Mathieu Devos <mathieu.dvs@gmail.com>

Skb and ieee80211 headers

12 messages

Mathieu Devos <mathieu.devos@ugent.be> Wed, Jul 31, 2013 at 11:39 AM
To: linux-wireless@vger.kernel.org

Hi,

I hope this is the right place to ask for a little bit of help as I'm currently beyond stuck on a challenge I'm trying to accomplish. I'm trying to write a "simple" LKM that properly uses a ieee80211 header to print information about the mac addresses (addr1->addr4) and later down the road try to send my own data.

I only need to get L2 working, no need for TCP/IP, just a proper ieee80211 based on input from skb would be huge for me.

So my issue: when placing the ieee80211 on my mac_header after I hook my skb from my wireless device (wlan0 on android - I9100) I get a huge amount of zero's and random(?) numbers when trying to print the addresses. This leads me to the first conclusion that mac_header is placed wrong when using 80211. After that I saw a lot of people just using the skb->data pointer. Now this gives even weirder issues for me and actually totally crashes my kernel.
So I went back to starting with printing as much info as possible.
This is a sample output after I hook my packet type:

Skb->dev->name: wlan0

Skb->head: 0xe1d37040

Skb->mac_header: 0xe1d372a9

Skb->data: 0x510 (!!!)

Skb->tail: 0xe1d37460

Skb->len: 617

Skb->hdr_len: 0

When trying to just capture this and only print a certain message when one of the addresses maches my dev->dev_addr I never get any data while the phone is connected and actively browsing the internet.

I'm aware that before I throw my hook some data is being changed around already in net/core/dev.c and in net/mac80211/rx.c The weird part is that these seem to be putting on ethernet headers (skb->protocol = eth_type_trans(skb, dev); AND kb_pull_inline(skb, ETH_HLEN);
eth = eth_hdr(skb);) on items that should be ieee80211 headers.

Any insights as to why my data header is in such a weird spot (nowhere between my head and my tail) or where I should call the

ieee80211_header on? I have tried working my way back from tail with len and adding another ETH_HLEN but while I get data, it never really matches my own mac addr so I'm assuming the data is still pretty wrong.

Added links:

https://github.com/mathieudevos/kernelmodules/blob/master/ethernet_test.c (my own program)
https://github.com/mathieudevos/linux_kernel_3.2.48 (used to get all the .c files from to acquire information)

If possible I'd like to write a small guide after these issues have been fixes for people who like me would like to get started with a basic LKM in the ieee80211 part of linux.

If this is not the place to ask these questions, please disregard me (hopefully it is) but all help would be welcome.

Kind regards,
Mathieu Devos

Arend van Spriel <arend@broadcom.com> Wed, Jul 31, 2013 at 12:08 PM
To: Mathieu Devos <mathieu.devos@ugent.be>
Cc: linux-wireless@vger.kernel.org

> On 07/31/2013 11:39 AM, Mathieu Devos wrote:
> > Hi,
> >
> > I hope this is the right place to ask for a little bit of help as I'm currently beyond stuck on a challenge I'm trying to accomplish. I'm trying to write a "simple" LKM that properly uses a ieee80211 header to print information about the mac addresses (addr1->addr4) and later down the road try to send my own data.
> >
> > I only need to get L2 working, no need for TCP/IP, just a proper ieee80211 based on input from skb would be huge for me.
> >
> > So my issue: when placing the ieee80211 on my mac_header after I hook my skb from my wireless device (wlan0 on android - I9100)

Not sure what you goal is, but what wireless device is that? You may just get 802.3 packets from the device.

Gr. AvS
[Quoted text hidden]

Mathieu Devos <mathieu.devos@ugent.be> Wed, Jul 31, 2013 at 12:28 PM
To: Arend van Spriel <arend@broadcom.com>
Cc: linux-wireless@vger.kernel.org

Hi,

It's an android smartphone (I 9100 - Samsung galaxy S2) so it does not have a normal ethernet 802.3 input even. I check before selecting the device that it's wireless (through ieee80211_ptr) and this properly returns the wlan0 device which should be on the 80211 standard.

My goal is to get the ieee80211_header properly on the skb with this

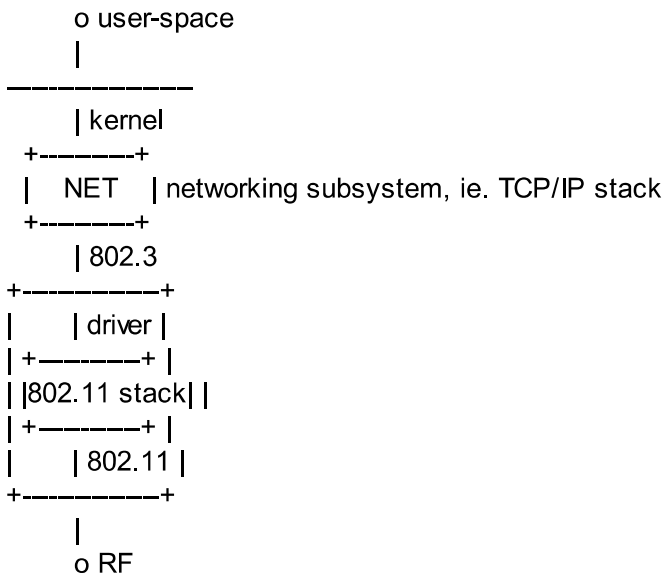
device, but some of the pointers and original data in the skb seem totally off. This leaves me clueless as to where to put this ieee80211_header.
I've tried putting it right on skb->head (wrong I know, but I was getting desperate), on skb->mac_header (also wrong, no idea why though), I went back from skb->tail with len and even added ETH_HLEN to that as well because you can see that before my hook gets activated: skb_pull_inline(skb, ETH_HLEN);
In the end I'm left with a header that is forced onto data but with a wrong origin pointer thus basically leaving me with all wrong data in the header.

Kind regards,
Mathieu Devos
[Quoted text hidden]

Arend van Spriel <arend@broadcom.com> Wed, Jul 31, 2013 at 1:05 PM
To: Mathieu Devos <mathieu.devos@ugent.be>
Cc: linux-wireless@vger.kernel.org

> On 07/31/2013 12:28 PM, Mathieu Devos wrote:
> > Hi,
> >
> > It's an android smartphone (I 9100 - Samsung galaxy S2) so it does not have a normal ethernet 802.3 input even. I check before selecting the device that it's wireless (through ieee80211_ptr) and this properly returns the wlan0 device which should be on the 80211 standard.

sigh Welcome in the world of protocol stacks, wireless, networking (choose your poison). Let me draw the picture.



The device hooks up to the networking subsystem as an ethernet device and as such it receives 802.3 packets. These are converted to 802.11 packets by the 802.11 stack. Now depending on your device that happens in the device driver or on the device itself. Another option is that this is done by mac80211 (kernel provided 802.11 stack), but that is probably not the case, but to be sure I ask again: what wireless device do you have in your galaxy S2?

Gr. AvS
[Quoted text hidden]

Mathieu Devos <mathieu.devos@ugent.be> Wed, Jul 31, 2013 at 2:39 PM
To: Arend van Spriel <arend@broadcom.com>

Cc: linux-wireless@vger.kernel.org

Hi,

The wireless chip is a Broadcast BCM4330 chip. After looking around a bit I found that this is a fullMAC and links to the driver on wireless.kernel: http://wireless.kernel.org/en/users/Drivers/brcm80211 and also links directly to android itself: https://android.googlesource.com/platform/hardware/broadcom/wlan

Still trying to learn a lot in this tightly packed world of protocol stacks, wireless and all the other poisons. Seems like I still have a long way to go. Thank you already for helping me out with these issues and taking the time to explain these things to me.

Kind regards,
Mathieu Devos
[Quoted text hidden]

Arend van Spriel <arend@broadcom.com> Wed, Jul 31, 2013 at 2:55 PM
To: Mathieu Devos <mathieu.devos@ugent.be>
Cc: linux-wireless@vger.kernel.org

> On 07/31/2013 02:39 PM, Mathieu Devos wrote:
> > Hi,
> >
> > The wireless chip is a Broadcast BCM4330 chip. After looking around a bit I found that this is a fullMAC and links to the driver on wireless.kernel: http://wireless.kernel.org/en/users/Drivers/brcm80211 and also links directly to android itself: https://android.googlesource.com/platform/hardware/broadcom/wlan

I suspected. The bcm4330 is indeed a fullmac device, which means the 802.11 stack is running on the device. The driver on Android is located under:

https://android.googlesource.com/kernel/samsung/+/android-samsung-3.0-ics-mr1/drivers/net/wireless/bcmdhd/

Not sure which android version you have.

The brcm80211 on wireless.kernel.org is the upstream linux driver.

Gr. AvS
[Quoted text hidden]

Mathieu Devos <mathieu.devos@ugent.be> Wed, Jul 31, 2013 at 3:45 PM
To: Arend van Spriel <arend@broadcom.com>
Cc: linux-wireless@vger.kernel.org

Alright,

Seems like I have fixed some issues while added some others. Since I assume that when my hook gets activated the data pointer should be at the start of the 802.3 header I casted an ethhdr (8023) on top of that and it seems that on my notebook this is handled correctly (I can actually check on my own mac addr and see that these frames are for/from me).
The issue with android: the data pointer (see previous mails) is totally off and causes a full blown kernel crash. Trying to set it manually leaves me with nothing really. Could more information about

skb->data manipulation be in those drivers and is there a reason why not everything is ended with the same data structure (thus standardizing the part where the hook comes in). My next issue is with mac_header that seems off just as much in both my notebook LKM and in my android device (when casting 8023 or 80211 headers on them they deliver totally wrong headers).
Somebody in the office told me that the 8023 header is added after the 80211 and thus to my logic when casting skb->data - sizeof(*wirelessheader) I should end at the start of the 80211 header. This however is wrong, are there more padding bytes present and if so, how much would I have to move the pointer or is do I have to use a whole other function for this.

A quick remark as to why I'm doing all this stuff: this is preparation for my master thesis where I'll be researching the possibility of RINA (instead of TCP/IP) in the wireless stack (android specifically). Since this is preparation work I'd love to end this with just having an sk_buff with correctly placed 802.3 header (works on my notebook, not on android device) and with a properly set up 802.11 header (so I can later on map functions from/to rina to 802.3/802.11).

Thanks again for the quick answers and the already provided help.

Kind regards,
Mathieu Devos
[Quoted text hidden]

Arend van Spriel <arend@broadcom.com> Wed, Jul 31, 2013 at 5:04 PM
To: Mathieu Devos <mathieu.devos@ugent.be>
Cc: linux-wireless@vger.kernel.org

> On 07/31/2013 03:45 PM, Mathieu Devos wrote:
> > Alright,
> >
> > Seems like I have fixed some issues while added some others. Since I assume that when my hook gets activated the data pointer should be at the start of the 802.3 header I casted an ethhdr (8023) on top of that and it seems that on my notebook this is handled correctly (I can actually check on my own mac addr and see that these frames are for/from me).

How does your hook work? Are you intercepting packets?

Regards,
Arend
[Quoted text hidden]

Mathieu Devos <mathieu.devos@ugent.be> Wed, Jul 31, 2013 at 5:11 PM
To: Arend van Spriel <arend@broadcom.com>
Cc: linux-wireless@vger.kernel.org

This is my hook: it gets called after I set dev to wlan0. I used a guide for this and it seems to work for my notebook where I'm able to find the 802.3 header but still no 802.11.

```
static int ptype_function(struct sk_buff *skb, struct net_device *dev,
struct packet_type *ptype, struct net_device *dev2);

static void throw_hook(struct net_device *dev){
  ptype.type = htons(ETH_P_ALL);
```

```
    ptype.func = &ptype_function;
    ptype.dev = dev;
    dev_add_pack(&ptype);
    printk(KERN_CRIT "Done setting up packet type");
}
```

All code can be found here:
android: https://github.com/mathieudevos/kernelmodules/blob/master/ethernet_test.c
notebook: https://github.com/mathieudevos/wifi_kernelmodules/blob/master/ethernet_test.c

Kind regards,
Mathieu Devos
[Quoted text hidden]

Mathieu Devos <mathieu.devos@ugent.be> Fri, Aug 2, 2013 at 9:49 AM
To: Arend van Spriel <arend@broadcom.com>
Cc: linux-wireless@vger.kernel.org

Hi,

An update is in order here I believe (and an apology because I resorted to asking my questions on IRC and not sticking to the mailing list, for that: sorry).
Let's start things off quickly with my main goal with this: the code that I'm trying to write is supposed to be an adapter for a new type of internet (RINA), this is in function of my master thesis (I'm new to Linux Kernel itself). So what exactly does this mean? I'd like to be able to hook packets (skb) in the kernel with an LKM and just use one layer (datalink layer) with the hardware layer being handled by the drivers and the hardware itself. Everything on top of that should be RINA specific code.
The specifics about the master thesis is that this all should be done on android, thus using the wireless stack. I believed that in an skbuff you had the formed 802.3 header through drivers but you still also had the original 802.11 header, this however, is now proven wrong, the 802.11 header is thrown out after the drivers are done with it thus it's impossible for me to map RINA to the 802.11 header. After resuming testing with this news I'm still left with a couple of questions however:

- Can I still use the added functionality of 802.11 as compared to 802.3 by talking directly to the device and setting it's fields? I'm thinking of using "struct wireless_device" and/or "struct wiphy" and the functions that come with it.

- Is it possible to find a general form of skb frame with correctly set headers? I have currently tested my code (https://github.com/mathieudevos/kernelmodules/blob/master/ethernet_test.c) and on one android device (galaxy S2 I9100, BCM4330 wireless chip) I had my 802.3 header by at my skb->head. On my own htc one X (htc endeavoru, wireless chip unknown still) I had my 802.3 header at my skb->mac_header and on my notebook (wireless 5100 AGN) this header was present on skb->data. Is it coincidence that these 3 are totally different and is there a function (that perhaps calls the drivers) to set these skb's in the correct prepared form? And if so, will this break when I void the layers on top of the mac layer.

While I know that a part of this still needs to be researched, for instance say it's possible to set certain fields of the device to use added 802.11 functionality I won't be setting the channel manually,

that's still left to the driver, but other added functions might be more useful for RINA.
What I'm hoping to achieve before really heading off into this new linux adventure is to be able to properly receive an SKB that has the same format on my devices and later down the road try to send set up an SKB and send that one myself.

I'd like to finish with another apology for not sticking to the mailing list, for that I'm truly sorry.

Kind regards,
Mathieu Devos
[Quoted text hidden]

Arend van Spriel <arend@broadcom.com> Wed, Aug 7, 2013 at 12:27 PM
To: Mathieu Devos <mathieu.devos@ugent.be>
Cc: linux-wireless@vger.kernel.org, "David S. Miller" <davem@davemloft.net>, Eric Dumazet <edumazet@google.com>

+ networking experts

On 08/02/2013 09:49 AM, Mathieu Devos wrote:

> Hi,
>
> An update is in order here I believe (and an apology because I resorted to asking my questions on IRC and not sticking to the mailing list, for that: sorry).
> Let's start things off quickly with my main goal with this: the code that I'm trying to write is supposed to be an adapter for a new type of internet (RINA), this is in function of my master thesis (I'm new to Linux Kernel itself). So what exactly does this mean? I'd like to be able to hook packets (skb) in the kernel with an LKM and just use one layer (datalink layer) with the hardware layer being handled by the drivers and the hardware itself. Everything on top of that should be RINA specific code.

So basically you want to be able to use RINA stack as a drop-in replacement for the TCP/IP stack keeping the netdev api unchanged so device drivers do not have to change.

> The specifics about the master thesis is that this all should be done on android, thus using the wireless stack.

I can not follow your line of thinking here. The android requirement (a stupid one if you ask me) does not restrict you to the wireless stack. There are android platforms with ethernet connectivity.

> I believed that in an skbuff you had the formed 802.3 header through drivers but you still also had the original 802.11 header, this however, is now proven wrong, the 802.11 header is thrown out after the drivers are done with it thus it's impossible for me to map RINA to the 802.11 header. After resuming testing with this news I'm still left with a couple of questions however:

An sk_buff is just a structure that makes it convenient to add or remove layer specific header (or tail) data moving from one layer to the other. An ethernet device with get an 802.3 packet, which is straightforward. A wireless device also gets an 802.3 packet so from the networking stacks' perspective there is no difference between a ethernet or a wireless device.

Looking at architecture shown in [1], I think RINA does not need to care about 802.11 either as the device

drivers have their own 802.11 stack, which takes care of converting 802.3 to 802.11 packets and vice versa (or use the kernel provided 802.11 stack, ie. mac80211 to take care of that).

> - Can I still use the added functionality of 802.11 as compared to 802.3 by talking directly to the device and setting it's fields? I'm thinking of using "struct wireless_device" and/or "struct wiphy" and the functions that come with it.

As said I do not think you should care about 802.11. The structures mentioned are data structures for configuration purposes on respectively wireless interface and wireless device level.

> - Is it possible to find a general form of skb frame with correctly set headers? I have currently tested my code (https://github.com/mathieudevos/kernelmodules/blob/master/ethernet_test.c) and on one android device (galaxy S2 I9100, BCM4330 wireless chip) I had my 802.3 header by at my skb->head. On my own htc one X (htc endeavoru, wireless chip unknown still) I had my 802.3 header at my skb->mac_header and on my notebook (wireless 5100 AGN) this header was present on skb->data. Is it coincidence that these 3 are totally different and is there a function (that perhaps calls the drivers) to set these skb's in the correct prepared form? And if so, will this break when I void the layers on top of the mac layer.

How generic do you want it to be? As said the sk_buff is a pretty dumb struct which is mostly protocol/stack independent. General advice here is to avoid dealing with the pointers directly and use skb function api.

The hook mechanism dev_add_pack() you are using in [2] is new to me, but it seems you are doing it while the packet is being processed in the network stack so the state of the sk_buff can be pretty unpredictable. Just remember you are probably not the only one handling this packet. The stack internals are something I tend to stay clear off. Maybe one of the networking experts can elaborate.

> While I know that a part of this still needs to be researched, for instance say it's possible to set certain fields of the device to use added 802.11 functionality I won't be setting the channel manually, that's still left to the driver, but other added functions might be more useful for RINA.

Examples of useful wireless specific functions? As I see it RINA is an IPC based network stack on top of some physical layer, ie. ethernet. The architectural picture in [1] even shows it can run on top of a TCP/IP stack using sockets.

> What I'm hoping to achieve before really heading off into this new linux adventure is to be able to properly receive an SKB that has the same format on my devices and later down the road try to send set up an SKB and send that one myself.

It seems to me that you should familiarize yourself with linux networking artifacts. Maybe [3] is a good read although it may be pretty outdated.

Regards,
Arend

[1] http://irati.eu/irati-first-phase-report-on-use-cases-requirements-analysis-updated-rina-specifications-and-high-level-software-architecture-available/
[2] https://github.com/mathieudevos/wifi_kernelmodules/blob/master/ethernet_test.c
[3] http://shop.oreilly.com/product/9780596002558.do

[Quoted text hidden]

Mathieu Devos <mathieu.devos@ugent.be> Wed, Aug 7, 2013 at 2:16 PM
Draft To: Arend van Spriel <arend@broadcom.com>

Cc: linux-wireless@vger.kernel.org, "David S. Miller" <davem@davemloft.net>, Eric Dumazet <edumazet@google.com>

Hi,

> So basically you want to be able to use RINA stack as a drop-in replacement
> for the TCP/IP stack keeping the netdev api unchanged so device drivers do
> not have to change.

Correct.

> I can not follow your line of thinking here. The android requirement (a
> stupid one if you ask me) does not restrict you to the wireless stack. There
> are android platforms with ethernet connectivity.

This is wrongfully stated by me, the implementation of the RINA library should be on the wireless stack and for this master thesis the testing was chosen to be done on the android kernel. Thus most likely going towards smaller devices so to see if rina is applicable on small devices with fewer resources than the average notebook. It's thus implementation first on wireless and then specifically on android and not the other way around.

[Quoted text hidden]