach(ifp, DLT_ATM_CLIP, T_ATM_LLC_MAX_LEN);

(void) splx(s);
}

and within atm_nif_detach(struct atm_nif *nif)

ifp = nif->nif_ifp;

s = splimp();

bpfdetach(ifp);
if_detach(ifp);
if_free(ifp);

(void) splx(s);
```

Holding a new netatm subsystem lock won't protect those calls so I'm not sure how to handle those. Other non-netatm code in the tree seems to not do any locking at all around those calls.

I believe these are now unnecessary since the ifnet lists now have their own locks.

These are really the only uses I've yet to convert so if someone can provide some pointers, I'd appreciate it. I'm pretty new to FreeBSD locking, either the old way or the new way. I'm still studying the code, including other network stacks and the netatm stack itself, but a pointer or two would be appreciated. I feel like it's mostly converted, though I've done no testing at all yet. Once I finish removing splimp(), I can test with the single subsystem lock, then move on to finer-grained locking where necessary.

I would look at the UDP and TCP protocol list locking as being somewhat similar to what you have here. UDP is the easiest to understand as it's mostly in one file, netinet/udp_usrreq.c.

Later,
George

freebsd-arch@xxxxxxxxxxxxx mailing list

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

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