mozilla

They did not know what hit them

Network Security Monitoring at Mozilla

"I bought you those servers to run NSM on them" <- Boss, 2012

New servers <- always cool

The big idea

NSM = Network Security Monitoring

Write arbitrary detection logic

Store metadata about connections

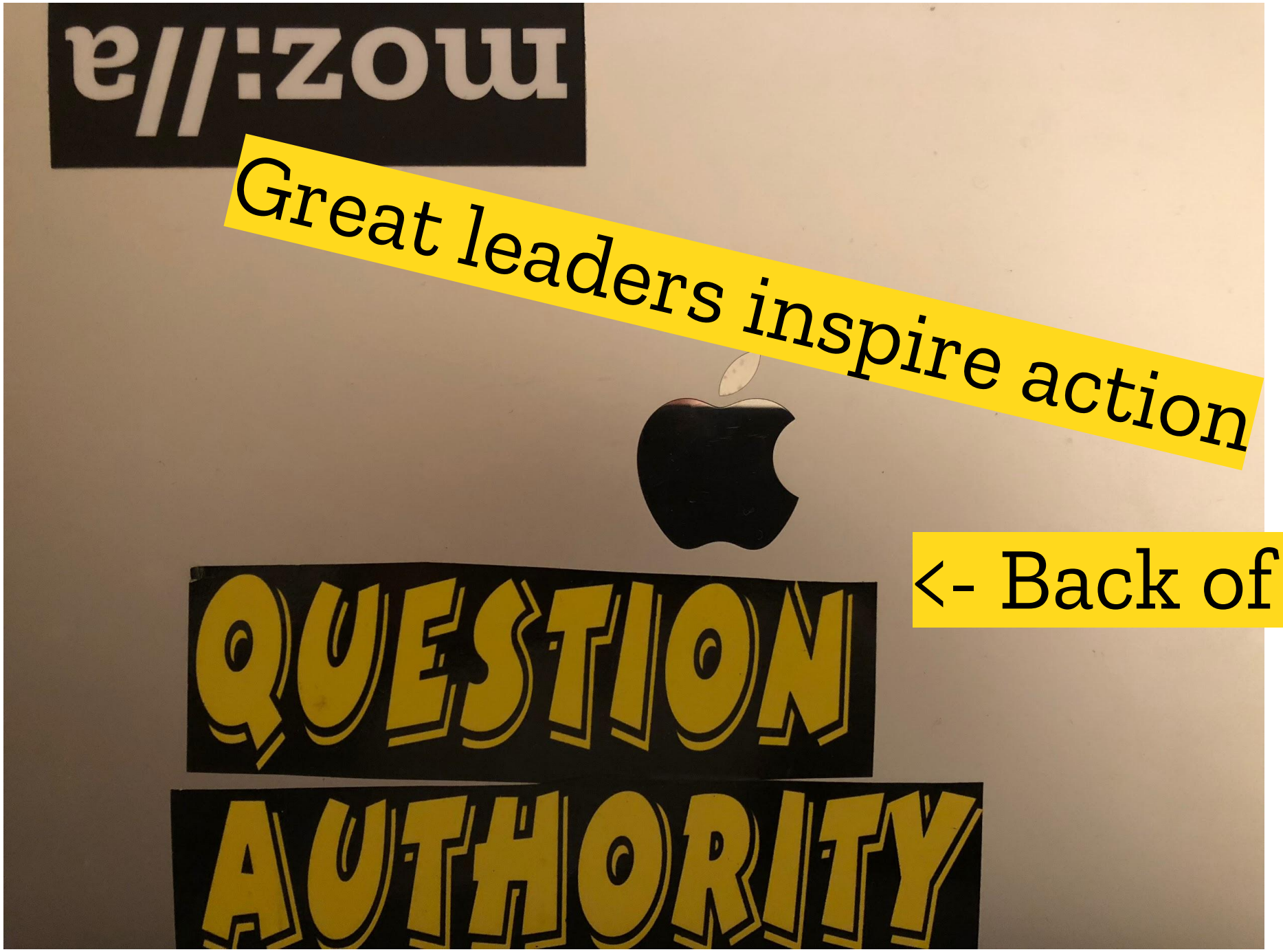
"You want to do IDS in 2012?"

"What is this bro/zeek that takes CPU from snort?"

Everything is encrypted

Not at our scale

Cannot be done



Great leaders inspire action

<- Back of my laptop

Not a silver bullet

Here is why we like it

Logs

Record threat actor's activity

DFIR

Past

Zeek

IOC

Match IOCs in a creative way

DFIR and detection

Past and present

Zeek

TTP

Do the TTP detection

Detection

Present

Zeek + Suricata

To answer

The most important question

Are we owned?

Mozilla's Threat Management response

To a new APT report

Zeek, Suri, Auditd, Syslog, application

Learn how to build a nice Zeek sensor

Learn how to improve what you have

Your monitoring is wrong ;)

"...but you promised AF_Packet!!"

AF_Packet

Mozilla NSM architecture

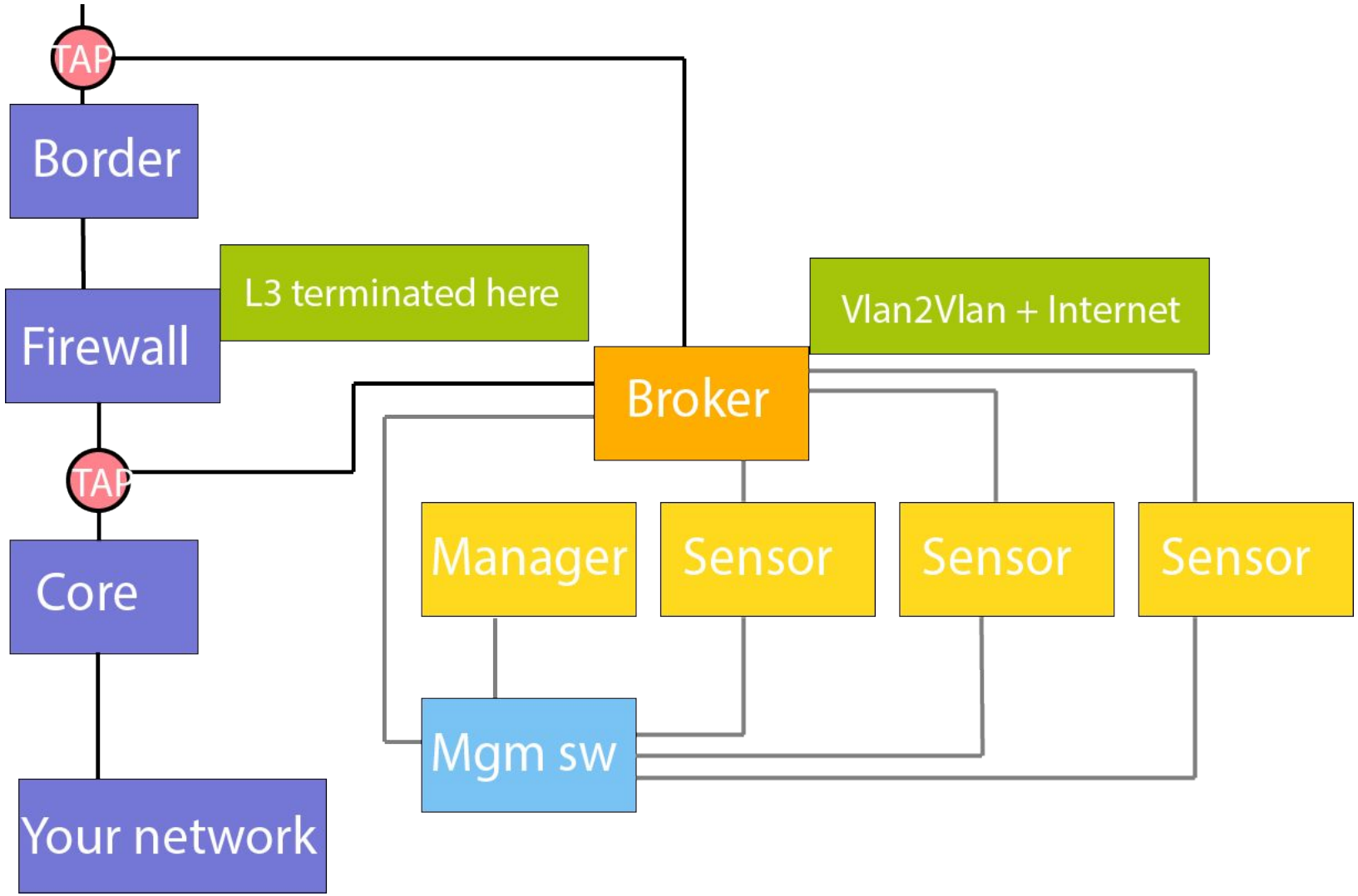
10 000 events / second

syslog-ng -> MozDef

ClearLinux

3 datacenters, 9 offices
AWS, GCE (??)

Europe, North America, Asia



Mozilla NSM Sensor (Mark VI ;)

CPU - 2x Intel Xeon

2 x 6 x 16GB DIMM <- all memory channels populated 1DPC

NUMA0 <- Intel X710-DA2 (i40e) / Mellanox ConnectX-4 Lx (mlx5)

NUMA1 <- Intel X710-DA2 (i40e) / Mellanox ConnectX-4 Lx (mlx5)

Mozilla + Suricata developers research

Hardware acceleration??

Maybe for bitcoin

Dual Xeons + Intel X710 + 128GB RAM

Suricata - 40Gbit/sec

No packet loss

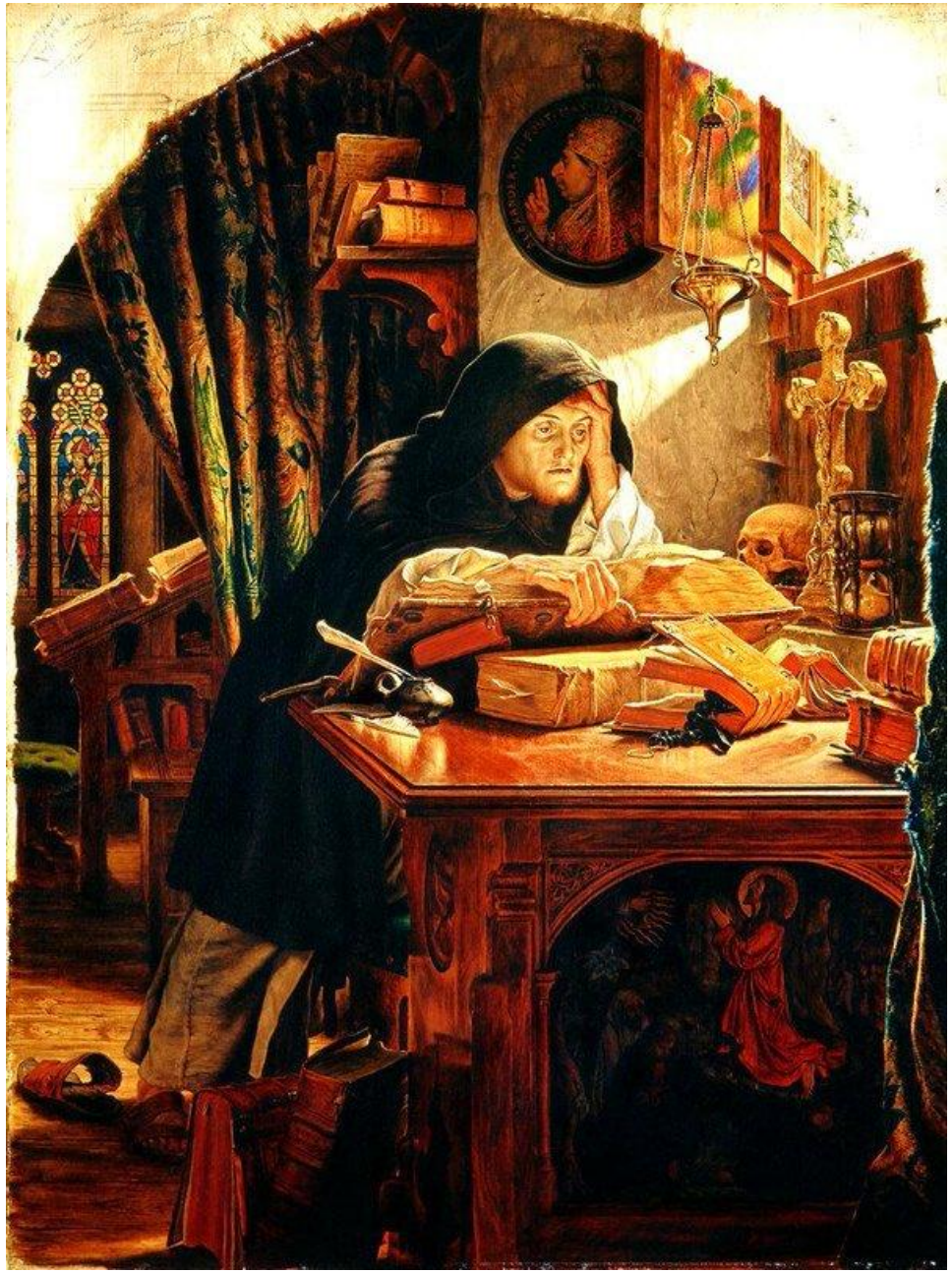
40 000 rules inspecting Vlan2Vlan traffic

Linux + AF_Packet

https://github.com/pevma/SEPTun

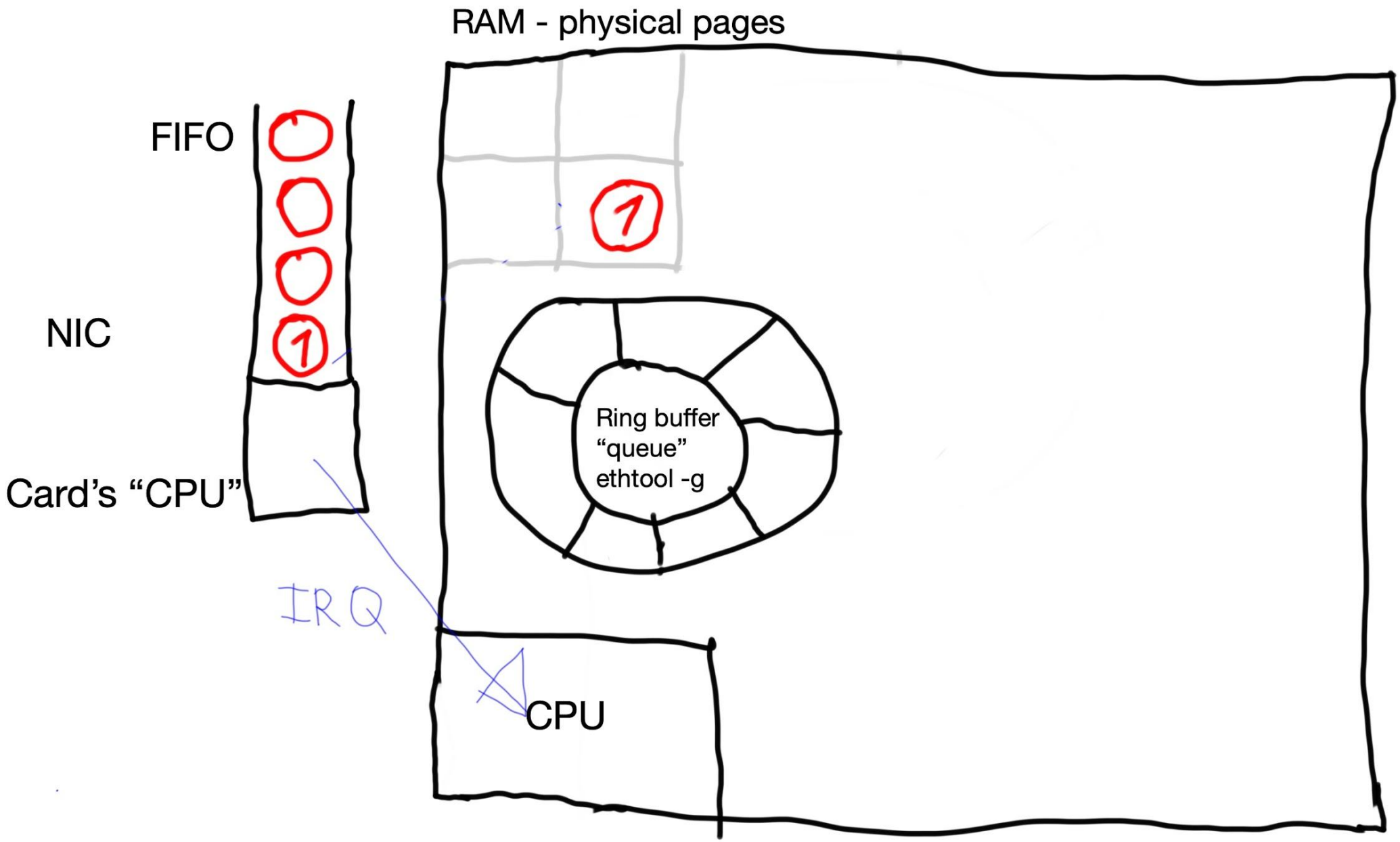
https://github.com/pevma/SEPTun-Mark-II

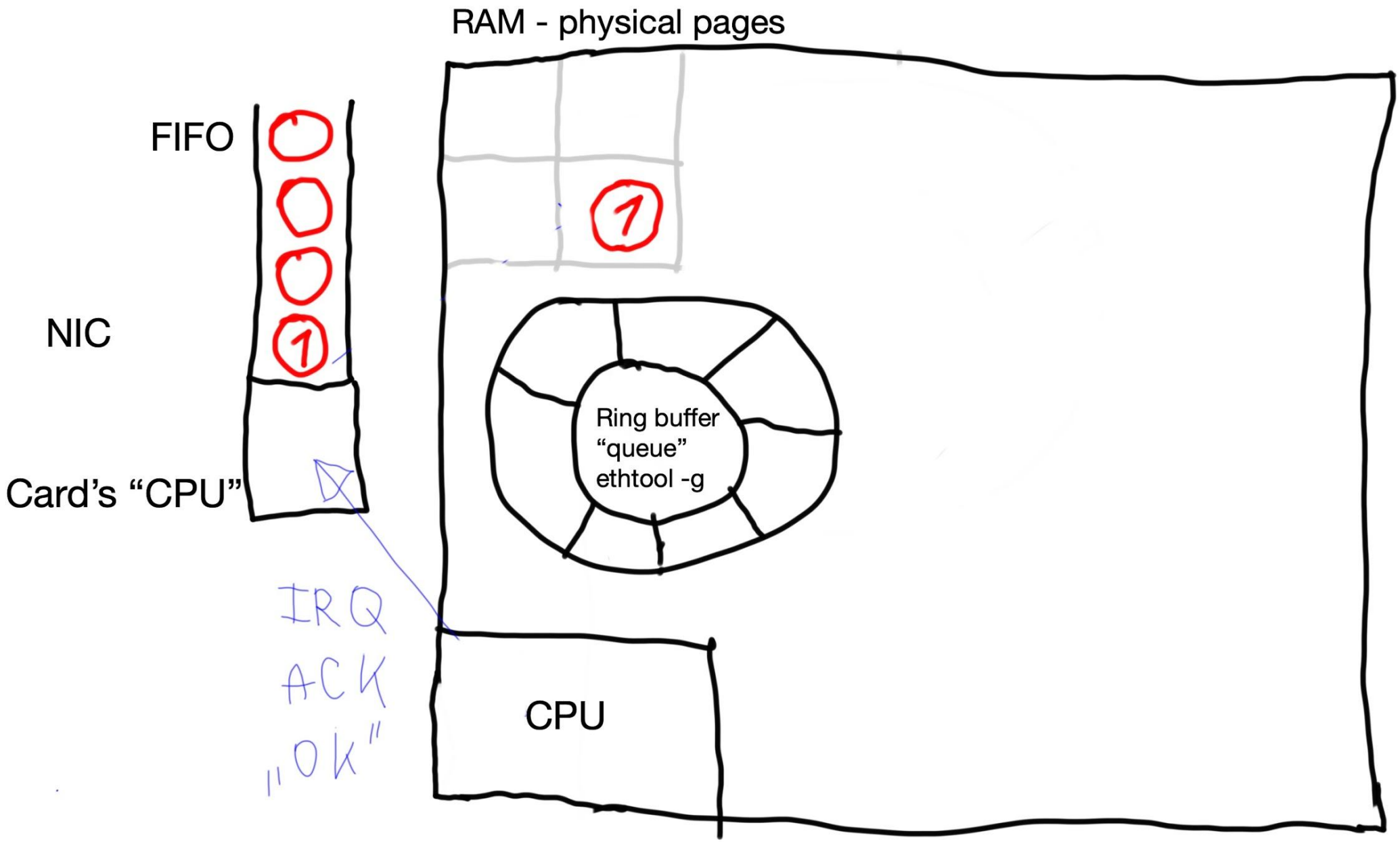
Developer looking at production logs after a regression with downtime.
Oil canvas, circa 1580

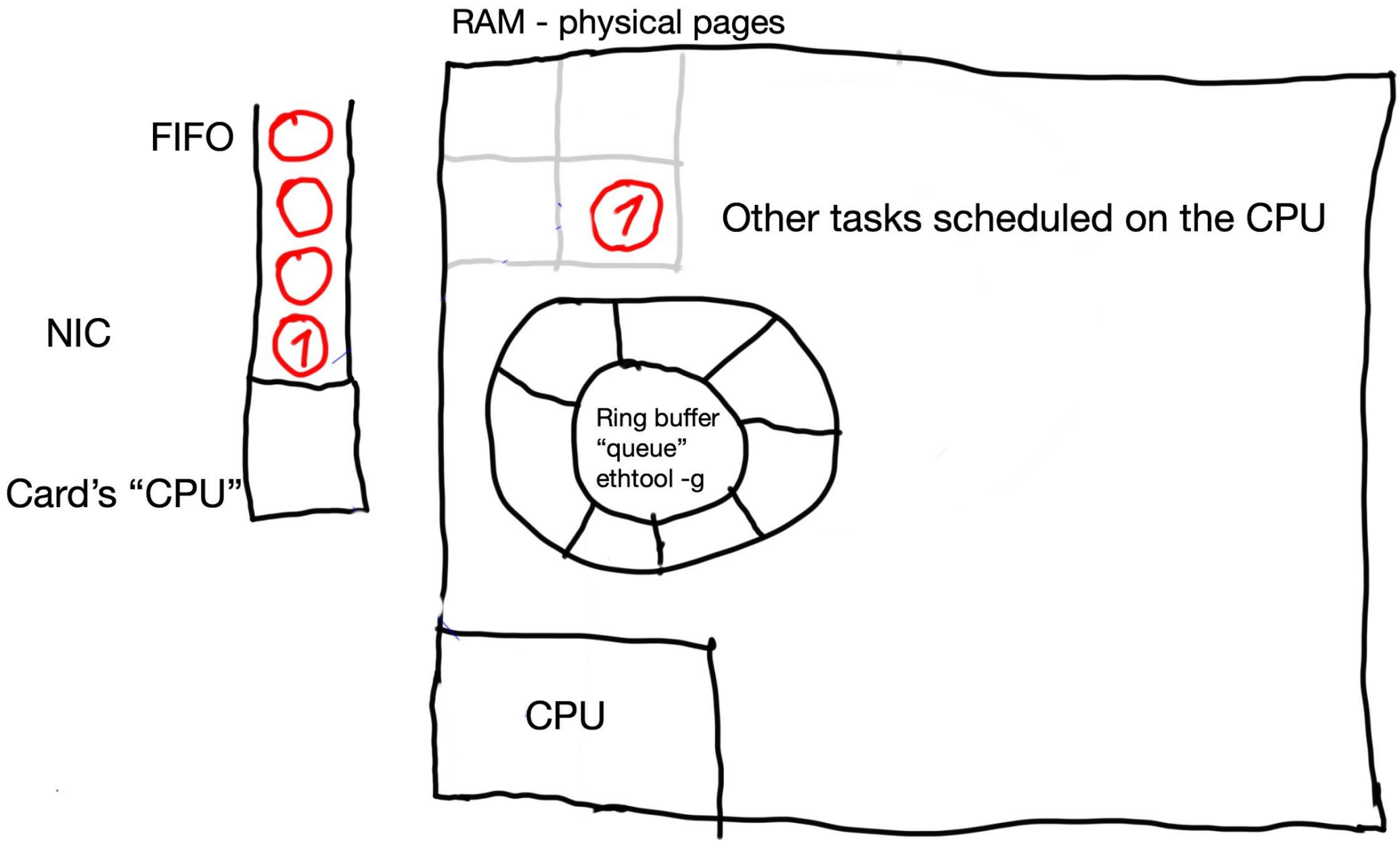


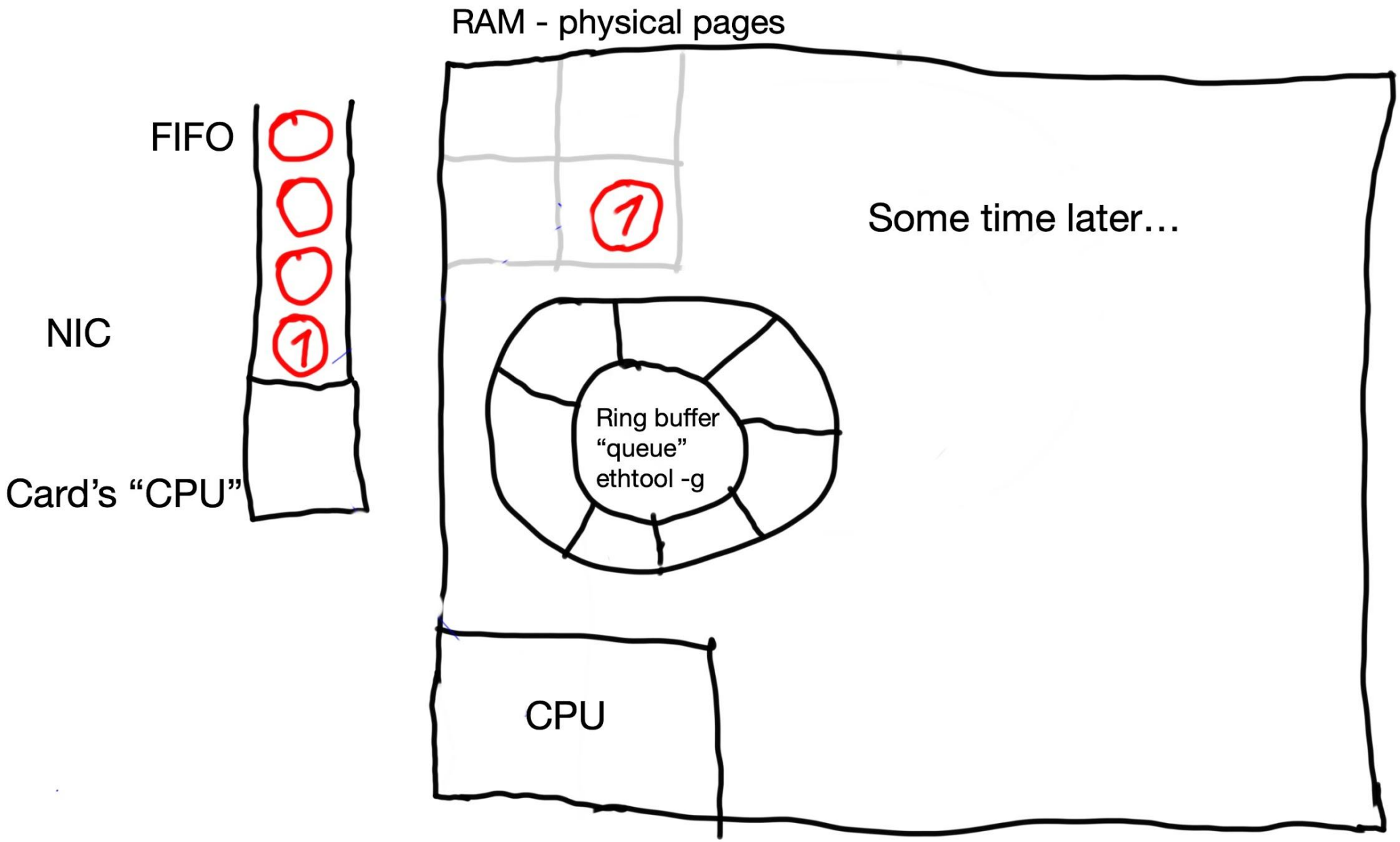
Overheard: looks like Michal

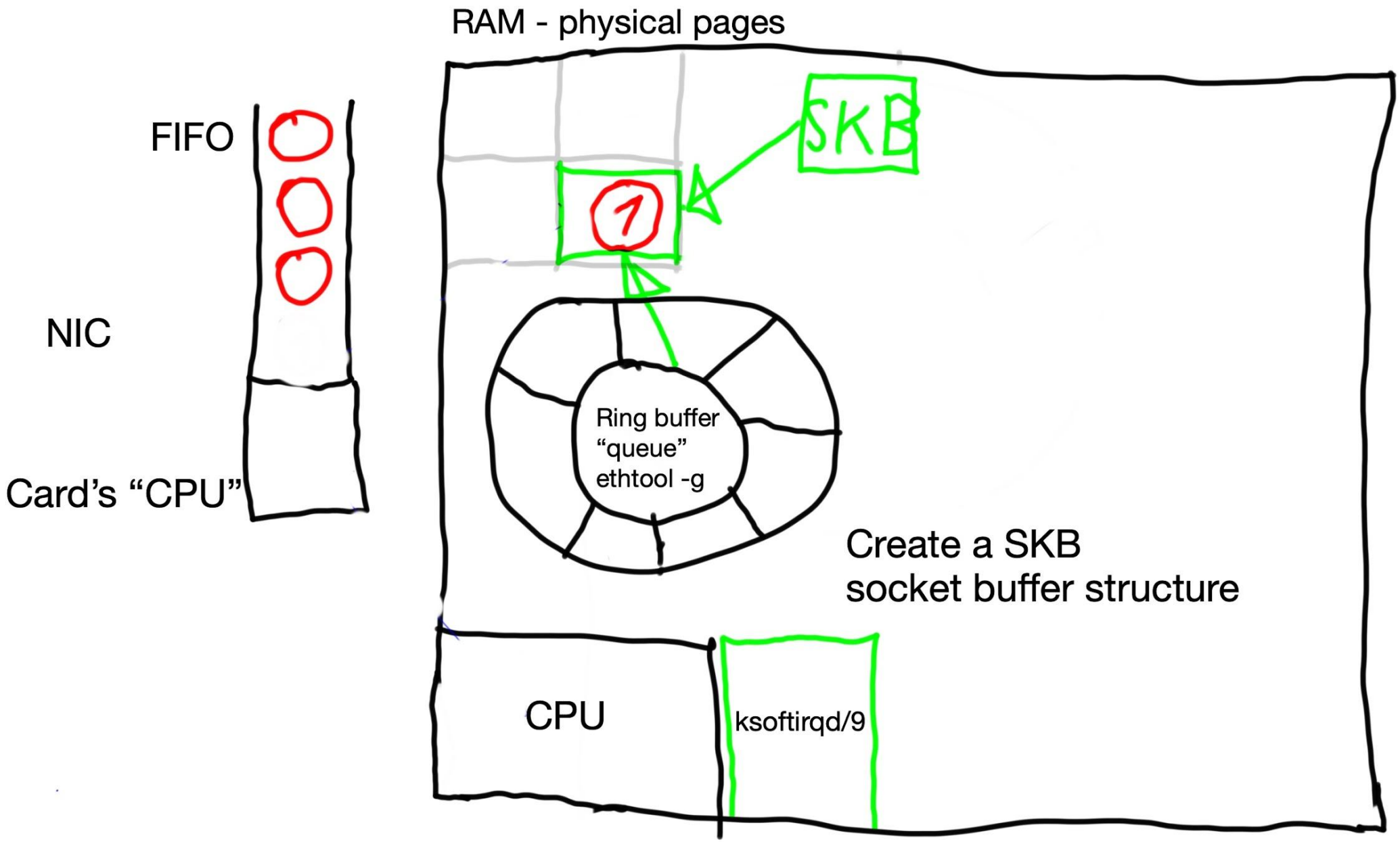
Modern OS - Linux 2.4+, Windows NT+, etc

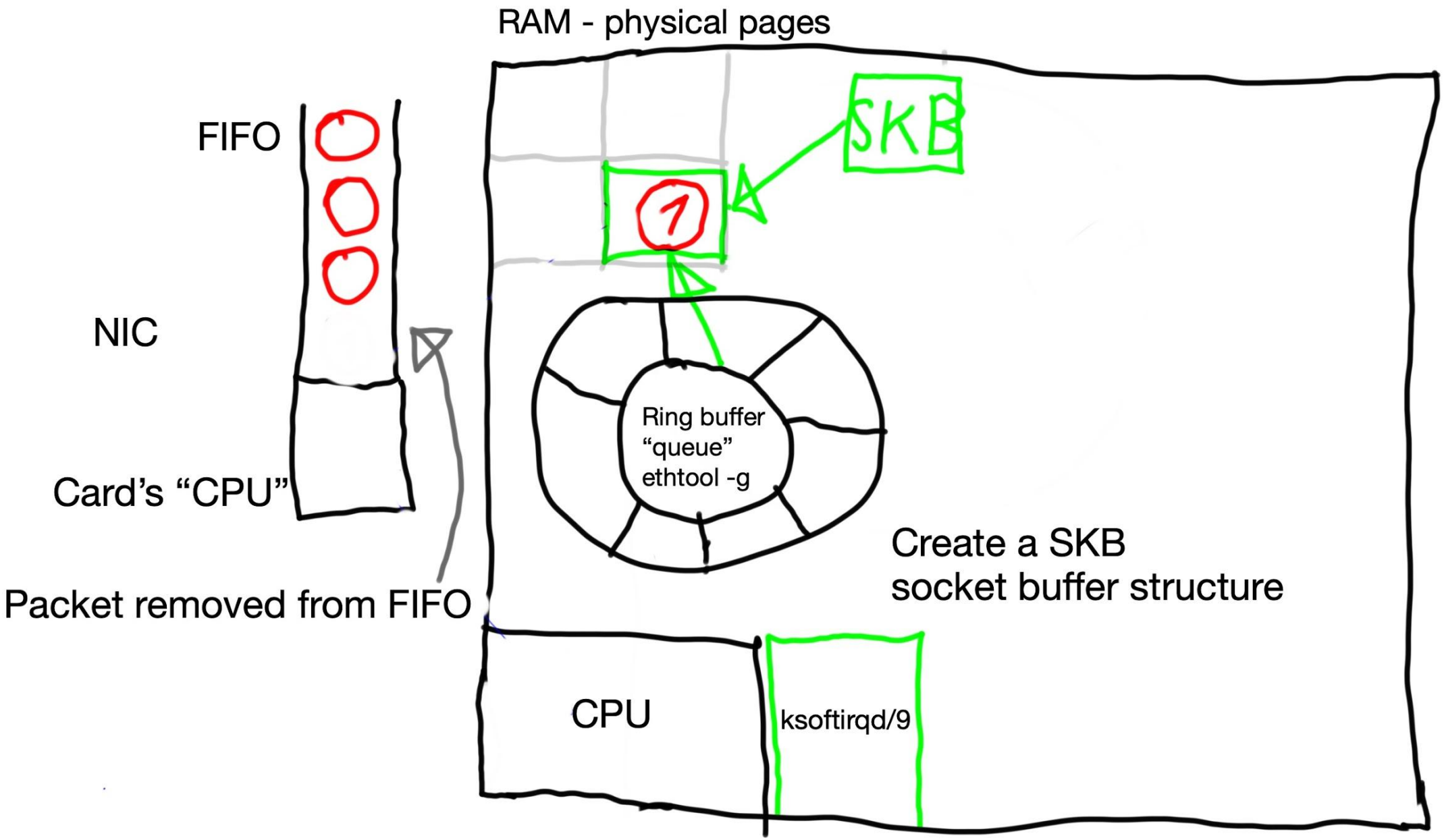

RAM - physical pages
FIFO
NIC
Card's "CPU"
IRQ
Ring buffer
"queue"
ethtool -g
CPU

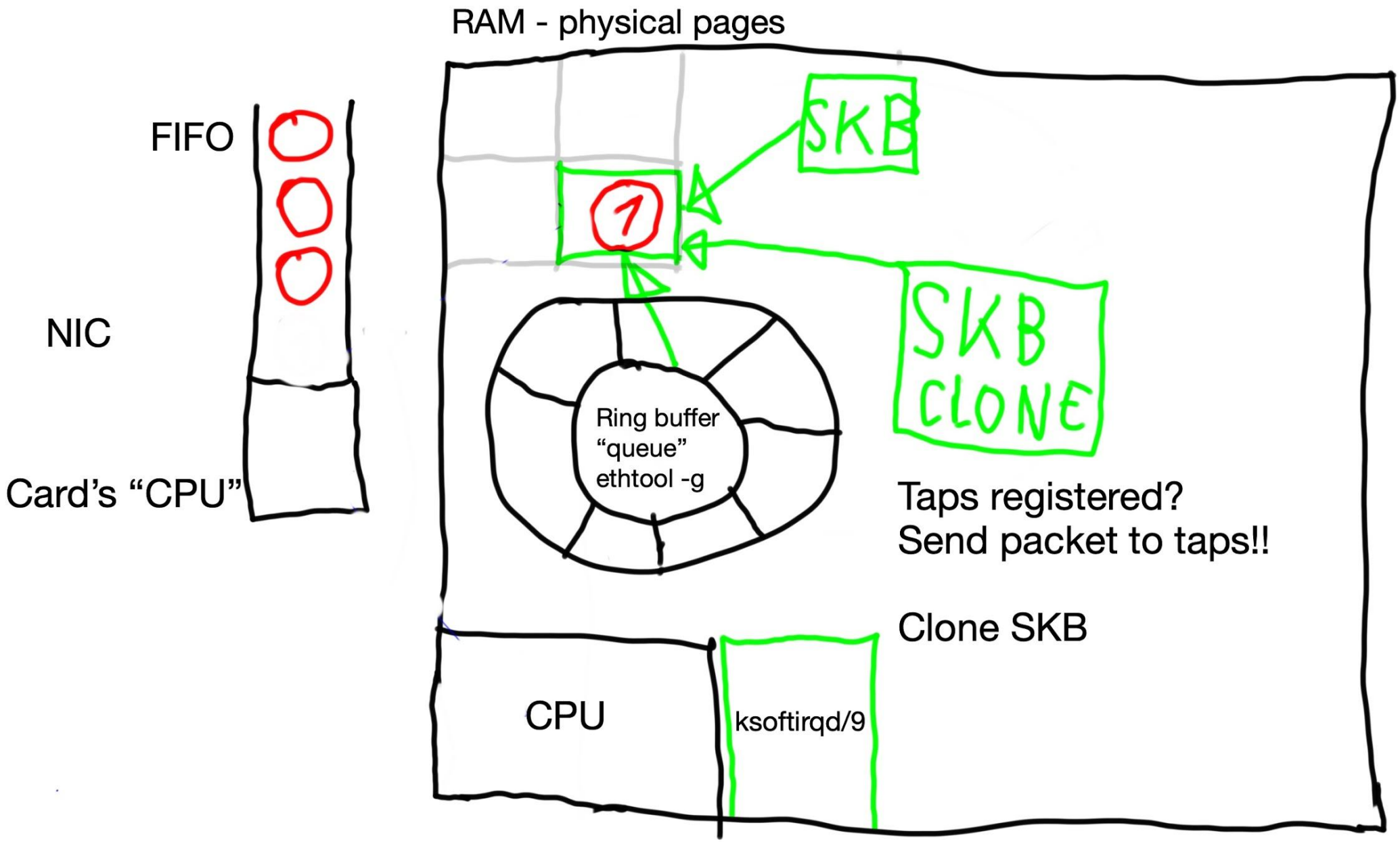

RAM - physical pages
FIFO
NIC
Card's "CPU"
Ring buffer
"queue"
ethtool -g
CPU
IRQ
ACK
"OK"

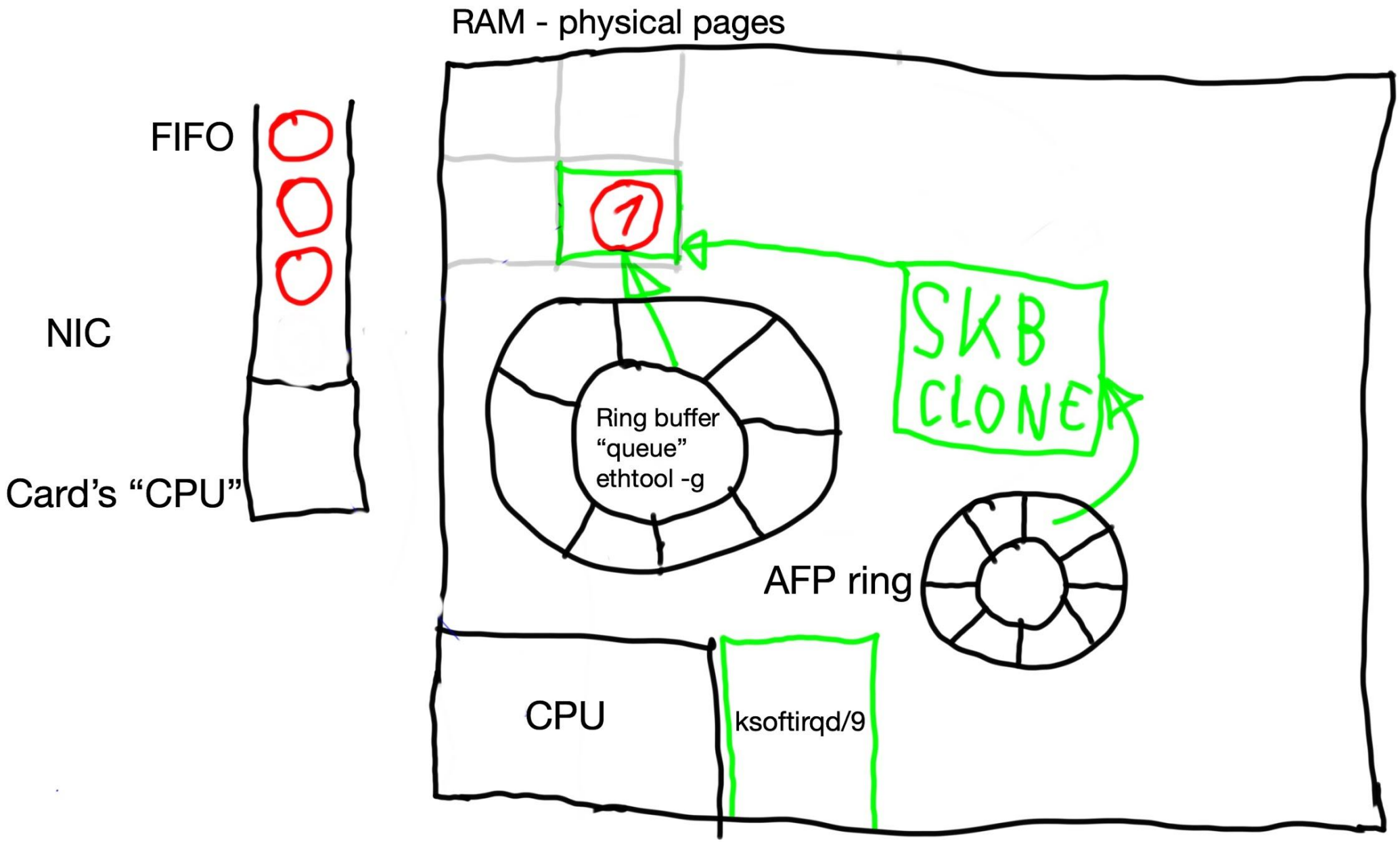

RAM - physical pages
FIFO
NIC
Card's "CPU"
Other tasks scheduled on the CPU
Ring buffer
"queue"
ethtool -g
CPU

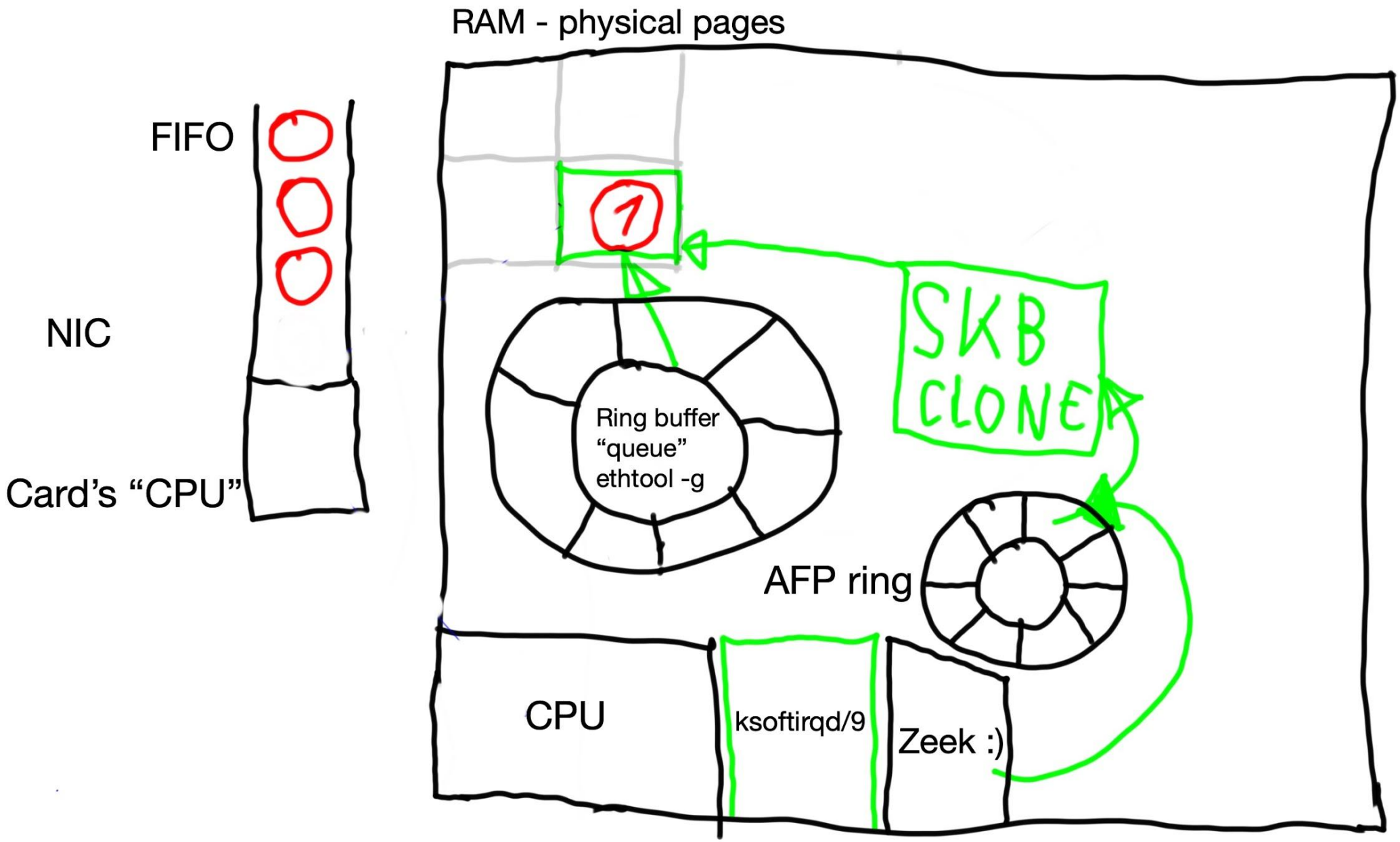

RAM - physical pages
FIFO
NIC
Card's "CPU"
Some time later…
Ring buffer
"queue"
ethtool -g
CPU


RAM - physical pages
FIFO
NIC
Card's "CPU"
SKB
1
Ring buffer
"queue"
ethtool -g
Create a SKB
socket buffer structure
CPU
ksoftirqd/9


RAM - physical pages
FIFO
NIC
Card's "CPU"
Packet removed from FIFO
SKB
1
Ring buffer
"queue"
ethtool -g
Create a SKB
socket buffer structure
CPU
ksoftirqd/9


RAM - physical pages
FIFO
NIC
Card's "CPU"
SKB
1
SKB
CLONE
Ring buffer
"queue"
ethtool -g
Taps registered?
Send packet to taps!!
Clone SKB
CPU
ksoftirqd/9


RAM - physical pages
FIFO
NIC
Card's "CPU"
1
Ring buffer
"queue"
ethtool -g
SKB
CLONE
AFP ring
CPU
ksoftirqd/9

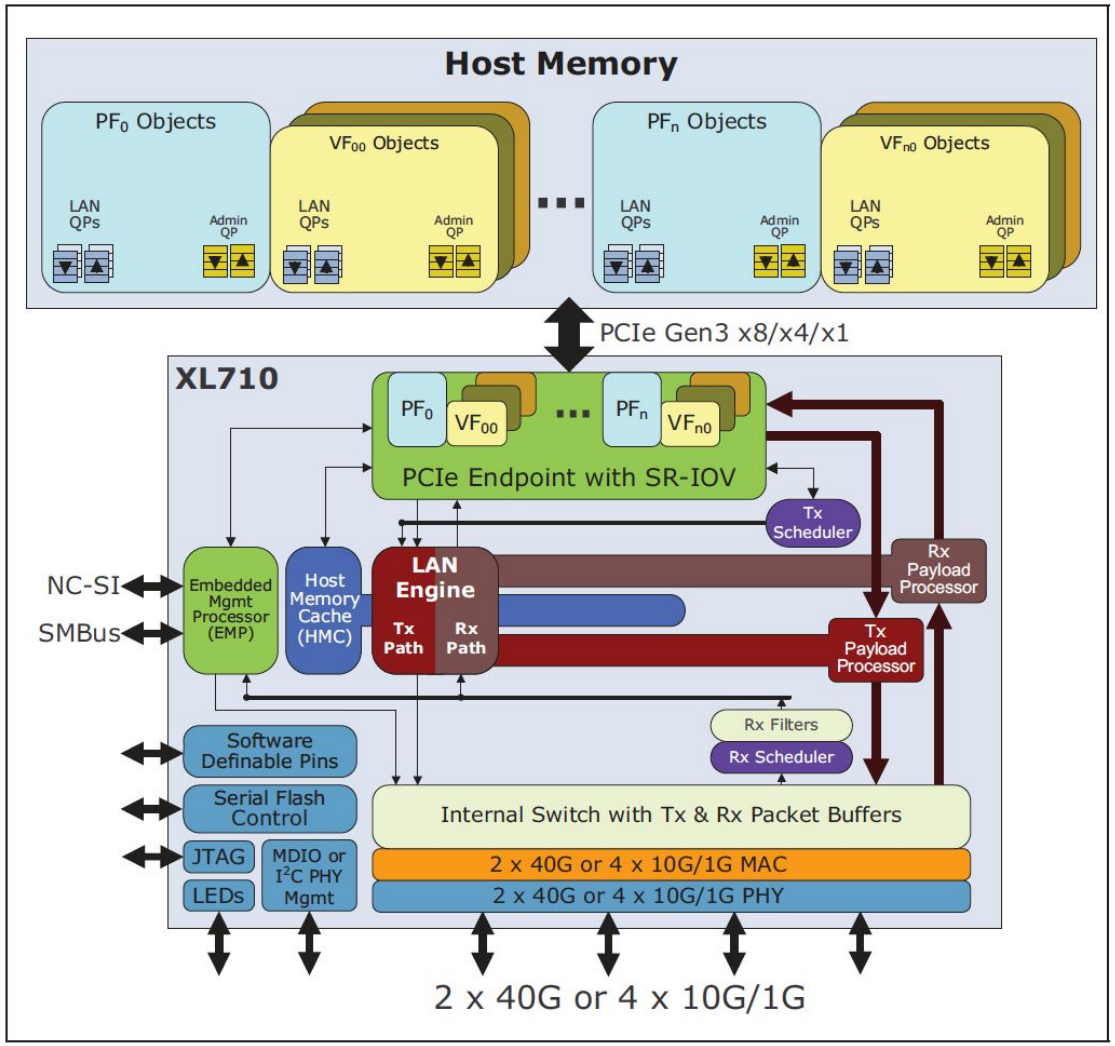

RAM - physical pages
FIFO
NIC
Card's "CPU"
1
Ring buffer
"queue"
ethtool -g
SKB
CLONE
AFP ring
CPU
ksoftirqd/9
Zeek :)

Modern cards datacenter in a box

X710 integrated managed switch

and 384 vNICs

And you can access all of this power :)



It is all about per-packet latency

It is NOT about zero copy!!

Netmap papers

Thanks Luigi Rizzo

What does eat time per packet?

Cache thrashing

Userspace -> kernel transitions

TLB thrashing

67ns to process a packet

200 cycles

Findings

Cache access timings, approximate

Local L3 - 20ns

Local RAM - 100ns

Remote L3 - 80ns

Remote RAM - 140ns

Findings

IPC - instructions per clock cycle

Before tuning - 0.7

After tuning - 2.7

Theoretical limit - 4.0

Intel DDIO

Packet arrives to card's FIFO

Card sends packets to the cache <- pre-warms the CPU cache

Hang-on to it!!

The Grand Plan - in English

Send all packets 10.1.2.3 <-> 8.8.8.8 to core 2

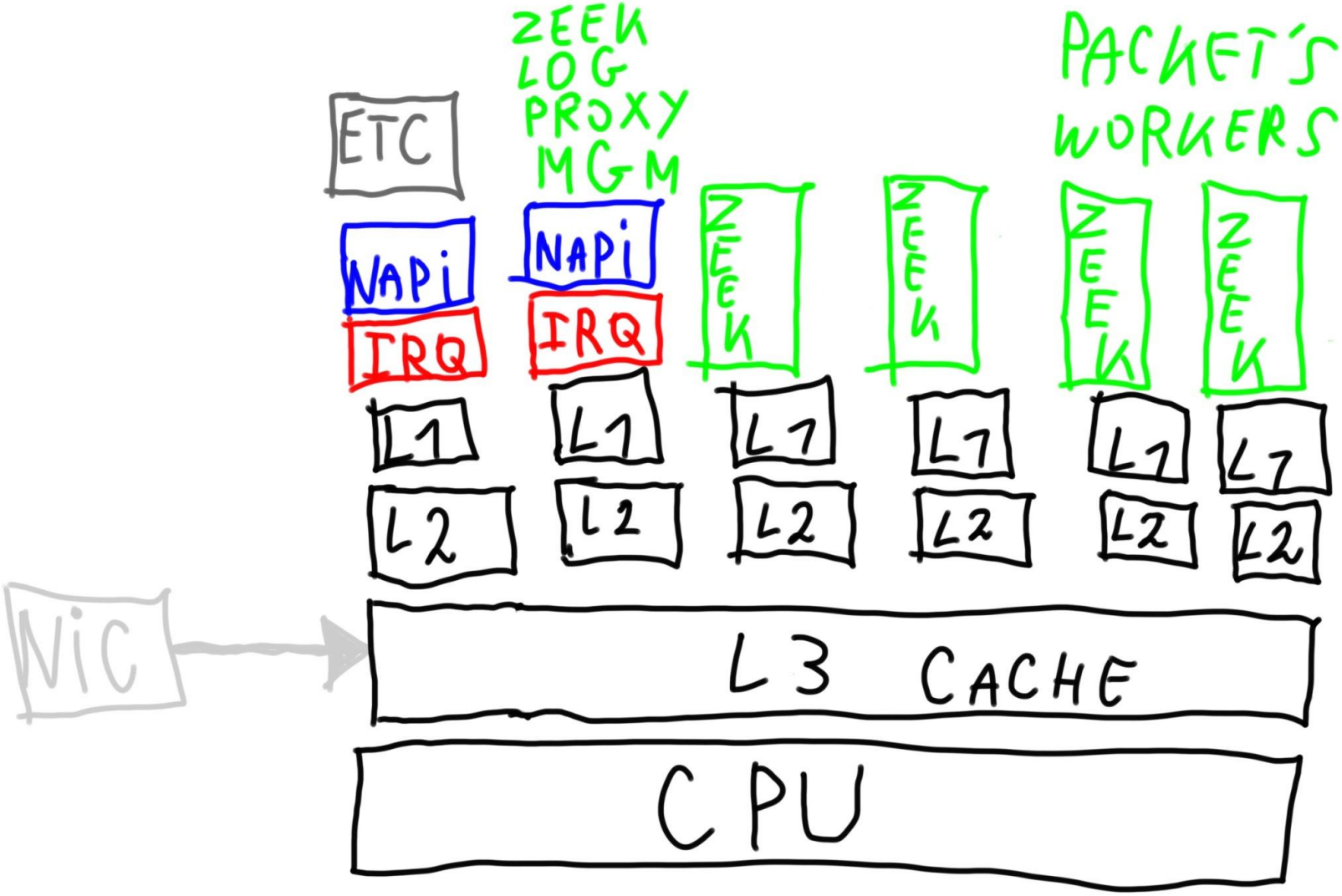
Zeek packets 10.1.2.3 <-> 8.8.8.8 on core 9

Dedicate cores for IRQ/SoftIRQ processing

Establish Zeek Worker cores

Achieve eternal happiness

The Grand Plan - in drawings (sorry ;)



Symmetric hashing

In software - AF_Packet - cluster_flow <- cannot configure

In software - AF_Packet - cluster_ebpf <- new hotness

In hardware - AF_Packet - cluster_qm

Software has fragmentation problems :(

Hardware is flexible :)

Who’s deciding? ATR? PF? RSS?

ATR - if enabled AND no Perfect Filters

Perfect Filters - if any

RSS - your fallback

ATR. Disable. It's out of order ;)

```
root@nsm1~ # ethtool --set-priv-flags enp17s0f0 flow-director-atr off
root@nsm1~ # ethtool --show-priv-flags enp17s0f0
Private flags for enp17s0f0:
MFP                     : off
LinkPolling             : off
flow-director-atr       : off
veb-stats               : off
hw-atr-eviction         : off
link-down-on-close      : off
legacy-rx               : off
disable-source-pruning  : off
disable-fw-lldp         : off
rs-fec                  : off
base-r-fec              : on
vf-true-promisc-support: off
```

NTuple AKA Too Perfect Filters

```
root@nsm1~ # ethtool -N enp17s0f0 flow-type udp4 src-port 514 action -1
Added rule with ID 1023
root@nsm1~ # ethtool -N enp17s0f0 flow-type udp4 dst-port 514 action -1
Added rule with ID 1022
root@nsm1~ # ethtool -n enp17s0f0
4 RX rings available
Total 2 rules

Filter: 1022
        Rule Type: UDP over IPv4
        Src IP addr: 0.0.0.0 mask: 255.255.255.255
        Dest IP addr: 0.0.0.0 mask: 255.255.255.25
        TOS: 0x0 mask: 0xff
        Src port: 0 mask: 0xffff
        Dest port: 514 mask: 0x0
        Action: Drop

Filter: 1023
        Rule Type: UDP over IPv4
        Src IP addr: 0.0.0.0 mask: 255.255.255.255
        Dest IP addr: 0.0.0.0 mask: 255.255.255.25
        TOS: 0x0 mask: 0xff
        Src port: 514 mask: 0x0
        Dest port: 0 mask: 0xffff
        Action: Drop
```

```
root@nsm1~ # ethtool -N enp17s0f0 flow-type udp4 dst-port 514 action -1
Added rule with ID 7679
root@nsm1~ # ethtool -N enp17s0f0 flow-type udp4 src-port 514 action -1
rmgr: Cannot insert RX class rule: Operation not supported
root@nsm1~ # ethtool -n enp17s0f0
4 RX rings available
Total 1 rules

Filter: 7679
        Rule Type: UDP over IPv4
        Src IP addr: 0.0.0.0 mask: 255.255.255.255
        Dest IP addr: 0.0.0.0 mask: 255.255.255.255
        TOS: 0x0 mask: 0xff
        Src port: 0 mask: 0xffff
        Dest port: 514 mask: 0x0
        Action: Drop
```

RSS - what is hashed?

```
root@nsm2~ # for i in udp4 udp6 tcp4 tcp6; do ethtool -n enp7s0f1 rx-flow-hash
$i; done;
UDP over IPV4 flows use these fields for computing Hash flow key:
IP SA
IP DA

UDP over IPV6 flows use these fields for computing Hash flow key:
IP SA
IP DA

TCP over IPV4 flows use these fields for computing Hash flow key:
IP SA
IP DA

TCP over IPV6 flows use these fields for computing Hash flow key:
IP SA
IP DA
```

RSS - how is it hashed?

```
RSS hash key:
6d:5a:6d:5a:6d:5a:6d:5a:6d:5a:6d:5a:6d:5a:6d:5a:6d:5a:6d:5a:6d:5a:6d:5a:6
d:5a:6d:5a:6d:5a:6d:5a:6d:5a:6d:5a:6d:5a
RSS hash function:
    toeplitz: on
    xor: off
    crc32: off
root@nsm2~ # ethtool -x enp7s0f1^C
```

ethtool -K enp7s0f1 ntuple on ; ethtool -K enp7s0f1 rxhash on

for i in tcp4 udp4 tcp6 udp6; do ethtool -U enp7s0f1 rx-flow-hash $i sd; done;

ethtool -X enp7s0f1 hkey \
6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6
D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A:6D:5A equal 4

RSS - how is the hash used?

```
root@nsm2~ # ethtool -x enp7s0f1
RX flow hash indirection table for enp7s0f1 with 6 RX ring(s):
    0:      0     1     2     3     4     5     0     1
    8:      2     3     4     5     0     1     2     3
   16:      4     5     0     1     2     3     4     5
   24:      0     1     2     3     4     5     0     1
   32:      2     3     4     5     0     1     2     3
   40:      4     5     0     1     2     3     4     5
   48:      0     1     2     3     4     5     0     1
   56:      2     3     4     5     0     1     2     3
   64:      4     5     0     1     2     3     4     5
   72:      0     1     2     3     4     5     0     1
   80:      2     3     4     5     0     1     2     3
   88:      4     5     0     1     2     3     4     5
   96:      0     1     2     3     4     5     0     1
  104:      2     3     4     5     0     1     2     3
  112:      4     5     0     1     2     3     4     5
  120:      0     1     2     3     4     5     0     1
```

Hashing consistency

cluster_flow may have problems with fragments

cluster_qm -> RSS

RSS cannot handle fragments (nothing can :)

Hash 3-tuple

Also true for your packet broker!!

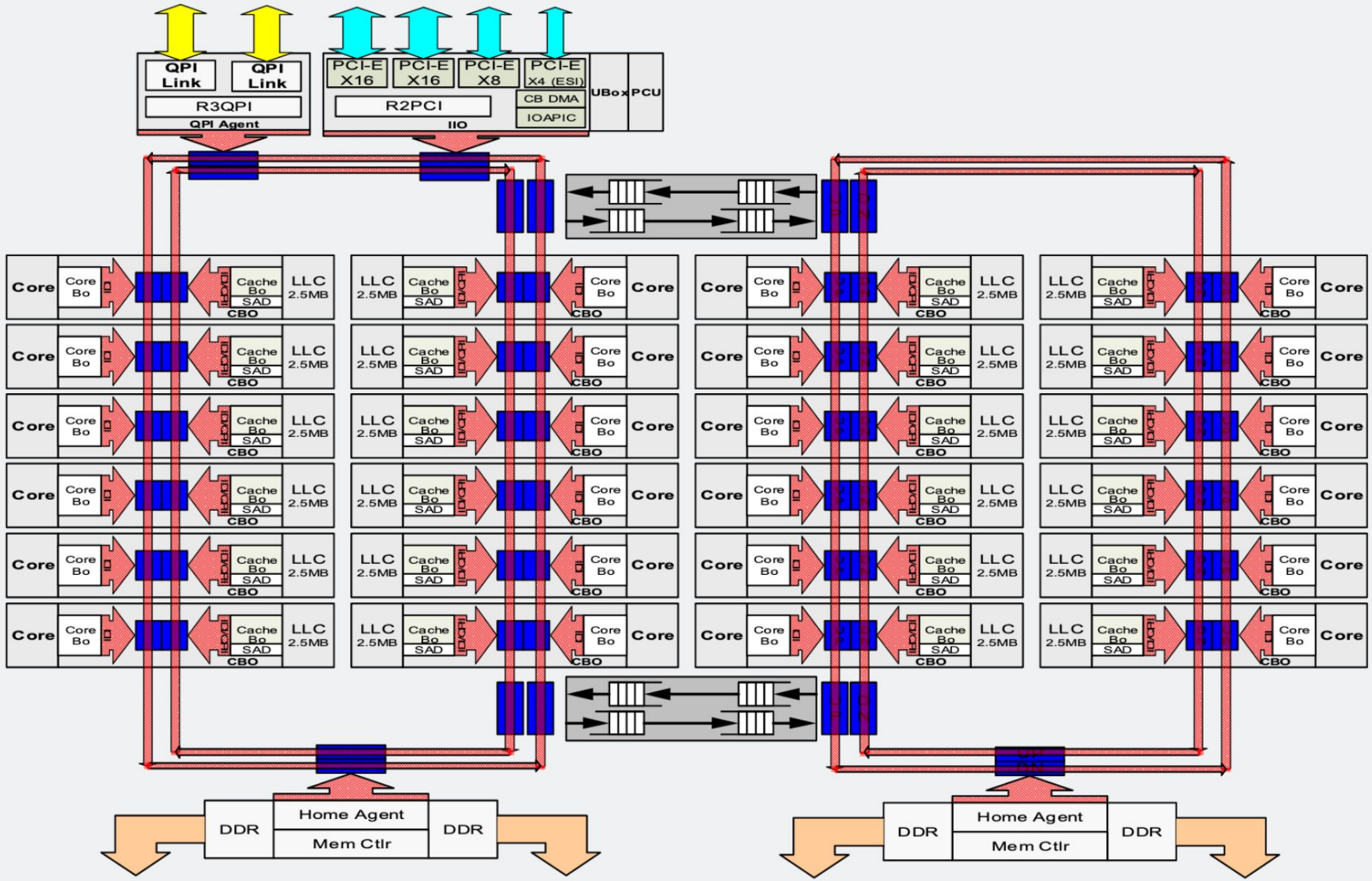
Findings

Smaller amount of faster cores <- good

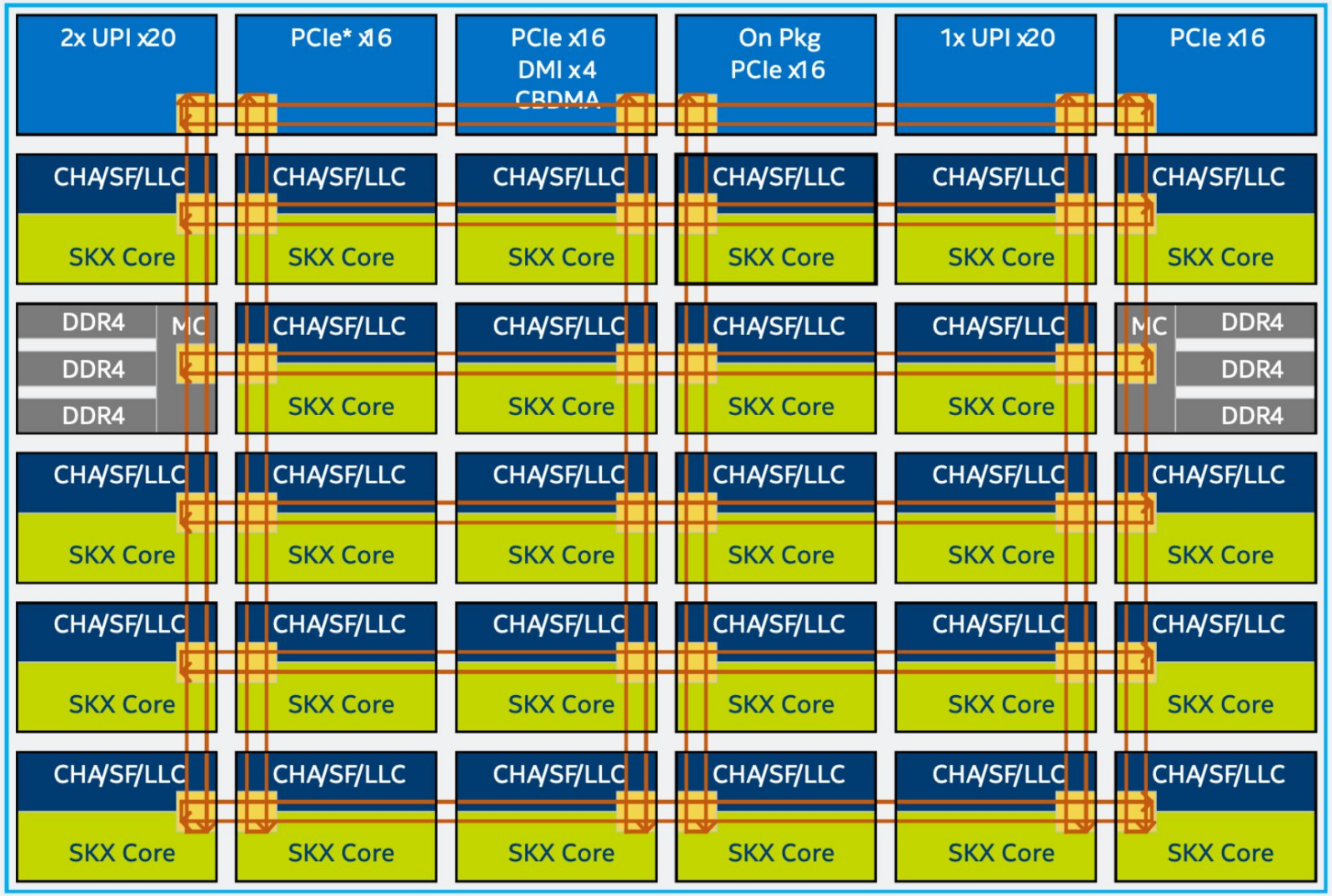
vs

High core count <- sometimes bad ;)

2016: INTEL® XEON® PROCESSOR E7 V4, 14NM (BROADWELL EX 24-CORE DIE)



2017: INTEL® XEON® SCALABLE PROCESSOR, 14NM (SKYLAKE-SP 28-CORE DIE)



CHA– Caching and Home Agent; SF– Snoop Filter; LLC– Last Level Cache
SKX Core– Skylake Server Core; UPI– Intel® UltraPath Interconnect

Findings

Cache coherence protocol

Use Early Snooping

Findings

Cluster On-Die

Sub NUMA Clustering

Disable

Findings

Limit C-states to C1

Leave C-states enabled for Turbo Boost

Disable P-states

Findings

Use HyperThreading for Zeek Workers logical cores

Findings

Use all memory channels. But there's more.

2DPC (2Rx8) - 2x 8GB / channel

(3DPC reduces frequency pre-Skyline)

Keep DIMMs at the same size

Use dual ranks (but don't sweat it + watch for frequency)

Findings

Lower the number of buffers

ethtool <ethX> rx 512

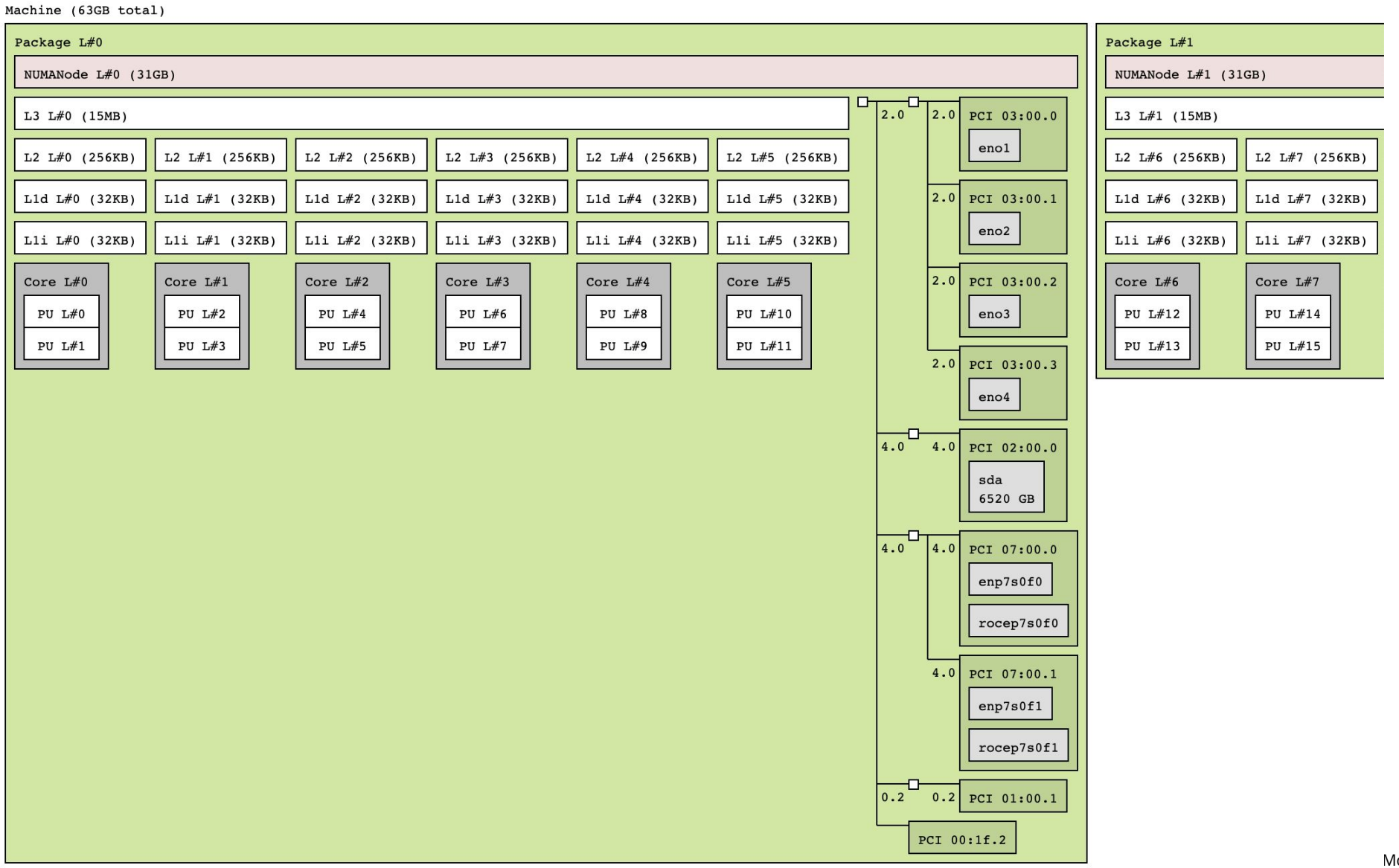
Discover the architecture

```
find /sys/devices/system/cpu/cpu0/cpuidle -name latency
-o -name name | xargs cat

numactl --hardware
lscpu

ls -ld /sys/devices/system/node/node*
cat /sys/devices/system/node/node0/cpulist
cat /sys/class/net/eth3/device/numa_node
egrep "CPU0|eth3" /proc/interrupts
```

lstopo --of svg -p --no-factorize > /tmp/o1.svg



Your checklist

ethtool -i <int> <- update firmware

mlxup for Mellanox

nvmupdate64e for Intel

Keep kernel updated

Use upstream driver. Forget sourceforge.

Configure the kernel

intel_iommu=off (or pt)

intel_idle.max_cstates=1 (or cpudmalatency.c)

pcie_aspm=off

isolcpus=4-21,32-48 <- reserve core 0-3 on each NUMA node
nohz_full=4-21,32-48 (<- does nothing for Zeek ;)
rcu_nocbs=4-21,32-48

Set IRQ and SoftIRQ affinity

```
root@nsm1~ # ./set_irq_affinity 0-3 enp17s0f0
IFACE CORE MASK -> FILE
=======================
enp17s0f0 0 1 -> /proc/irq/77/smp_affinity
enp17s0f0 1 2 -> /proc/irq/78/smp_affinity
enp17s0f0 2 4 -> /proc/irq/79/smp_affinity
enp17s0f0 3 8 -> /proc/irq/80/smp_affinity
```

Configure Zeek

```
mpurzynski@nsm2~ $ sudo cat /etc/zeek/config/node.cfg
[nsm2-manager]
type=manager
host=localhost
pin_cpus=0

[nsm2-logger]
type=logger
host=localhost
pin_cpus=1

[nsm2-proxy1]
type=proxy
host=localhost
pin_cpus=0

[nsm2-af_packet-enp7s0f1]
type=worker
host=localhost
lb_procs=6
lb_method=custom
interface=af_packet::enp7s0f1
af_packet_fanout_id=101
af_packet_fanout_mode=AF_Packet::FANOUT_QM
af_packet_buffer_size=134217728
pin_cpus=0,1,2,3,4,5,6
```

When 4 is the new 8 and 8 is the new 16

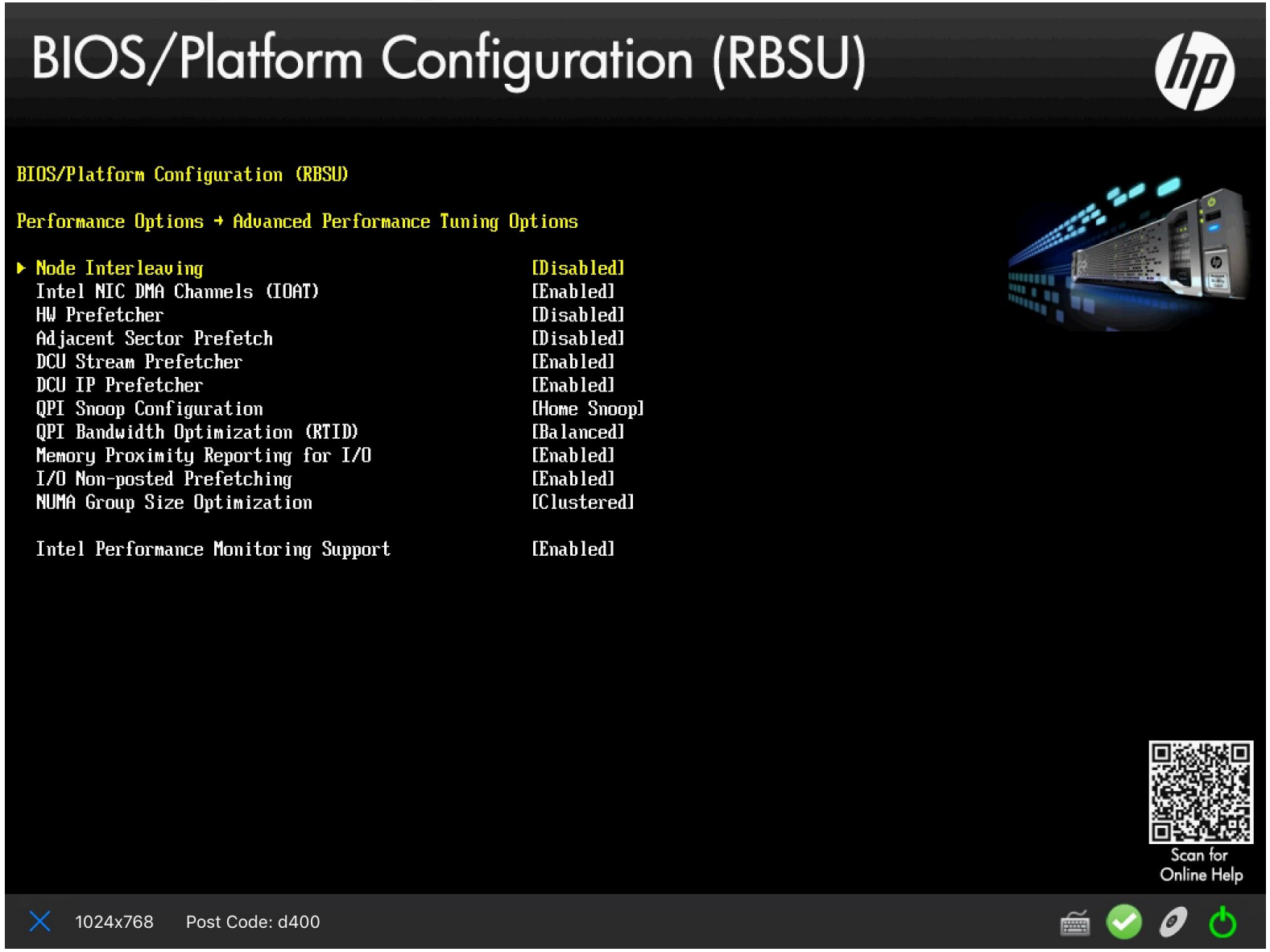
Is your PCIe v3.0 slot x8?

Some x8 slots are x4 electrically and x8 mechanically

Some x16 slots are x8 electrically and x16 mechanically

Is your PCIe slot v3.0?

Disable monkey data prefetchers



Interrupt moderation

ethtool -C ethX adaptive-rx off adaptive-tx off rx-usecs 84 tx-usecs 84

start with 84us ~ 12 000 int/sec

if rx_dropped - cpu too slow or not enough buffers (ethtool -G) to hold packets for 84us or too low interrupt rate

if cpu utilization not maxed - 62usec to service buffers faster and have less descriptors (so less cache trashing)

Are my sensors dropping packets?


```
{"ts":1569970982.589008,"ts_delta":300.000054,"peer":"nsm1-af_packet-enp17s0f0-4","gaps":38907,"acks":147799,"percent_lost":26.324265}
{"ts":1569970982.712347,"ts_delta":300.000004,"peer":"nsm1-af_packet-enp17s0f0-2","gaps":37629,"acks":107405,"percent_lost":35.034682}
{"ts":1569970982.722461,"ts_delta":300.000004,"peer":"nsm1-af_packet-enp17s0f0-3","gaps":48390,"acks":158835,"percent_lost":30.465577}
{"ts":1569970982.76303,"ts_delta":300.000526,"peer":"nsm1-af_packet-enp17s0f0-1","gaps":26400,"acks":117336,"percent_lost":22.499489}
{"ts":1569971282.589062,"ts_delta":300.000054,"peer":"nsm1-af_packet-enp17s0f0-4","gaps":32629,"acks":136170,"percent_lost":23.961959}
{"ts":1569971282.712673,"ts_delta":300.000326,"peer":"nsm1-af_packet-enp17s0f0-2","gaps":37184,"acks":111760,"percent_lost":33.271296}
{"ts":1569971282.724242,"ts_delta":300.001781,"peer":"nsm1-af_packet-enp17s0f0-3","gaps":34031,"acks":146676,"percent_lost":23.201478}
{"ts":1569971282.763072,"ts_delta":300.000042,"peer":"nsm1-af_packet-enp17s0f0-1","gaps":39984,"acks":117426,"percent_lost":34.050381}
```


“Something is dropping somewhere”

What is my packet drop rate?

```
root@nsm1~ # ethtool -S enp17s0f0 | egrep 'drop|illegal|fault|error' | egrep -v "tx_"
     rx_errors: 0
     rx_dropped: 8444
     rx_length_errors: 0
     rx_crc_errors: 0
     port.rx_dropped: 0
     port.rx_crc_errors: 0
     port.illegal_bytes: 0
     port.mac_local_faults: 0
     port.mac_remote_faults: 0
     port.rx_length_errors: 0
root@nsm1~ # ethtool -S enp17s0f0 | egrep 'rx_packet'
     rx_packets: 11233814105
     veb.tc_0_rx_packets: 0
     veb.tc_1_rx_packets: 0
     veb.tc_2_rx_packets: 0
     veb.tc_3_rx_packets: 0
     veb.tc_4_rx_packets: 0
     veb.tc_5_rx_packets: 0
     veb.tc_6_rx_packets: 0
     veb.tc_7_rx_packets: 0
root@nsm1~ #
```

What is my packet drop rate?

```
root@nsm1~ # ./softnet_stat.sh 2>/dev/null
cpu        total     dropped   squeezed  collision
  0 3260743040           0         21          0
  1 1930474146           0         23          0
  2 2759716965           0         14          0
  3 2407128437           0         24          0
```

Pro-tip: ignore dropped, watch if squeezed is growing

Wait what?

softnet stats “dropped” -> out of per-CPU backlog

Ain’t no backlog without RPS

RPS?!?!

Talk to me later ;)

What is my packet drop rate?

@load misc/stats

stats.log <- only AF_Packet!!

pkts_proc
bytes_recv
pkts_dropped
pkts_link

When 2x 40 is 50

Your X710 / X722 - 2x 40Gbit = 1x 50Gbit

And X510 / 520 / 540 can do only 8M - 10M pps

Myths

Linux network stack is not zero copy and is slow

Need to bypass!!

Answer

Not true from many years

Myths

Linux network stack is not multithreaded everywhere (pf_ring)

Answer

Not true from many years

Myths

I need to process 40 / 100Gbit and 60M pps

Answer

40Gbit interfaces vs 40Gbit/sec of traffic

Not all traffic is equal <- drop early

Average packet size (IMIX) - >900 bytes -> much less PPS

Myths

Cross-NUMA talk is bad because of bandwidth

Nope. Bandwidth is plenty (over 100 Gbit/sec). Latency kills ya.

Hmmm... guess how I know?!?!

Mistakes

"I will make every buffer BIG"

...and cause tons of cache misses

Mistakes

"So I have this 4 CPU 384GB RAM with 128 cores"

And a cache miss almost 100% of time

eBPF - AF_XDP

Netronome + XDP = hardware bypass

Fully programmable - L2-L7

40 / 100Gbit (from 500USD)

ARM11

48 flow processing cores

60 packet processing cores

480 threads

8GB DDR3 packet buffer

Fast. Reliable. Cheap

Fast. Reliable. Cheap

Choose two?

Have $$$?

Buy appliance

Have time? Need flexibility?

Build one

You can build a flexible & high-performance sensor

With commodity hardware

@michalpurzynski

https://github.com/mozilla/zept

You can build a flexible & high-performance sensor

With commodity hardware

*with some learning

Thank You