Production Snabb

Simple, fast software networking functions with Snabb

21 June 2017 – SDN Barcelona

Andy Wingo wingo@igalia.com

@andywingo
hey
network
hackers

Agenda:
» Snabb, a VNF workbench
» Recent developments
» Some batteries included: Snabb in practice
Commodity hardware is capable of high-performance networking

- 1 core at 15MPPS: 65ns/packet

What software to put on the hardware?
The teleology of open source: “one day this will all run Linux”

Conventional wisdom: if I walk the racks of a big ISP, it’s probably all Linux
linux?  The teleology of open source: “one day this will all run Linux”

Conventional wisdom: if I walk the racks of a big ISP, it’s probably all Linux

Q: The hardware is ready for 10 Gbps on a core. Is Linux?
The teleology of open source: “one day this will all run Linux”

Conventional wisdom: if I walk the racks of a big ISP, it’s probably all Linux

Q: The hardware is ready for 10 Gbps on a core. Is Linux?
A: Nope
why
not
linux

Heavyweight networking stack
System/user barrier splits your single network function into two programs
Associated costs both at development-time and run-time
Cut Linux-the-kernel out of the picture; bring up card from user space

- tell Linux to forget about this PCI device
- mmap device’s PCI registers into address space
- poke registers as needed
- set up a ring buffer for receive/transmit
- profit!
user-space networking

Multiple open source user-space networking projects having success

Prominent examples:

- Snabb (2012)
- DPDK (2012)
- VPP/fd.io (2016)

(Is this SDN? :))

How do software network functions work?
aside

Snabb aims to be rewritable software

The hard part: searching program-space for elegant hacks

“Is that all? I could rewrite that in a weekend.”
A snabb program consists of a graph of apps.
Apps are connected by directional links.
A snabb program processes packets in units of breaths.
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
breaths

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
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 of included PcapFilter

```plaintext
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
```
```c
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...
inventory

apps: software components that developers compose into network functions

programs: complete network functions

**bold**: new in 2016/2017

*italics*: not yet merged to mainline
app
catalog:
i/o

Intel i210/i350/82599/XL710
Mellanox ConnectX-4/5
VirtIO host and guest
UNIX socket
Linux: tap and “raw” (e.g. eth0)
Pcap files
app catalog: l2

Flooding and learning bridges

VLAN insert/filter-and-remove/mux

ARP / NDP
app
catalog:
l3

IPv4/v6 fragmentation and reassembly
IPv4/v6 splitter
ICMPv4/v6 echo responder
Control plane delegation (nh_fwd)
(No routing yet)
app catalog: transport  

IPsec ESP  

Lightweight 4-over-6 AFTR  

“Keyed IPv6 Tunnel” (draft-tr-mkonstan-keyed-ipv6-tunnel-01)
app catalog: monitoring

Netflow capture and export

L7 monitor / filter (using libndpi)

pcap filter (with machine-code backend)
Many workload generators
programs

$ git clone \\n    https://github.com/SnabbCo/snabb
$ cd snabb
$ make
$ src/snabb
Usage: src/snabb <program> ...

This snabb executable has the following programs built in:
  lisper
  lwaftr
  packetblaster
  pci_bind
  snabbmark
  snabbnvf
  snabbvmx
  snsh
  top
  wall

For detailed usage of any program run:
  snabb <program> --help
program: packet blaster

Generally useful tool: fill TX buffer of NIC with packets and transmit them over and over again

snabb packetblaster replay \ packets.pcap 82:00:1

Measures received (return) traffic too

Easily saturates 10G links
program: lwaftr

“Lightweight 4-over-6”: RFC 7596
Snabb-implemented border router for lw4o6
IPv4 for entire countries!
Remarkable deployment report from OTE engineer Kostas Zordabelos, April 2017:
https://www.youtube.com/watch?v=EEpUWieTr40&t=1h46m
Why Snabb?
Fast, fluid development
RFC only finalized during development
Good speed
Open source
Cheap
program: nfv

Host switch providing network connectivity to QEMU instances

“Original” Snabb app

Like Open vSwitch with DPDK datapath, or OpenContrail

OpenStack integration never landed... but the market has moved on

(Has the market moved on from classic NFV?)
Idea: Snabb data plane, external control and management planes
Contributed by Juniper engineer Marcel Wiget
Possibility to delegate to Juniper vMX to determine next hops; or to an image with Linux

Juniper Tech Club, March 2017:
https://www.youtube.com/watch?v=N_CjXgyrUcY
snabb snabbvmx lwaftr --help
program: snabbwall

L7 firewall that optionally uses nDPI
http://snabbwall.org/

Collaboration between Igalia and NLnet foundation
Landed upstream in 2017
program: ipfix

Prototype NETFLOW collector and exporter (v9 and IPFIX)

Currently only 5MPPS, working on single-core improvements then moving to RSS

Pending to land upstream
program: l2vpn

Alexander Gall’s L2 VPN over IPv6
Pending to land upstream; used in production AFAIU
Ideal Snabb use case: programmer-operator builds bespoke tool
programs: your vnf here

Snabb upstream open to include new network functions
Repository will grow as people build new things
Igalia can build one for you :)}
From prototype to production: what do you need?

(Re)configurability

State monitoring
YANG is great!!

Native YANG support in Snabb

- Load and serialize textual configurations
- Compiled compilations (useful for big routing tables)
- Incremental update
- State query
snabb config

App & link graph a function of config
Update config? Diff graphs, apply incremental changes

Carefully built to scale
- Fast-paths for some incremental updates, e.g. add lwAFTR softwire
- Config/state query avoids touching data plane process
- Updates cause minimal change
- Subquery built-in
snabb config

Command-line tool, snabb config
NETCONF via Sysrepo bridge
Other configuration agents possible
100G in production Snabb

Multiple coordinated data-plane processes

Horizontal scaling via BGP/ECMP:
terabit lw4o6 deployments

Performance x-ray: where to focus effort to improve speed?

[Your cool hack here!]

Work in progress!
thanks!

Make a thing with Snabb!

git clone https://github.com/SnabbCo/snabb

cd snabb

make

wingo@igalia.com

@andywingo
oh no here comes the hidden track!
Storytime! Modern x86: who’s winning?
Clock speed same since years ago
Main memory just as far away
“We need to do work on data... but there’s just so much of it and it’s really far away.”

Three primary improvements:

- CPU can work on more data per cycle, once data in registers
- CPU can load more data per cycle, once it’s in cache
- CPU can make more parallel fetches to L3 and RAM at once
Networking folks can win too

Instead of chasing zero-copy, tying yourself to ever-more-proprietary features of your NIC, just take the hit once: DDIO into L3.

Copy if you need to – copies with L3 not expensive.

Software will eat the world!
Networking folks can win too

Once in L3, you have:

- wide loads and stores via AVX2 and soon AVX-512 (64 bytes!)
- pretty good instruction-level parallelism: up to 16 concurrent L2 misses per core on haswell
- wide SIMD: checksum in software!
- software, not firmware