Experiment Spec — Đánh giá hiệu năng datapath mạng Kubernetes

So sánh Mode A (kube-proxy baseline) vs Mode B (Cilium eBPF kube-proxy replacement)

> Mục tiêu của spec này là làm cho thí nghiệm **reproducible**, **fair**, có **bằng chứng** và có thể **giải thích cơ chế** (không chỉ "ra số").

---

1) Mục tiêu (Objectives)

1. Đo và so sánh hiệu năng đường đi mạng (datapath) cho traffic nội bộ Kubernetes:
   - Service-to-Service qua **ClusterIP**
   - Hành vi khi **tải cao** và **connection churn**
   - Overhead khi bật **NetworkPolicy** (off → on)
2. Trả lời được 3 câu hỏi chính:
   - Mode B có cải thiện **tail latency (p95/p99)** so với Mode A không?
   - Ở tải cao và churn, mode nào **ổn định** hơn (error rate/timeout)?
   - Khi bật policy, overhead tăng thế nào và có **evidence** về enforcement?

---

2) Phạm vi và giả thuyết (Scope & Hypotheses)

Phạm vi

- In-cluster traffic (client pod → Service → server pods)
- Không đo Internet ingress/egress hay cross-region.
- Focus vào datapath Service + Policy.

Giả thuyết

- **Mode B** (eBPF datapath) có thể giảm overhead/giảm tail latency khi tải cao.
- **Mode B** có thể nhạy với cấu hình (kube-proxy replacement, policy), và overhead observability (Hubble) cần được kiểm soát.

---

3) Môi trường thực nghiệm (Environment)

3.1 Hạ tầng (AWS/EKS)

3.1 Hạ tầng (AWS/EKS)

- Platform: **AWS EKS**
- AZ: **1 AZ** (giảm nhiễu cross-AZ; ghi rõ vì sao trong Threats to validity)
- Node group: **Managed Node Group**, cố định **min=desired=max=3**, không autoscale khi đo
- Instance type: **t3.large (2 vCPU, 8GB RAM)** đồng nhất giữa các node
- Không chạy workload ngoài scope benchmark trong lúc đo

> Lưu ý: t3.large là burstable → cần theo dõi dấu hiệu cạn CPU credit/tail latency méo.

3.2 Thành phần trong cluster

- **Mode A (Baseline)**: kube-proxy mặc định EKS (iptables/ipvs tùy cấu hình)
- **Mode B (eBPF)**: Cilium kube-proxy replacement (kube-proxy-free) + Hubble
- Observability: Prometheus + Grafana (kube-prometheus-stack hoặc tương đương)
- Benchmark tool: Fortio (in-cluster) *(có thể thay bằng k6 nếu bạn chọn k6)*

3.3 Workload benchmark (tối giản nhưng đúng bài)

- `server`: HTTP echo/httpbin
  - replicas cố định
  - có requests/limits (để tránh noisy scheduling)
- `client`: pod chạy Fortio
- Service type: **ClusterIP**
- Namespace: `workload-bench`

---

4) Biến thực nghiệm & tiêu chí đánh giá (Variables & Metrics)

4.1 Biến độc lập (Independent variables)

- **Mode**: A vs B
- **Scenario**: S1, S2, S3 (mục 6)
- **Load level**: L1, L2, L3 (được calibrate trước)

4.2 Metric bắt buộc (Primary metrics)

- Latency: **p50 / p95 / p99** (ms)
- Throughput/RPS
- Error rate (HTTP errors, timeouts, connection errors)

4.3 Metric để giải thích cơ chế (Explainability metrics)

Tối thiểu chọn 1–2 nhóm sau (đủ để "giải thích", không cần tham):

- Node CPU breakdown: user/system/**softirq**
- Pod CPU/memory (server/client/cilium)
- Network drops/retransmits (nếu có)
- **Hubble flows** (Mode B): verdict FORWARDED/DROPPED, drops, policy decisions

---

5) Kiểm soát nhiễu (Controls) + Fairness Checklist

5.1 Controls (nguyên tắc)

- Cố định: số node, instance type, replicas, resource requests/limits
- Benchmark **in-cluster** (giảm nhiễu Internet)
- Có warm-up, duration cố định, và lặp **≥ 3 runs/case**
- Không nâng cấp/đổi cấu hình giữa các runs trong cùng mode
- Không chạy 2 mode đồng thời trên cùng hạ tầng (tránh nhiễu tài nguyên)

5.2 Fairness checklist (đưa vào report/slide để chống bắt bẻ)

Giữ **giống nhau** giữa Mode A và Mode B:

- EKS/K8s version, node AMI/kernel
- Instance type, số node, AZ, scaling policy (fixed 3 nodes)
- Workload YAML: replicas, requests/limits, image, env
- Fortio params: duration, warm-up, concurrency/QPS, payload size
- Monitoring stack (Prometheus/Grafana) cấu hình tương đương
- Cách thu thập artifacts và cách tổng hợp số liệu

Chỉ khác nhau:

- Datapath: kube-proxy (A) vs Cilium eBPF kube-proxy replacement (B)
- (Nếu có) policy engine khác biệt do Cilium (nhưng phải đảm bảo semantics tương đương khi so sánh S3)

---

6) Kịch bản đo (Scenarios)

S1 — Service Baseline

Mục đích: baseline latency/throughput khi chỉ đi qua Service.

- Policy: off hoặc allow-all

- Policy: off hoặc allow-all
- Kỳ vọng: error rất thấp, tail latency ổn định

S2 — High-load + Connection Churn

Mục đích: stress ở tải cao, nhiều connection mở/đóng liên tục.

- Tăng concurrency + QPS
- Có churn (ví dụ request ngắn, nhiều kết nối mới; keepalive có thể off tùy kịch bản)
- Kỳ vọng: tail latency (p95/p99) tách biệt rõ, error rate có thể tăng

S3 — NetworkPolicy Overhead (off → on)

Mục đích: đo overhead khi bật policy enforcement và chứng minh enforcement có thật.

- Bước 1: policy off/allow-all → chạy như S1
- Bước 2: apply policy → chạy lại cùng tải

Nâng độ khó vừa đủ (khuyến nghị để ăn điểm)

- **S3a (simple)**: 1 policy rule đơn giản
- **S3b (complex)**: policy nhiều rule hơn (vd 10–20 rule hoặc nhiều selectors) → giúp bạn thấy "overhead tăng theo complexity" và viết phân tích tốt hơn.

---

7) Load levels (L1/L2/L3) + Calibration plan

7.1 Định nghĩa load levels

- **L1 (Light)**: ổn định, gần như 0 error, p99 thấp
- **L2 (Medium)**: xuất hiện tail latency rõ nhưng chưa gãy
- **L3 (High)**: gần ngưỡng gãy (tail tăng mạnh), error vẫn trong mức chấp nhận

7.2 Calibration (bắt buộc làm trước khi chạy chính thức)

Mục tiêu: chọn L1/L2/L3 bằng dữ liệu (tránh "chọn cảm tính").

- Chạy sweep tăng dần load (QPS hoặc concurrency) trên **cùng 1 mode** (thường làm trên Mode A trước)
- Ghi lại: QPS/concurrency vs p99 vs error rate
- Chốt 3 điểm L1/L2/L3 và **đóng băng tham số** cho toàn bộ thí nghiệm

Deliverable calibration

- 1 hình nhỏ: (load → p99) hoặc (load → error rate)

- Bảng tham số L1/L2/L3 (c, q, duration, payload…)

---

8) Methodology — Quy trình chạy chuẩn cho mỗi case

Mỗi tổ hợp: **(Mode × Scenario × Load level)** chạy theo pipeline:

1. **Pre-check**
   - `kubectl get nodes/pods` đảm bảo healthy
   - Mode B: `cilium status`, `hubble status` OK
   - Prometheus targets UP
2. **Warm-up**
   - chạy 30–60s warm-up (không ghi số liệu chính thức)
3. **Measurement**
   - duration cố định (ví dụ 120s)
   - ghi output Fortio/k6 (latency quantiles, RPS, errors)
4. **Repeat**
   - lặp **≥ 3 runs**
   - nghỉ ngắn 30–60s giữa runs (tránh nhiệt/credit/noise)
5. **Collect artifacts**
   - thu logs/metrics snapshot theo mục 9

---

9) Artifacts & Evidence (bằng chứng bắt buộc)

9.1 Cấu trúc thư mục

`results/<mode>/<scenario>/<load>/<run_timestamp>/`

9.2 Tối thiểu phải có (per run)

- `bench.log` : stdout Fortio/k6
- `metadata.json` : thông tin cấu hình run (mode/scenario/load/params/versions/timestamps)
- `hubble.log` : flows/verdict (Mode B; S3 đặc biệt quan trọng)
- `grafana/` : ảnh dashboard (latency/RPS + node CPU/softirq + network)
- `manifests/` : YAML thực tế đã apply (deploy/service/policy) hoặc `kubectl get ... -o yaml`

> Thầy thường tin kết quả khi bạn có “bench output + dashboard + flow evidence”.

---

10) Tổng hợp số liệu & so sánh (Aggregation & Comparison)

10.1 Tổng hợp

- Với mỗi (Mode, Scenario, Load):
  - lấy **median** của p50/p95/p99, RPS, error rate từ ≥3 runs
  - ghi thêm **min–max** (hoặc std) để thể hiện biến động

10.2 So sánh A vs B

- Tính chênh lệch:
  - Δ% p99 (quan trọng nhất), Δ% p95, Δ% p50
  - Δ% RPS / throughput
  - Δ error rate
- Kèm giải thích bằng evidence:
  - node CPU/softirq, saturation signals
  - drops/retransmits (nếu có)
  - Hubble verdict (đặc biệt ở S3)

---

11) Appendix — Versions & Config (bắt buộc ghi rõ)

Tạo 1 bảng (1 trang) trong report/appendix:

- EKS/K8s version: `...`
- Node AMI / kernel: `...`
- Cilium version: `...`
- kube-proxy replacement mode: `...` (strict/partial/...)
- Hubble enabled: `yes/no` + sampling (nếu có)
- Prometheus/Grafana stack version: `...`
- Fortio/k6 version: `...`
- Workload image version: `...`

---

12) Threats to Validity (nguy cơ sai lệch) + cách giảm thiểu

- **Burstable instances (t3.large):** CPU credit có thể làm tail latency méo
  - giảm: duration hợp lý, nghỉ giữa runs, theo dõi CPU/softirq, ghi chú bất thường
- **Noisy neighbor trên AWS:** nền cloud có biến động
  - giảm: lặp ≥3, dùng median + min/max, chạy xen kẽ A/B theo phiên

- **Observability overhead:** Prom/Grafana/Hubble tạo overhead
  - giảm: giữ monitoring stack tương đương giữa 2 mode; chỉ khác datapath
- **1 AZ vs 2 AZ:** chọn 1 AZ giảm nhiễu nhưng không đại diện cross-AZ
  - ghi rõ giới hạn phạm vi và lý do lựa chọn

---

13) Definition of Done (đủ điều kiện chốt report)

Bạn coi như "done" khi có:

- Calibration L1/L2/L3 + 1 hình minh họa
- Bảng tổng hợp A vs B cho S1/S2/S3 (p50/p95/p99/RPS/errors)
- Evidence: dashboard + (Mode B) hubble flows cho S3
- Appendix versions/config + Threats to validity rõ ràng