Snabb: Open Source Meets Dataplane

RIPE77, October 2018, Amsterdam

Andy Wingo | wingo@igalia.com | @andywingo
Why? The problem solved by Snabb
How? Snabb from the ground up
What? What’s in the box
Who? Snabb in the wild
why?

Ever want to deploy a new RFC, but no vendor is selling it?

Maybe it’s not even an RFC yet?

10 years ago – too bad

Now – open source software + commodity servers
software?  User-space data planes
Avoid the kernel, handle all data in user-space
Snabb, DPDK, VPP (fd.io)
user
space

Tell Linux to forget about NIC
Mmap NIC’s PCI registers into address space

Read and write memory == read and write PCI registers

Poke registers as needed to bring up NIC

Set up a ring buffer for RX/TX
Busy-loop to take packets from RX, process, send to TX
advantage You get the whole packet

No hazard for straying off device-supported hot-path

Program using whatever technology you want: C, Rust, Lua, Scheme, ...

“It’s just programming”

Hire anyone you want to modify the programs
limits

Limited by PCI bandwidth
Limited to ~10-50Gbps/CPU core (parallelization possible)
Tangential to containerization / kubernetes / openstack hellscape
Goal: “rewritable software”

The hard part: searching program-space for elegant hacks

“Is that all? I could rewrite that in a weekend.”
in a nutshell

A snabb program consists of a graph of apps.
Apps are connected by directional links.
A snabb program processes packets in units of breaths.
program code

Instantiate apps
Declare links
Breathe
local Intel82599 =
    require("apps.intel.intel_app").Intel82599
local PcapFilter =
    require("apps.packet_filter.pcap_filter").PcapFilter

local c = config.new()
config.app(c, "nic", Intel82599, {pciaddr="82:00.0"})
config.app(c, "filter", PcapFilter, {filter="tcp port 80"})
config.link(c, "nic.tx -> filter.input")
config.link(c, "filter.output -> nic.rx")

engine.configure(c)

while true do engine.breathe() end
snabb is written in lua

- Short and sweet programs
- LuaJIT does the heavy lifting
- High-performance just-in-time compilation, applied to networking domain
- Lua all the way down – packet processing, not just configuration
Each breath has two phases:

- Inhale a batch of packets into the network
- Process those packets

To inhale, run *pull functions* on apps that have them

To process, run *push functions* on apps that have them
# Pull function for built-in Intel82599 app

```lua
function Intel82599:pull ()
    for i = 1, engine.pull_npackets do
        if not self.dev:can_receive() then break end
        local pkt = self.dev:receive()
        link.transmit(self.output.tx, pkt)
    end
end
```
# Push function for built-in PcapFilter app

```lua
function PcapFilter:push ()
    while not link.empty(self.input.rx) do
        local p = link.receive(self.input.rx)
        if self.accept_fn(p.data, p.length) then
            link.transmit(self.output.tx, p)
        else
            packet.free(p)
        end
    end
end
```
packets
and
links

struct packet {
    uint16_t length;
    unsigned char data[10*1024];
};

struct link {
    struct packet *packets[1024];
    // the next element to be read
    int read;
    // the next element to be written
    int write;
};
// (Some statistics counters elided)
voilà

At this point, you can rewrite Snabb
(Please do!)
But you might want to use it as-is...
unboxing

$ git clone \
  https://github.com/snabbco/snabb
$ cd snabb
$ make
$ ./src/snabb

What’s in there?

How are people using it?
apps

I/O: Intel i210/i350/82599, Mellanox ConnectX4/5, TAP, AF_PACKET, AF_XDP, vhost/virtio, pcap...

L2: ARP, NDP, learning bridge, l2vpn...

L3: IPsec, ICMP, fragmentation...

+: IPFIX, lwAFTR, DPI, firewall, pflang...

Apps: learning bridge, NIC
yang

App graph as function of YANG-modelled configuration

Run-time config/state query, reconfigure

Multi-process

Statistics aggregation

https://snabbco.github.io/#ptree
libraries

LPM, JSON, fast raw hash tables, protocol stack, timer wheel, profiling, packet match domain-specific language compilers, NUMA/CPU binding, RRD files...

https://snabbco.github.io/
no full router yet

Some support for receiving routes from Linux
We would love to flesh this out!
snabb in the wild

See lightning talk “8 ways network engineers use Snabb” for more examples
exploratory analysis

Flexibility, expressiveness, and rapid development of scapy, the speed to run live

A large CDN uses Snabb in this way internally
layer 2
vpn

# github.com/alexandergall/snabb
# l2vpn branch

$ snabb l2vpn l2vpn.conf

RFC 4664 layer 2 learning bridge over IPv6

Built by SWITCH network engineer Alexander Gall because what he needed wasn’t on offer

In production linking academic sites in Switzerland
ipsec

Vita: https://github.com/inters/vita

Secure VPN between sites, IPSec, 1-10 Gbps/core

Funded by NLnet Foundation
Lightweight 4-over-6 AFTR: processes all IPv4 traffic for a network

YANG-enabled, runtime reconfigurable

Multi-process: one instance can manage many NICs in a machine

See K. Zorbadelos (OTE) at RIPE76: https://ripe76.ripe.net/archives/video/30/
join us!

https://github.com/snabbco/snabb
snabb.slack.com (see Github page for join link)

wingo@igalia.com, @andywingo

Happy hacking!