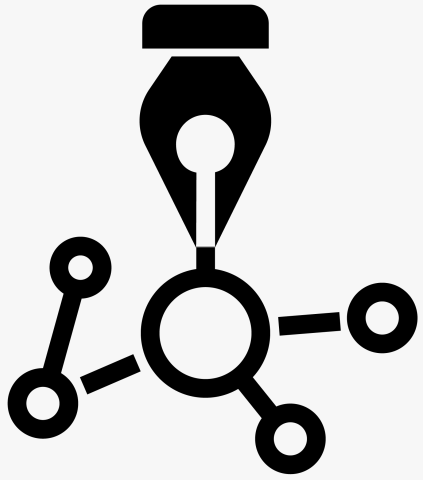


netpen.io

Visual editor and API for network environments script generation

https://www.netpen.io

Eyal Birger

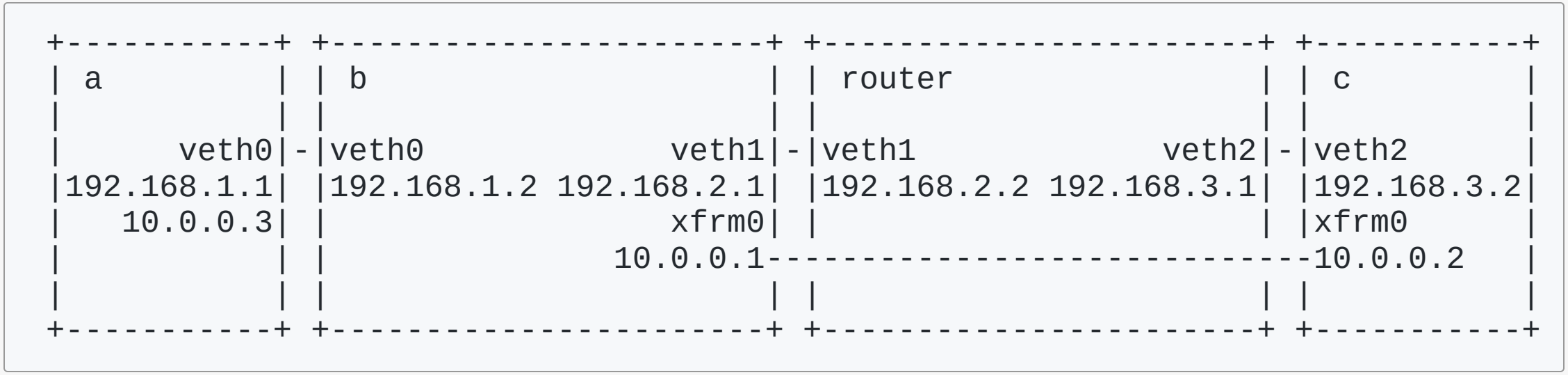
Netdev 0x15 - July 2021

Simulating Network Setups

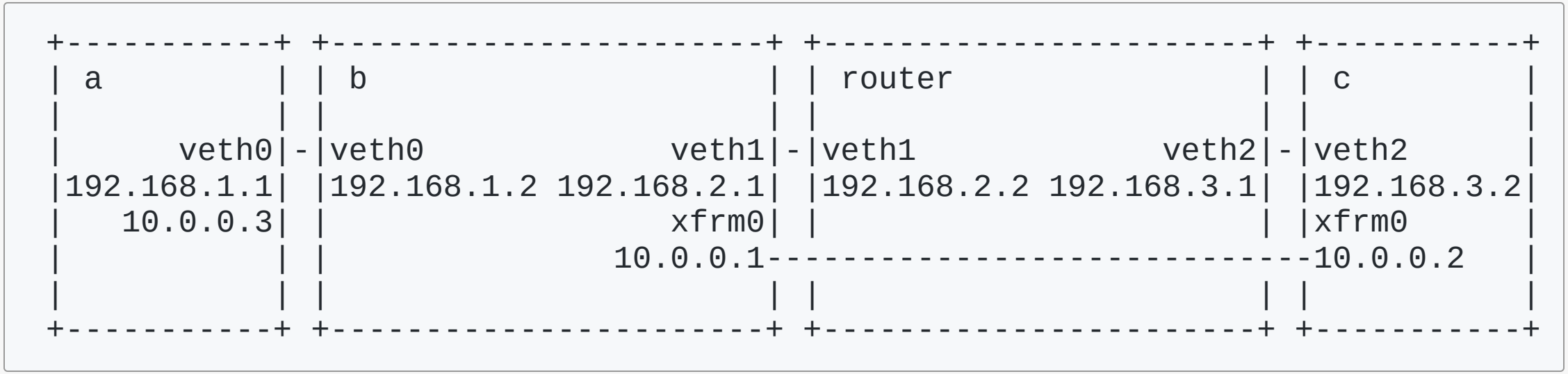
Original incentive

System using point to point IPsec connections was dropping large UDP packets.

Original incentive



Original incentive



Needs

- Network namespaces setup and configuration (e.g. forwarding)
- VETH devices, addresses, routing
- XFRM configuration (simplest possible - no IKE!)
- MTU setup

Network scenarios and BASH

Creating (complex) network topologies using ad-hoc BASH scripts is commonplace.

Networks are commonly setup using network namespaces connected by VETH devices.

A cursory check shows thousands of emails containing "ip netns" on the netdev mailing list.

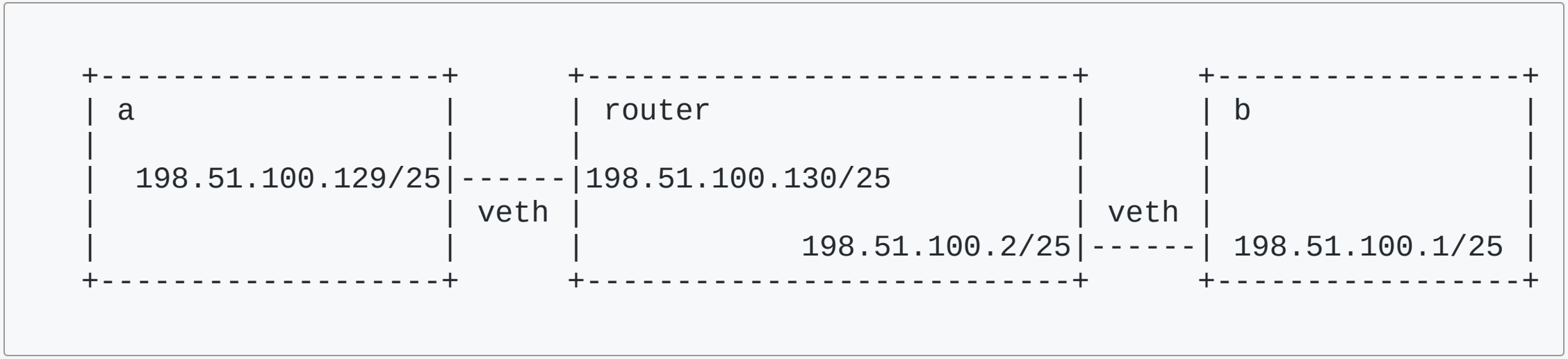
Easily configure Linux networking scenarios

Goals

- Easy to use
- Automatic routing and address assignment
- Support advanced networking features
- Facilitate collaboration

API First

Simple Routing Scenario



Router - YAML

```
items:
  - netns:
      name: a

  - netns:
      name: b

  - netns:
      name: router

  - subnet:
      name: default
      cidr: 198.51.100.0/24

  - veth:
      name: atorouter
      dev1:
        netns: netns.a
        subnets:
          - subnet.default
      dev2:
        netns: netns.router
        subnets:
          - subnet.default

  - veth:
      name: btorouter
      dev1:
        netns: netns.b
        subnets:
          - subnet.default
      dev2:
        netns: netns.router
        subnets:
          - subnet.default
```

API

HTTP `POST`, e.g:

```
curl --data-binary "@examples/router.yml" https://api.netpen.io/v1/bash
```

Router - BASH

```
#!/bin/bash

cat << "EOF"
 ____                _
|  _ \   ___   _   _ | |_   ___  _ __
| |_) | / _ \ | | | || __| / _ \| '__|
|  _ < | (_) || |_| || |_ |  __/| |
|_| \_\ \___/  \__,_| \__| \___||_|


EOF


cat << EOF
+-----------+-----------------------------------+
| Namespace | IPv4                              |
+-----------+-----------------------------------+
|     a     | 198.51.100.129/25 (atorouter.dev1) |
+-----------+-----------------------------------+
|     b     | 198.51.100.1/25 (btorouter.dev1)   |
+-----------+-----------------------------------+
|   router  | 198.51.100.130/25 (atorouter.dev2) |
|           | 198.51.100.2/25 (btorouter.dev2)   |
+-----------+-----------------------------------+
EOF


sysctl -w net.ipv4.route.mtu_expires=15


# netns
ip netns add router
ip netns exec router sysctl -w net.ipv4.conf.all.forwarding=1
ip netns exec router sysctl -w net.ipv6.conf.all.forwarding=2
ip netns add a
ip netns exec a sysctl -w net.ipv4.conf.all.forwarding=1
ip netns exec a sysctl -w net.ipv6.conf.all.forwarding=2
ip netns add b
ip netns exec b sysctl -w net.ipv4.conf.all.forwarding=1
ip netns exec b sysctl -w net.ipv6.conf.all.forwarding=2

# veth
ip link add btorouter.dev1 netns b type veth peer name btorouter.dev2 netns router
ip -net b link set btorouter.dev1 up
ip  -net b addr add 198.51.100.1/25 dev btorouter.dev1
ip -net router link set btorouter.dev2 up
ip  -net router addr add 198.51.100.2/25 dev btorouter.dev2
ip link add atorouter.dev1 netns a type veth peer name atorouter.dev2 netns router
ip -net a link set atorouter.dev1 up
ip  -net a addr add 198.51.100.129/25 dev atorouter.dev1
ip -net router link set atorouter.dev2 up
ip  -net router addr add 198.51.100.130/25 dev atorouter.dev2

# routes
ip  -net a route add 198.51.100.0/25 dev atorouter.dev1 metric 30 via 198.51.100.130 src 198.51.100.129
ip  -net b route add 198.51.100.128/25 dev btorouter.dev1 metric 40 via 198.51.100.2 src 198.51.100.1
```

Current configuration items

- Network Namespaces
- VETH
- Bridge
- VLAN
- MACVLAN
- Team
- XFRM transport
- Tunnels
  - XFRM (xfrmi and vti)
  - IPIP
  - VXLAN
  - GRE
  - WireGuard
- Virtual Routing and Forwarding (VRF)

UI

Collaboration

Under the hood

Implementation

- Backend: Python
  - Address configuration - separate routable devices and assign addresses based on L2/L3 connectivity
  - Routing - shortest path selection between routeable devices
- Frontend: React

Development & Contribution

- Github project page: https://github.com/ebirger/netpen
- Local development

```
git clone https://github.com/ebirger/netpen.git
cd netpen
make dev
```

Future Work

- eBPF
- TC filters
- Netfilter rules (iptables/nftables)
- Policy routing
- Light Weight Tunneling
- More device types

Thank You