ons

NORTH AMERICA

OPEN NETWORKING //
Enabling Collaborative
Development & Innovation

Cloud Native Network Tracing with Clovisor

Stephen Wong, FutureWei Technologies, Inc.

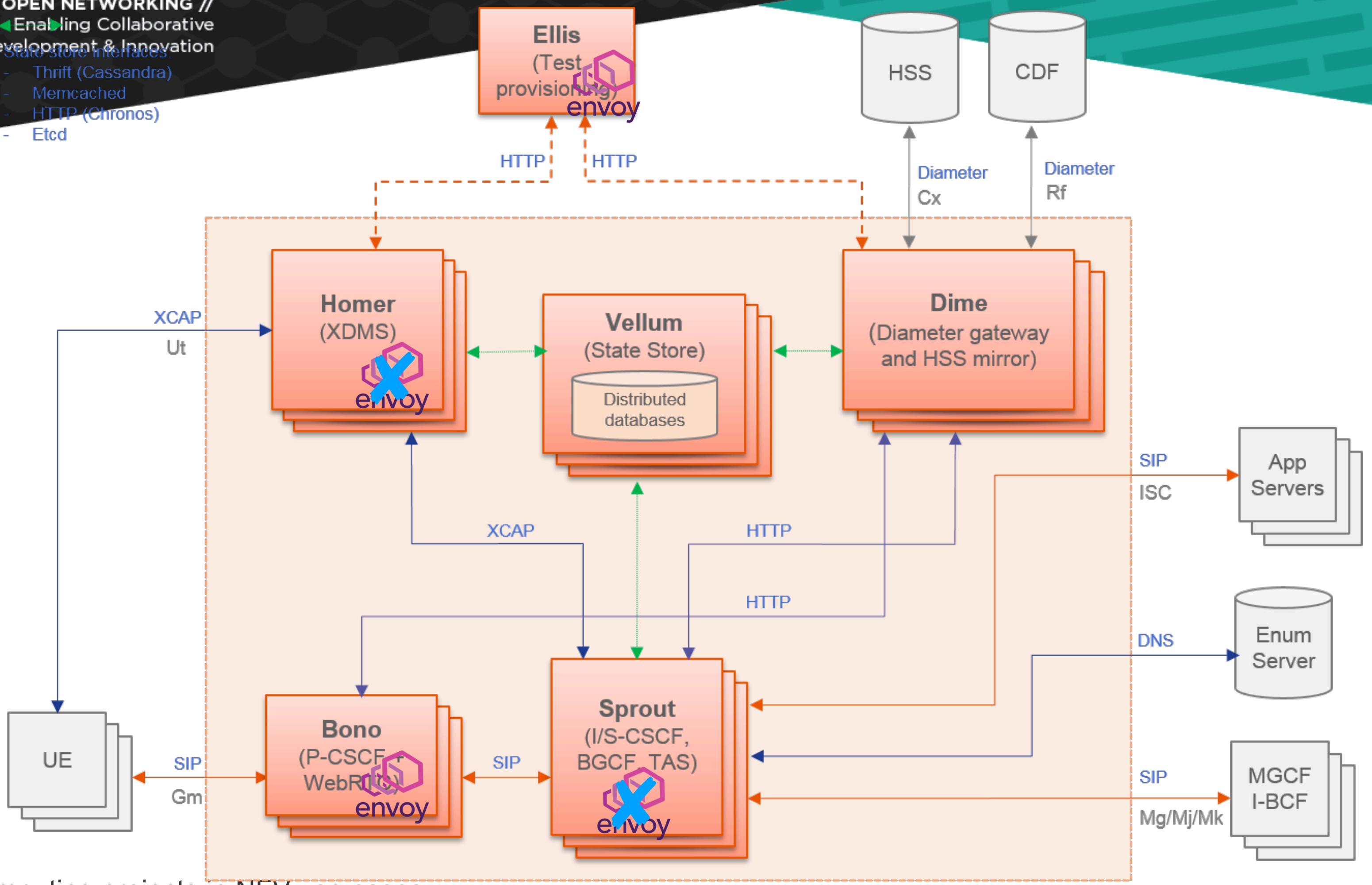
Hosted By
THE LINUX FOUNDATION | LF NETWORKING

Project Clover

1. Aims to integrate cloud native computing related projects for NFV use cases
    a. Integration point of cloud native projects is the application —- need to find a network function that is microservice-tized, k8s friendly
2. Since its inception, Clover has focused on examining evaluating Istio for service mesh orchestration for NFV control plane w.r.t. ease of operations and deployment
    a. Traffic management and policy control
    b. Visibility and Telemetry

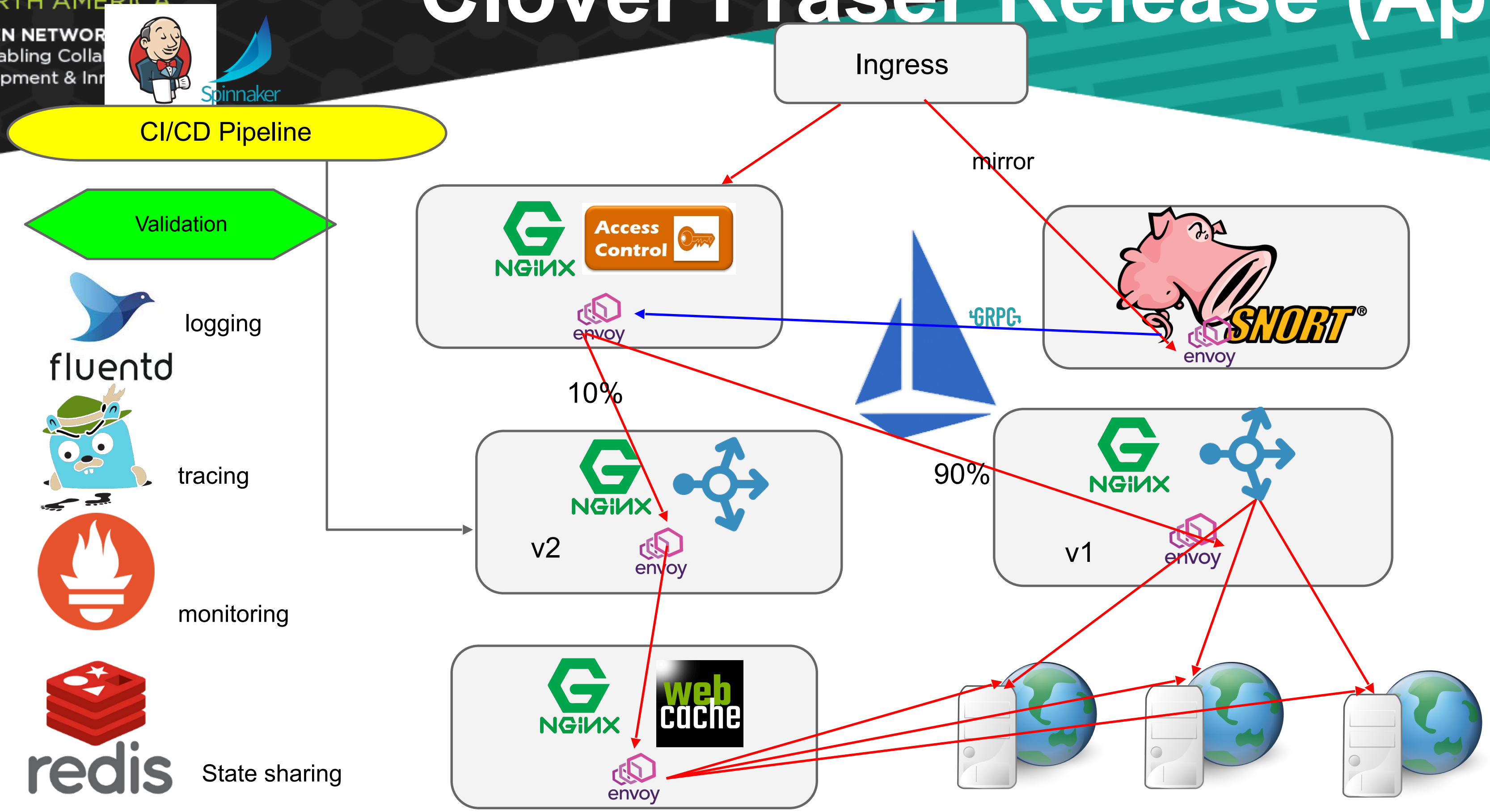
Original Intent for OPNFV Clover Fraser

State store interfaces:
- Thrift (Cassandra)
- Memcached
- HTTP (Chronos)
- Etcd



- Integrating cloud native computing projects in NFV use cases
- Originally wanted to utilize Clearwater and integrate it in Istio, failed:
  1. Istio / Envoy drops connections due to them being headless services
  2. Zero visibility into unsupported protocols

Clover Fraser Release (Apr’18)



1. Built a simple HTTP based network function to demonstrate running containerized NF on Istio (0.6)
2. Essentially built a simplistic example A-B testing with Istio route-rules and tracing data
3. The acceptance criteria were based on correctness and performance, both of these could be obtained just via tracing data
4. Implementation of the sample HTTP NF gave a good view of “perfect” app —- that is, the spans were correlated by this app preserving HTTP header

What were learned?

- Istio is great on many fronts — and for Clover, we found its visibility and telemetry to be essential
- Application network tracing: Envoy's network traces reveal more on application behaviors: it is transactional (request / response), reveals APIs (HTTP or gRPC), and the duration calculates the entire transaction
- Grouping spans into a single trace via request ID and trace ID —- full path of traversal of micro services on a particular request for the application

What more is needed?

- Istio's tracing data works best with HTTP. For NFV use cases, support to analyze / decode more network protocols, even potentially proprietary ones, will be critical
- Due to NFV common deployments, more raw networking info including more IP or TCP header fields

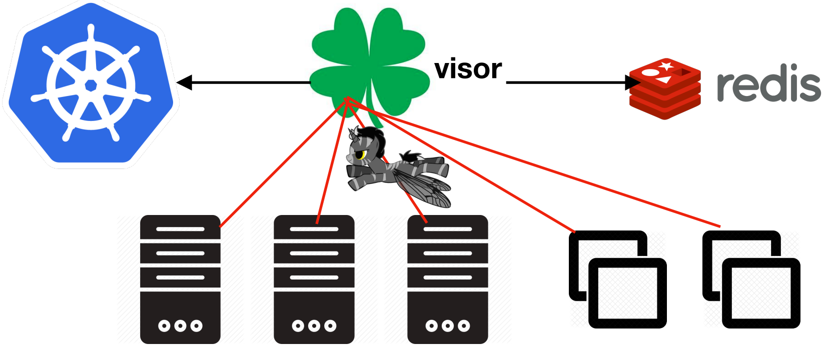
Clovisor: Clover's Network Tracing Module

1. Cloud Native:

a) Cloud Provider Independent
- Bare-metal servers, GKE, EKS…etc

b) CNI Plugin Agnostic
- All CNI plugins should work unless such plugin does kernel bypass

c) CPU Architecture Independent
- Any architecture supported by Linux (x86, ARM…etc), code currently tested with kernel versions 4.14 and 4.15



2. Design Principals:

a) Minimal Configurations
- Detect change in k8s cluster pod/service states

b) Minimal disruption to Packet Flow
- Utilizes eBPF to perform seamless integration, and will **NOT** modify traffic flow

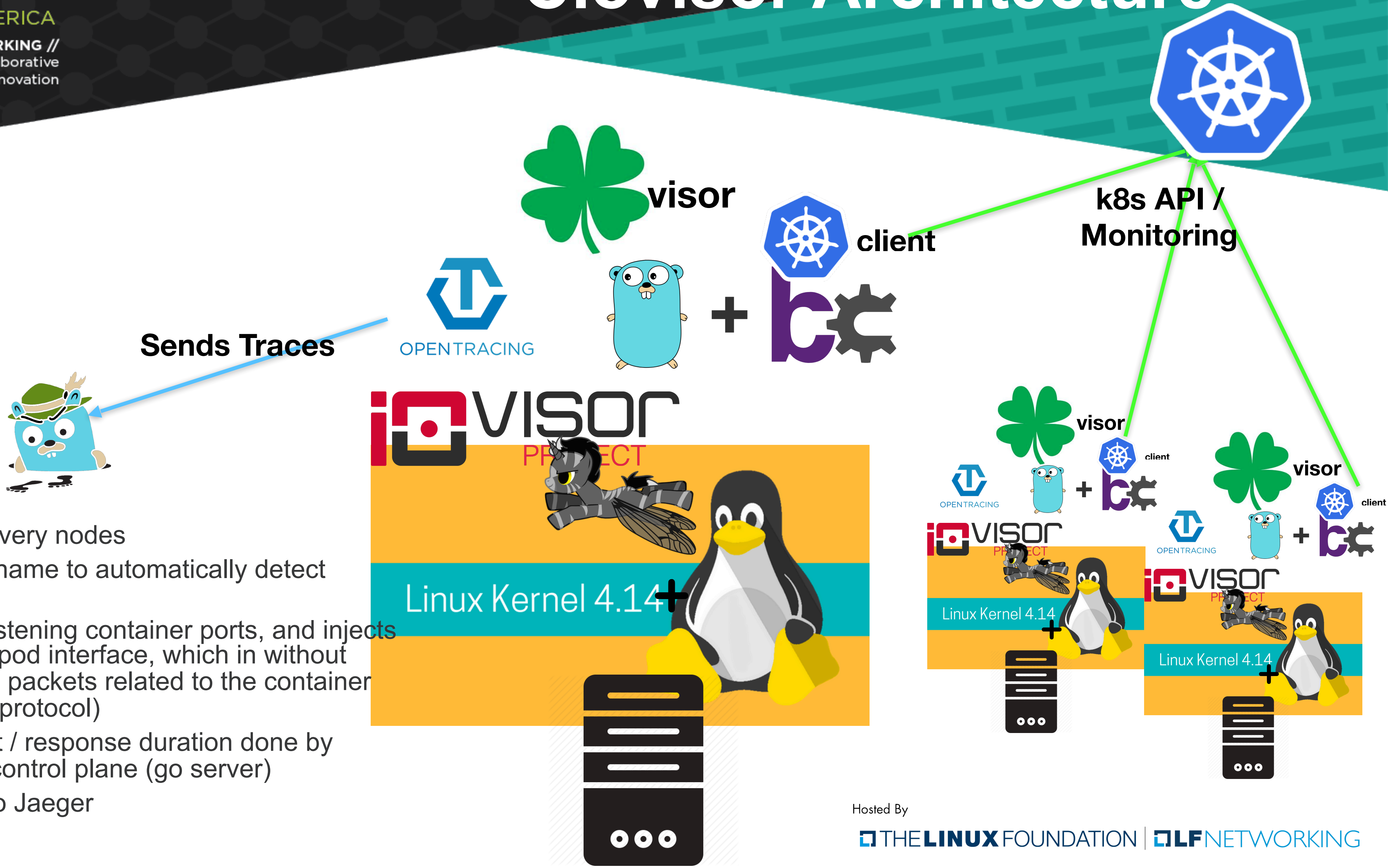
c) Scale-out Architecture
- DaemonSet —- linearly scale on each node in cluster

3. In-depth Integration with Cloud Native Ecosystem Projects:

a) Integrates with Kubernetes and OpenTracing -> Jaeger

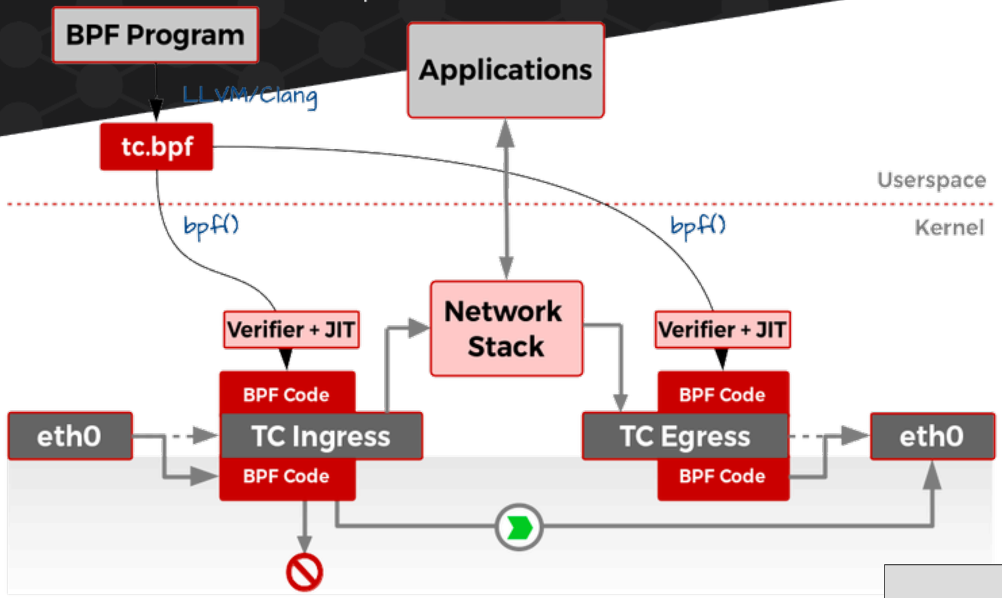
b) Future: use fluentd to collect logs (packet dump), and exposes metrics to Prometheus

Clovisor Architecture



1. DaemonSet: runs an instance on every nodes
2. With k8s client, reads service port name to automatically detect service port protocol
3. Detects pod creation, extracts all listening container ports, and injects BPF program on ingress/egress of pod interface, which in without configuration automatically trace all packets related to the container ports (service port name to specify protocol)
4. New session detection and request / response duration done by kernel, everything else is done on control plane (go server)
5. Traces sent via OpenTracing API to Jaeger

IOVisor / eBPF

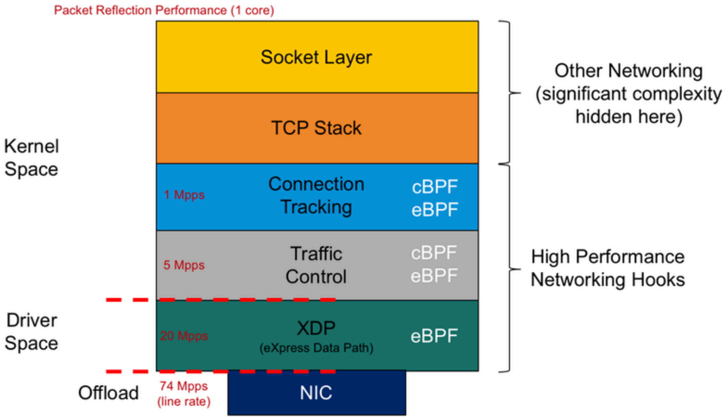


1.eBPF:

a) Inject bytecodes to kernel trace points / probes
  - Event driven model

b) Networking: tc
  - Utilizes Linux **tc** (traffic control) to inject bytecode on ingress and egress direction of a network interface

c) Verifier / JIT (just-in-time compiler)
  - Verifier ensures bytecode does NOT crash kernel

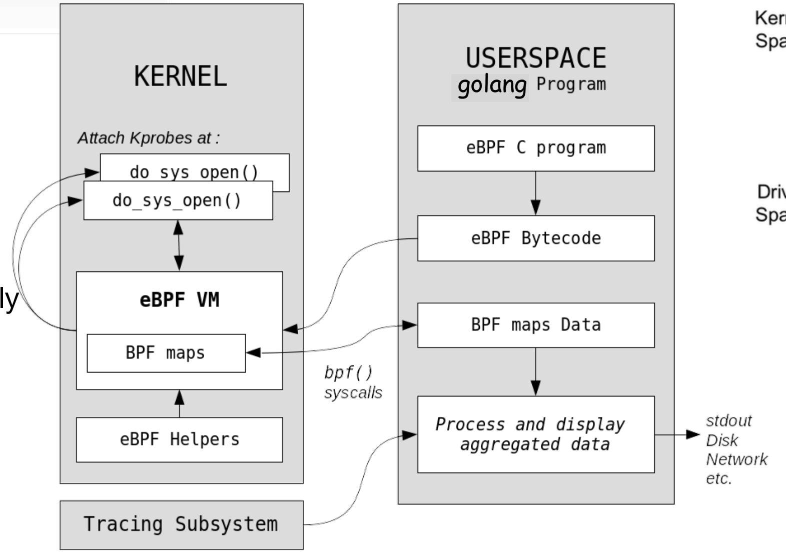


2.IOVisor bcc:

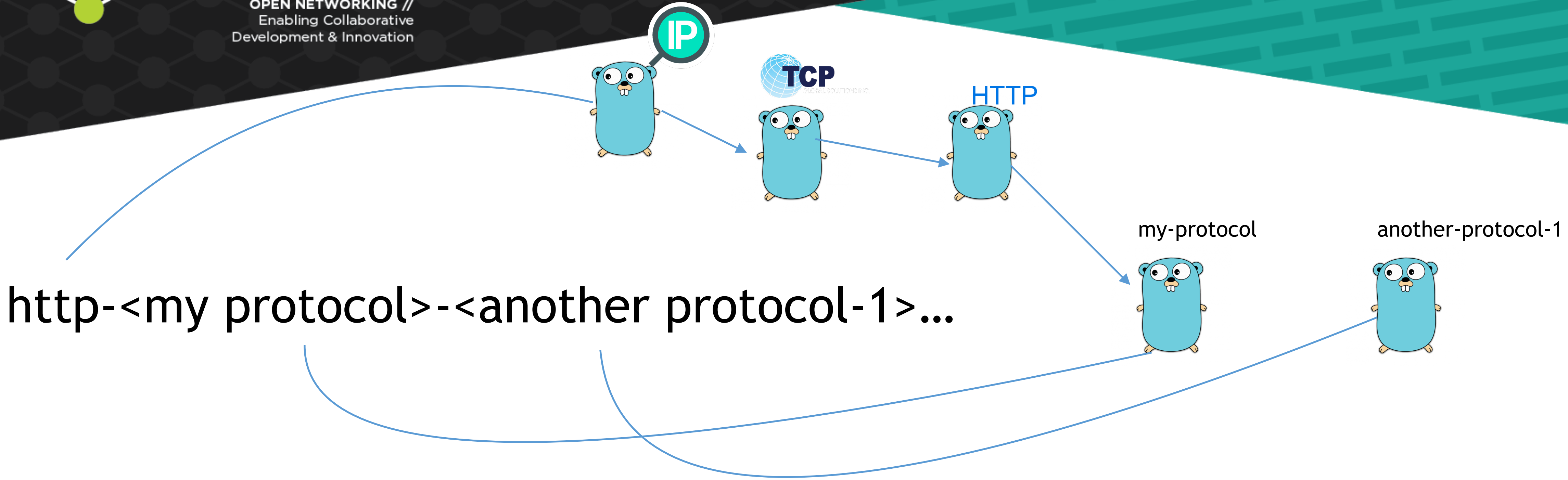
a) Ease of eBPF Development
  - Helper functions, kernel API wrappers…etc

b) Dynamic Validation and Compilation
  - Userspace eBPF code written in 'C' is dynamically verified (static analysis) and compiled

c) gobpf
  - Golang interface for userspace code —- much more performant than Python



Stackable Protocol Analyzer Plugin

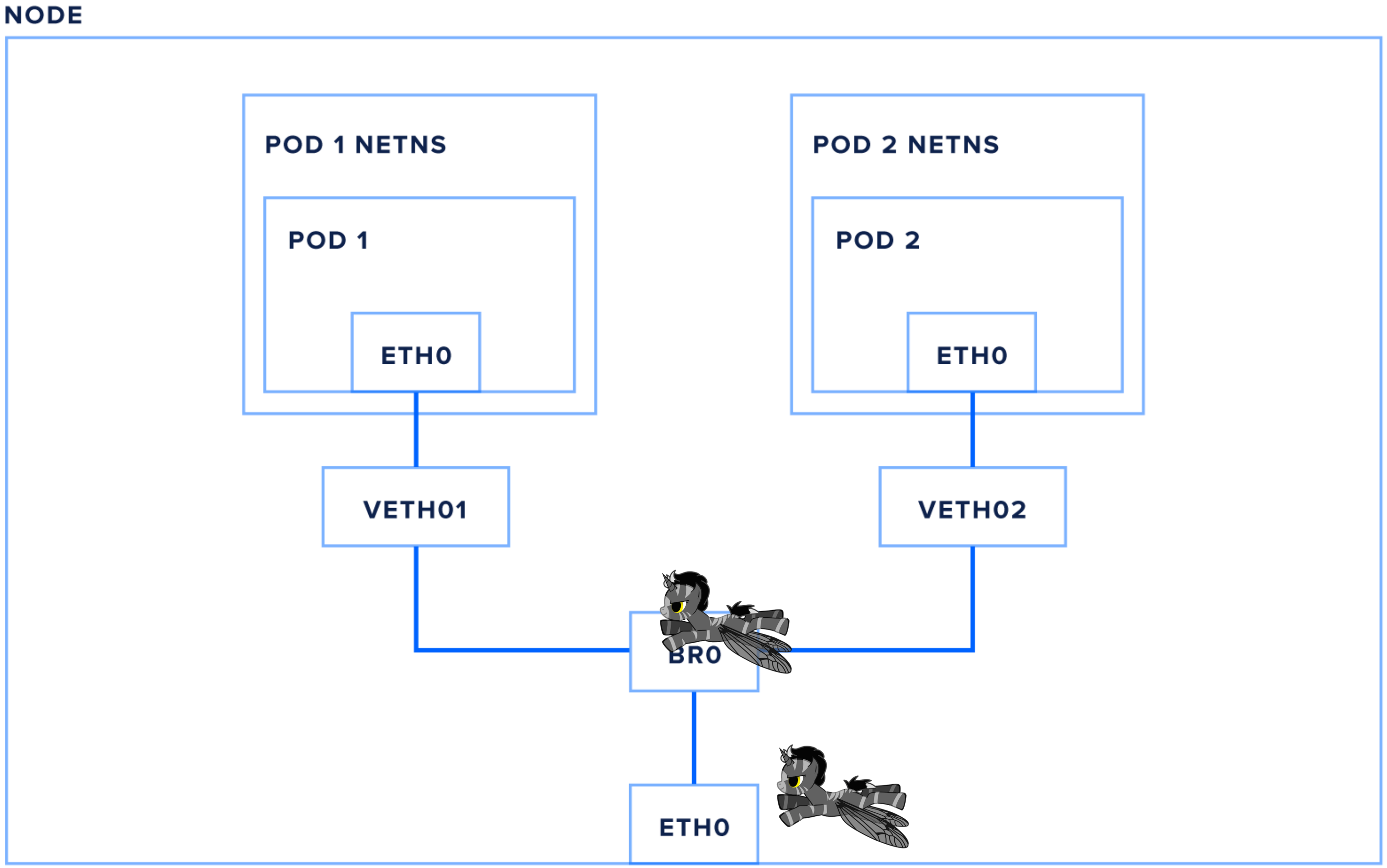


http-<my protocol>-<another protocol-1>...

- Clovisor supports a protocol stack inspired model to allow user to implement their own protocol analyzer plugin library
- Essentially user can extend Clovisor traces to also include her own proprietary protocol, or adding more fields to the trace for existing supported protocols
- It also allows extending information on IP/TCP/UDP layers

Node Interface Tracing

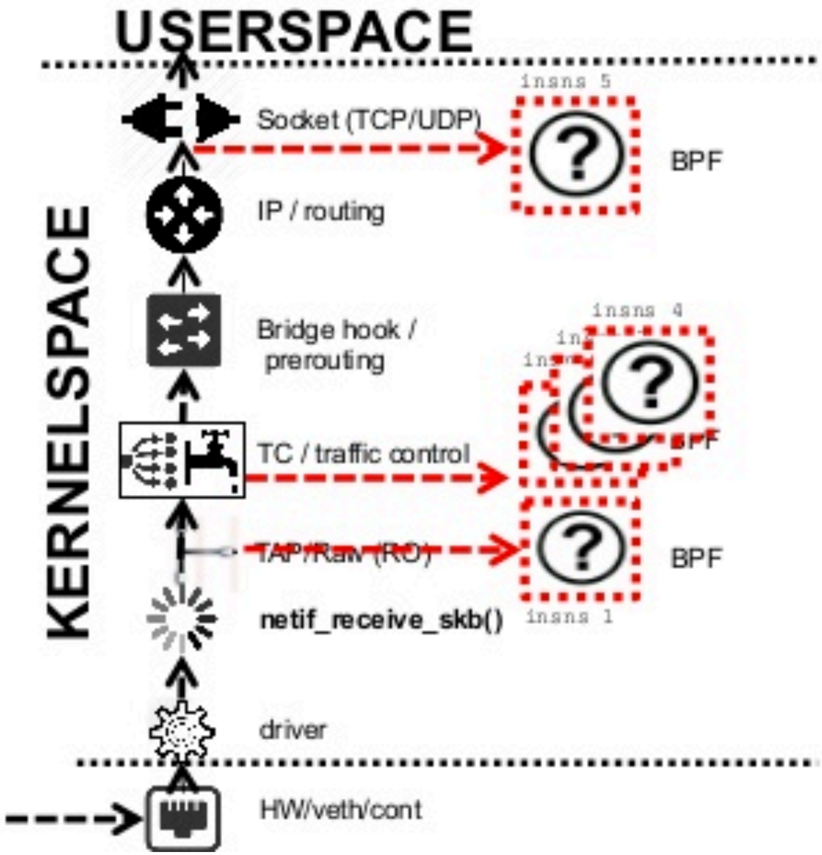
1. Clovisor offers a second point of tracing by tapping into node interfaces
2. Essentially Clovisor injects BPF programs to track pod sessions that are ingress or egress on the node interfaces
3. On NFV use cases, node interfaces are precious resources, the extra trace info from these interfaces can be used for rate limiting policies, providing insight on microservice utilization of bandwidth —- mapping all the way to application



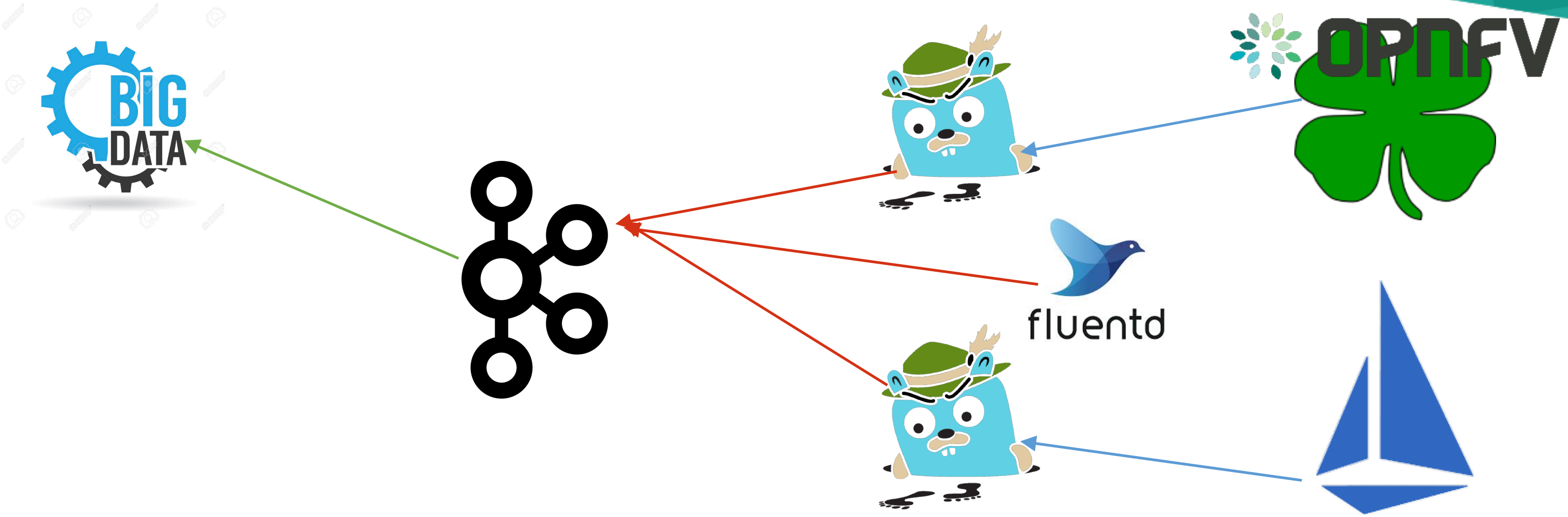
ons
NORTH AMERICA
OPEN NETWORKING //
Enabling Collaborative
Development & Innovation

Socket Visibility

- Various kprobes allow BPF developers to tag into different points of socket setup/teardown
- On various states of connections, on when datagram is sent, or when packet is being written / transferred
  1. kprobe__tcp_v[4|6]_connect / kretprobe__tcp_v[4|6]_connect
  2. kprobe__skb_copy_datagram_iter
  3. kprobe__tcp_sendmsg / kretprobe__tcp_sendmsg
- This is useful essentially for encryption cases
  1. Istio-auth as centralized authenticator, and Envoy to en/decrypt
  2. kTLS



Correlation Engine



- Since data comes from different sources, Clovisor needs a higher level analytic engine to correlate these trace datas into a single view
  1. Correlate Clovisor network traces with Istio/Envoy traces on HTTP sessions via the request-id and trace-id fields in HTTP header
  2. For non-HTTP, Clovisor needs different types of correlation. Possible: extract application (event) logs from fluentd, and run log analysis to correlate application events to (a) correlate Clovisor traces, and (b) correlates spans into a single trace

Summary

1. Clovisor is built to offer network tracing on cloud native application and can be used to augment Istio/Envoy tracing
2. Clovisor allows user defined protocol analysis over different protocols on protocol stack
3. Clovisor offers three points of visibility:
   a. Pod ingress / egress
   b. Node interfaces ingress / egress
   c. Application socket
4. Clovisor has built-in correlation engine to correlate traces with other data sources

If Interested, please join us on **clover-project**@slack

Thank You!!!

ons

NORTH AMERICA

OPEN NETWORKING //
Enabling Collaborative
Development & Innovation