FreeBSD/VPC

Virtual Private Cloud support (fka SDN)

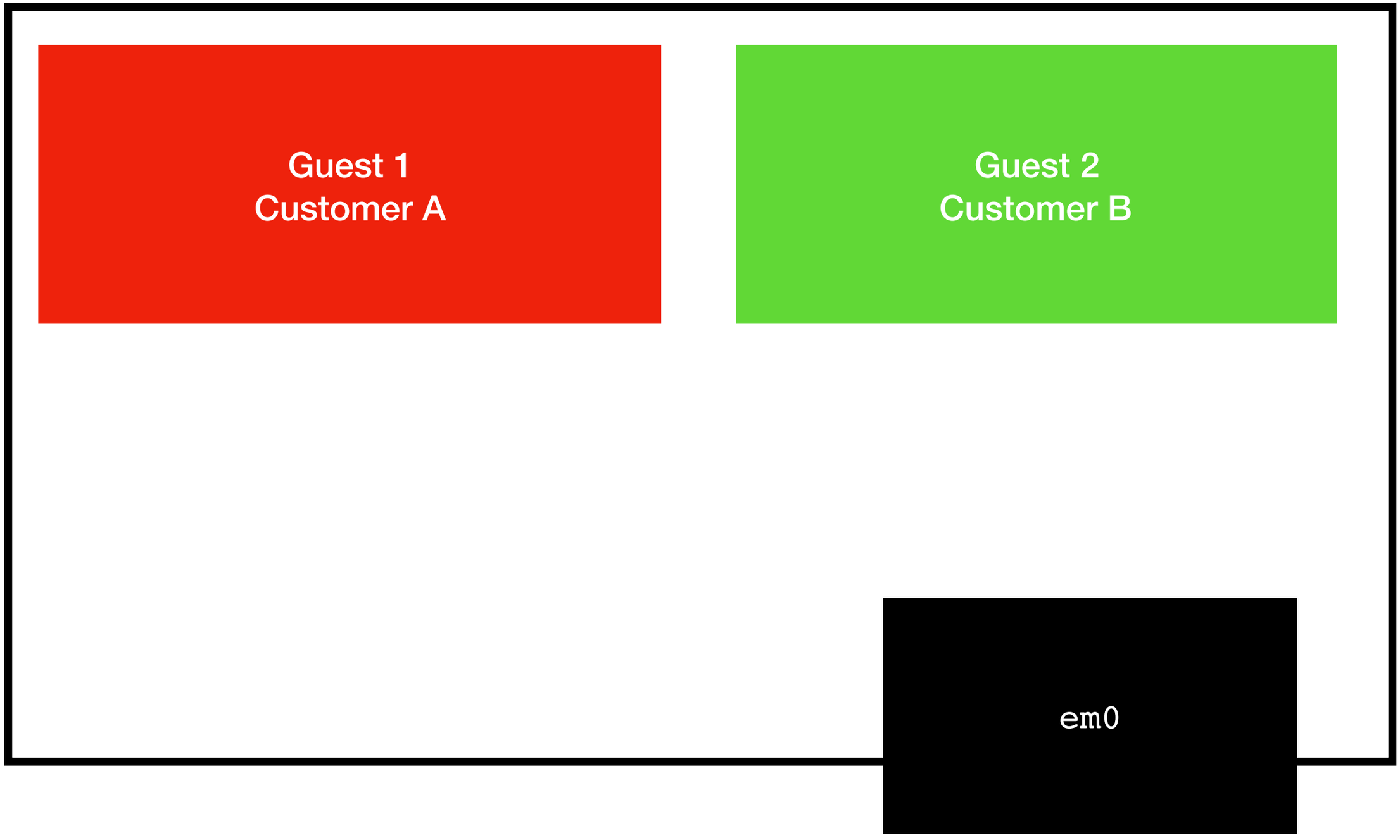
Virtualization Status

- `bhyve(4)` is a stable, performant hypervisor
- Network isolation is not core to `bhyve(4)` today
- Use of `VNET(9)` for manipulating FIBS for `tap(4)` interfaces is possible, but limited and not performant

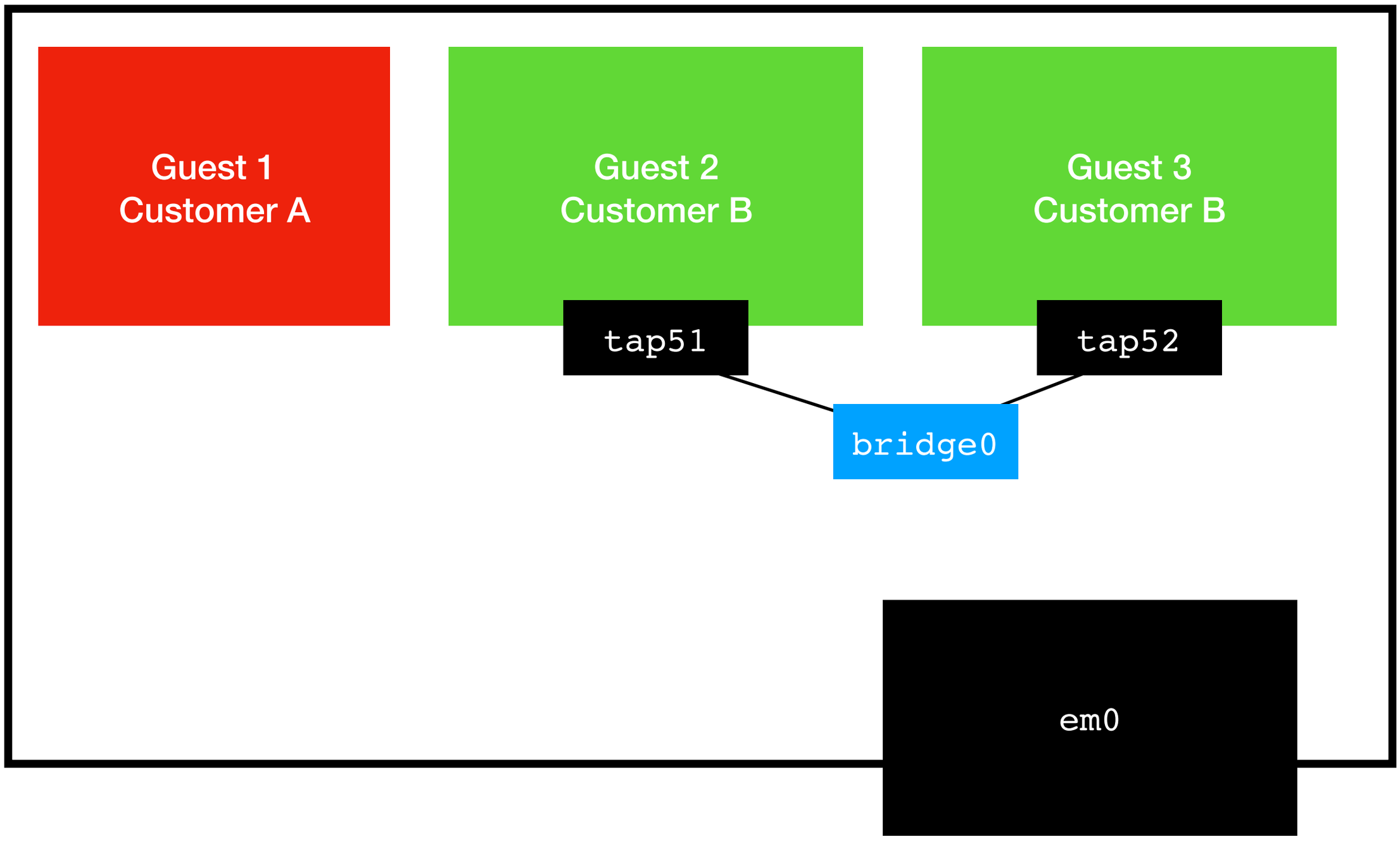
Problem

- `bhyve(4)` guests run customer workloads
- Cloud providers need a single FIB for the underlay network
- Guests run in isolated overlay networks
- How do you map guests to their respective overlay network?

Guest Workloads



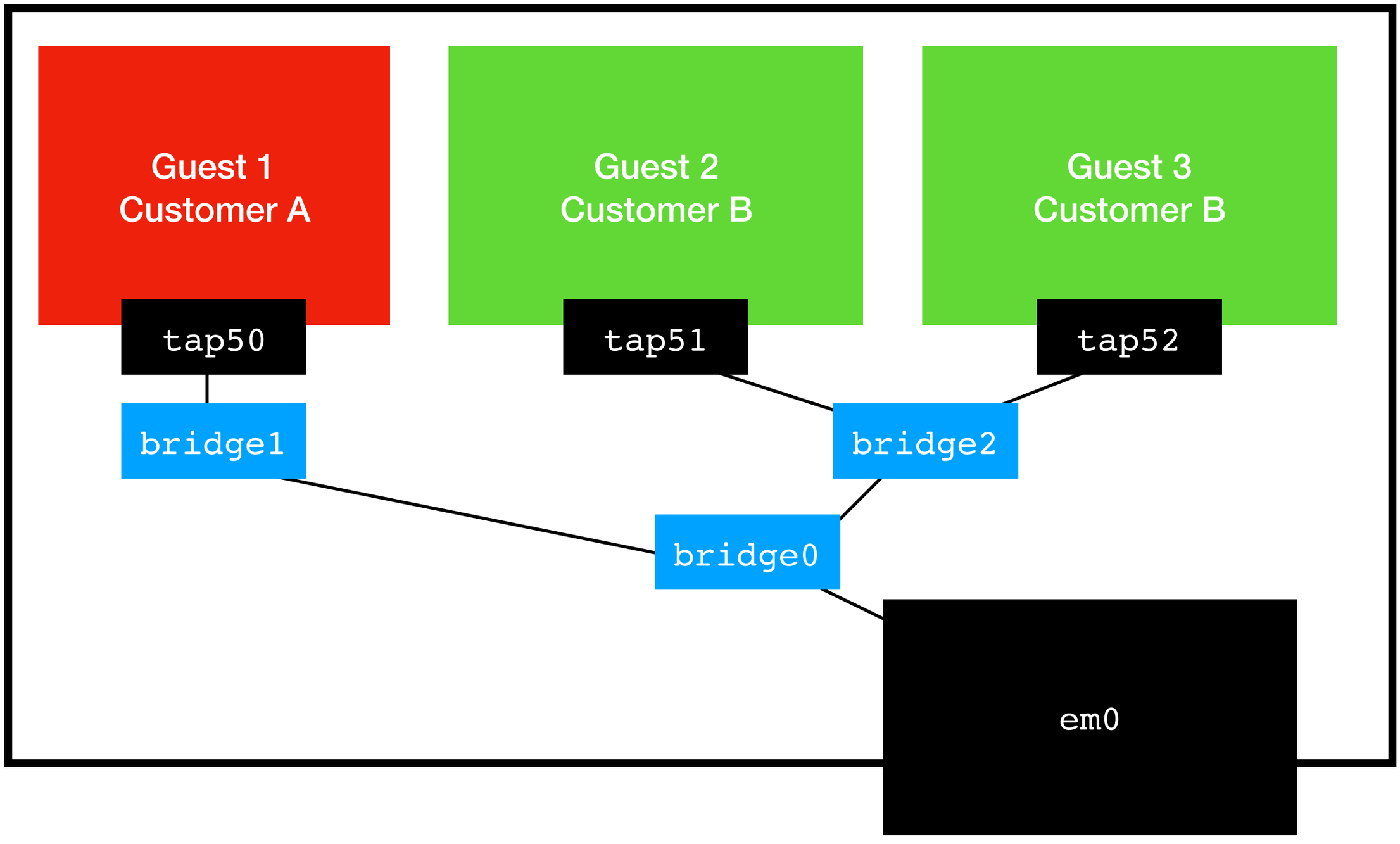
Guest Workloads



FreeBSD

- `bhyve(4)` guests run customer workloads
- Cloud providers need a single FIB for the underlay network
- Guests run in isolated overlay networks
- How do you map guests to their respective overlay network?

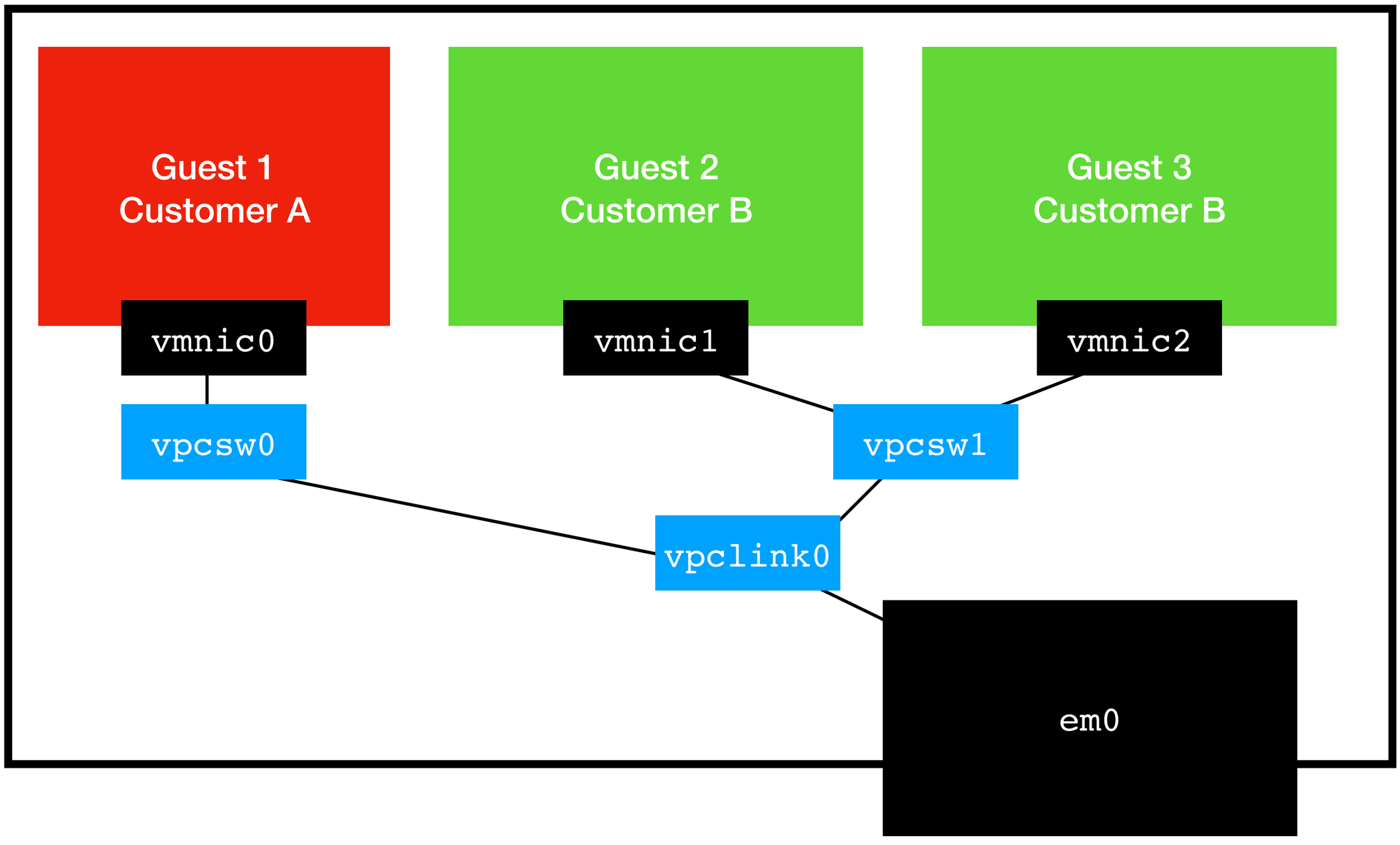
if_bridge(4) Approach



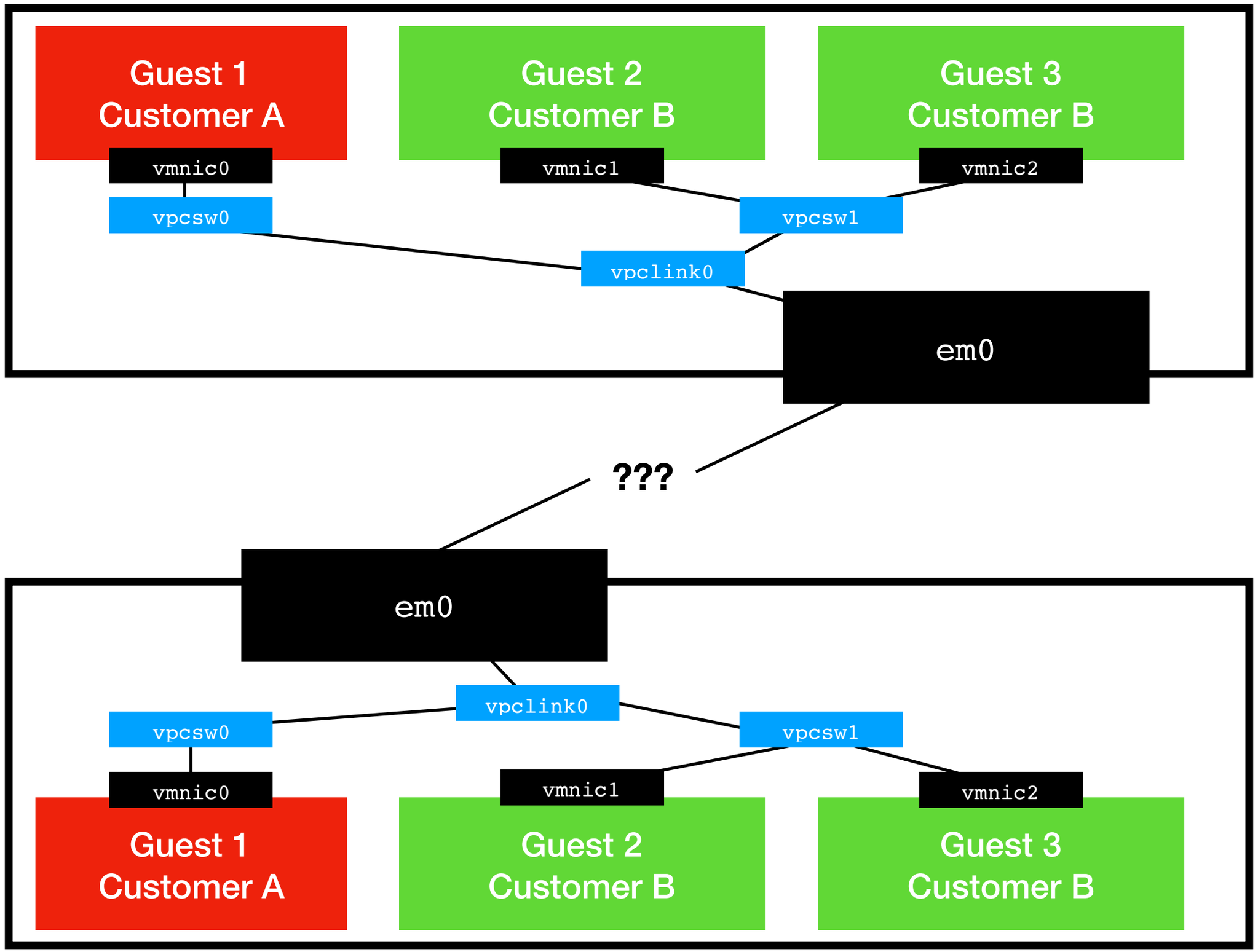
Problems with Current Tools:
`tap(4)`/`bridge(4)`/`vxlan(4)`/`VNET(9)`

- `tap(4)` is slow
- `bridge(4)` is slow
- `vxlan(4)` sends received packets through `ip_input()` twice (i.e. "sub-optimal")
- `VNET(9)` virtualizes underlay networks, not overlay networks
- How do you ARP across machines?
- How do you perform `vxlan(4)` encap?

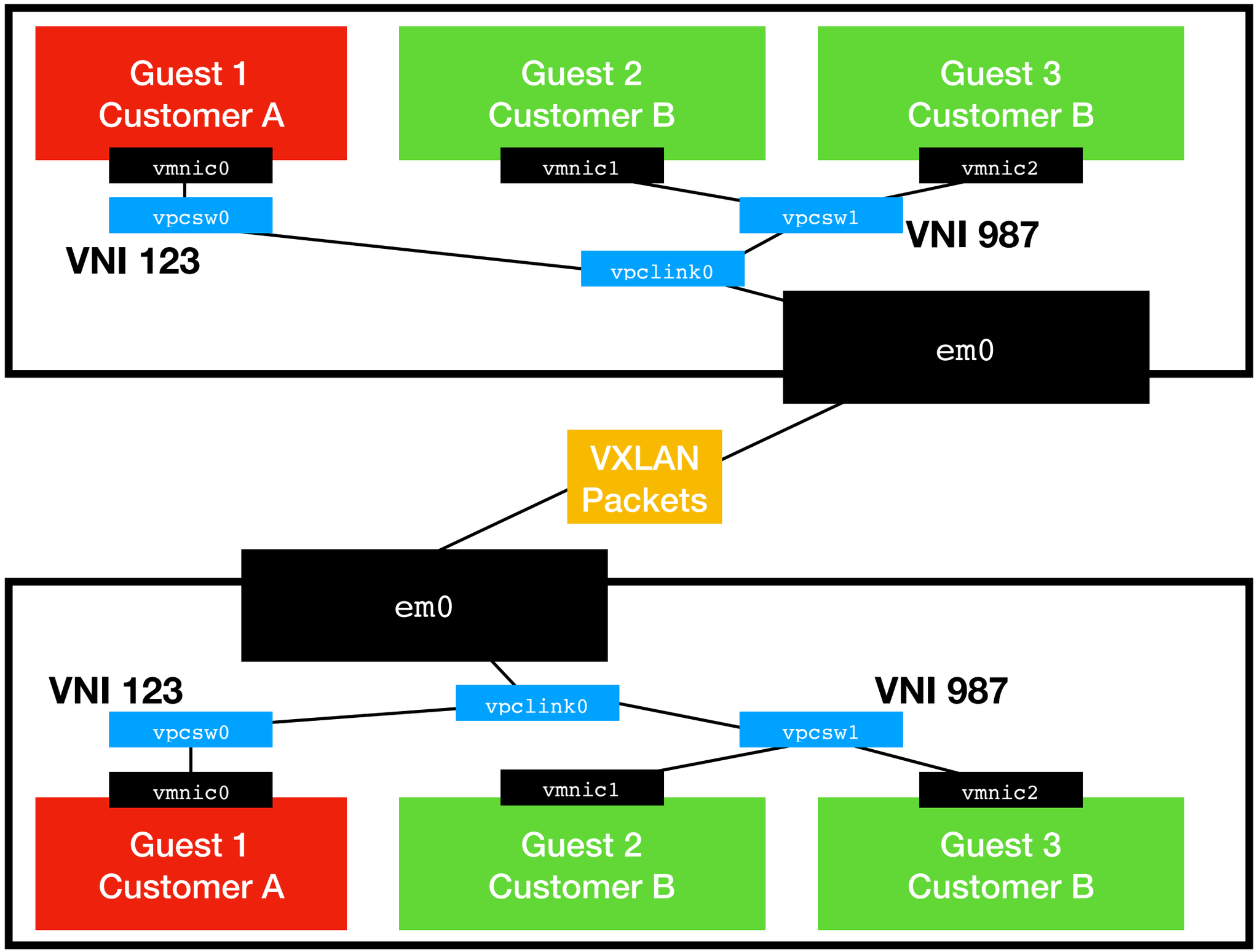
FreeBSD/vpc


Guest 1
Customer A
Guest 2
Customer B
Guest 3
Customer B
vmnic0
vmnic1
vmnic2
vpcsw0
vpcsw1
vpclink0
em0

FreeBSD/vpc Multi-Host



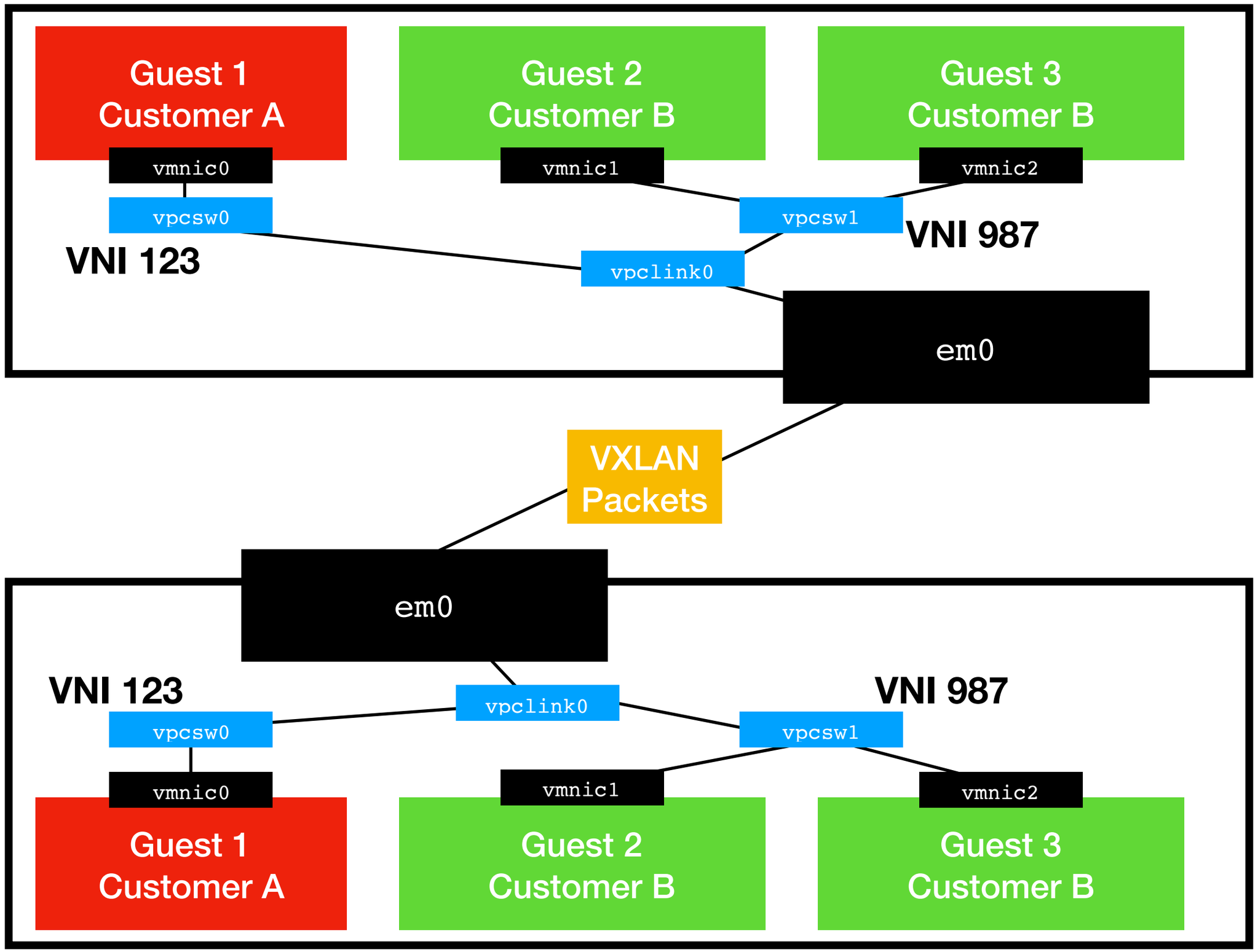
FreeBSD/vpc Multi-Host



VXLAN to the Rescue

- Encapsulates all IP packets as UDP
- Adds a preamble to IP packet
- Tags packets and with a VXLAN ID, known as a VNI
- VXLAN is similar to VLAN tagging, but embeds tagging in the IP header, not in the L2 frame

FreeBSD/vpc Multi-Host



`vpc(4)` Interfaces

- `vpcsw(4)` - switches packets - one packet per customer, multiple subnets supported in the same switch

- `vmnic(4)` - dedicated guest NIC, looks like a virtio network device to guests

- `vpcp(4)` - plugs `vmnic(4)` ports into `vpcsw(4)` switches

- `vpci(4)` - Non-`bhyve(4)` interface, usable in `jails(2)`

- `ethlink(4)` - Performs unencapsulated packet forwarding, wraps a cloned or physical ethernet interface

- `vpclink(4)` - Performs VXLAN encapsulation

New System Calls

- `vpc_open(2)` - Creates a new VPC descriptor
- `vpc_ctl(2)` - Manipulates VPC descriptors
- Capsicum-like, intended for privilege separation
- Intended for idempotent tooling
- Makes aggressive use of UUIDs as operator handles to be compatible with Triton

Ongoing Work

- Firewalling
- Routing
- NAT
- Userland Control Plane (including setup and teardown of `bhyve(4)` guests via something not a shell script)

Code

- Kernel: https://github.com/joyent/freebsd/tree/projects/VPC

- Kernel Libraries: https://github.com/joyent/freebsd/tree/projects/VPC/libexec/go/src/go.freebsd.org/sys/vpc

- Userland tooling: https://github.com/sean-/vpc

Questions?