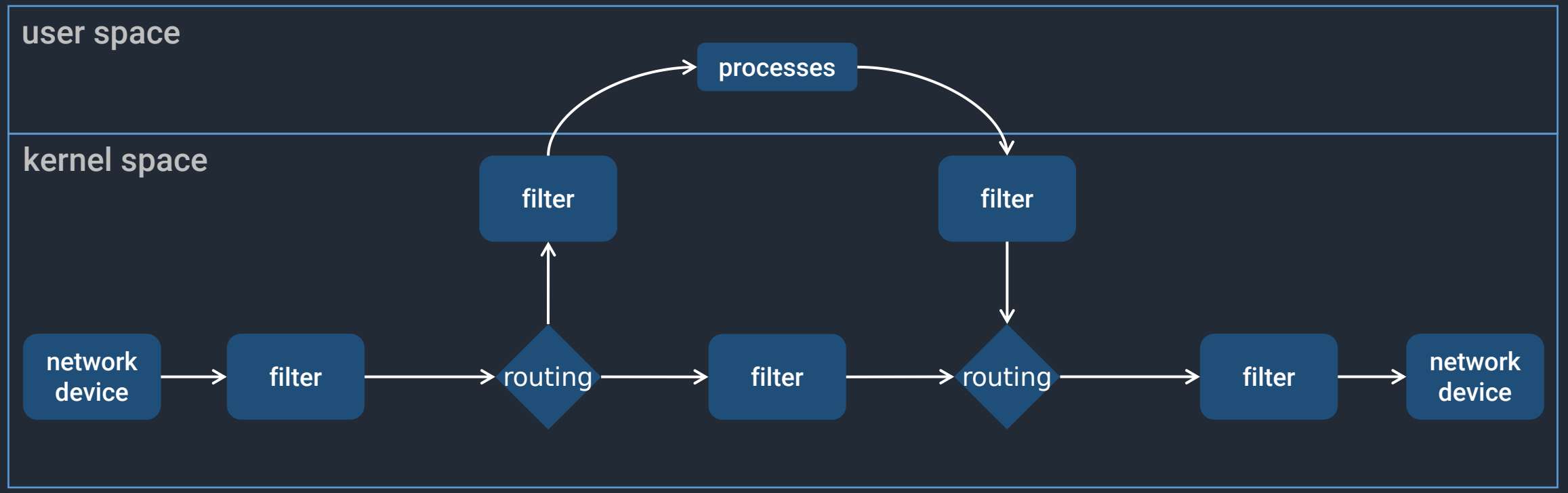
Deep Packet Inspection

Leonard Seibold

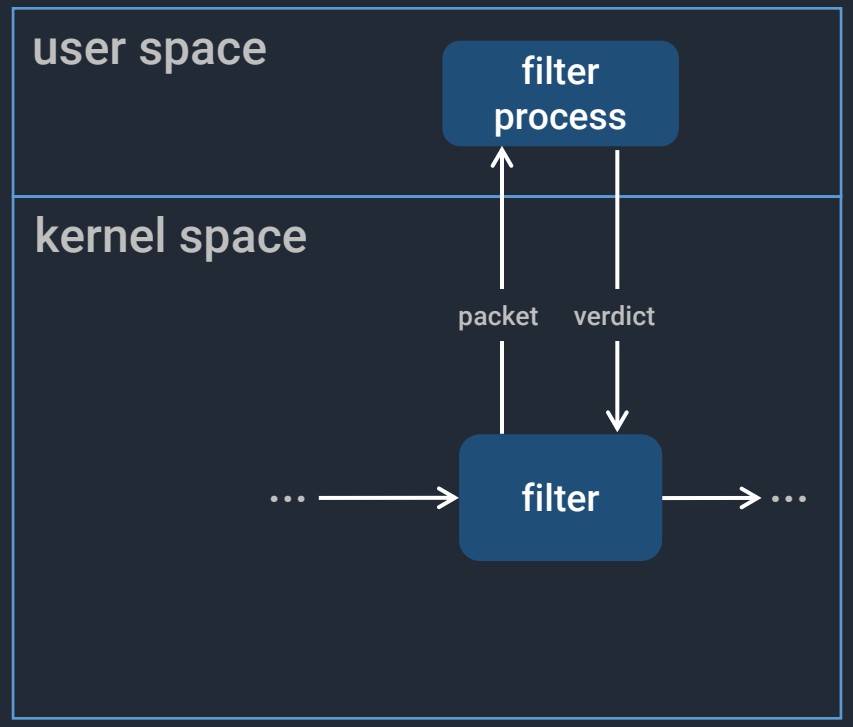
Netfilter API

- Linux API for packet filtering, NAT, mangling
- can be used to apply filters at different stages of packet traversal



Filter in user space

- kernel module often impractical
  - difficult to develop, debug
  - normal userland libraries (e.g. for Layer 7 Interpretation) can‘t be used
- Move filter to user space?
  - call user space function from kernel space (Upcall)



Communication with Unix Domain Sockets

- used for inter-process communication
- can use filesystem as address space
  - socket can be bound to a file
- connection based, full-duplex
- in this case: preserves user/kernel space separation, memory is not shared

```
01 unsigned int hook_func(void *priv,
02                        struct sk_buff *skb,
03                        const struct nf_hook_state *state) {
04
05     unsigned char ans[sizeof(unsigned int)];
06
07     sck_h->send_msg(sck_h, skb->data, skb->data_len);
08
09     if (sck_h->state == Error_Send) {
10         // Error handling (client disconnected)
11         return NF_ACCEPT;
12     }
13
14     sck_h->recv_msg(sck_h, ans, sizeof(unsigned int));
15
16     if (sck_h->state == Error_Recv) {
17         // Error handling (client disconnected)
18         return NF_ACCEPT;
19     }
20
21     return *((unsgined int *) ans);
22 }
```

Why this doesn‘t work...

- netfilter hook is called in critical section
- the kernel thread can‘t be scheduled
- socket IO is async, we need to wait for the answer
- `recv_msg` call leads to kernel panic as scheduler is confused

```
01 unsigned int hook_func(void *priv,
02                        struct sk_buff *skb,
03                        const struct nf_hook_state *state) {
04
05     unsigned char ans[sizeof(unsigned int)];
06
07     sck_h->send_msg(sck_h, skb->data, skb->data_len);
08
09     if (sck_h->state == Error_Send) {
10         // Error handling (client disconnected)
11         return NF_ACCEPT;
12     }
13
14     sck_h->recv_msg(sck_h, ans, sizeof(unsigned int));
15
16     if (sck_h->state == Error_Recv) {
17         // Error handling (client disconnected)
18         return NF_ACCEPT;
19     }
20
21     return *((unsgined int *) ans);
22 }
```

New Idea

- send packet to filter process, then drop it
- when the verdict is ready, reintroduce packet to netfilter

- Netfilter is already capable of this, there is even a userland library that implements socket communication to filter from user space

What's next?

- implement a filter in user space
- how big is the performance impact?

- different architecture: use Unix pipes to propagate packet/verdict through user space