BPF cgroup infra enhancements for Kubernetes like environments

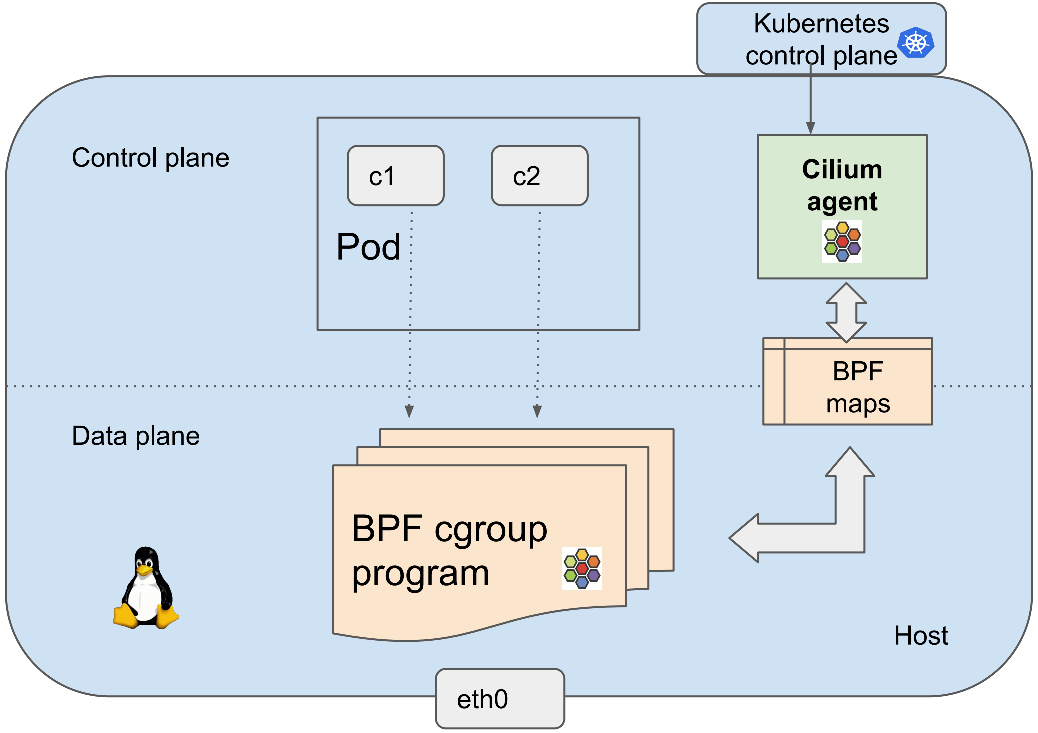
Aditi Ghag, Isovalent

Background: Traffic control at cgroup (socket) layer in Cilium

- Socket-LB: Load balancing for service vip to backend
    - Mount cgroup2 fs
    - **BPF_PROG_TYPE_CGROUP_SOCK_ADDR** attached at the cgroup root
- Source IP and port in socket addr not available during program execution
- Common identification across control and data plane for fine-grained traffic control
    - Selectively skip socket-LB
    - Policy enforcement
    - Tracing

Cgroup ID as shared context between control and data plane

Cgroup ID: Unique value for cgroup v2 hierarchy



Maintains pod and container cgroup paths and respective cgroup ids

Retrieves cgroup-id using BPF helpers:

```
u64
bpf_get_current_cgroup_id(void):
```
container cgroup ID

```
u64
bpf_get_current_ancestor_cgroup_id(
int ancestor_level):
```
pod cgroup ID

Problems: Inconsistent cgroup hierarchies

- Pod/container cgroup paths are platform dependent
  - /kubepods.slice/kubepods-besteffort.slice/kubepods-besteffort-**pod9ac48755_3968_48e4_b9dc_6d4b69f3bb42**.slice/**cri-containerd-3baf66ee56a52a8765c3deb2444315411a888fa3e2f8f7ddd75e9ded3c34425e**.scope
  - /kubelet/kubepods/**pod4841248b-fc2f-41f4-9981-a685bf840ab5/d8f227cc24940cfdce8d8e601f3b92242ac9661b0e83f0ea57fdea1cb6bc93ec**
- Variable fields like **pod QoS**, ID, container runtime strings encoded in paths
  - /kubepods.slice/kubepods-**pod9aa48755_3968_48e4_b9dc_6d4b69f3bc23**.slice/**cri-containerd-3caf66ee56a52a8765c3deb2444315411a888fa3e2f8f7ddd75e9ded3c34425e**.scope
- We discussed this issue with Kubernetes upstream
  - Recommended container runtimes to pass pod cgroup paths to CNIs like Cilium
  - ✅ Cilium control plane has reliable access to pod cgroup paths
- Data plane unable to retrieve **pod** cgroup IDs deterministically

Limitations and proposals to extend BPF cgroup infra

```
u64 bpf_get_current_ancestor_cgroup_id(int ancestor_level):
```
pod cgroup ID

```
Description: The helper is useful to implement policies based on
             cgroups that are upper in hierarchy than immediate
             cgroup associated with the current task.
```

Hindrance: Ancestor level is with respect to the root so it isn't constant!

Alternative: Cgroup local storage associated with root hierarchy where BPF cgroup programs are attached X

Proposal: Allow ancestor levels with respect to current tasks => retrieve pod level cgroup ID (one level up ancestor) from container task

1. Pass negative ancestor levels to current helper?
2. Introduce new kfunc?
3. Get current hierarchy level?