Getting back what was lost in the era of high-speed software packet processing

Marcelo Abranches[1], Oliver Michel[2] and Eric Keller[1]

(1) University of Colorado, Boulder. (2) Princeton University

Hotnets, 2022. Austin, TX.

Why should we care about Linux Networking?

- Kernel functions
  - Firewall, forwarding
- Management tools
  - Iproute2, iptables, …
- Control Plane
  - FRR, StrongSwan

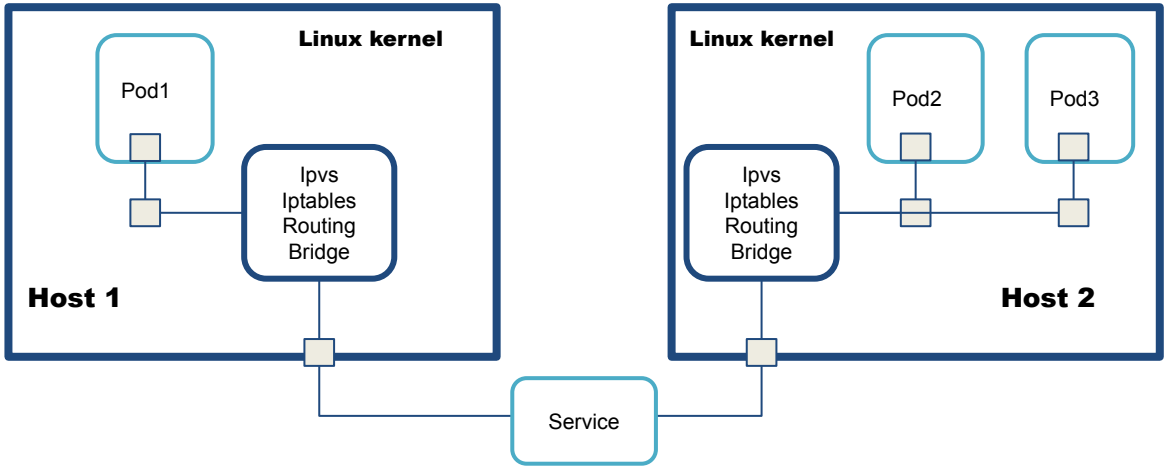
Linux Networking in Practice

Example: Deployment with 3 Pods (containers) and a Service

kubernetes

>kubectl apply -f ./my-manifest.yaml

```
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: hostname-101-deployment
spec:
  replicas: 3
  selector:
    # Like saying "Make sure there are three pods running
    # with the label app = hostname and version = v101"
    matchLabels:
      app: hostname
      version: v101
  template:
    metadata:
      labels:
        # The 'app' label is used by both the service
        # and the deployment to select the pods they operate on.
        app: hostname
        # The 'version' label is used only by the deployment
        # to control replication.
        version: v101
    spec:
      containers:
        - name: nginx-hostname
          image: kubegoldenguide/nginx-hostname:1.0.1
          ports:
            - containerPort: 80
```



Linux Networking is Rich

Linux Networking is Rich

But, Linux Networking is Slow

Alternate Pipelines Emerged

Bypass Linux networking to gain performance

Kernel Bypass

- DPDK

In-kernel Network Bypass

- XDP/eBPF

Alternate Pipelines Emerged

Bypass Linux networking to gain performance

Kernel Bypass

- DPDK

In-kernel Network Bypass

- XDP/eBPF

Lose Linux's ecosystem

Need to reimplement services

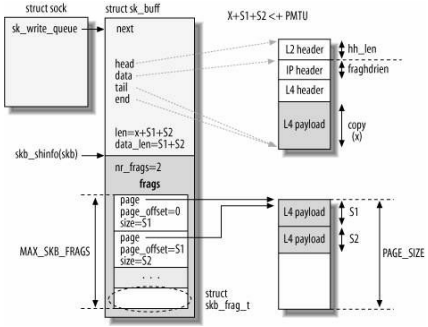
Can we just make Linux network faster?

Why is Linux Slow?

Because it’s general

Why is Linux Slow?

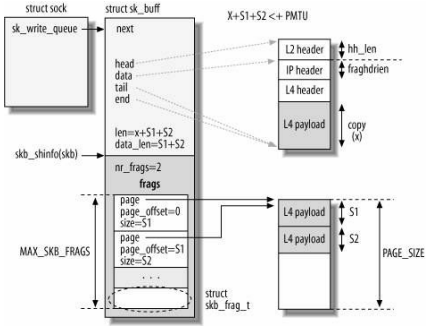
Because it's general



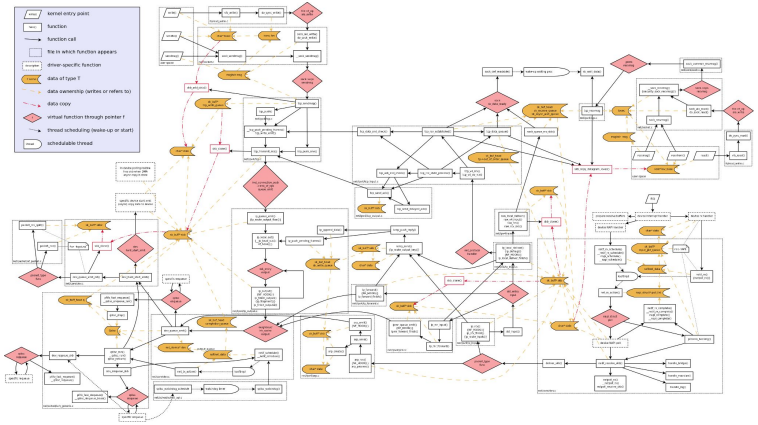
Parsing, allocating and populating complex data structures (e.g., skb)

Why is Linux Slow?

Because it's general



Parsing, allocating and populating complex data structures (e.g., skb)



Long critical path of all functions and corner cases

Insights for Redesigning Linux

1. Linux mixes fast / slow path processing in a single data path
   - Instead, instantiate fast path in a specialized execution environment

1. Not everything is needed all the time
   - Instead, compose a minimal data plane automatically based only what is needed

Is this redesign even possible today?

What we need is a fast-path execution environment that …

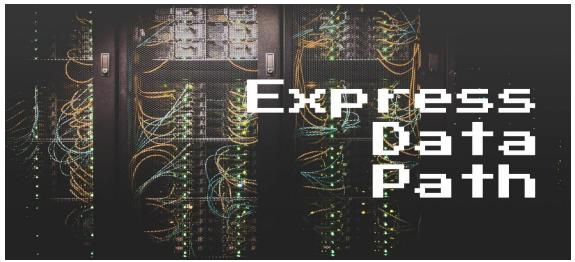
1. Is efficient
2. Enables secure/dynamic code injection
3. Enables interaction with Linux

What we need is a fast-path execution environment that …

1. Is efficient

2. Enables secure/dynamic code injection

3. Enables interaction with Linux

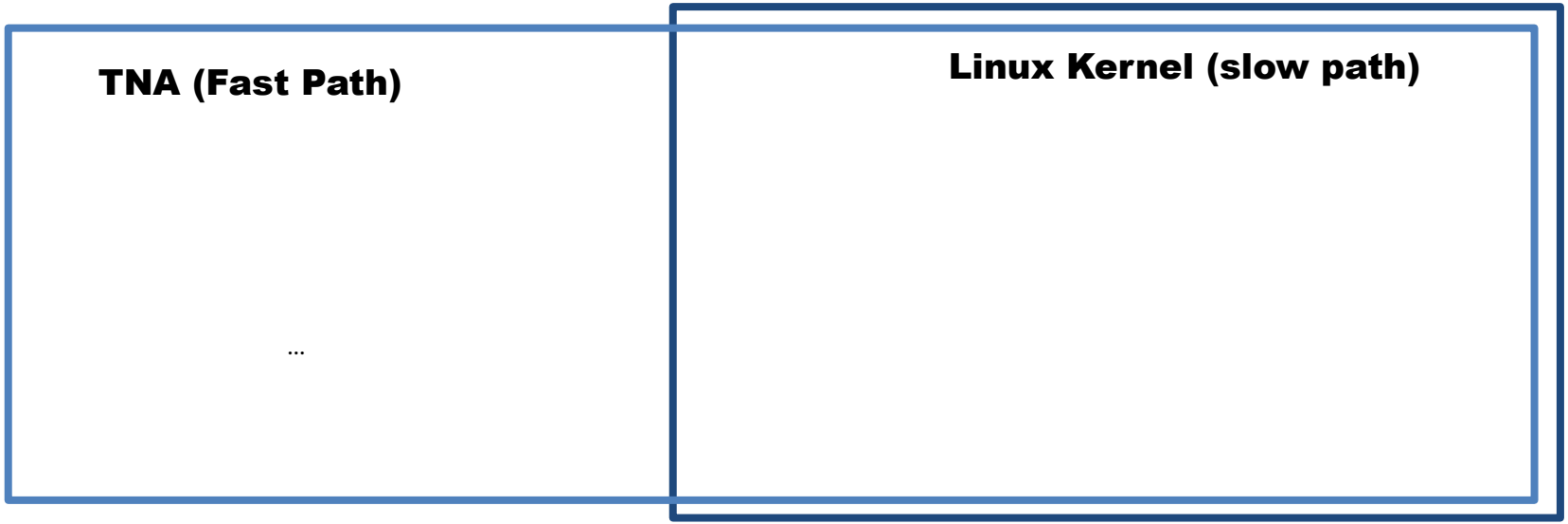


Three challenges remain

1. Break down Linux network processing
   - Fast and slow path
2. Make this redesign transparent to the rest of the system
   - Leverage Linux's ecosystem
3. Dynamically create a fast path
   - Light and supports what is configured

Introducing Transparent Network Acceleration (TNA)

Break down Linux network processing



Break down Linux network processing

Linux provides completeness of processing

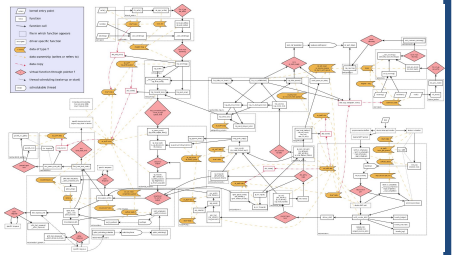
TNA (Fast Path)

...

Linux Kernel (slow path)

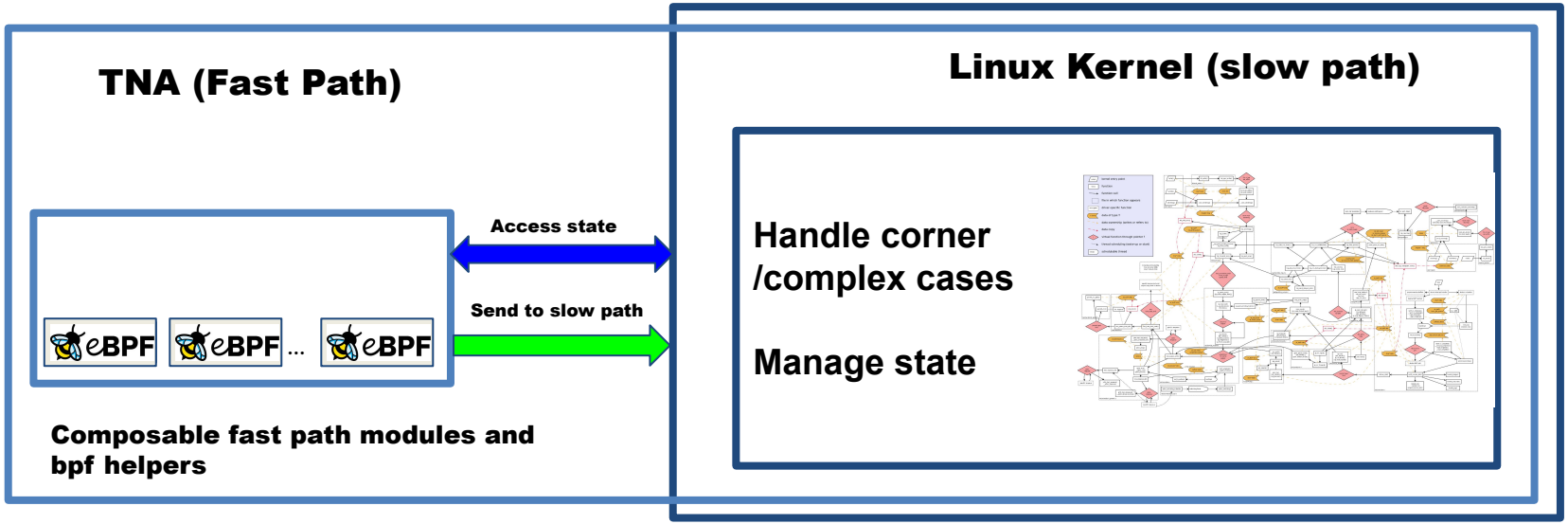
Handle corner /complex cases

Manage state



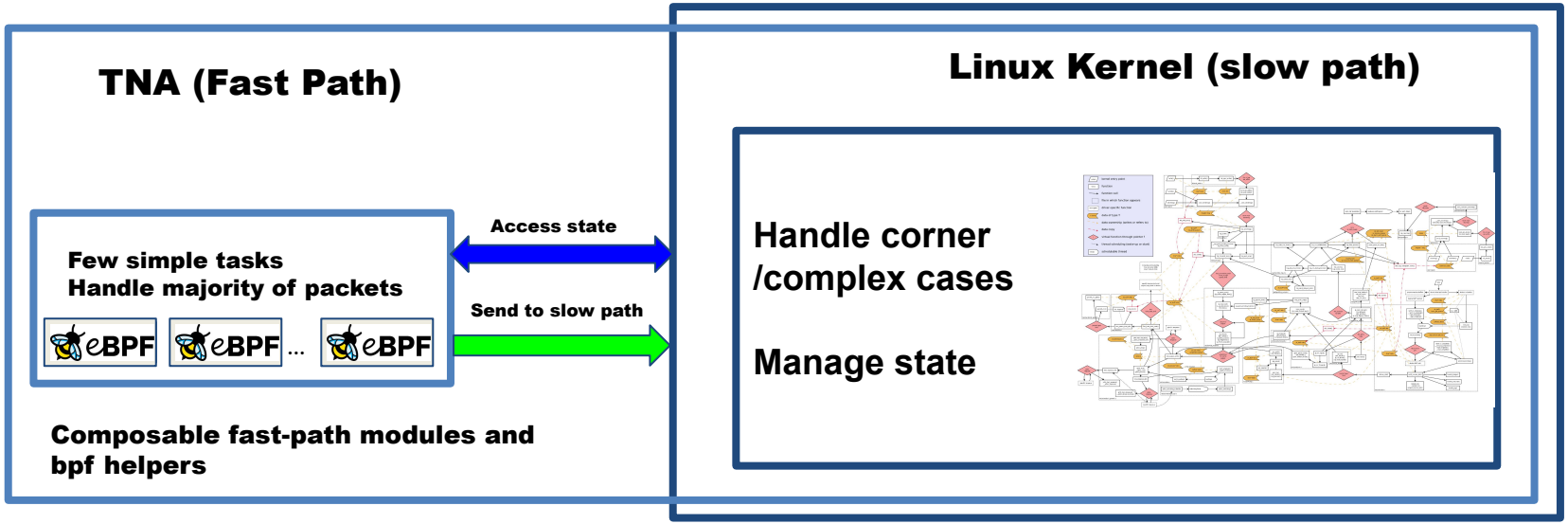
Break down Linux network processing

Linux provides completeness of processing



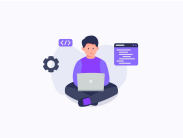
Break down Linux network processing

Linux provides completeness of processing
TNA allows processing common case packets with minimal overheads



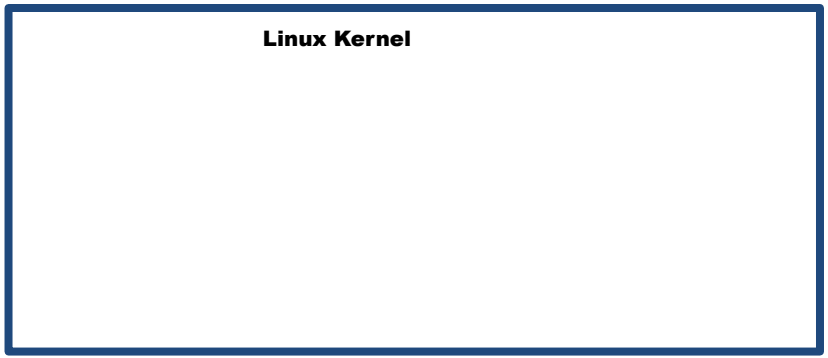
Make this redesign transparent to the rest of the system

User (or tool) configures Linux network



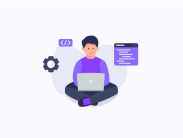
ANSIBLE

kubernetes



Make this redesign transparent to the rest of the system

User (or tool) configures Linux network

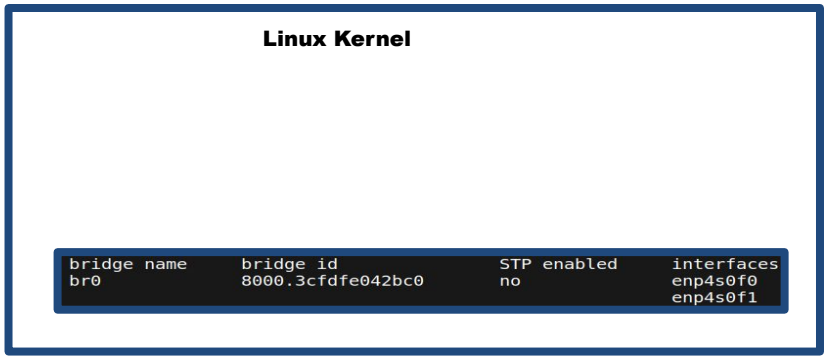


ANSIBLE

kubernetes

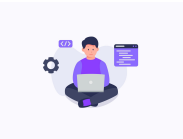
Create a bridge

```
>brctl addbr br0
>brctl addif br0 enp4s0f0
>brctl addif br0 enp4s0f1
```



Make this redesign transparent to the rest of the system

User (or tool) configures Linux network

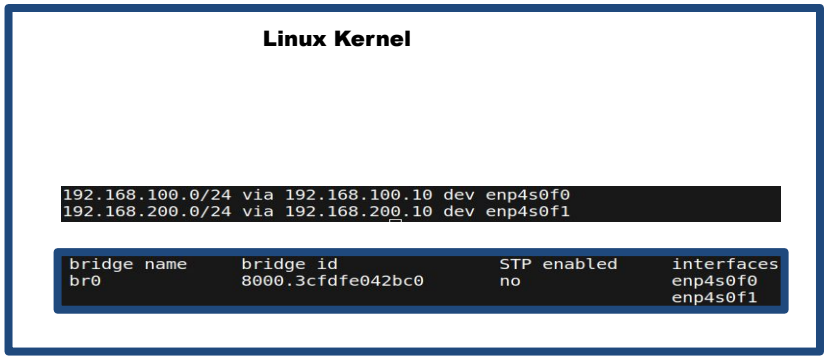


ANSIBLE

kubernetes

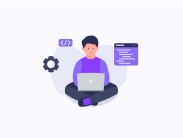
Manipulate routes

>ip route add 192.168.200.0/24 via 192.168.200.10
>ip route add 192.168.100.0/24 via 192.168.100.10



Make this redesign transparent to the rest of the system

User (or tool) configures Linux network



ANSIBLE

kubernetes

Add filtering rules

>iptables -d 192.168.100.100 -A FORWARD -j DROP
>iptables -d 192.168.200.100 -A FORWARD -j DROP

Linux Kernel

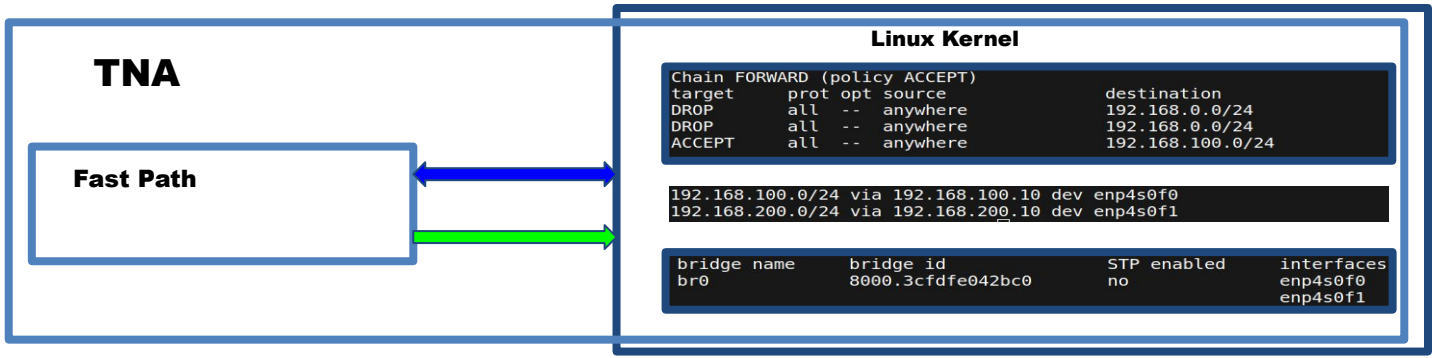
```
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             192.168.0.0/24
DROP       all  --  anywhere             192.168.0.0/24
ACCEPT     all  --  anywhere             192.168.100.0/24
```

```
192.168.100.0/24 via 192.168.100.10 dev enp4s0f0
192.168.200.0/24 via 192.168.200.10 dev enp4s0f1
```

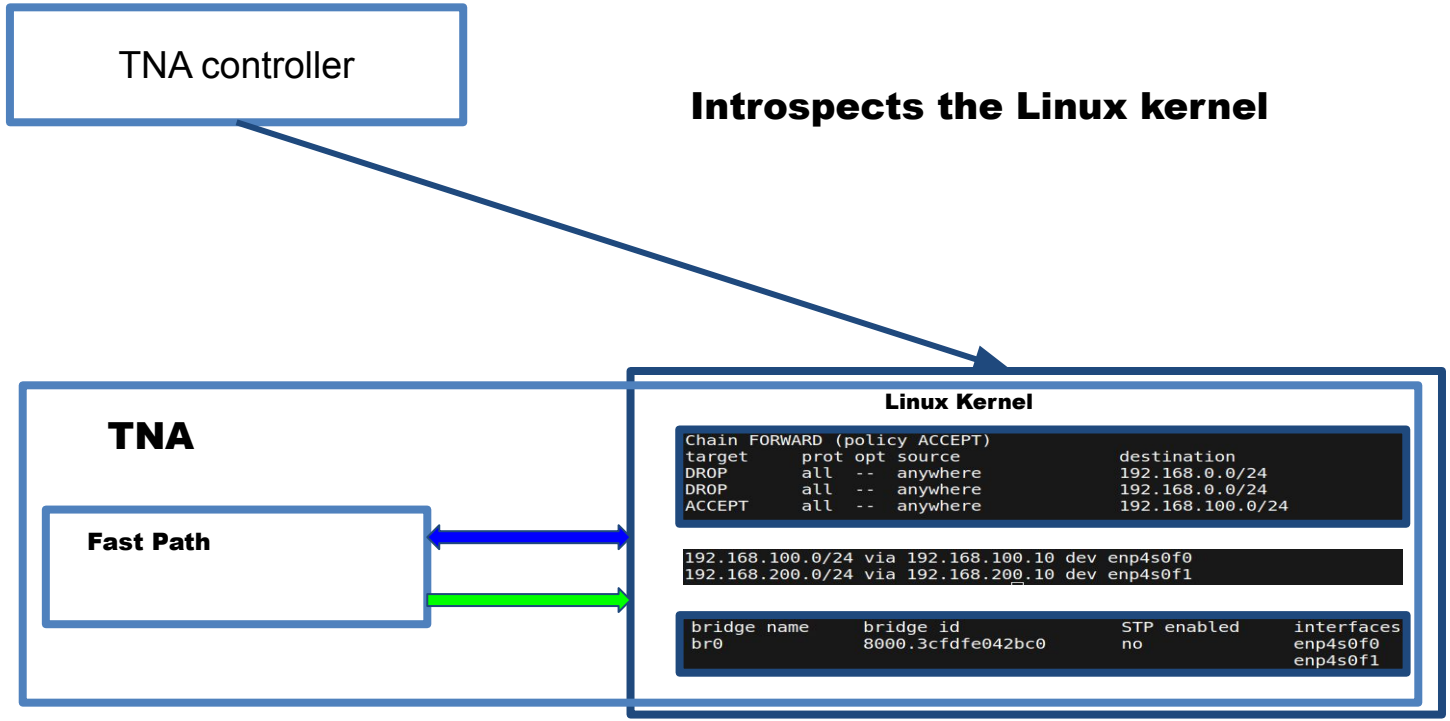
```
bridge name     bridge id               STP enabled     interfaces
br0             8000.3cfdfe042bc0       no              enp4s0f0
                                                        enp4s0f1
```

Dynamically create a fast path

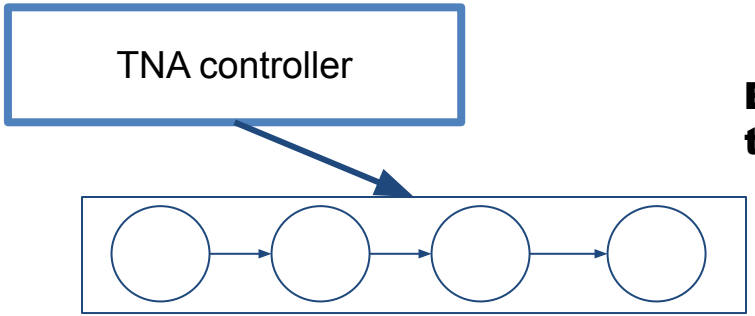
TNA controller



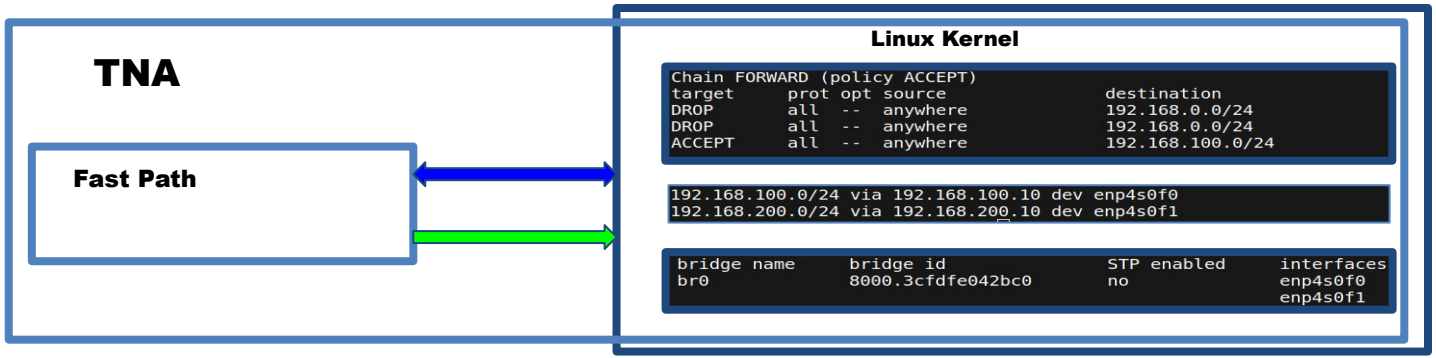
Dynamically create a fast path



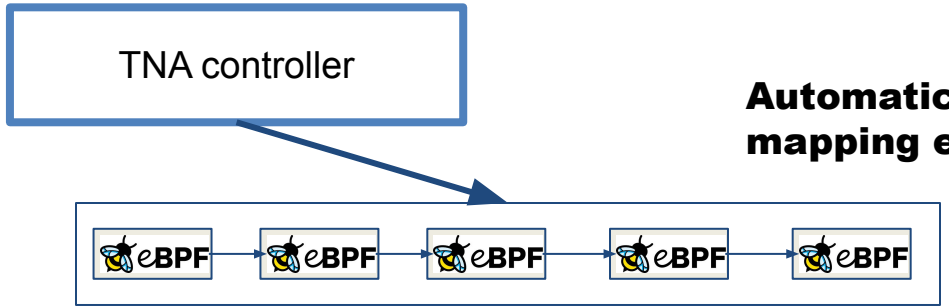
Dynamically create a fast path



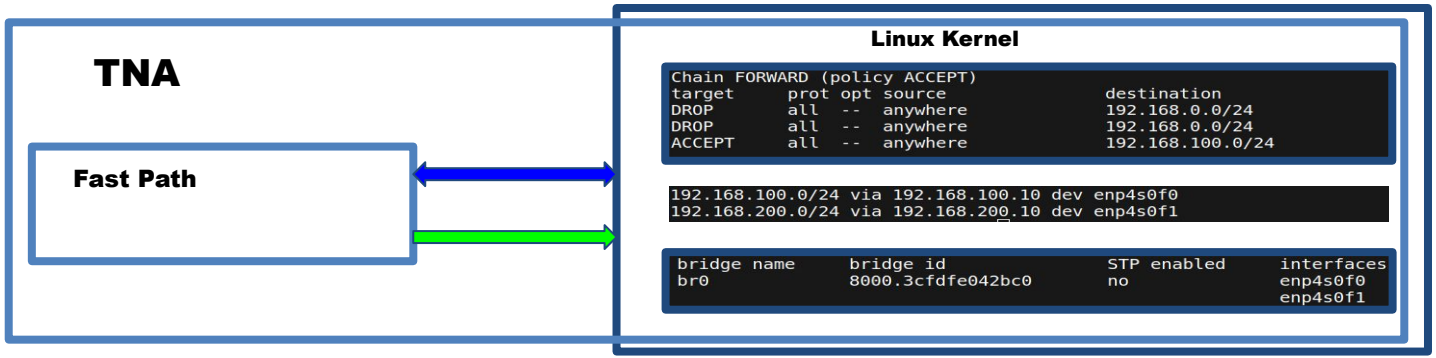
Builds a dependency graph representing the configured services



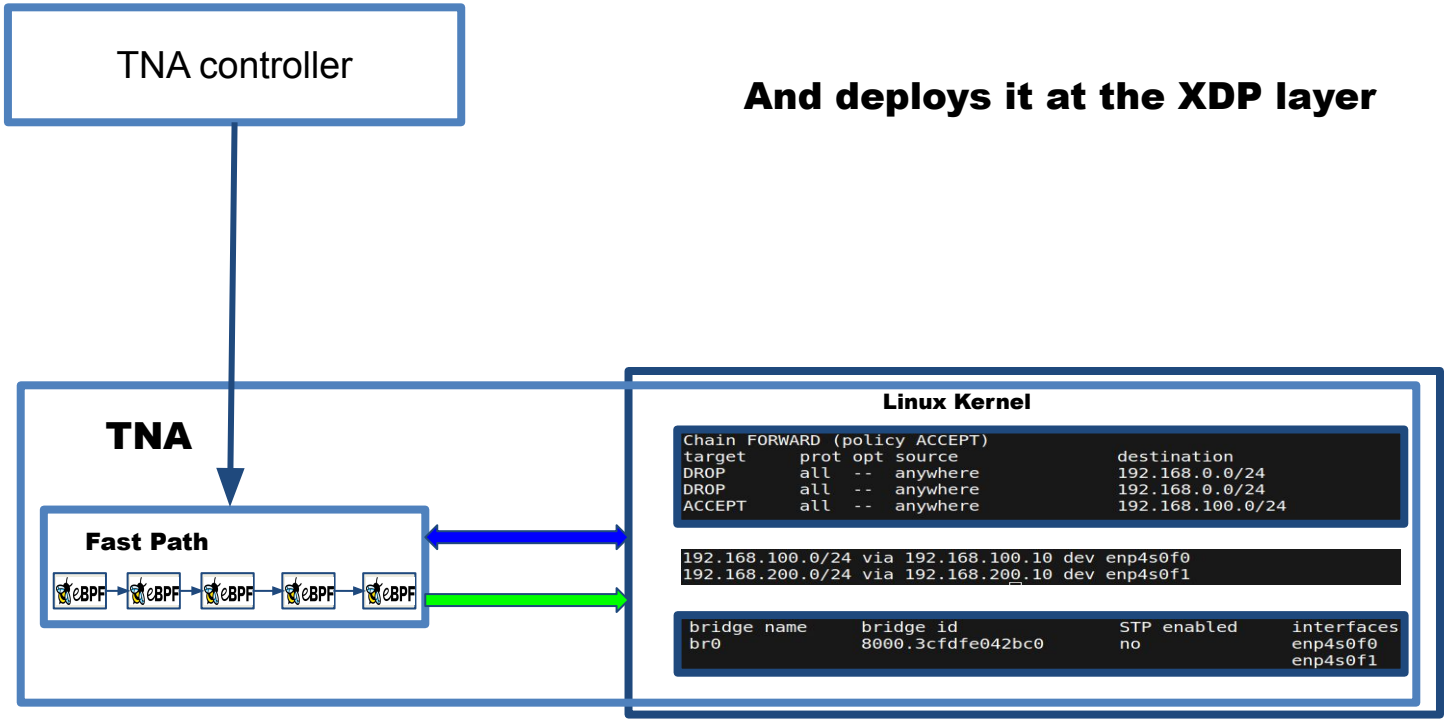
Dynamically create a fast path



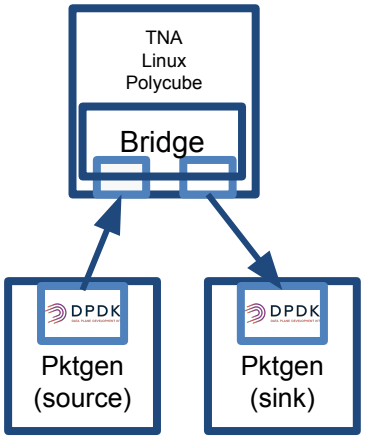
Automatically composes a fast path by mapping each node into one or more FPMs



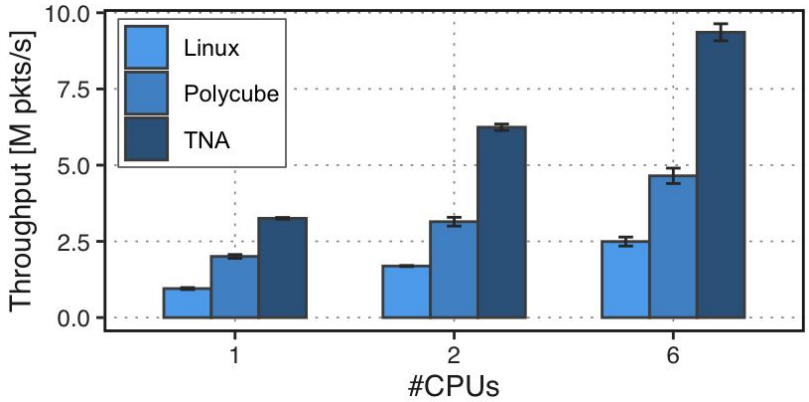
Dynamically create a fast path



Preliminary evaluation



3.4-3.8x faster than Linux and
1.6-2x faster than Polycube



Conclusion

- We propose a redesign of the Linux network stack
  - Make it faster
- This is realizable with technology currently available on the Linux kernel

Future Work

- Comprehensive analysis of Linux network stack
  - Decompose/accelerate more subsystems
- How to ensure the correctness of the data plane
- Explore debugging mechanisms

Thanks!