FreeBSD Interface API (IfAPI)

BY JUSTIN HIBBITS

As some may be aware, Juniper uses its own custom network stack with FreeBSD—forked long ago—so it only superficially resembles the current FreeBSD network stack. There is state in the current FreeBSD stack that doesn't exist in Junos, and vise-versa.

The Why and How

Junos is big, very big. Updating FreeBSD is a monumental task. To make it easier, Juniper split the FreeBSD component out into its own separate repository, leaving Juniper enhancements in its separate repository. This causes a dilemma—how can we keep drivers in FreeBSD, but the netstack elsewhere? As part of the FreeBSD split project, the original DrvAPI was born. With this, drivers could exist in the FreeBSD repository, while keeping the Junos netstack separate.

But what is the netstack? Where do we draw the line between netstack and the rest? The initial approach was "all directories in sys/ prefixed **`net`**" which works well. However, recently the **`netlink`** component was added, which really isn't part of the network stack conceptually, so that was crossed out. Now the netstack consists of **`net`**, **`net80211`**, **`netgraph`**, **`netinet`**, **`netinet6`**, and **`netpfil`**. Keeping the details to only the network stack also hides the details from the core kernel. Some changes were needed for other parts of the kernel to accommodate the IfAPI, including NFS rootless (boot) and mbuf handling.

Juniper split the FreeBSD component out into its own separate repository.

Design

The current design of the IfAPI is simply accessors and iterators. This was chosen as the most expedient way to convert drivers and hide the **`struct ifnet`**, though it is far from the best way. Conversion was largely mechanical, and Juniper provided a shell script to handle the bulk of the conversion in **`tools/ifnet/convert_ifapi.sh`**. Obviously, this may miss some conversions, such as those where **`ifp`** is a member of another structure or is named something else like **`foo_ifp`**, but it does handle most cases.

As for iterators, the initial implementation was based upon Gleb Smirnoff's **`if_foreach_lladdr()`**, using callbacks when iterating over a given type. This was applied to both if_addr and if_t, where iterating over the interfaces only iterates over the current VNET. More re-

cently, a new iterator API was added, allowing iterating with a more traditional loop; the requirement being you must call `if_iter_finish()` or `ifa_iter_finish()` to clean up any state that set up for the iteration, including freeing any memory the implementation may have allocated (not done with the FreeBSD network stack, but could be done by other stacks).

Benefits

Decoupling device drivers from the network stack details brings benefits beyond Juniper's source code management. With a stable ABI, a single device driver can be used with multiple different network stacks. For instance, one image for all computers in a data center could select–at boot time—a different network stack based on the execution profile; a high-performance limited stack for some devices and a full stack for others, all using the same network drivers.

Another, smaller benefit is that driver changes and netstack changes can occur simultaneously without affecting one another. Before the IfAPI, any changes made to the `struct ifnet` required rebuilding all device drivers. Going forward, with the `struct ifnet` being completely private, any change to the structure requires rebuilding only the files that directly reference it, resulting in a shorter debug cycle.

With a stable ABI, a single device driver can be used with multiple different network stacks.

Where Next?

IfAPI is just step 1, there is still more to do to properly abstract away the network stack. Gleb Smirnoff proposed using KOBJ interfaces to allow a more pluggable netstack and fully decouple the netstack from the rest. This would even allow replacing the netstack at runtime (`kldunload`/`kldload`). Taking this further, we could potentially allow multiple netstacks, assigning different devices to different stacks. With that, there could even be the possibility of moving interfaces between netstacks dynamically.

Conclusion

The IfAPI is only phase one in an effort to decouple network drivers from the inner workings of the network stack. With further work, multiple network stacks could be in use—and even reloadable network stacks.

JUSTIN HIBBITS was foolishly entrusted with a FreeBSD commit bit in 2011 for his obsession with PowerPC. Since then, he's focused mostly on PowerPC and other embedded architectures. He currently works for Juniper Networks, working on all things FreeBSD kernel related, and continues his passion for low-level development and exotic architectures.