

Re: sosend/soreceive consistency improvements

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2006-07/msg00087.html>

- *From:* Brooks Davis <brooks@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 24 Jul 2006 12:32:05 -0500
-

On Sun, Jul 23, 2006 at 07:57:56PM +0100, Robert Watson wrote:

As part of cleanups, locking, and optimization work, I've been looking at the socket send and receive paths.

In the past, work was done do allow the uio/mbuf chain send and receive paths (sosend, soreceive) to be pluggable for a protocol, so that the protocol could provide substitute implementations. This is not, generally, currently used, although I recently changed UDP to use an optimized datagram send routine. This pluggability is made possible by virtue of each protocol providing its own pru_sosend() and pru_soreceive() methods in the protocol switch.

There's another side to the pluggability, however -- the socket consumers in the kernel, of which there are quite a few -- obviously the socket system calls, but also netgraph, distributed file systems, etc. Some of these consumers have been modified to call so->so_proto->pr_usrreqs->pru_soreceive and ...->pru_sosend, but it turns out many haven't. New references to sosend() and soreceive() periodically get encoded into consumers -- presumably because they are easy to spell, and in fact are generally functionally identical. But not always! It turns out that the NFS code isn't using the optimized UDP send path via sosend_dgram(), because it's calling sosend() directly.

Rather than continue in this "in between state", in which the uio/mbuf chain sosend and soreceive are reached via the protocol switch in each occurrence, I propose a change: sosend() and soreceive() will now be the formal APIs for sending and receiveing on sockets within the kernel, as is the case with many other so*() functions, and they will perform the protocol switch dereference. The existing functions are renamed to sosend_generic() and soreceive_generic(), and in most cases are never referenced by protocols since our protocol domain registration already uses sosend() and soreceive() as the defaults today. The new code strikes me as quite a bit more readable, and likely easier for socket consumers to use.

Any thoughts and/or objections?

Re: sosend/soreceive consistency improvements

Makes sense to me. Is there an measurable performance impact? I wouldn't really expect much if any, but it's probably worth a check. The function is a fairly obvious target for inlining if there is any.

-- Brooks

Attachment: [pgpGxVts3008u.pgp](#)

Description: PGP signature