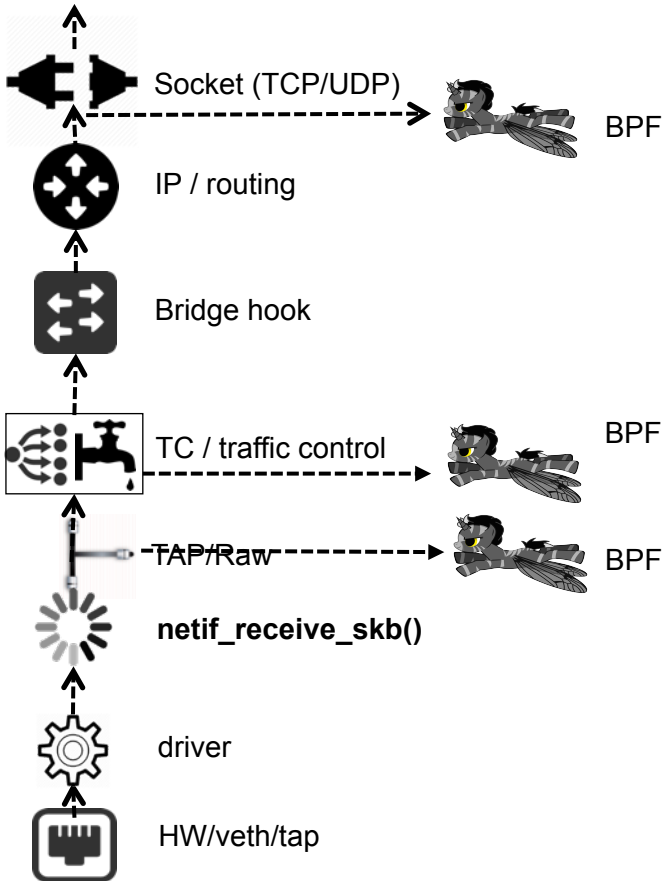
# BPF networking examples

2015 aug 20

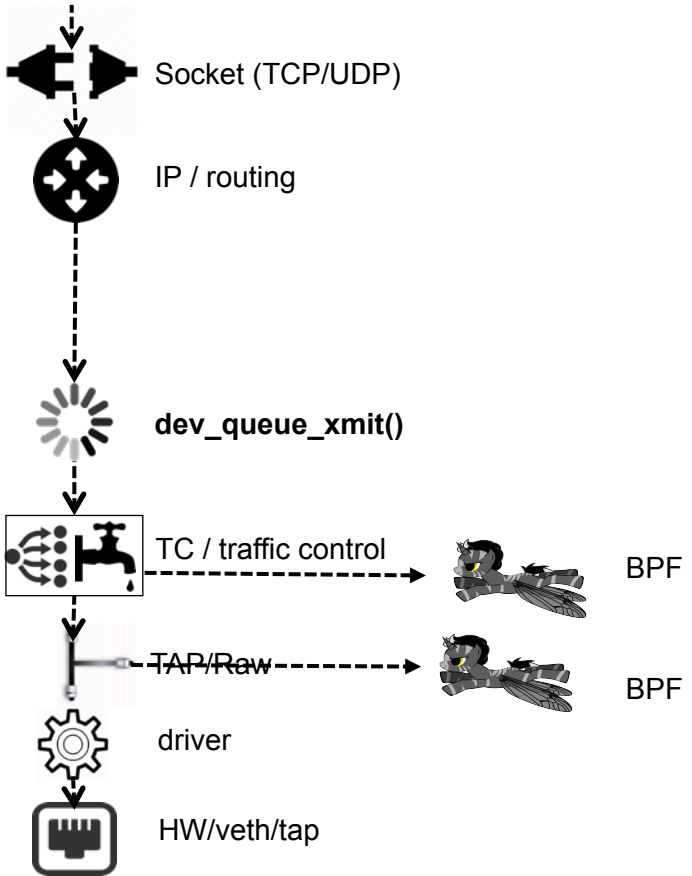
# Hooking into the Linux networking stack (RX)

- BPF programs can attach to sockets or the traffic control (TC) subsystem
- sockets: TCP, UDP, netlink, af_unix, raw
- This allows to hook at different levels of the Linux networking stack, providing the ability to act on traffic that has or hasn't been processed already by other pieces of the stack

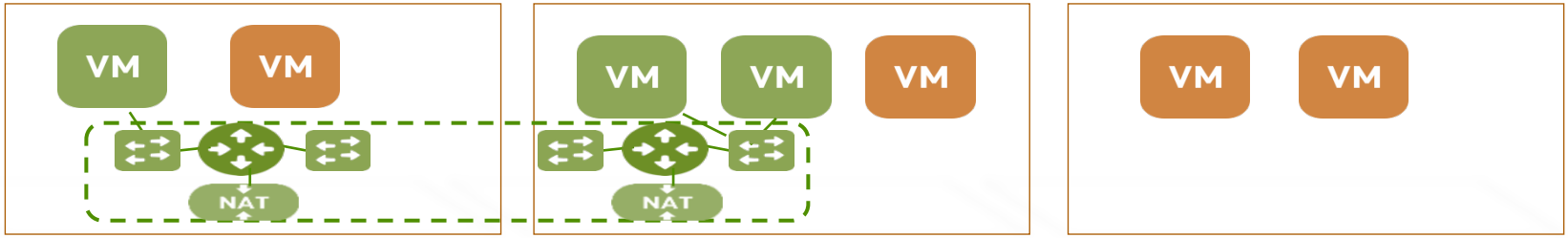


# Hooking into the Linux networking stack (TX)

- Opens up the possibility to implement network functions at different layers of the stack

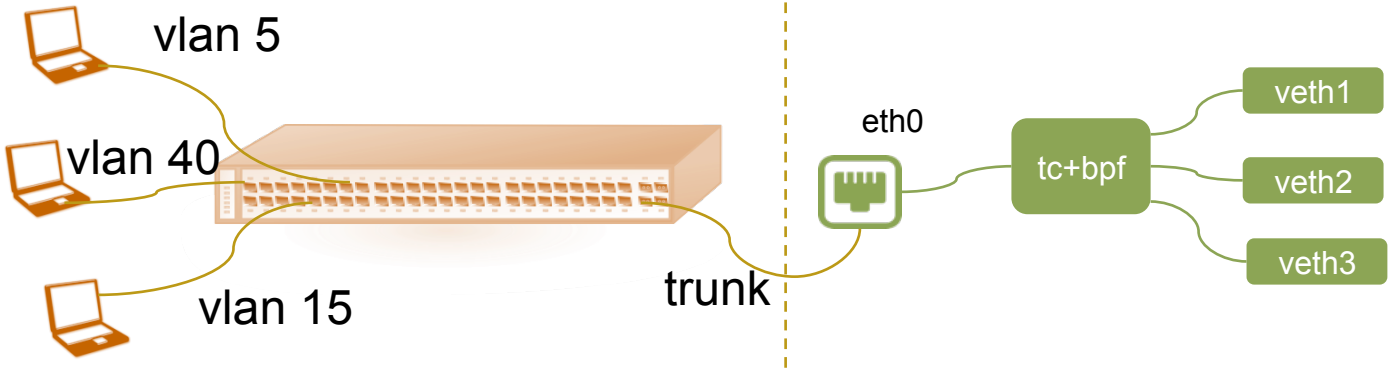


# Use BPF to build distributed virtual topology out of existing linux bridges and routers



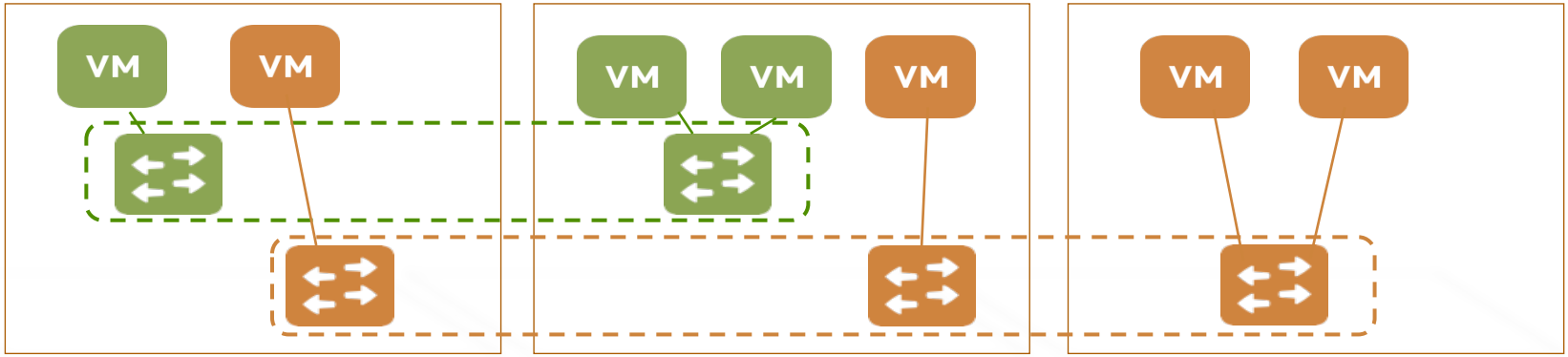

= BPF

# physical to virtual gateway in BPF

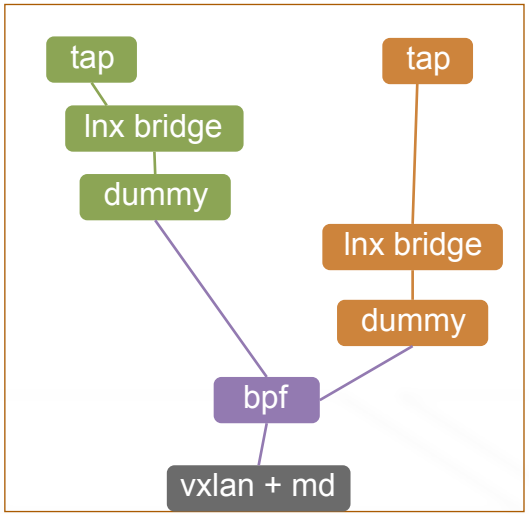


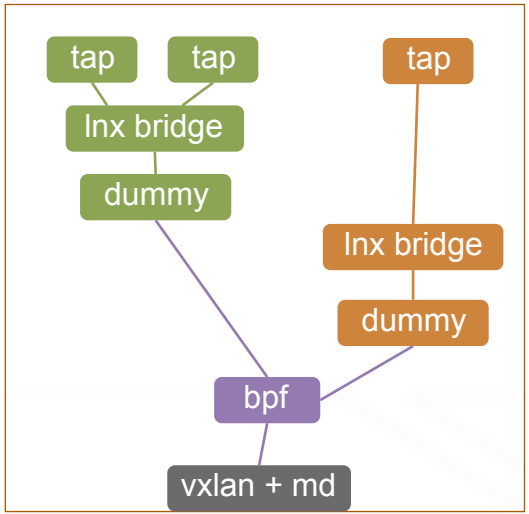

phys2virt: src mac redirect + vlan learning + vlan_pop
virt2phys: ifindex based redirect + vlan_push

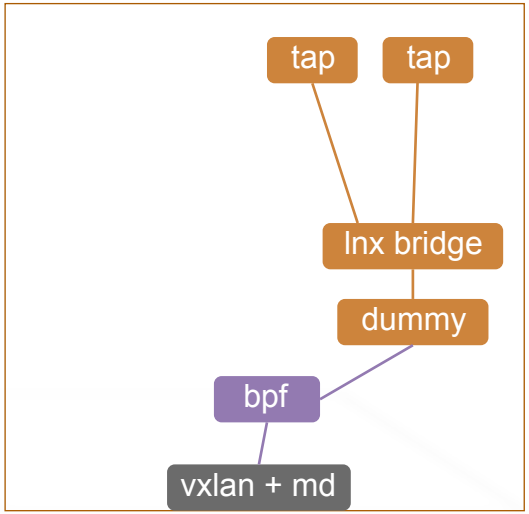
# Distributed bridge with BPF



# Distributed bridge with BPF and multicast

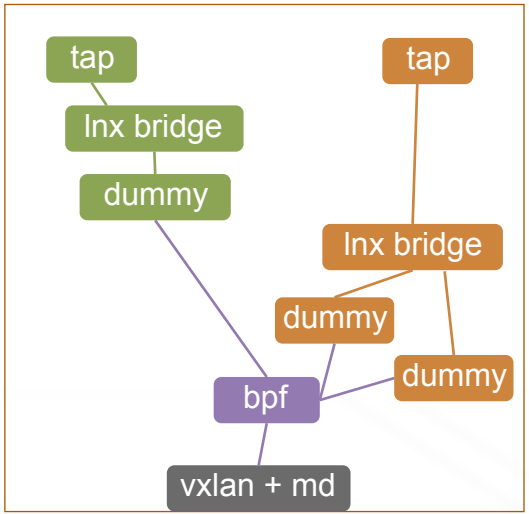


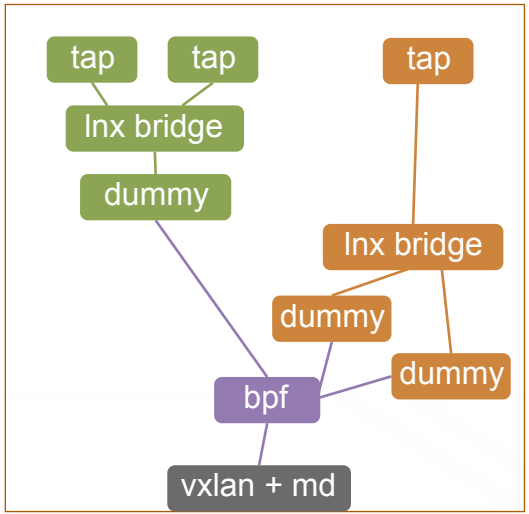



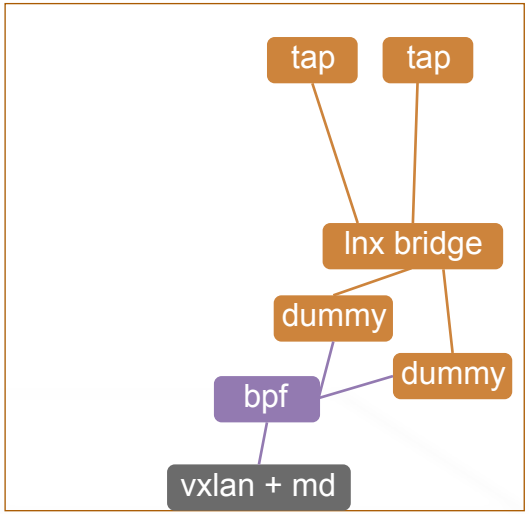




BPF program does
- ingress: redirect {vni} -> {ifindex}, learn {src_mac, ifindex} -> {remote_ip, vni}
- egress: add metadata {dst_mac, ifindex} -> {remote_ip, vni}, redirect to tunnel
  use multicast for unknown dst_mac

# Distributed bridge with BPF (no multicast)

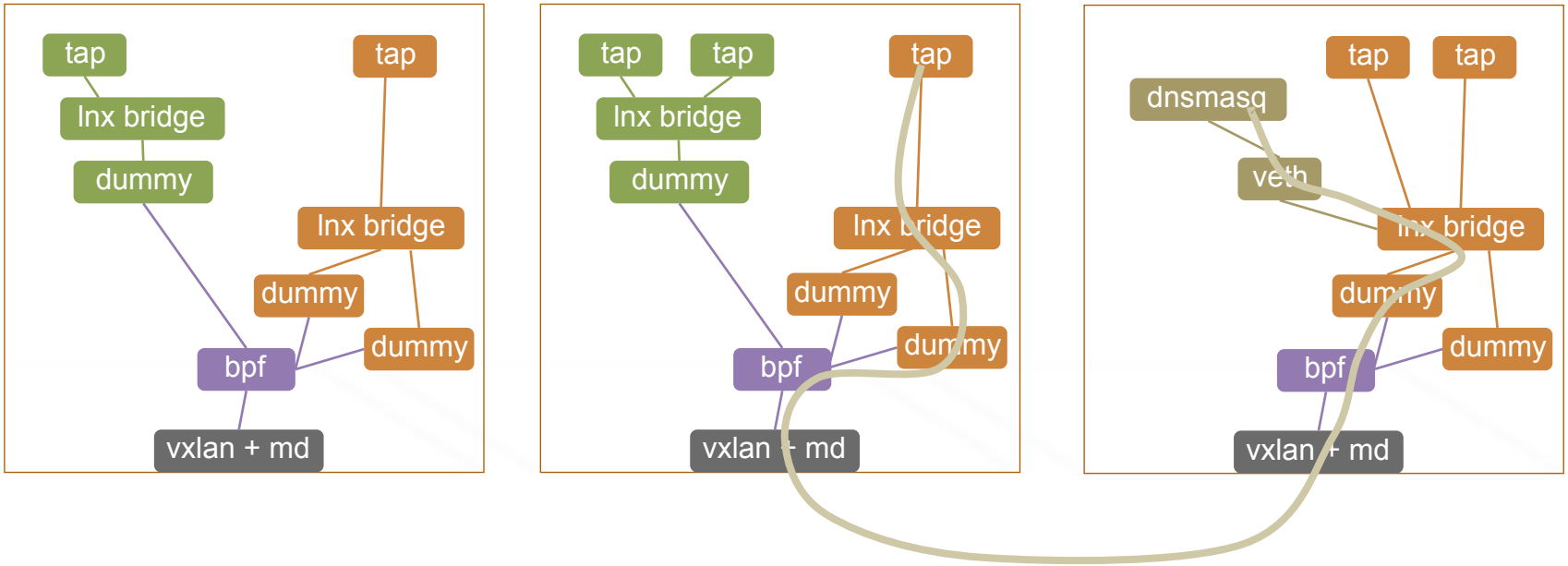


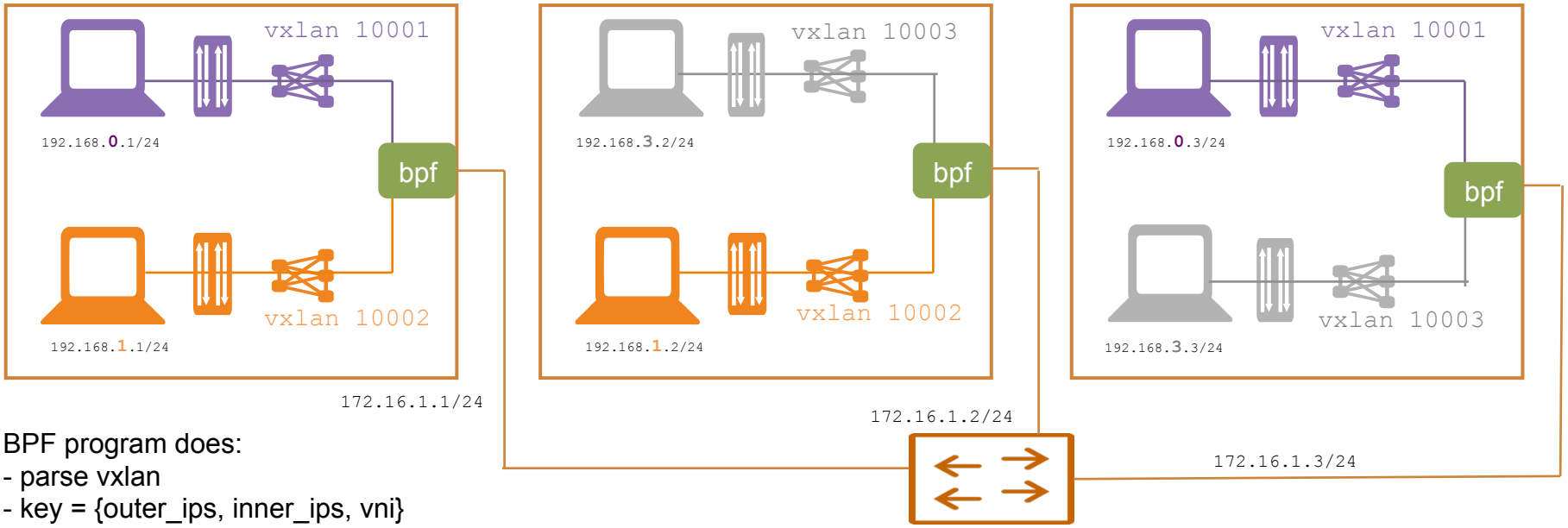

BPF program does
- ingress: redirect {vni, remote_ip} -> {ifindex}
- egress: add metadata {ifindex} -> {remote_ip, vni}, redirect to tunnel

# Distributed bridge with BPF (no multicast) and DHCP




packet flow of DHCP request/reply

# tunnel analytics with BPF




BPF program does:
- parse vxlan
- key = {outer_ips, inner_ips, vni}
  value = map[key]
  value.rx_pkts ++
  value.rx_bytes += skb->len
  value.tx_pkts ++
  value.tx_bytes += skb->len

# tunnel analytics with BPF

- BPF program collects TX/RX stats, stores them in a map
- user space python does:

```
def stats2json(k, v):
    return {
        "vni": int(k.vni),
        "outer_sip": str(IPAddress(k.outer_sip)),
        "outer_dip": str(IPAddress(k.outer_dip)),
        "inner_sip": str(IPAddress(k.inner_sip)),
        "inner_dip": str(IPAddress(k.inner_dip)),
        "tx_pkts": v.tx_pkts, "tx_bytes": v.tx_bytes,
        "rx_pkts": v.rx_pkts, "rx_bytes": v.rx_bytes,
    }
while True:
    result_total = []
    for k, v in stats.items():
      result_total.append(stats2json(k, v))
    with open("./chord-transitions/data/tunnel.json.new", "w") as
        json.dump(result_total, f)
```

- chord-transitions converts json to chord diagram