BATMAN and the LinuxKPI: Running Linux drivers on FreeBSD 🦇

Aymeric Wibo
A little bit about myself

- CS student at UCLouvain, in Belgium 🇧🇪
- Part-time at Bnewable (energy storage solutions startup) ⚡
- Interested in computer graphics (and BSD).
- **GSoC 2023 student.**
Dog (Bubbles)
What is the focus of this talk?

- The LinuxKPI & the state of affairs w.r.t. porting Linux drivers to FreeBSD.
- Will end with a case study on porting BATMAN to FreeBSD (my GSoC project).
What is the LinuxKPI?

- Kinda just a bunch of C headers which map Linux kernel functions (the KPI) to FreeBSD ones.
- **Internal kernel functions**, not syscalls, that's (part of) the Linuxulator's job.
- Headers rooted in `sys/compat/linuxkpi/common/include`.
**Example function:** `get_random_u32_below`

In `sys/compat/linuxkpi/common/include/linux/random.h`:

```c
static inline u32
get_random_u32_below(u32 ceil)
{
    return (arc4random_uniform(ceil));
}
```

`get_random_u32_below` is the Linux function, `arc4random_uniform` is FreeBSD's equivalent.
Let's port a simple Linux driver!!

Original Linux code:

```c
#include <linux/module.h>

static int __init demo_init(void) {
    pr_info("Hello from this demo module!\n");
    return 0;
}

module_init(demo_init);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Aymeric Wibo");
MODULE_DESCRIPTION("Super simple linux driver");
MODULE_VERSION("0.0.0");
```
Let's port a simple Linux driver!

```c
#include <linux/kernel.h>
#include <linux/module.h>

static int __init demo_init(void) {
    pr_info("Hello from this demo module!\n");
    return 0;
}

module_init(demo_init);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Aymeric Wibo");
MODULE_DESCRIPTION("Super simple linux driver");

#if defined(__linux__)
    MODULE_VERSION("0.0.0");
#endif

#if defined(__FreeBSD__)
    MODULE_VERSION(demo, 0);
    MODULE_DEPEND(demo, linuxkpi, 1, 1, 1);
#endif
```
.PATH: .

KMOD= demo
SRCS= demo.c

SRCS+= ${LINUXKPI_GENSRCs}
CFLAGS+= ${LINUXKPI_INCLUDES}

.include <bsd.kmod.mk>
$ kldload ./demo.ko
$ dmesg | tail -n 1
Hello from this demo module!

It works!
What can LinuxKPI do for us today?

- More than print hello world!
- Full network drivers (e.g. `iwlwifi(4)`).
- Full graphics drivers (e.g. `graphics/drm-kmod` with `i915`, `radeon`, `amd`).
🌟 Adding new stuff
Update to the demo module

```c
#include <linux/kernel.h>

static int __init demo_init(void) {
    hypothetical_function("Hello from this demo module!\n");
    return 0;
}
```
Doesn't build!

demo.c:4:2: error: call to undeclared function 'hypothetical_function'; ISO C99 and later do not support implicit function declarations [-Wimplicit-function-declaration]
   4 |     hypothetical_function("Hello from this demo module!\n"");
     |     ^
1 error generated.
*** Error code 1
**Add stub**

In `linux/kernel.h`:

```c
static inline void hypothetical_function(const char *str)
{
    pr_debug("TODO: %s\n", __func__);
}
```
What does this do on Linux?

$ modprobe ./demo.ko
$ dmesg | tail -n 1
!eludom omed siht morf olleH
What does this do on FreeBSD?

$ kldload ./demo.ko
$ dmesg | tail -n 1
todo: hypothetical_function
Does this matter?
Yes? Implement observed behaviour 🧐

```c
static inline void hypothetical_function(const char *str)
{
    ssize_t const len = strlen(str);
    for (ssize_t i = len - 2; i >= 0; i++)
        pr_info("%c", str[i]);
    pr_info("\n");
}

⚠️ Can't be isomorphic to GPL code!
```
What about now?

```bash
$ kldload ./demo.ko
$ dmesg | tail -n 1
!eludom omed siht morf olleH
```

It works!
Questions related to LinuxKPI?
BATMAN: A case study
A little bit of background...
What's a BATMAN?

- Very annoying to Google.
- Better Approach to Mobile Ad-hoc Networking.
- Essentially OSPF/OLSR, but optimized for big wireless meshes (i.e. constantly changing topology/link quality, &c).
What is it used for?

- Driving force is Freifunk.
- Community wireless mesh networks.
Quick history of BATMAN

- Freifunk were feeling limited by OLSR – topology changes are expensive!
- Initially, userspace `batmand`.
- Then, `batman-adv` kernel module (BATMAN IV & V).
- BMX6/7 → offshoot.
802.11? New Internet?

Traces?

\[ \sum_{k \in F} \left( \frac{r_k}{r_m} \right) + |F| \]

\[ \sum_{k \in F} \left( \frac{r_k}{r_m} \right) \]

WCET?

\[ W_{\text{WCET}} = \text{Constant} \]

Theoretical bandwidth:

\[ B_T = \min \left( B(\text{chain}) \cdot \text{hop} \right) \]

AUV hops = 6, Distance = 5, \( B_T = 2 \text{ Mbps} \)

\[ W_{\text{trans}} \]

\[ W_{\text{mod}} \]
Kitti's hog-nosed bat
How does BATMAN work?

- Only cares about local changes in topology.
- Layer 2 routing protocol → everything above gets wrapped.
How does BATMAN work?

Example Tee Cee Pee packet:

Frame 1077: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface bridge1, id 0  
  Type: Unknown (0x4305)  
  Packet Type: BATADV_UNICAST (64)  
  Version: 15  
  Time to Live: 50  
  TT Version: 2  
  Destination: NetApp_91:93:b7 (00:a0:98:91:93:b7)  
  Type: IPv4 (0x0800)  
Internet Protocol Version 4, Src: 10.0.0.3, Dst: 10.0.0.1  
How does BATMAN work?

- Provided by kernel module `batman-adv` on Linux (`batman_adv` on FreeBSD).
- So-called "hard" vs "soft" interfaces (e.g. `vtnet0` vs `batadv0`).
- Sends out "echolocation" packets (ELP) to discover neighbours (new in BATMAN V) (hard).
- Sends out OGM(2) packets to discover best routes (soft).
How to set up a BATMAN network on FreeBSD?

```
$ kldload batman_adv
$ ifconfig vtnet0 mtu 1532 up
$ ifconfig batadv create ra BATMAN_V
   batadv0
$ ifconfig vtnet0 master batadv0
$ ifconfig batadv0 inet 10.0.0.1/24
```

MTU of 1532 to give room to BATMAN header.
How to set up a BATMAN network through the Linuxulator?

Exactly the same way as on Linux!

```bash
$ ip link set up mtu 1532 dev eth0
$ ip link add name bat0 type batadv
$ ip link set dev eth0 master bat0
$ ip link set up dev bat0
$ ip addr add 10.0.0.1/24 dev bat0
```
Quick demo of a BATMAN 🚀
What are the major things that had to be done to get batman-adv ported?
What had to be done to get `batman-adv` ported:

`ifconfig /Netlink`

- `batadv` cloner with `ra` parameter (only on Netlink, no `ioctl`).
- No modification of other settings (yet, need Generic Netlink) – only defaults.
- Add concept of setting the "master" interface to FreeBSD and FreeBSD's Netlink.
- Thanks `melifaro@`!
What had to be done to get **batman-adv** ported: **Linuxulator**

- Just the conversion from **batX** (Linux) to **batadvX** (FreeBSD) → `ifname_bsd_to_linux_ifp`. 
What had to be done to get batman-adv ported:

Changes to batman-adv

• Goal: keep this source code as similar as possible to Linux!
• Not always possible though 😕
• Guard namespace collisions, e.g.:

```c
#if defined(__FreeBSD__)
    soft_iface = linux_dev_get_by_index(net, ifindex);
#else
    soft_iface = dev_get_by_index(net, ifindex);
#endif
```
What had to be done to get **batman-adv** ported:

Changes to **batman-adv**

- Biggest modification to **batman-adv** needed by far is to add the cloner for the soft interface:

```c
static struct if_clone_addrreq_v2 batadv_ifc_addrreq = {
    .version = 2, /* For netlink callbacks. */
    .flags = IFC_F_AUTOUNIT,
    .match_f = batadv_softif_ifc_match,
    .create_f = batadv_softif_ifc_create,
    .destroy_f = batadv_softif_ifc_destroy,
    .create_nl_f = batadv_softif_ifc_create_nl,
    .modify_nl_f = batadv_softif_ifc_modify_nl,
    .dump_nl_f = batadv_softif_ifc_dump_nl,
};
```
What had to be done to get batman-adv ported:

Changes to batman-adv

- On Linux, this is `struct rtnl_link_ops batadv_link_ops`, but would require deep changes to/integration with FreeBSD network code.
- Sometimes supporting something with the LinuxKPI is not a worthwhile tradeoff!
What had to be done to get **batman-adv** ported:

**LinuxKPI** *(struct net_device)*

- Linux's equivalent to *struct ifnet*.
- Made it the same as *struct ifnet*.
- Fields in common aliased, other Linux stuff bolted on at the end.
- Means *struct net_device* can be passed around as *struct ifnet* (and vice-versa).
struct net_device *
linuxkpi_alloc_netdev_ifp(size_t priv_len, u_char type,
    void(*setup_func)(struct net_device *))
{
    struct net_device *ndev;
    if_t ifp;

    ndev = malloc(sizeof(*ndev) + priv_len, M_NETDEV, M_NOWAIT);
    if (ndev == NULL)
        return (ndev);
    ifp = (if_t)ndev;

    linuxkpi_init_dummy_netdev(ndev);
    if_fill_domain(ifp, type, IF_NODOM);
    ndev->has_ifp = true;

    memset(ndev->drv_priv, 0, priv_len);
    setup_func(ndev);

    return (ndev);
}
What had to be done to get `batman-adv` ported:

**LinuxKPI** (`struct net_device`)

- `linuxkpi_dev_queue_xmit` (equivalent to `ifp->if_output`).
- `linuxkpi_netif_rx` (equivalent to `ifp->if_input`).
- The `struct net_device` is just cast to `struct ifnet` to get the `ifp`. 
What had to be done to get `batman-adv` ported:

**LinuxKPI** (`struct sk_buff`)

- Linux's equivalent to `mbuf`.
- Way more complex and has a ton of random functions (`skb_*`).
- Not currently backed by `mbuf` – data is being copied back and forth (don't sue me 😐).
- Big function is `linuxkpi_skb_from_mbuf`, the other way is easier.
Many more things!

That you'll have the pleasure of discovering though my upcoming reviews :)
The future?

- `batman_adv` in ports!
- Wi-Fi support!! (Only `sys/net/if_ETHERSUBR.c` interfaces at the moment).
- Backing `sk_buff`'s with `mbuf`'s.
- Upstreaming LinuxKPI changes.
- `batctl`?
- Demo mesh.
Call to action

- This was actually not too bad, even though I was intimidated by the LinuxKPI at first.
- If you have a Linux driver you want to port, it could be worth trying!
- Important for FreeBSD's continued relevancy in certain domains.
That's all folks 🐰

Questions?
Reach out to me 💌

- Have a beer! 🍺
- FreeBSD email: obiwac@freebsd.org
- Website: https://obiw.ac
- GitHub: https://github.com/obiwac
- Discord: @obiwac