Deep Packet Inspection

Leonard Seibold

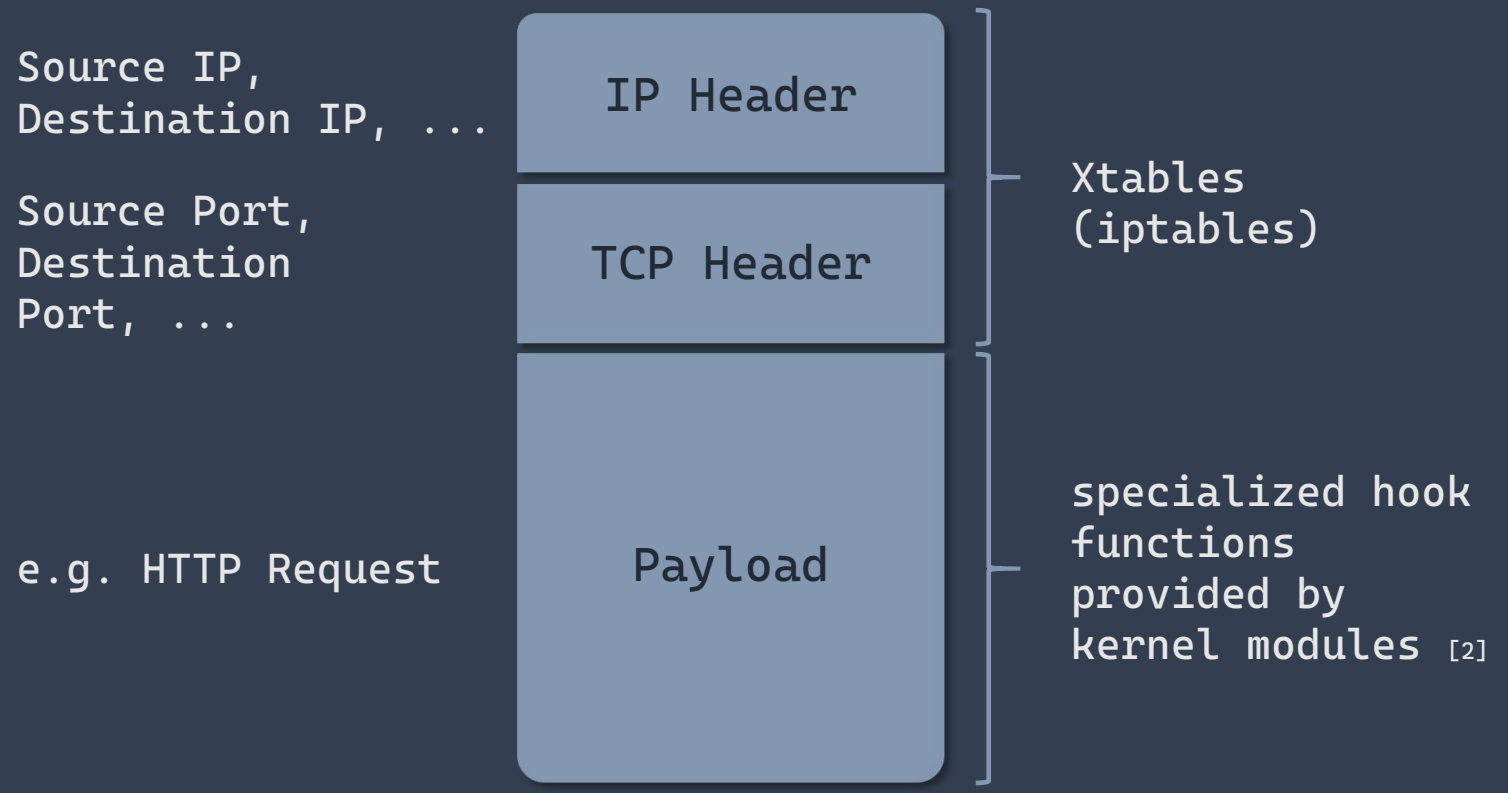
Agenda

- Motivation
- Architecture
- Orchestration with UNIX Pipes
- Live Demo
- Limits and Alternatives
- What‘s next?

> *Deep packet inspection (DPI) or packet sniffing is a type of data processing that inspects in detail data being sent over a computer network, and usually takes action by blocking, re-routing, or logging it accordingly.*
>
> — Wikipedia (Deep packet inspection) [1]

Motivation

- Deep Packet Inspection
  - inspect packet payloads, not just header info
  - allows for sophisticated firewalls, intrusion detection, ...
  - iptables only provides simple rules
  - parse application layer protocol

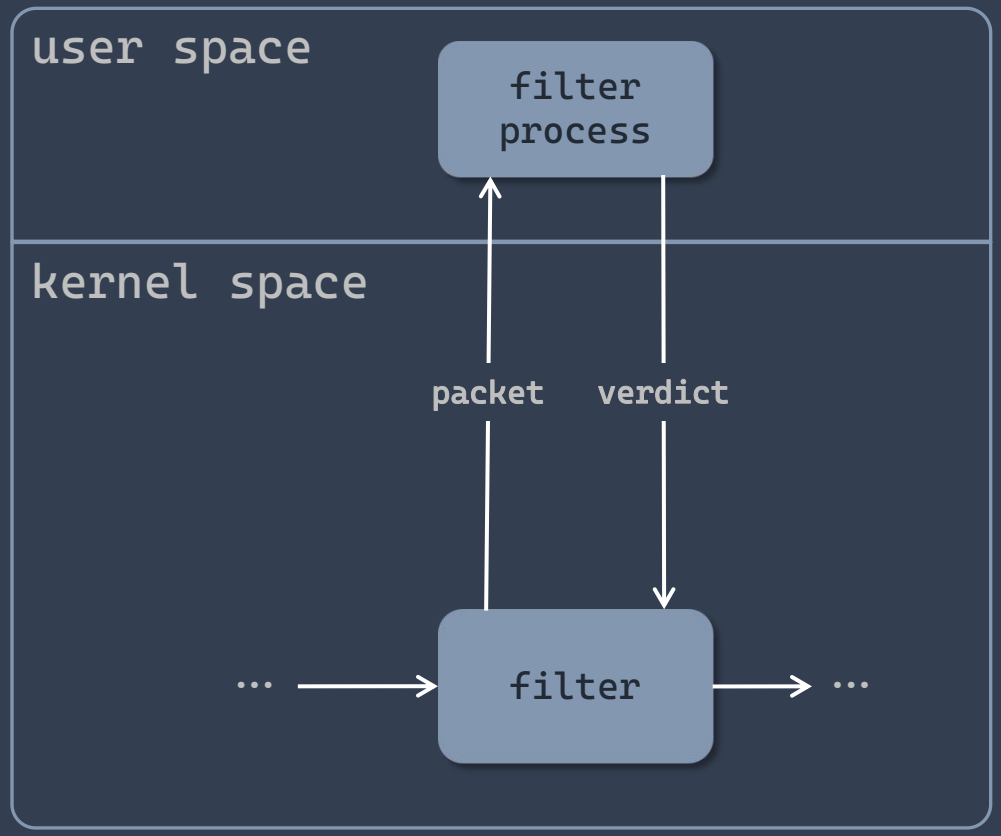


Motivation

Userspace Filtering

- Kernel space development is strenuous
  - debugging is difficult
  - bugs often cause the kernel to panic
  - development restricted to C90/C99
  - userland libraries (e.g. for application layer interpretation) can‘t be used

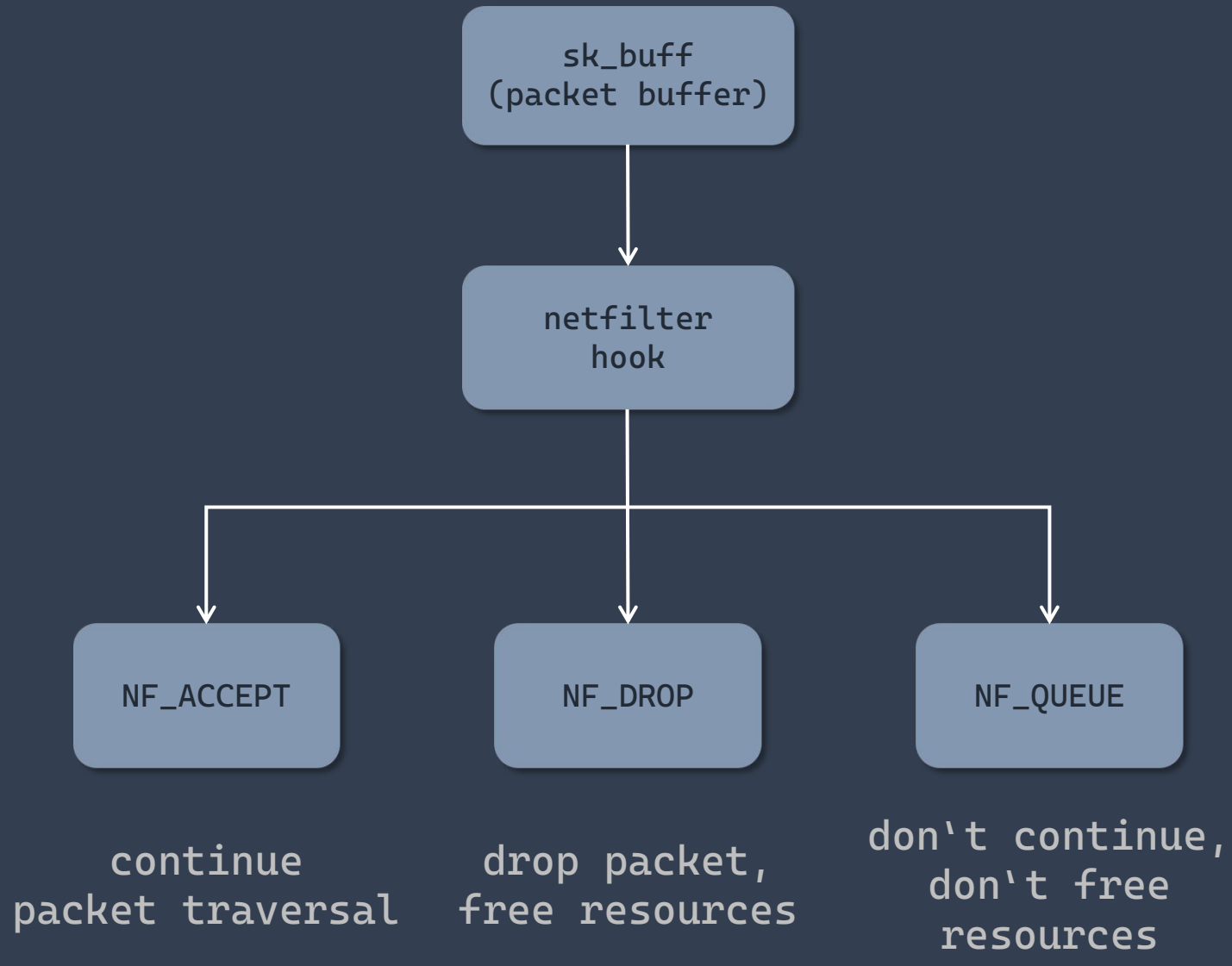
→ move filter to user space



Architecture

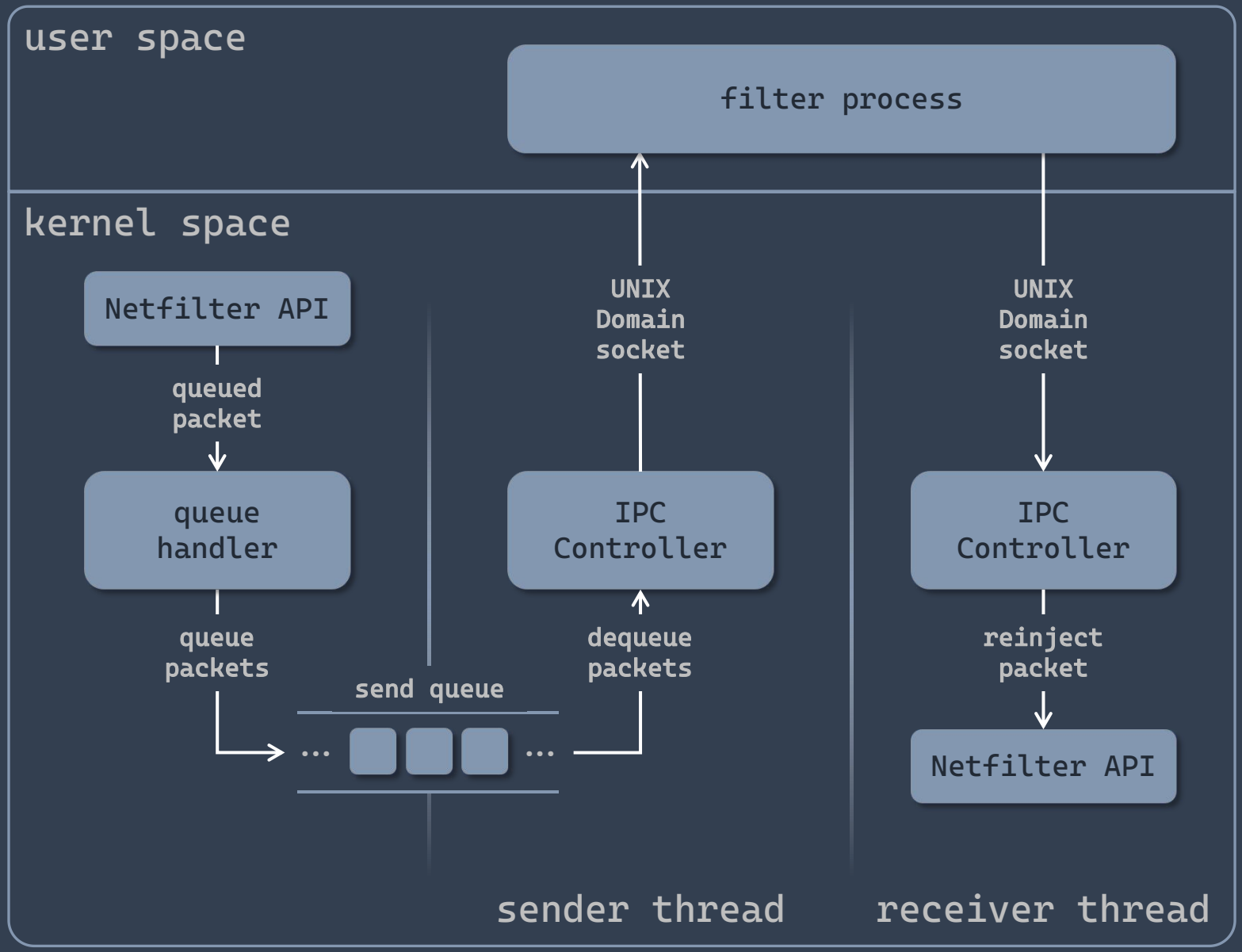
- Netfilter API allows us to register hooks to filter packets
  - used by Xtables
- Can be used to perform filter operations, but can‘t call `schedule()`
- iptables supports `NF_QUEUE` as a target:

```
$ iptables -A INPUT -p \
    tcp --dport 80 -j  \
    QUEUE
```



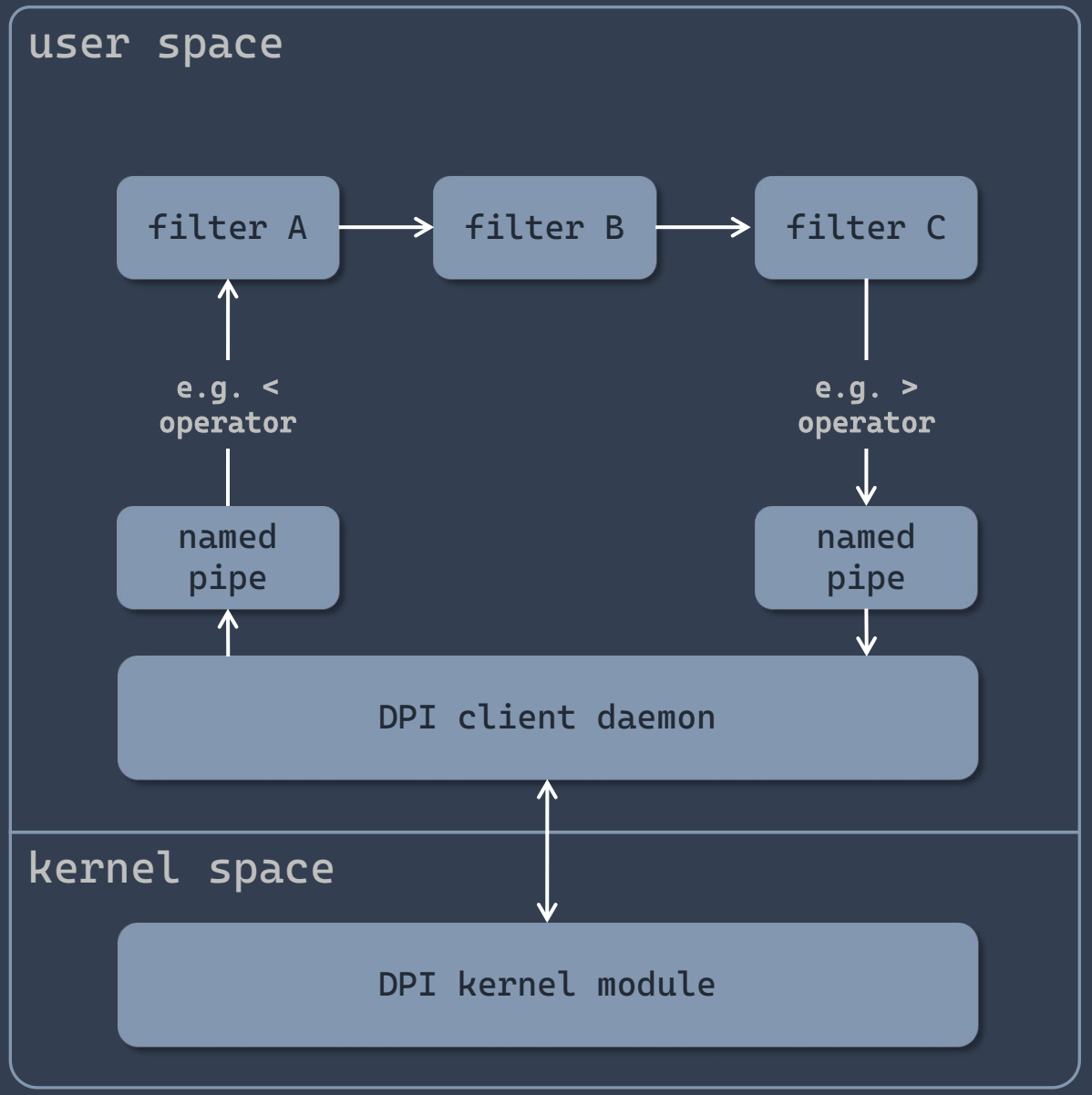
Architecture

- Netfilter API allows us to register a „queue handler“
  - gets called when a hook returned `NF_QUEUE`
  - *can* call `schedule()`
- UNIX domain socket provides IPC between kernel module and user space process
- only one client at a time for simplicity
  - less synchronization
  - no conflicting verdicts



Orchestration with UNIX pipes

- combine different filter programs
- follows UNIX paradigm
  - write small, specialized filters that can be reused and combined with others instead of one complex, unflexible one

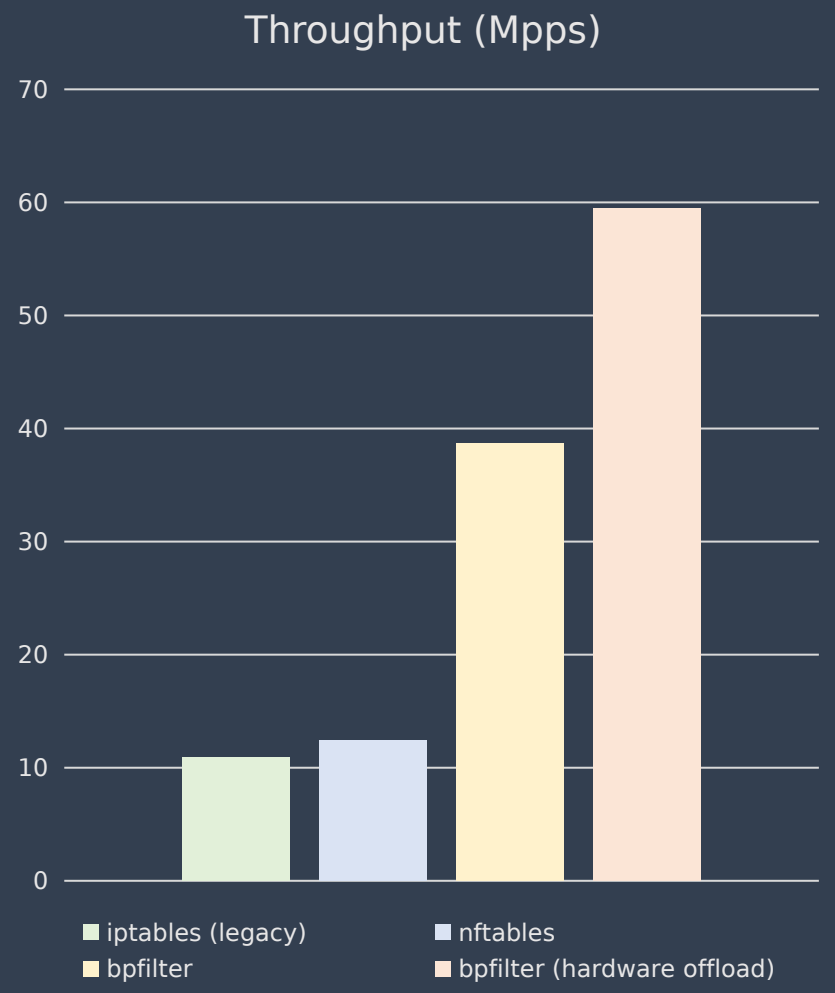


Live Demo

Limits and Alternatives

- big performance impact
  - a lot of scheduling, many syscalls, ...
  - packet data gets copied around alot
- data manipulation can be difficult due to checksums
- deep packet inspection is only useful if packet payload is unencrypted

- Alternative (the future?): eBPF [3][5]
  - special purpose VM directly running in kernel space
  - can filter packets before ordinary packet traversal even starts
  - allows for „quick dropping“



[5]

What‘s next?

- improve stability and performance
  - remove unnecessary heap allocations
  - fix weird nullpointer derefenciations and memory leaks
- cut out the middleman
  - write/read named pipes directly from kernel
  - sophisticated socket communication is also provided by existing nfnetlink_queue subsystem [4]
- expose more packet metadata to user space
  - network device, network namespace, ethernet frame data, ...
- more detailed analysis of performance impact
  - write a more complex filter that parses a application layer protocol (e.g. HTTP)

Sources

[1] https://en.wikipedia.org/wiki/Deep_packet_inspection (03.08.2020, 21:37)

[2] https://inai.de/documents/Netfilter_Modules.pdf
(by Jan Engelhardt, Nicolas Bouliane; rev. 03.07.2012) (03.08.2020, 21:37)

[3] https://lwn.net/Articles/740157/ (03.08.2020, 21:41)

[4] https://www.netfilter.org/projects/libnfnetlink/ (03.08.2020, 21:49)

[5] https://www.netronome.com/blog/frnog-30-faster-networking-la-francaise/ (03.08.2020, 21:54)

Project Sourcecode

github.com/zortax/deep-packet-inspection