Envoy 调试流量的常用技巧

Lizan Zhou (@zlizan)
2020-10-28

tetrate

Intro

- Envoy senior maintainer
- Istio networking/security WG lead
- Founding Engineer @ Tetrate
- Twitter/WeChat: @zlizan

We're hiring! https://cloudnative.to/job/tetrate/

Agenda

- What is Envoy
- Envoy design philosophy
- Intro to Envoy Config
- Life of a Request
- Debugging feature
- Demo

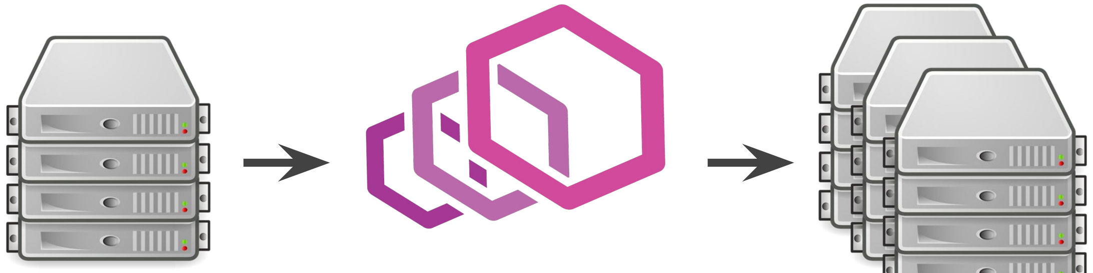
State of microservice networking in industry

- **Languages** and frameworks.
- **Protocols** (HTTP/1, HTTP/2, gRPC, databases, caching, etc.).
- **Infrastructures** (IaaS, CaaS, on premise, etc.).
- Intermediate **load balancers** (AWS ELB, F5, etc.).
- **Per language libraries** for service calls.
  - Inconsistent **observability** output (stats, tracing, and logging).
  - Implementations (often partial) of **retry**, **circuit breaking**, **rate limiting**, **timeouts**, and other distributed systems best practices.
  - **Authentication** and **Authorization**.
- **Libraries** are incredibly **painful to upgrade** (Think CVEs).

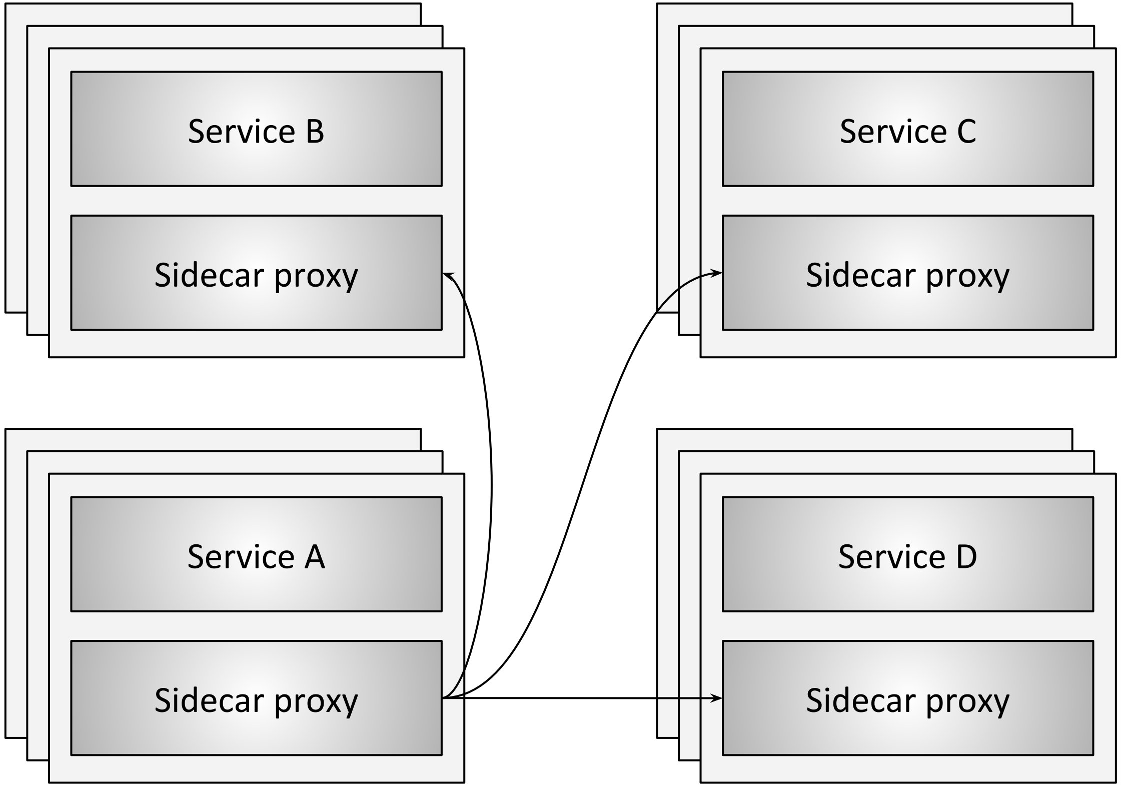
What is Envoy?

The network should be transparent to applications.

When network and application problems do occur it should be easy to determine the source of the problem.



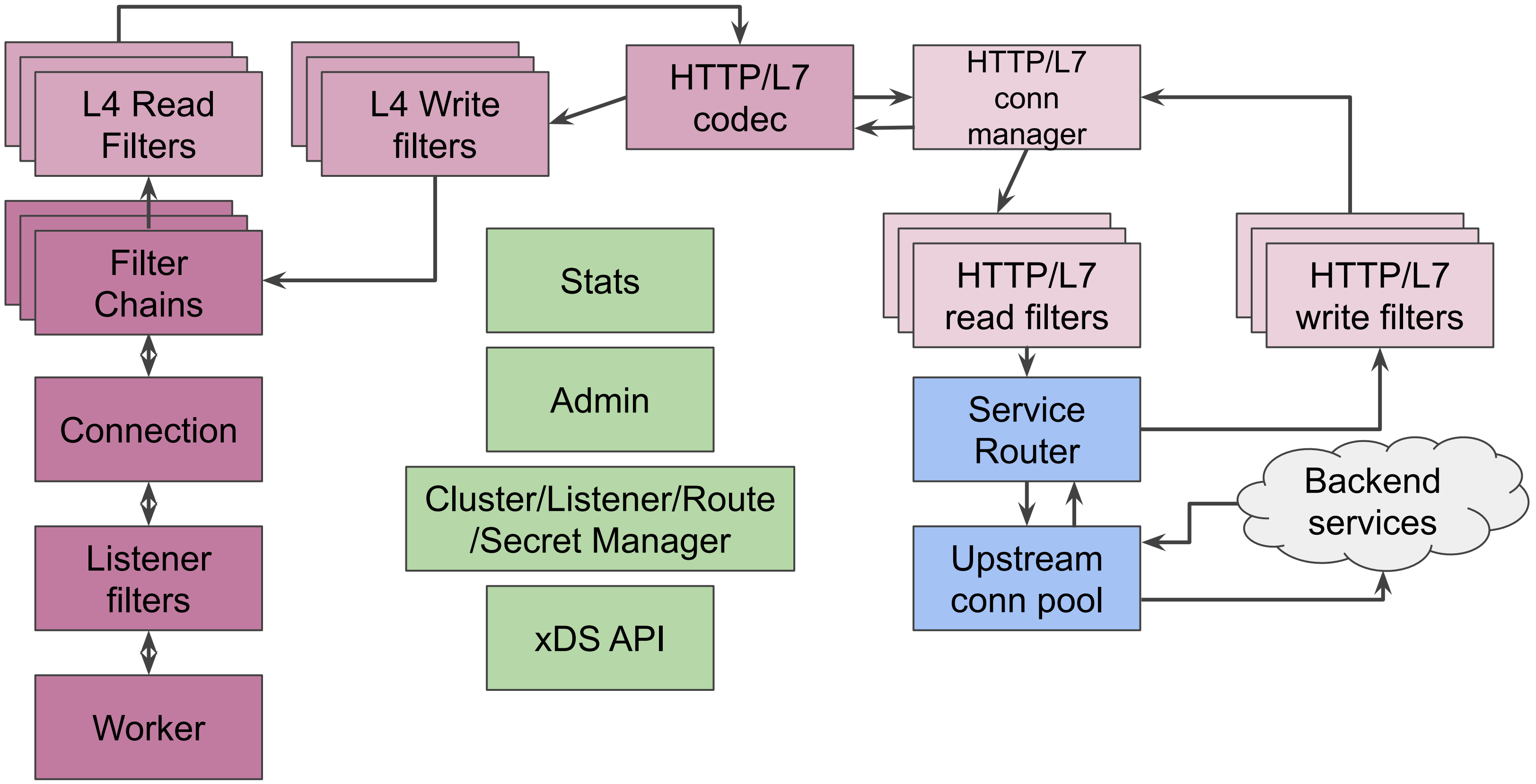
Service mesh refresher



Envoy design goals

- Out of process architecture
- High performance / low latency code base
- **L3/L4 filter architecture**
- **HTTP L7 filter architecture**
- HTTP/2 first
- Service discovery and active/passive health checking
- Advanced load balancing
- **Best in class observability** (stats, logging, and tracing)
- AuthN/AuthZ
- Edge proxy

Envoy architecture



Extension and pluggability

Envoy is designed to have multiple extension point. E.g.:

- L4/L7 filters
- Access loggers
- Tracers
- Health checkers
- Transport sockets
- Retry policy
- Resource monitors
- Stats sink

Envoy Config (xDS)

```yaml
static_resources:
  listeners:
  - address:
      socket_address:
        address: 0.0.0.0
        port_value: 8080
    filter_chains:
    - filters:
      - name: envoy.filters.network.http_connection_manager
        typed_config:
          "@type": type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionManager
          codec_type: auto
          stat_prefix: ingress_http
          route_config:
            name: local_route
            virtual_hosts:
            - name: backend
              domains:
              - "*"
              routes:
              - match:
                  prefix: "/service/1"
                route:
                  cluster: service1
              - match:
                  prefix: "/service/2"
                route:
                  cluster: service2
          http_filters:
          - name: envoy.filters.http.router
            typed_config: {}
```

Introduction to Envoy Data Plane API

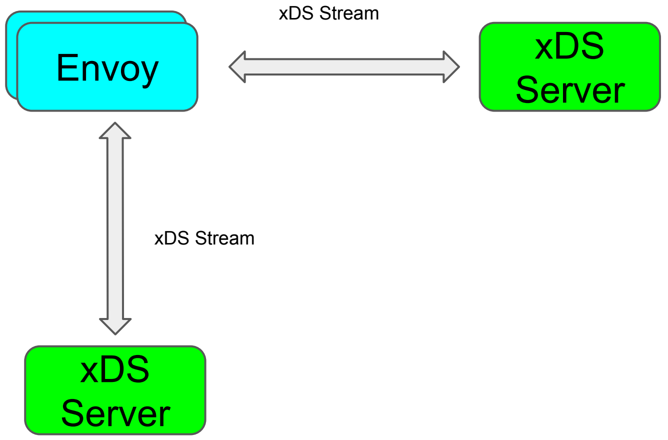
- The interface to manage Envoy
- We can pass the configurations to Envoy as JSON / YAML.
- These configurations illustrate as Protocol Buffers in Envoy.
- Optimized for machine generation.

What is xDS?

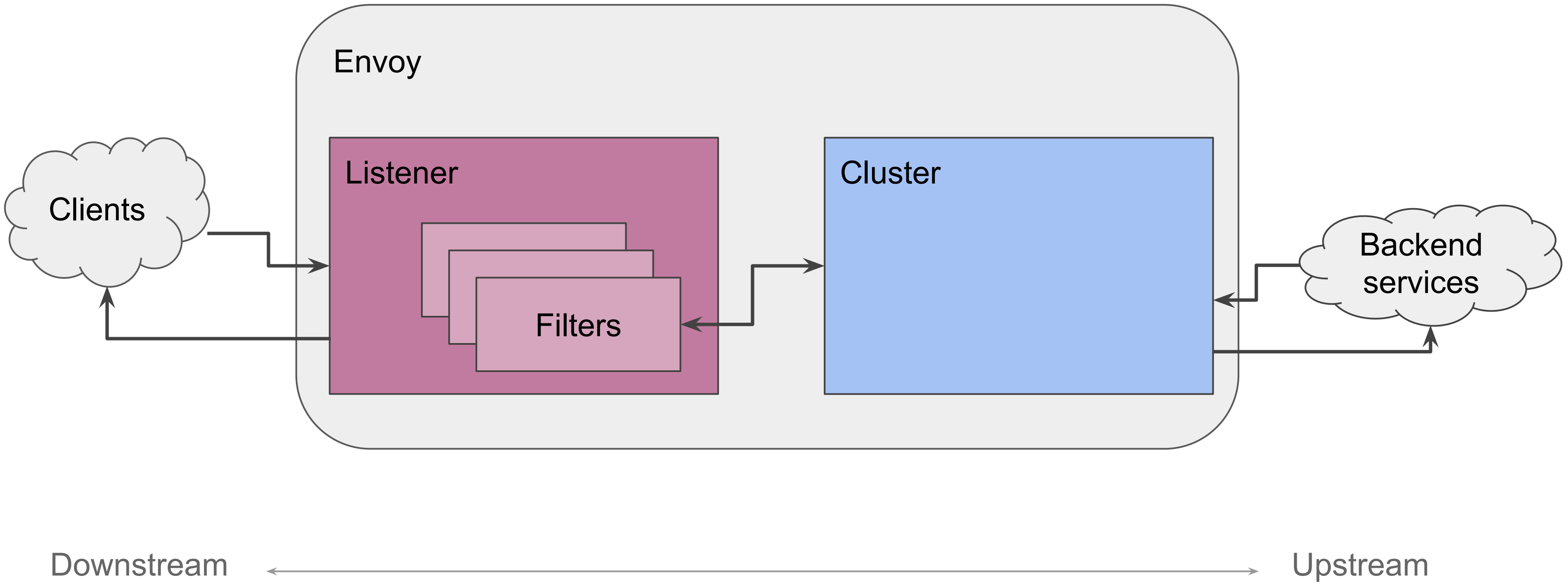
- The ability to update configurations of data plane without stopping.
- A characteristic functionality on Envoy.
- x Discovery Service
  - LDS (Listener Discovery Service)
  - CDS (Cluster Discovery Service)
  - EDS (Endpoint Discovery Service)
  - SDS (Secret Discovery Service)
  - RDS (Route Discovery Service)
  - RTDS (Runtime Discovery Service)
  - etc...

xDS Protocol

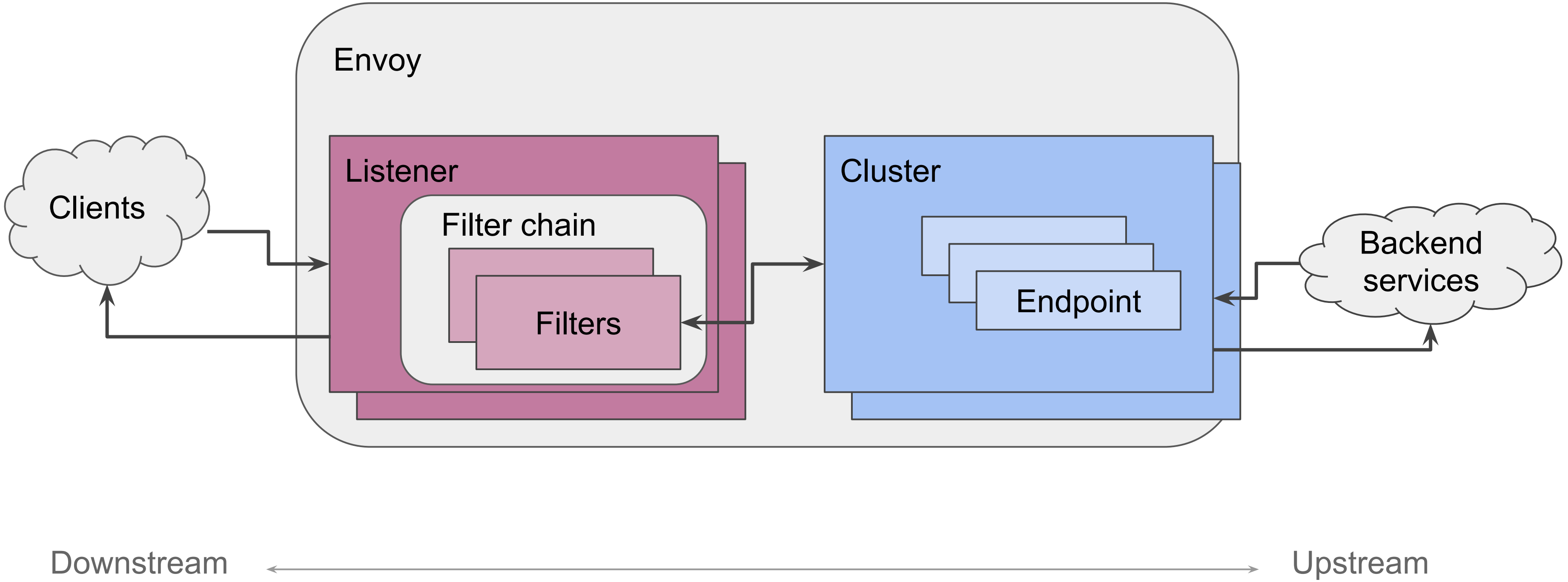
- gRPC / REST / File based protocol is used for xDS.
  - Especially, bidirectional gRPC streaming is applied for gRPC approach.
  - Polling is used for REST.



xDS Resources

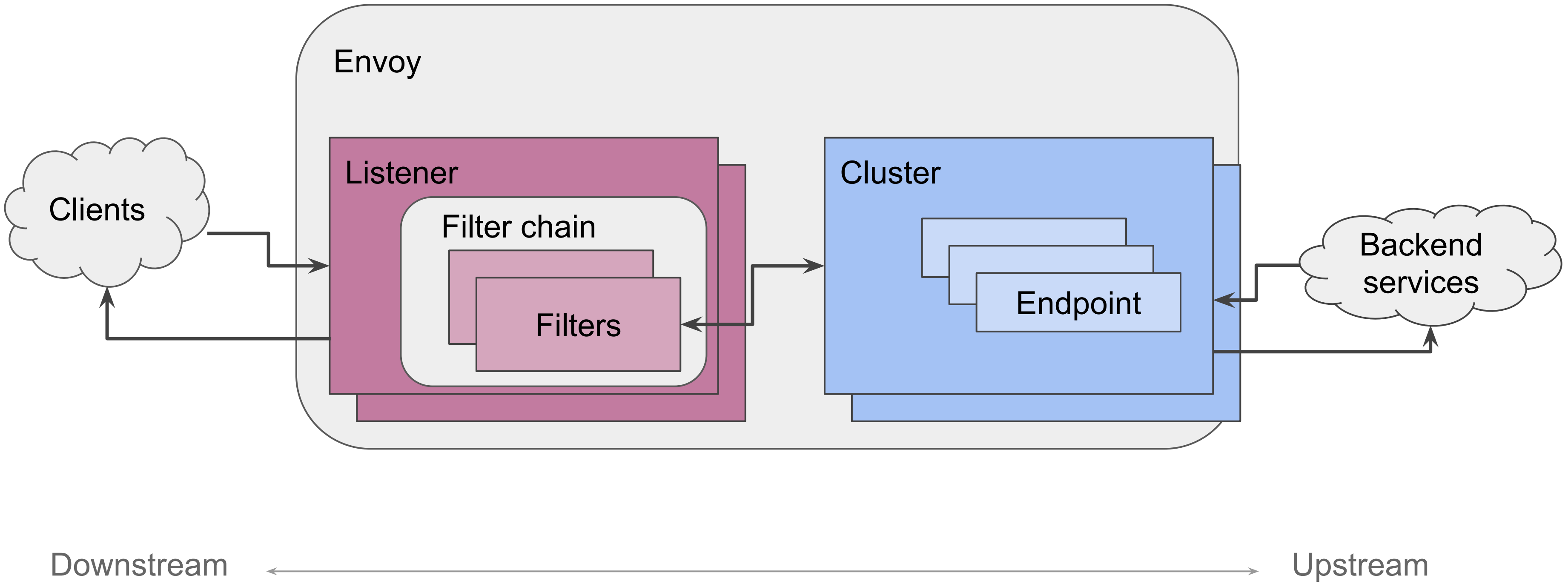


xDS Resources



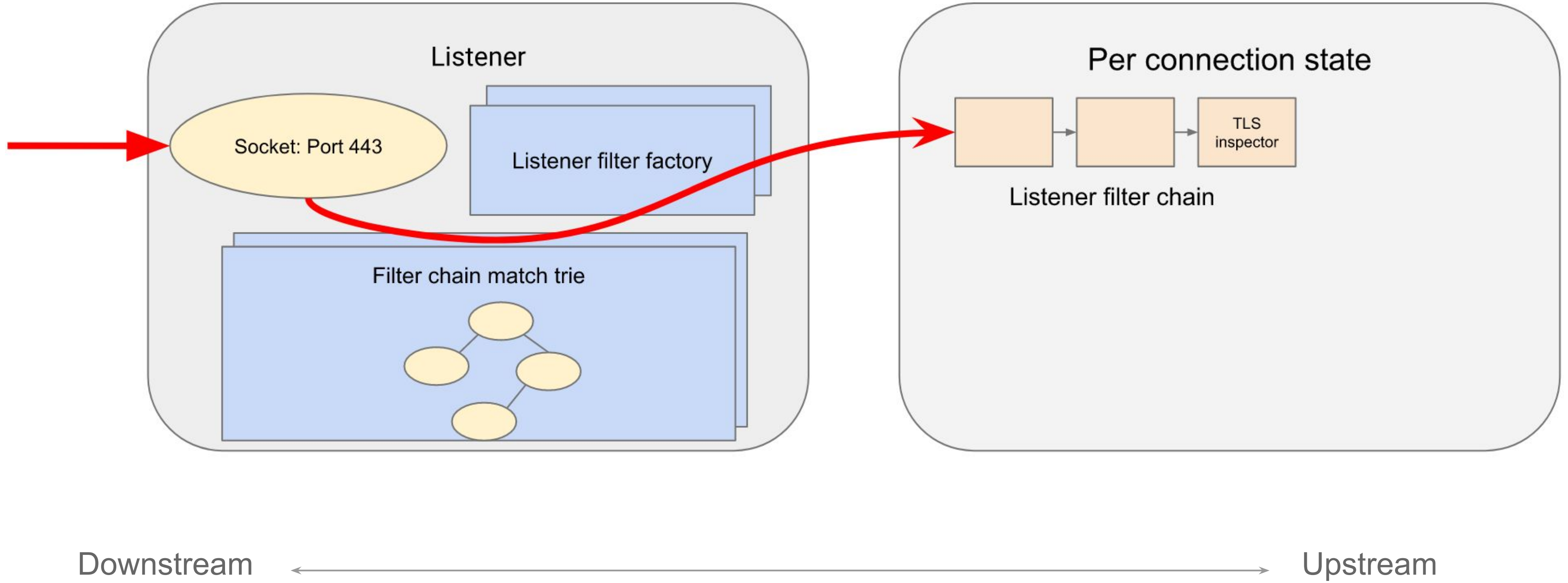
Using Envoy to Debug

Life of a Request

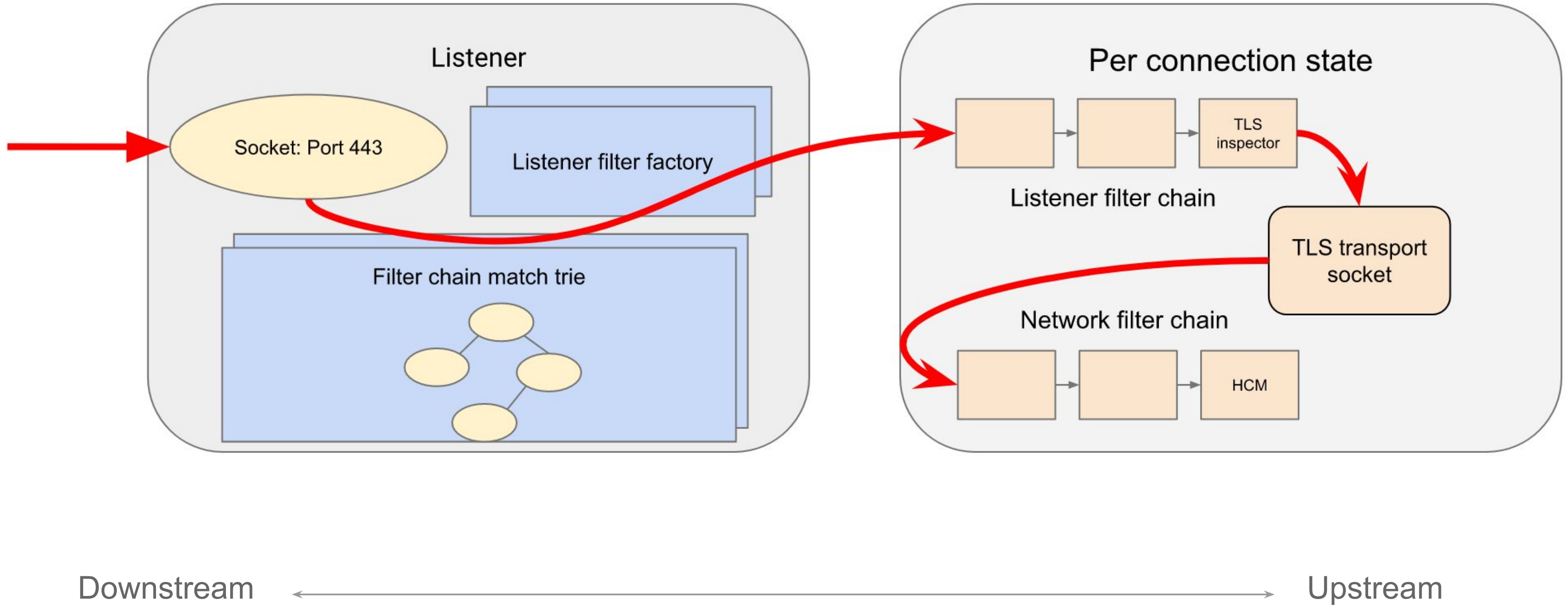


Life of a Request

tetrate

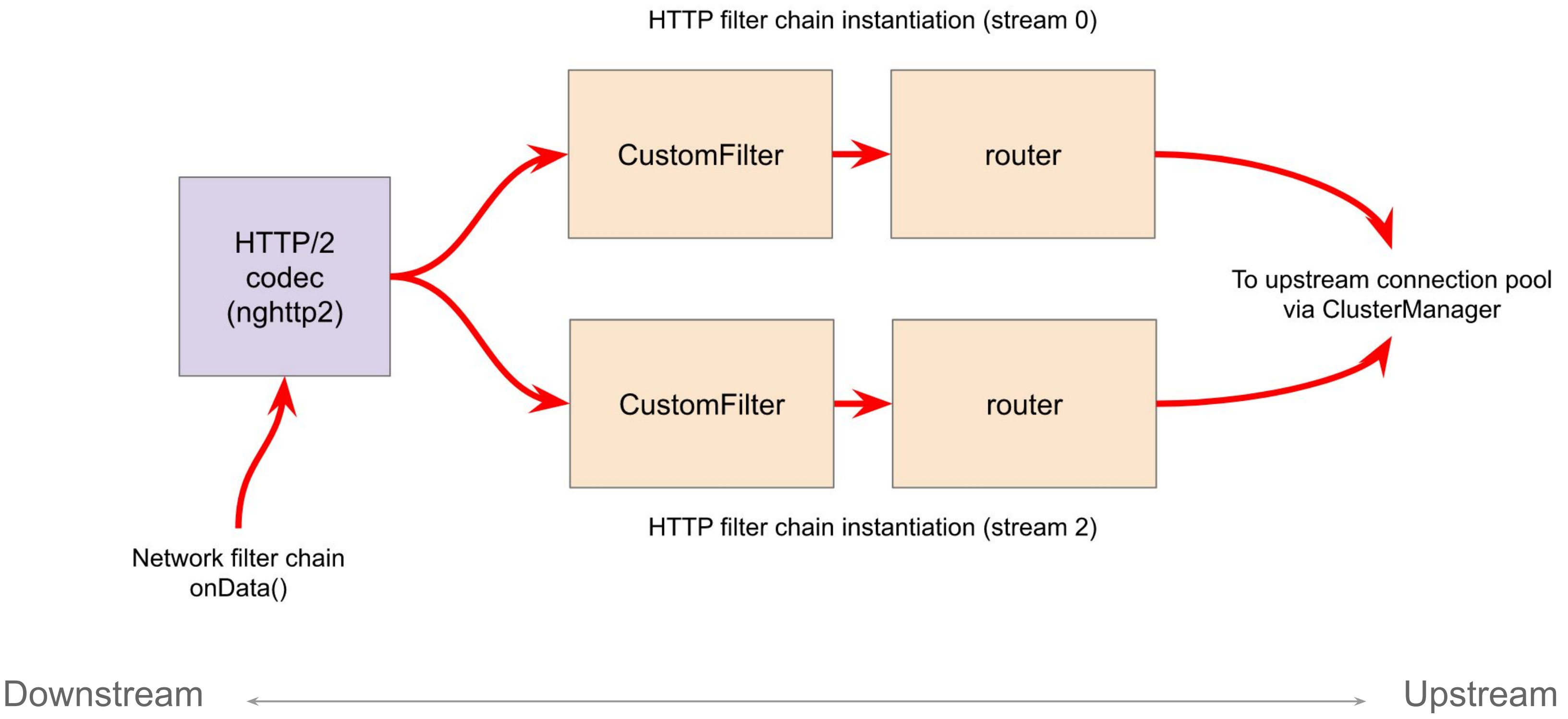


Life of a Request

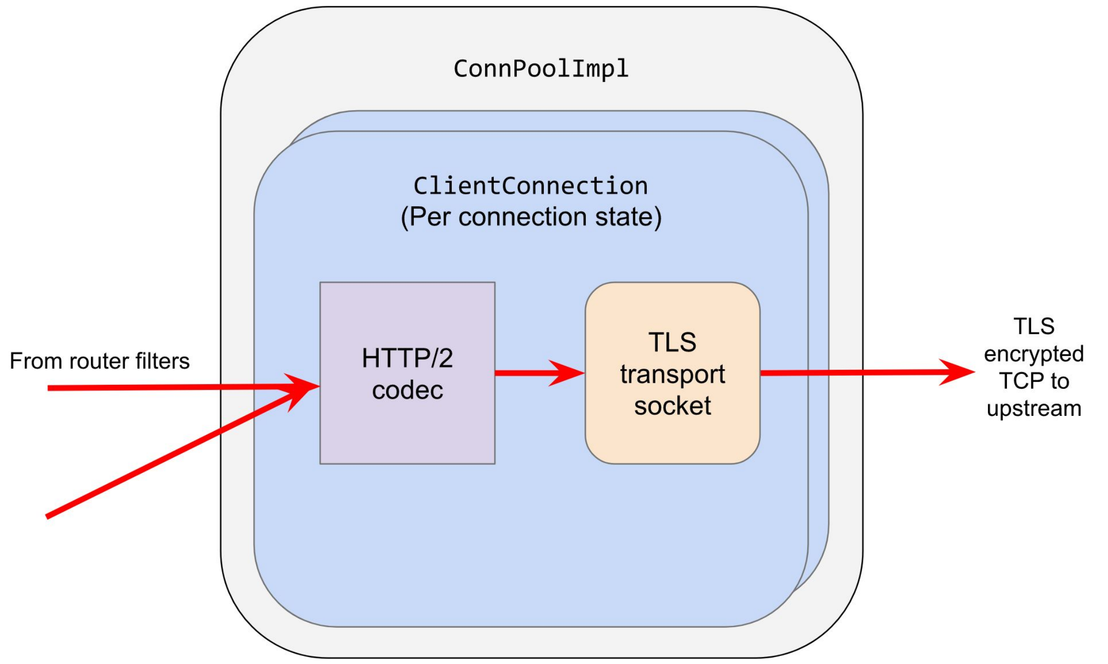


Life of a Request

tetrate


HTTP filter chain instantiation (stream 0)
CustomFilter
router
HTTP/2 codec (nghttp2)
To upstream connection pool via ClusterManager
CustomFilter
router
HTTP filter chain instantiation (stream 2)
Network filter chain onData()
Downstream
Upstream

Life of a Request



Downstream ⟷ Upstream

Debugging traffic

For TCP / HTTP / gRPC

- Access log
- Stats
- Debug log
- HTTP Tapping
- Transport socket tapping

Access log

- Can be logged by
  - Listener
  - HTTP Connection Manager
    - Router
  - TCP Proxy
  - ... (other extensions may implement)

Access log

tetrate

Default format:

```
[%START_TIME%] "%REQ(:METHOD)% %REQ(X-ENVOY-ORIGINAL-PATH?:PATH)%
%PROTOCOL%" %RESPONSE_CODE% %RESPONSE_FLAGS% %BYTES_RECEIVED%
%BYTES_SENT% %DURATION% %RESP(X-ENVOY-UPSTREAM-SERVICE-TIME)%
"%REQ(X-FORWARDED-FOR)%" "%REQ(USER-AGENT)%" "%REQ(X-REQUEST-ID)%"
"%REQ(:AUTHORITY)%" "%UPSTREAM_HOST%"
```

Access log

For better debuggability:

- **`%GRPC_STATUS%`**
  - HTTP status for gRPC is ALWAYS 200
- **`%UPSTREAM_TRANSPORT_FAILURE_REASON%`**
  - To debug upstream TLS connection error
    - Certificate expired
    - SAN validation failure

More:
https://www.envoyproxy.io/docs/envoy/latest/configuration/observability/access_log/usage

Stats

- Retrieve from admin endpoint

```
curl http://127.0.0.1:8001/stats
```

HTTP connection manager stats:

https://www.envoyproxy.io/docs/envoy/latest/configuration/http/http_conn_man/stats

Cluster stats

https://www.envoyproxy.io/docs/envoy/latest/configuration/upstream/cluster_manager/cluster_stats

Understanding stats

Important stats:

- `http.*.downstream_cx_*`
- `http.*.downstream_rq_*`
- `cluster.*.upstream_cx_*`
- `cluster.*.upstream_rq_*`
- `cluster.*.circuit_breakers.*`

Debug log

- NOT for PRODUCTION usage
- Enable via admin endpoint
- Dump all traffic in log (INCLUDE sensitive data, be careful)
- Most dangerous, most useful

Tapping

tetrate

Tapping

- Extract part of traffic for debug

HTTP Tap

Tap implemented as HTTP filter

- Advanced matching
- Streamed to admin endpoint
- Demo

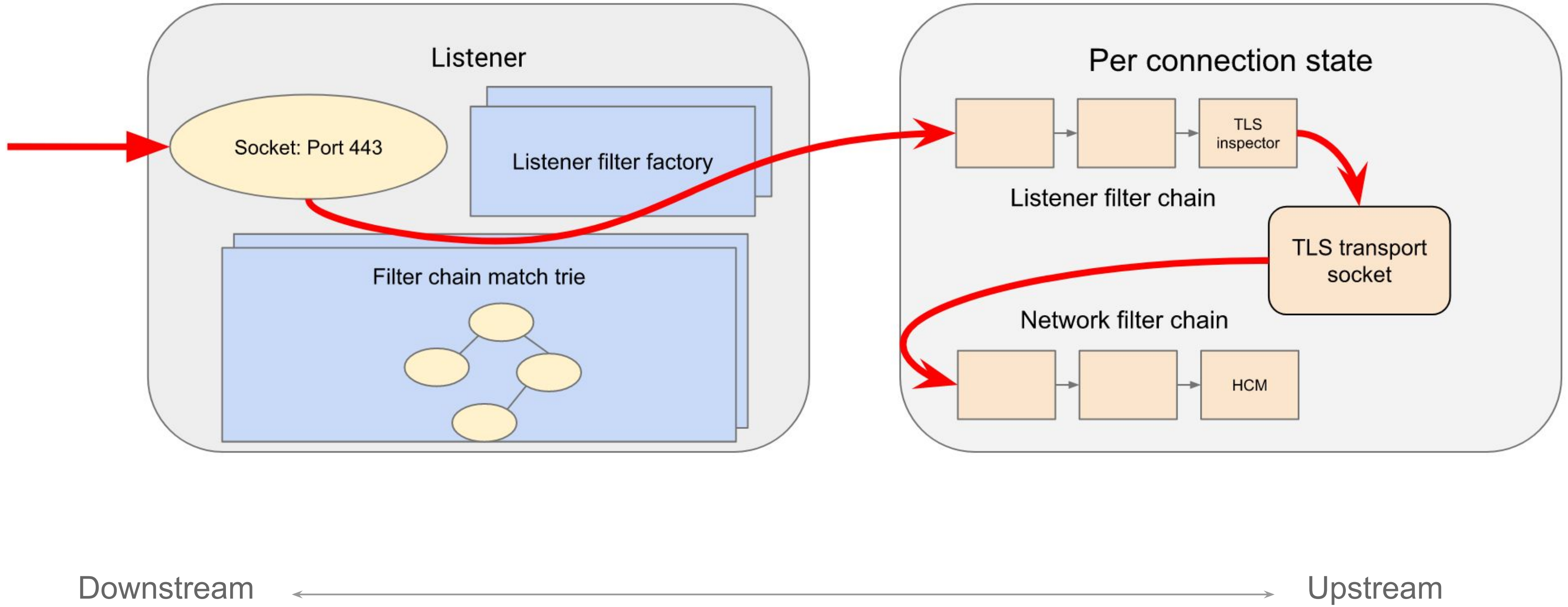
Tap transport socket

Transport socket

- Primarily used for TLS
- TAP is implemented to debug what is reading/writing from/to a socket

Life of a Request (recap)

tetrate



Demo

Q&A

tetrate

Thank you for coming!

WeChat/Twitter: zlizan

GitHub: lizan