Apiary: Distributed Programming and Lifecycle Management for eBPF

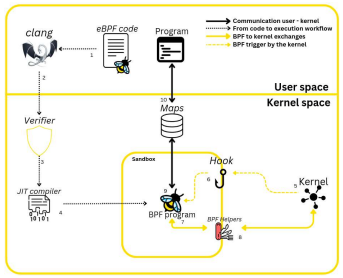
Pragna Mamidipaka, Theophilus A. Benson
Carnegie Mellon University

Carnegie Mellon University
Electrical & Computer ENGINEERING

sigcomm 2025 COIMBRA
Networking Women

Brief on eBPF

Why it's a superpower for the kernel



Source - https://blog.pocok.dev/articles/ebpf-guide

- ✓ Safe, flexible way to run code inside the Linux kernel.
- ✓ Used in networking, observability, and security.
- ✓ Customize and extend the kernel without rebuilds.

1

eBPF in practice

Powerful, but fragile

The good

Hyperscalers run tens to hundreds of eBPF programs per server – Huge performance wins!

The bad

Complex functionality spans multiple kernel hook points – so solutions are inherently distributed.

The ugly

A. Writing and maintaining eBPF programs is now extremely challenging.
B. Incidents and regressions routinely occur as programs grow.

2

Apiary

Distributed systems abstractions for eBPF

Components

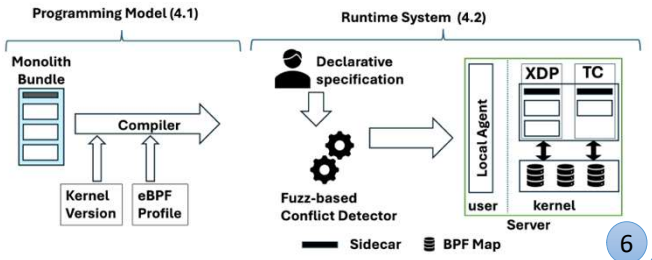
- **Bundle:** Classifier + enforcer actors.
- **Apiary APIs:** Simple primitives for coordination.
- **Compiler:** Translates bundle specs into eBPF programs, maps, and metadata.
- **Local Agent:** deploys **sidecars**.

Decoupled development from runtime issues!

Properties

- ✓ **Coordination, consistency, ordering:** Enforced by sidecars.
- ✓ **Conflict detection:** Declarative specs + fuzzing explore unsafe interactions.
- ✓ **Performance optimization:** Compiler picks best kernel-local IPC; supports program merging, migration.

Correct and Efficient by design!



6

The *mismatch* and why it exists

Per-program tooling vs multi-hookpoint functionality

- ❖ **Modern eBPF applications have outgrown per-program infrastructure.** Use cases span multiple hookpoints and need holistic coordination
- ❖ **Language and verifier constraints** Programs must be small, and cross-program interactions are not validated.
- ❖ **Reassembly of state is ad-hoc** Classification, state sharing, and ordering must be rebuilt manually in each program, making shared context across programs very **brittle**.

3

Key insight

You need centralized control

- − Teams ship overlapping programs that could interfere with each other
- − Program wise updates can create version skew across the pipeline
- + Centralized coordination maintains accurate cross-hookpoint state.

It's possible to get it in eBPF

State and program logic already separated via maps, allowing central visibility and control.

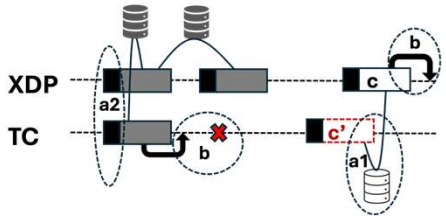
Lift coordination, consistency, conflict resolution to a runtime so that developer solely focuses on eBPF logic.

5

Development challenges

(a) Distributed coordination (b) Conflict resolution (c) Consistent updates

Grey and white boxes are distributed eBPF programs from different teams. The black boxes are the classifier component which is redundantly replicated across eBPF programs.



4

- ➢ **Scenario** For TCP connect tracing, developers duplicate classifier and track using custom logic. **Apiary makes it a bundle :** classifier defines a conn which enforcers reuse.
- ➢ **Update** : sidecars route all packets of a flow to the same version.
- ➢ **Ops** : Declare invariants (ordering/coverage), Apiary fuzzes permutations to flag conflicts before production.

7