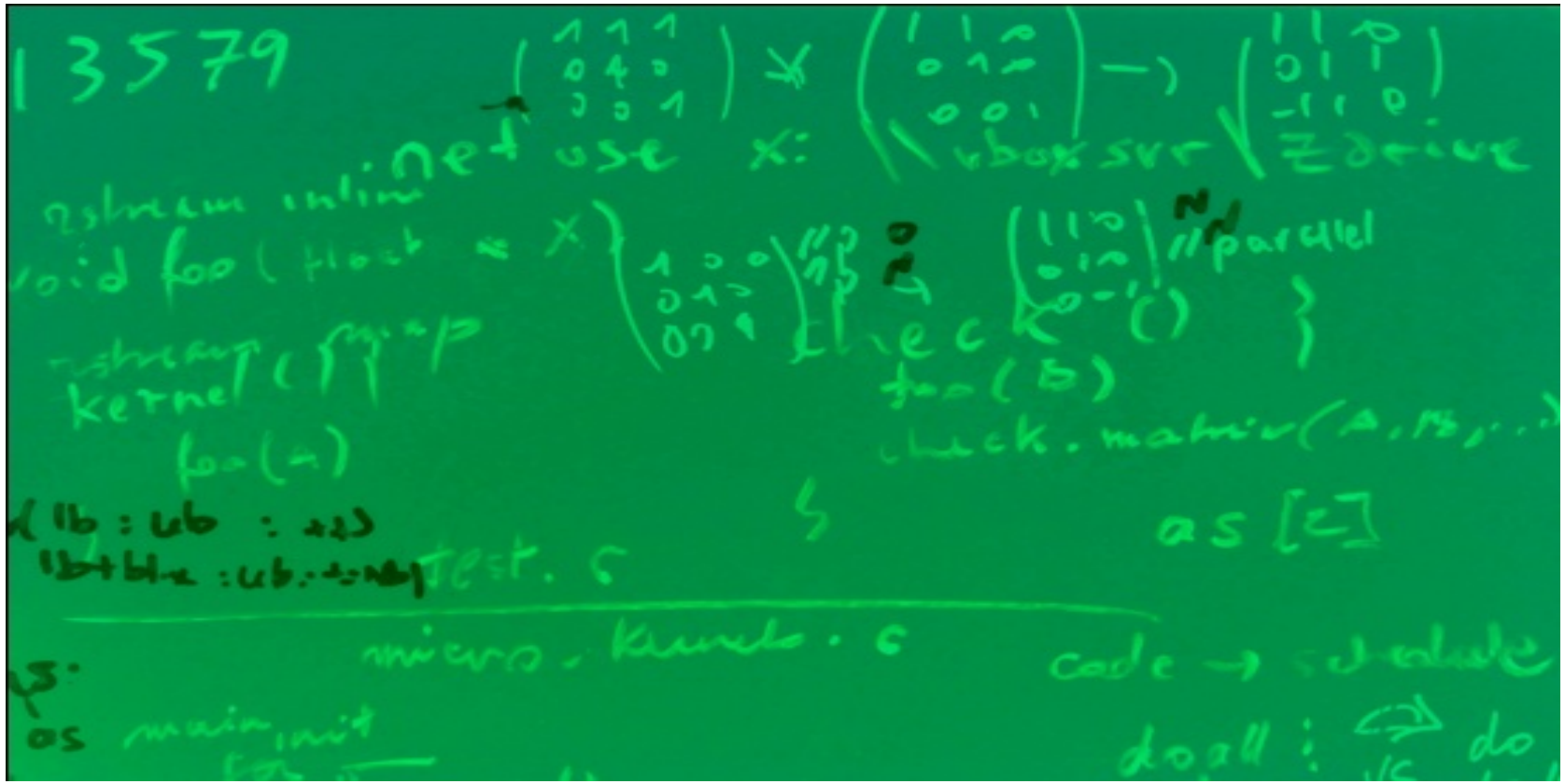
Running Bro in the Cloud at Scale



About:

Alan Commike

Reservoir Labs: Commercial Bro systems; HW, VM, services

commike@reservoir.com

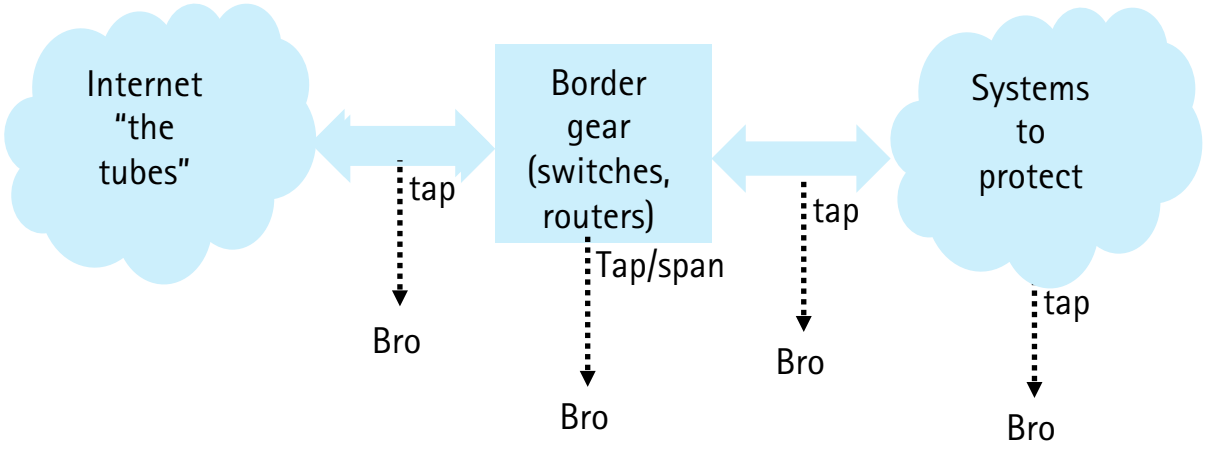
Intro

Three Sections

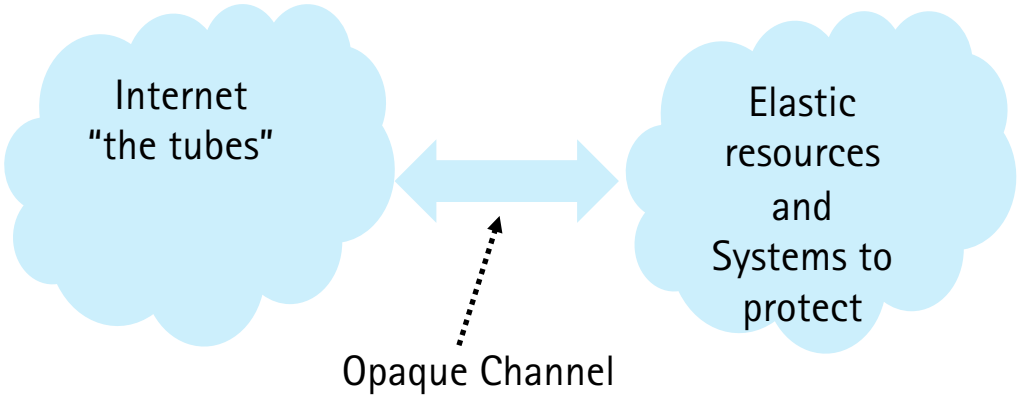
- The Cloud: accessing and distributing packets
- Scaling a virtual Bro instance: vertical and horizontal
- Measuring performance in cloudy environments

What makes “the Cloud” different?

Not the cloud



The cloud



The

What type of cloud

Cloud can have many definitions:

- Your own the infrastructure, have full access
  - Similar to traditional networks, level of visibility determines tap points
  - Virtual to virtual visibility requires more work
- Hosted, on-/off-prem, limited access to underlying infrastructure
  - Lack of infrastructure access
  - Security policies
  - vNIC vs real NIC and drivers

Identity can be much more complex as VMs/containers move, scale up/down.

Do what in the cloud?

- Protect services in the cloud
  - VMs
  - Virtual switches / overlay network
- Run a virtual Bro, issues?
  - Packet delivery
  - Virtual NICs
  - Bro scaling

Bro for cloud-scale apps too?

- Data center migration to the cloud: On-prem principles still apply
  - Watch the border
  - Watch the core or at least important segments
  - Understand topology and services to look for anomalies
- Cloud scale apps: This is different
  - Simple “micro service” communication patterns
  - All SSL
  - No users and typical user services
  - **Service Level Identity**
    - IPs/ports/protocols no longer only indicators

Cloud Visibility

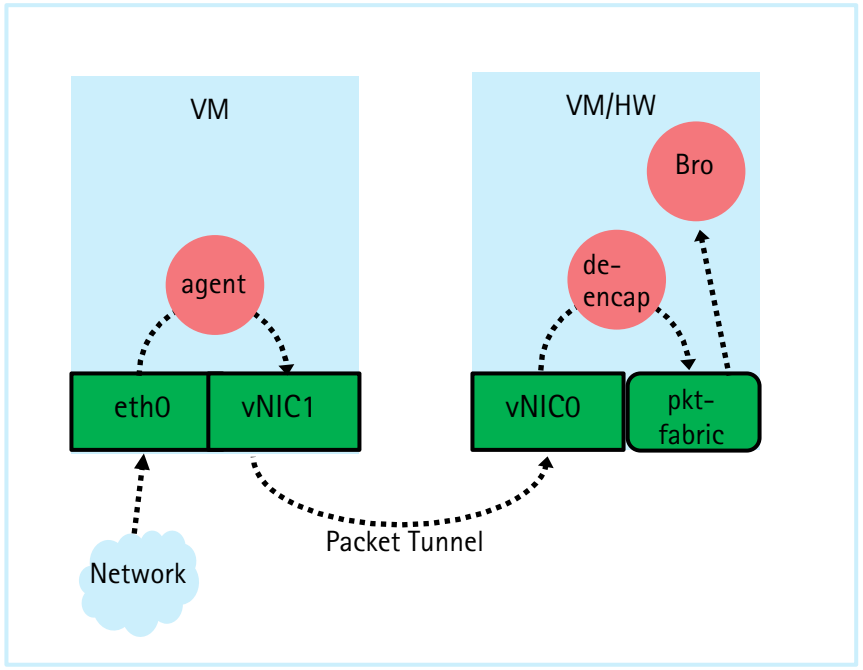
Otherwise said as, “**how do I see the packets on my VMs**?”

- Large commercial elastic cloud vendors
  - Do not provide a “tap” service
  - Do not allow fully promiscuous interfaces

- Solutions
  - Node agents
  - Spans or mirrors on virtual switches

Agents

- Insert an agent into VM/Container
  - Agent “taps” internal vNICs
  - Forwards packets elsewhere for processing



DIY Agent and forwarder:

to tap:

```
tcpdump -i eth0 -s0 -w - | nc my_bro_ip 5555
```

to aggregate:

```
ip link add pkt-fabric type dummy
ifconfig pkt-fabric up
ifconfig pkt-fabric 192.168.1.2
nc -l -k 5555| tcpreplay -i pkt-fabric -
```

```
ip link set pkt-fabric up
ip addr add 192.168.1.2 dev pkt-fabric
```

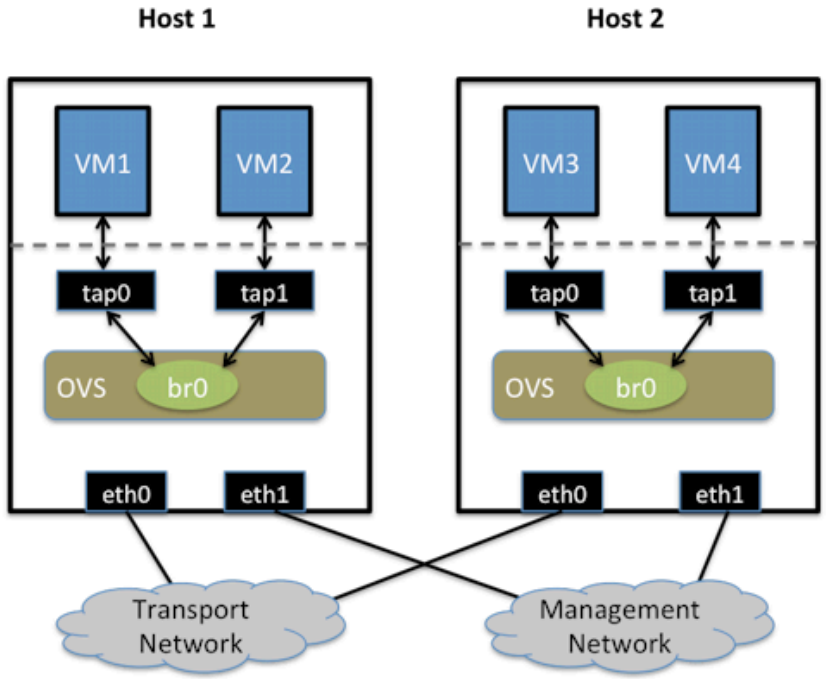
Agents

- Various commercial solutions for agent-based cloud tap/agg
  - Gigamon / Ixia
  - Similar tap/agg functionality as Gigamon / Ixia HW products
  - Same principles as tcpdump/nc/tcpreplay
  - Other vendors too

Virtual Switches

openvswitch

- One of the most popular OSS virtual switches
- OpenStack TAAS (tap-as-a-service)
  - Multi-tenant aware tap service
- SDN/OpenFlow group/select tables
  - Assuming have access to infrastructure



OVS 2.8 Docs: http://docs.openvswitch.org/en/latest/howto/tunneling/

Scaling Bro…

The story so far

- Data in cloud is “tapped”
  - Agents are forwarding data
  - Switches are mirroring data
- A tap/agg “fabric” is in place to shuttle packets to one or more nodes
- Bro is run on the nodes

How to run a Bro on incoming packets?

Hybrid setup

Back-haul packets to on-prem

- Packets sent to physical HW
- Setup Bro as normal
- Tunneled tap/agg fabrics need to de-encapsulate
  - GRE or other tunnel protocols de-encap in switches

Example:

- Gigamon agents, gigamon fabric to physical HW
- Corelight, Reservoir, roll-your-own Bro

Virtualized Bro

Add Bro instances to elastic cloud resources

- BPFs
- RSS
- Multi-NIC through vswitch
- Kernel AF_PACKET
- Netfilter, nftables, eBPF?

BPFs

Simple and old-school

- vNIC set as promiscuous
- All workers read from same vNIC, all get copy of every packet
- Kernel BPFs ensure each worker only sees 1/nth flow
- base/frameworks/pack-filter/utils.bro has sample filter:
  - v4_filter = fmt("ip and ((ip[14:2]+ip[18:2]) - (%d*((ip[14:2]+ip[18:2])/%d)) == %d)", num_parts, num_parts, this_part);

Bro makes this extremely simple:

```
@load load-balancing.bro
```

Down the rabbit hole: BPFs and eBPFs

BPF – The BSD Packet Filter, 1993 USENIX conference

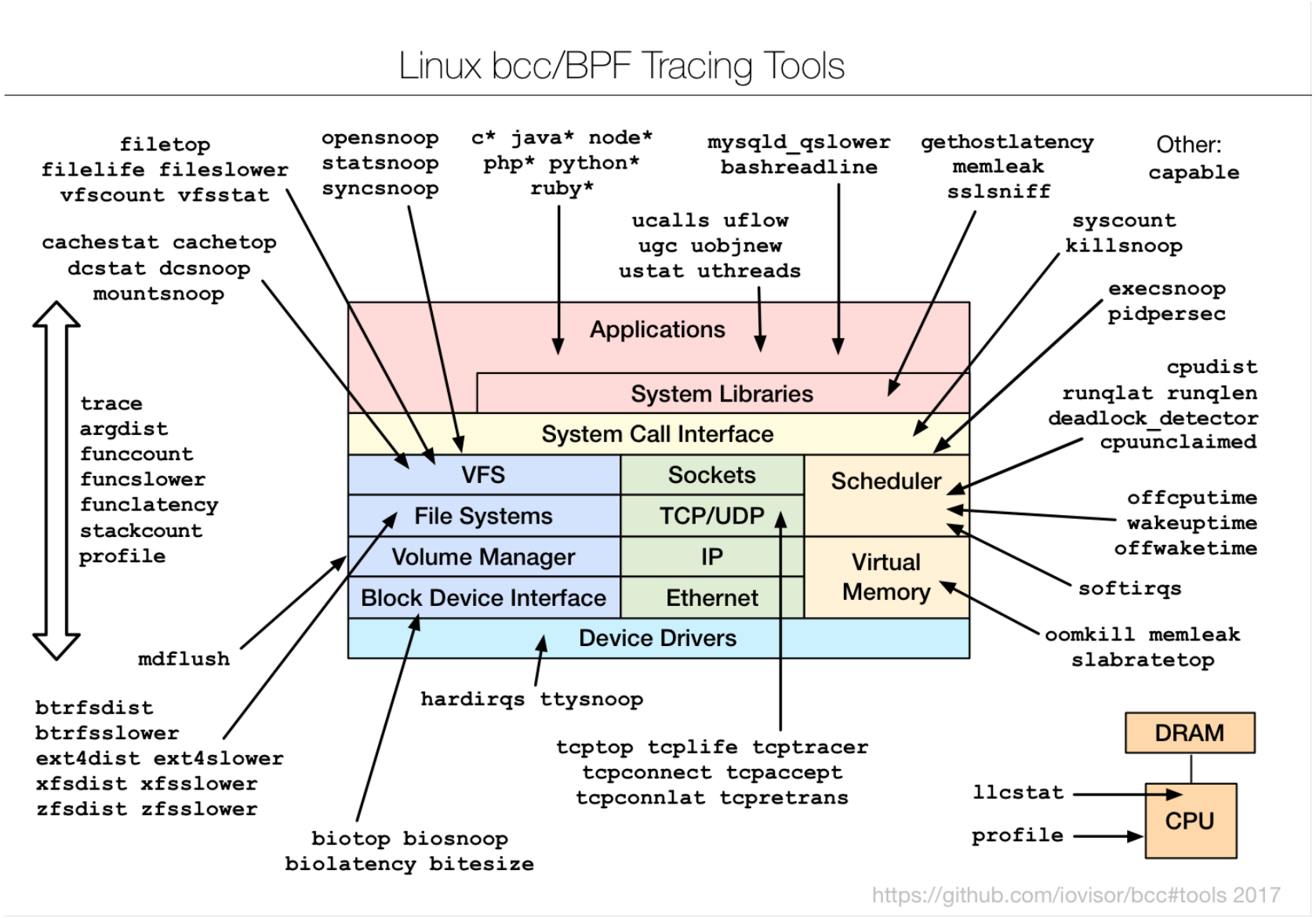
- In kernel BPF virtual machine
- A filter is a ”program” run on the VM
- Higher level language in libpcap/tcpdump, compiles down to BPF

eBPF – enhanced BPF

- Universal in-kernel virtual machine (as stated in the bpf man page)
- LLVM back-end
- Ability to hook and instrument in-kernel

```
[root@rscope]# tcpdump -d "port 80"
(000) ldh      [12]
(001) jeq      #0x86dd          jt 2    jf 10
(002) ldb      [20]
(003) jeq      #0x84            jt 6    jf 4
(004) jeq      #0x6             jt 6    jf 5
(005) jeq      #0x11            jt 6    jf 23
(006) ldh      [54]
(007) jeq      #0x50            jt 22    jf 8
(008) ldh      [56]
(009) jeq      #0x50            jt 22    jf 23
(010) jeq      #0x800           jt 11    jf 23
(011) ldb      [23]
(012) jeq      #0x84            jt 15    jf 13
(013) jeq      #0x6             jt 15    jf 14
(014) jeq      #0x11            jt 15    jf 23
(015) ldh      [20]
(016) jset     #0x1fff          jt 23    jf 17
(017) ldxb     4*([14]&0xf)
(018) ldh      [x + 14]
(019) jeq      #0x50            jt 22    jf 20
(020) ldh      [x + 16]
(021) jeq      #0x50            jt 22    jf 23
(022) ret      #65535(023) ret      #0
```

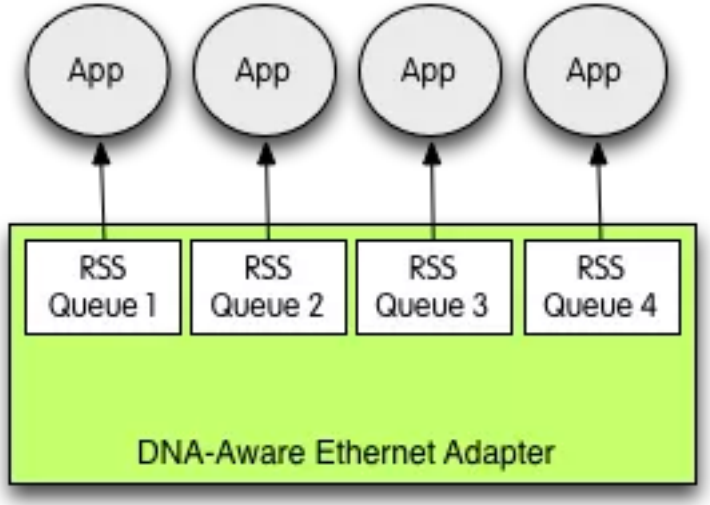
Down the rabbit hole: BPFs and eBPFs



RSS

RSS: Receive Side Scaling

- Often used in real hardware to distribute flows to multiple queues
  - Myricom libpcap, PF_RING/DNA/ZC, Intel x710
- Also works with virtio and vmxnet3 in virtual world



http://www.ntop.org/pf_ring/hardware-based-symmetric-flow-balancing-in-dna/

Virtual RSS

- Tell hypervisor to add multiple queues per virtual NIC (libvirt: virtio device, queues=N)
- Tell guest to use multiple queues (Linux: ethtool –L)
- Each queue gets associated with a CPU
- Pin workers to CPUs

Multi-NIC

Virtual Switch can expose many vNICs

- Create N vNICs
- Flow hash over N vNICs
- N Bro workers read from N vNICs

Examples:

- Feed incoming packets to Openvswitch
- Use OpenFlow group/select tables to hash over flows
- Requires OVS 2.7+ and Netronome extension for 5-tuple hash
  - add-group command, selection_method=dp_hash

AF_PACKET

AF_PACKET is in-kernel (Linux) packet delivery mechanism

- ***Hardware agnostic!***
- PF_RING, netmap, and others require specific drivers/NICs
- AF_PACKET all in-kernel, agnostic to vNIC
- DPDK acceleration, including through virtual switch
- Still some growing pains: kernels, patches, etc.

Elasticity and Scalability

It’s the cloud, unlimited resource!

- Vertical scalability: Bro on a single node
- Horizontal scalability: More Bro nodes
  - Tree of Bros
  - Distribute traffic across all Bros
  - Dynamically scale more Bros when load goes up/down
  - Assume failures

Measuring Bro…

Measurement of a vBro

- Many places for packet drops
- Need multiple measurement points
- Variability due to host/hypervisor/guest interactions

A dropped packet defined

- A dropped packet
  - Prior layer had no room for packet
  - Generally, ring buffer full: HW or SW
- Examples
  - HW gets packet off the wire, internal buffers full. HW packet drop
  - SW ring is full, SW drops packet.
    - Depending on arch, backpressure may then also cause HW drops
  - Host delivers to hypervisor, hypervisor to guest
    - Drops on host NIC, drops on guest vNIC

Measurement

- Understand packet path
- Each buffer, ring, FIFO, is an opportunity to drop a packet
- Try to measure at each point
- Validate with upstream switch/tap agg

- Example:
  - Define a set timeframe: a few minutes
  - Switch port delivers X packets or tcpreplay pcap with X packets
  - Add all drop points, all receive points. Does it add up?
  - When it doesn't add up, there's a buffer in the path missing
  - This is hard!

Measurement in the cloud

- Limited visibility upstream
- Use any info available for upstream “truth”
- Often Netflow(ish) or similar is available (AWS VPC Flow Logs)
- Difficult to find true drop rates

END

commike@reservoir.com

https://www.reservoir.com/r-scope-vm-preview/

(and also: We’re hiring!)