KRISTOF PROVOST

A PACKET'S JOURNEY THROUGH PF

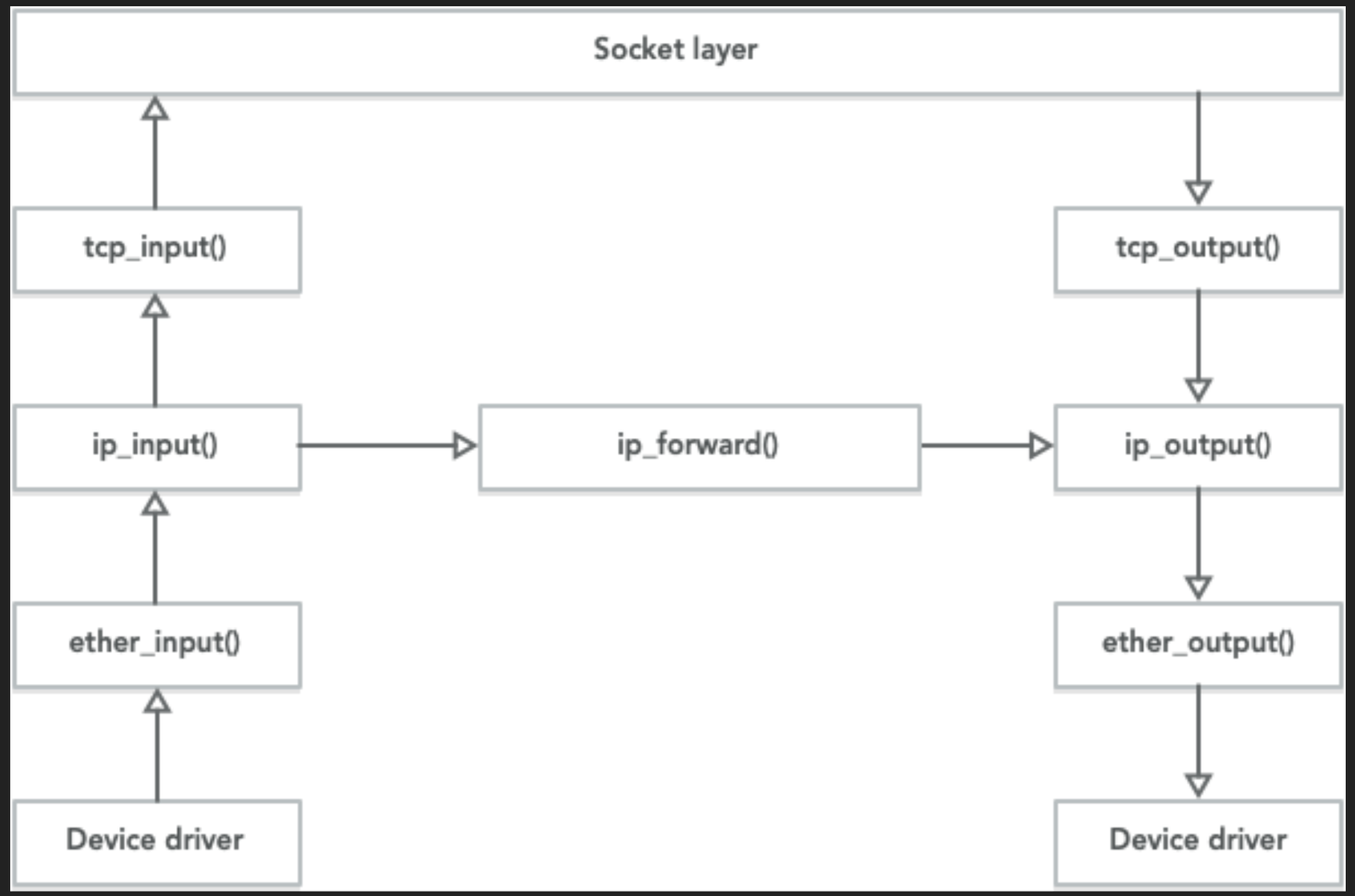
WHO AM I?

- Kristof Provost
- kp@FreeBSD.org
- pf (in FreeBSD) maintainer since 2015
  - “Hmm, IPv6 fragmentation handling isn’t great. I bet I could fix that!”
  - And in pfSense since 2021
    - Thanks, Netgate!

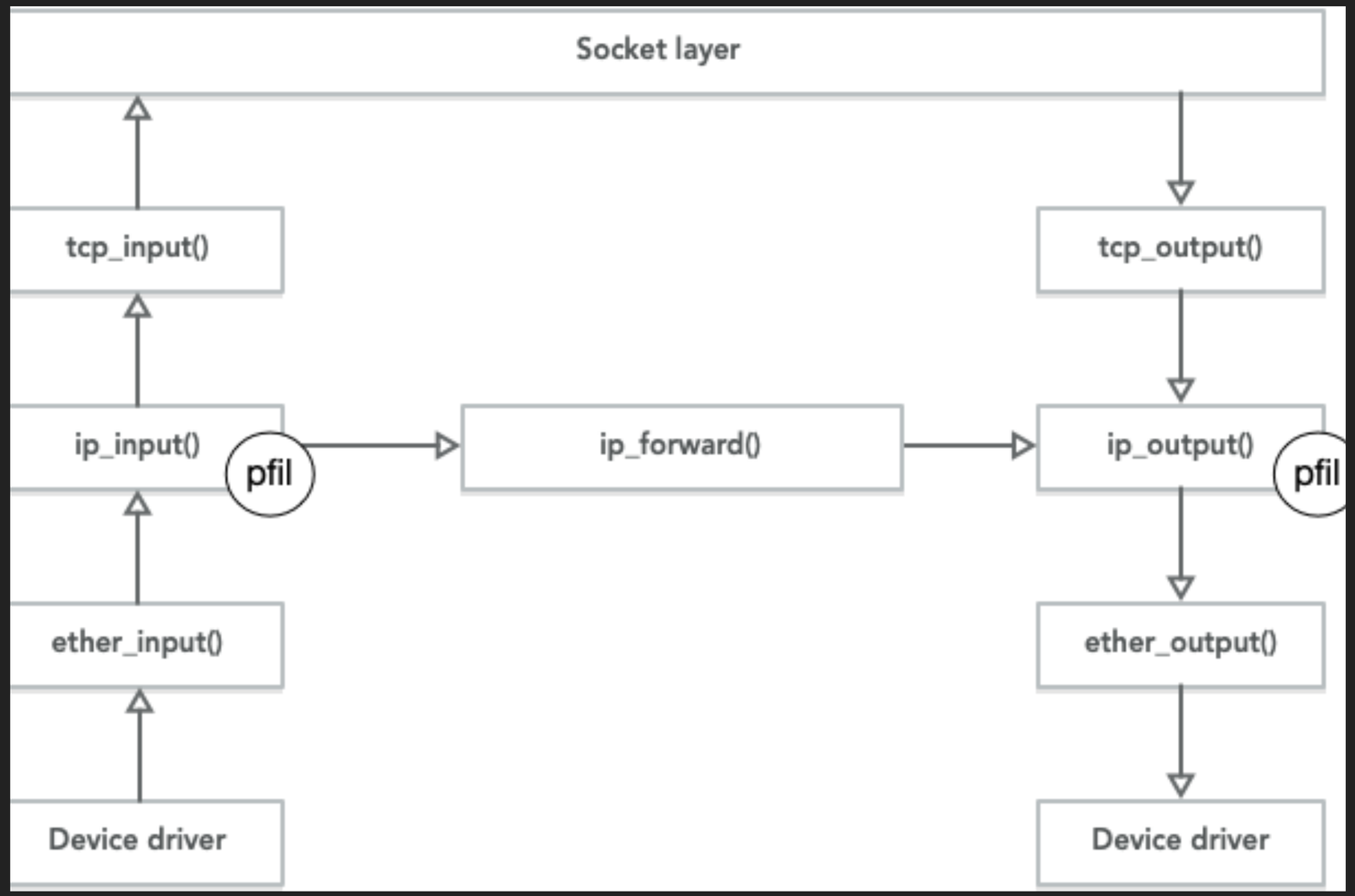
INTRODUCTION

- Based on FreeBSD main as of today(-ish)
- See also “A Packet’s Journey Through the OpenBSD Network Stack”
  - Alexander Bluhm
  - https://www.youtube.com/watch?v=Kn2XEW4Qre0
  - https://2024.eurobsdcon.org/slides/eurobsdcon2024-alexander_bluhm-a_packets_journey.pdf

TL;DR: THE NETWORK STACK



TL;DR: THE NETWORK STACK



KEY CONCEPTS

- States
  - pf is a stateful firewall[*]
  - Even for stateless protocols (i.e. UDP)
- Rules
  - i.e. what policy are we apply to packets (or connections!)

- [*] Except when not. pf on layer 2 is stateless

30,000 FT OVERVIEW

- pf_test()
- pf_setup_pdesc()
  - Parse packet
  - Normalise packet
    - i.e. reassembly
- pf_test_state_<protocol>()
  - (TCP, UDP, SCTP, ICMP, Other)
  - Find state
  - Or pf_test_rule()

30,001 FT OVERVIEW

- Output handling
  - pass
  - drop
  - route-to
  - af-to
- IPv6 special case
  - Re-fragment

IMPLICATIONS

- Test for state first
- Evaluate rules only if no state is found

- So if rules change, existing connections keep passing
  - ‘block all’ may not be block everything immediately!
  - Flush or kill states to actually terminate them

MORE IMPLICATIONS

- State lookup is performance critical
- How does this work?
  - Hash table
    - With linked list of states in each hash row
      - net.pf.states_hashsize
  - Key
    - Src/dst IP
    - Src/dst port (or ICMP type/code)
    - Address Family
    - Protocol

CONTROL PLANE

- How the user configures pf and get information out of it
- Interface to userspace
  - ioctl
  - ioctl + nvlist
  - netlink
    - Hopefully the only option in the future
- Somewhat abstracted by libpfctl
- pfctl

LOCKING

- Rules lock
  - Read/write lock
    - (Read-mostly), and therein lies yet another story
- State lock
  - Per hash-row
    - Another reason for net.pf.states_hashsize to be well dimensioned

LOCKING PFSYNC

- Used to be locked with a single mutex
- pfsync locking is now per-bucket
  - Buckets collect state updates for a number of states, based on their ID hash
  - Performance improvement from 30 to 100%
- Tuneable with
  - net.pfsync.pfsync_buckets
  - defaults to 2x ncpu

ETHERNET

- FreeBSD-unique feature
- (Very) basic filtering on Layer 2
  - Mostly so we can look at MAC addresses for captive portal scenarios
- Stateless

```
ether pass quick proto 0x0806
ether pass quick from 00:01:02:03:04:05
ether pass tag captive
```

SCTP

- Very TCP-like, but with multiplexed flows
- And multihoming
- Hence special case handling
  - Parse SCTP header to find ASCONF chunks
- Set up states for all multi homed options
- Also unique to FreeBSD
  - Not aware of another open-source firewall that handles SCTP multihoming

COUNTERS

- What do they mean?
  - Where do they live in the code?
  - Surprising performance implications

COUNTERS (2/2)

State Table	Total	Rate
current entries	0	
searches	301	150.5/s
inserts	0	0.0/s
removals	0	0.0/s
Counters		
match	301	150.5/s
bad-offset	0	0.0/s
fragment	0	0.0/s
short	0	0.0/s
normalize	0	0.0/s
memory	0	0.0/s
bad-timestamp	0	0.0/s
congestion	0	0.0/s
ip-option	0	0.0/s
proto-cksum	0	0.0/s
state-mismatch	0	0.0/s
state-insert	0	0.0/s
state-limit	0	0.0/s
src-limit	0	0.0/s
synproxy	0	0.0/s
map-failed	0	0.0/s
translate	0	0.0/s

COUNTERS (3/3): RULE COUNTERS

```
block drop in log inet all
  [ Evaluations: 871131     Packets: 127454     Bytes: 14161624     States: 0      ]
  [ Inserted: uid 0 pid 0 State Creations: 0      ]
```

DTRACE: USEFUL PROBE POINTS

- pf:purge:state:rowcount
  - Useful for monitoring hash table usage
- pf:ioctl:ioctl:error & pf:ioctl:function:error
  - Useful to pinpoint ioctl failures
- pf:sctp:multihome:{test, add, remove}
  - For SCTP multihome monitoring
- pf:{ip,ip6}:route_to:{entry, drop, output}
  - route-to/reply-to/dup-to debugging

QUESTIONS?