GALETTE: a Lightweight XDP Dataplane on your Raspberry Pi

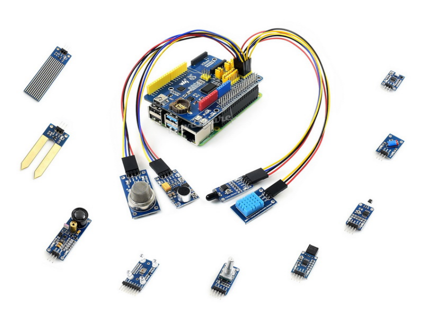
Kyle A. Simpson, Chris Williamson, Douglas J. Paul, Dimitrios P. Pezaros
kylesimpson1@acm.org
FelixMcFelix *https://mcfelix.me*

13th June, 2023

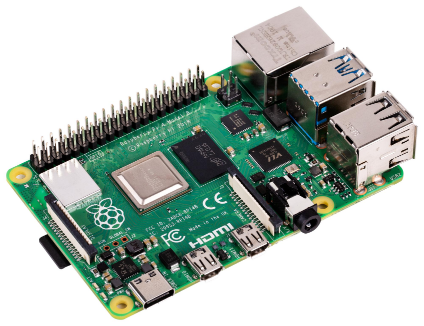
University of Glasgow

Securing Sensor & IoT Networks

- Security – ingress/egress packet processing by *network functions*.
    - IP layer – Firewalls, DPI, ACLs...
    - Middleboxes a bad fit.
    - Needs to be reconfigurable – attacks and security context evolve.
- Ideally in-situ.
    - Dynamic/retrofitted.
    - But limited space + power in the field.
    - Physically vulnerable!



Fast, cheap, and secure IoT Defence – pick 3?



- Single-board compute like RPis are small, capable, affordable! Cheap!
    - See also: NUCs (££), Jetsons (£££).
    - *Linux-based*: Easy(/ier) to target and write for. We also get kernel network stack advancements.
    - Different CPU architectures.
- Sensor networks have low data rates; a good fit.
- Project goals:
    - Fast! Low-latency, quickly reconfigurable.
    - Secure! efficient NFV code gen from *memory-safe languages*.

Galette's Research Objectives

1. What specialisations does XDP Function Chaining need to best suit SBCs?
    - 'Acceptably' low-latency packet-processing, without pushing CPU/power draw too high?
2. How do we make eBPF + native compile from memory-safe systems languages easy? And portable across 'native'?
    - One Rust program per NF $\implies$ compiles to eBPF + `$PLATFORM`.
    - Simple, dynamic chain format.
    - Fast reconfiguration.
3. How much better is it [power, perf, lat]?
    - With/without polling.

Background

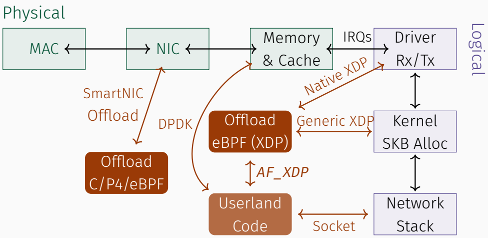
Limits of existing SFC

- 'Best' low latency processing (DPDK) is expensive – CPU and power.
    - ...IFF you have HW support (NUCs)
- No powerful hardware offloads or acceleration.
    - FPGA hats/daughterboards 'off-path'
- Devices physically vulnerable, no ECC memory.
- ...So, how to reconcile with cheap & portable SBCs?

eBPF: What and Why?

- Simple register machine VM (user-written) code, derived from BPF.
- Modern use – Kernel hooks, perf instrumentation, debugging
- JIT compiled
- Kernel-verified
  - Bounds-checked pointer accesses
  - Program size limited, no unbounded loops
  - Syscalls (*eBPF helpers*) exposed based on hook point

Network stack improvements: XDP



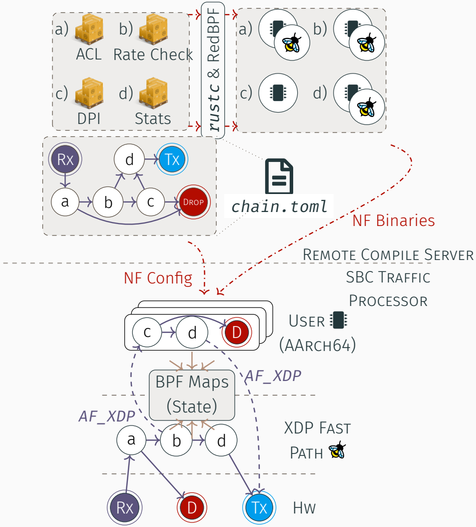
- eBPF hook attached to packet ingress
- Variations on hook $\in$ {Offload, Driver, Generic}
  - Perf degrades gracefully according to driver support
- Hook can modify & inspect packets before forwarding to Linux stack, sending straight to (another) NIC, or drop.
- Since 2019: *AF_XDP* stack bypass!

Q1: Specialising ***AF_XDP*** SFC to SBCs

Concrete design differences

- **Problem:** Mismatch of HW queues to physical cores:
  - Soln: load balance or place high-latency NFs in userland.
  - …also, don't pass packets back to k-space.
- **Problem:** XDP hooks only on ingress (*for now*):
  - Soln: load balance or place high-latency NFs in userland?
  - Write an individual NF *once*, compile for both envs, and replicate NFs as needed.

Design: Bird's eye view



- Two-tier approach—XDP & User.
- Composable NFs – graph structure.
- Critical or high performance NFs go into XDP:
  - Early results – low latency for most packets.
- Rare 'slow-path' still kernel bypass:
  - Expensive & proprietary code.
  - Only for candidate attack traffic.
- Reconfigurable, dynamic.

How does this differ from other frameworks?

In Security? SafeBricks[1], AuditBox[2] or similar.

- ...No SGX support in devices of interest.

In eBPF/XDP space? Polycube[3]!

- Built around datacentres – we often have just one HW queue for a NIC.

---

[1] Poddar *et al.*, 'SafeBricks: Shielding Network Functions in the Cloud'.
[2] Liu *et al.*, 'Don't Yank My Chain: Auditable NF Service Chaining'.
[3] Miano *et al.*, 'A Framework for eBPF-Based Network Functions in an Era of Microservices'.

How do we upcall to userland?

- **Problem:** Can send packet over *AF_XDP*, but *no context on what the next (callee) NF is.*
    - Polycube's solution inadequate: one discrete userland component per *cube*.
- Soln: Adjust headroom of packets, write in ID and action of caller.
- ...might be a ***memcpy***, but ideally only paid on packets who need it.

Q2: Easy Joint-Compile (eBPF + Native) from Rust 🦀

Skeleton details

- Consistent NF API for both XDP/userland.
- Rust compiler should be able to enforce...
  - *#![forbid(unsafe_code)]* (or similar cargo tooling) on NF module crates,
  - all NF branches specified.
- All compilation on external server.
  - SBC too constrained.
  - If compile-server is TEE-equipped, can attest compiler/code etc. following SotA!

```rust
#![no_std]
pub enum Action {
  Left,
  Right,
  Up,
  Down,
}

pub fn packet(bytes: impl Packet) -> Action {
  let addr_lsb_idx = 14 +
  match pkt.slice_from(12, 2) {
    Some(&[0x08, 0x00]) => 19, //v4
    Some(&[0x86, 0xDD]) => 39, //v6
    _ => {return Action::Left},
  };

  match pkt.slice_from(addr_lsb_idx, 1)
   .map(|v| v[0] % 2) {
     Some(0) => Action::Left,
     Some(1) => Action::Right,
     Some(2) => Action::Up,
     Some(3) => Action::Down,
     _ => unreachable!(),
  }
}
```

mod.rs: Load balance on dest addr

?? Chain example

< In lieu of a demo... >

Q3: Performance

Setup

?? Baselines

?? What Machines

?? Waht NFs.

?? WHy

?? Results?

content...

High-level

?? better at these things

If you want more detailed data, please check out our paper

Takeaways:

Cheap NFs: SBCs for packet processing.

Low-latency and fast: XDP path for majority of traffic, early & cheap anomaly checks, power savings.

Secure: PUFs for device, server, and function chain attestation.

Easy to write: *native and XDP* portable NFs in Rust.

Questions?

kylesimpson1@acm.org

FelixMcFelix *https://mcfelix.me*

University of Glasgow

NETLAB
NETWORKED SYSTEMS RESEARCH LABORATORY
University of Glasgow | School of Computing Science