High-performance traffic encryption on x86_64

RIPE78

Max Rottenkolber

max@mr.gy

*@eugeneia_*

Vita

whoami

Max Rottenkolber <max@mr.gy>

Open source hacker, working on Snabb since 2014

Consulting on software networking (in userspace), protocols, optimization...

Vita

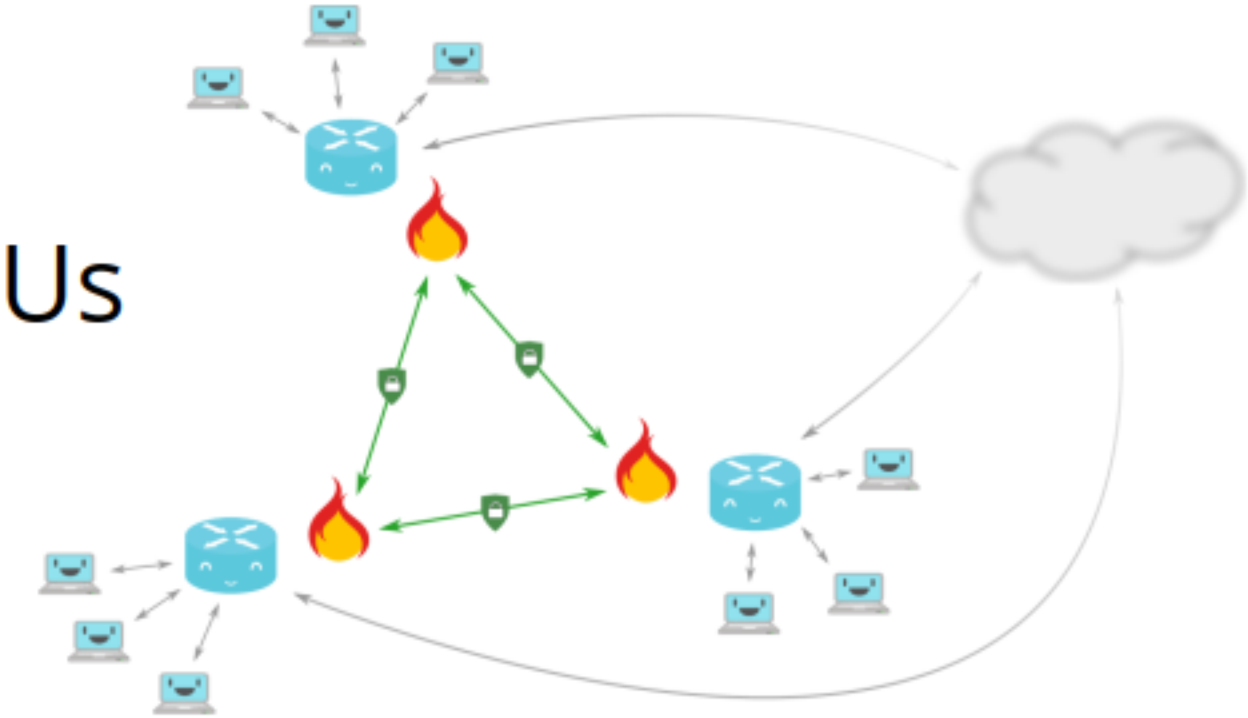
inter—
stellar

Vita

Vita is a high-performance site-to-site VPN gateway

Fully open source (and hackable!)

Runs on generic x86_64 server CPUs



Vita

Based on snabb

Written in a high-level language (Lua) RAPTORJIT

Made possible by nlnet FOUNDATION

snabb

```
while not link.empty(input) do

   local p = link.receive(input)

   if ipv4_ttl(p) > 0 then
      link.transmit(output, p)
   else
      link.transmit(time_exceeded, p)
   end

end
```

Vita

~3 Mpps per core on a modern CPU (duplex)

...or ~5 Gbps of IMIX traffic per core

100 Gbps at minimum packet size on a ~50 core box? :-)

How?

In Snabb-land we like to write software that is both fast and simple

…and we don’t like vendor lock-in

No QuickAssist, crypto cards… Only x86_64!

How?

For crunching numbers (encryption): AES-NI, AVX2 (optimized AES-GCM implementation written in DynASM)

```
function ghash_mul(Dst, gh, hk, t1, t2, t3)
  | vpclmulqdq xmm(t1), xmm(gh), xmm(hk), 0x11
  | vpclmulqdq xmm(t2), xmm(gh), xmm(hk), 0x00
  | vpclmulqdq xmm(t3), xmm(gh), xmm(hk), 0x01
  | vpclmulqdq xmm(gh), xmm(gh), xmm(hk), 0x10
  | vpxor xmm(gh), xmm(gh), xmm(t3)
  ...
```

How?

For route lookups (longest prefix match):

Optimized Poptrie implementation (again, DynASM)

```
function lookup (Dst, Poptrie, keysize)
   if Poptrie.direct_pointing then
      -- v = extract(key, 0, Poptrie.s)
      local direct_mask = bit.lshift(1ULL, Poptrie.s) - 1
      -- v = band(key, direct_mask)
      | mov v_dw, dword [key]
      | and v, direct_mask
   ...
```

How?

RaptorJIT + FFI

(simple and fast implementation of IPsec ESP)

```
esp_head = ffi.typeof[[
   struct {
     uint32_t spi;
     uint32_t seq_no;
   } __attribute__((packed))
 ]]
```

```
esp_tail = ffi.typeof[[
   struct {
     uint8_t pad_length;
     uint8_t next_header;
   } __attribute__((packed))
]]
```

How?

Problem: can not parallelize SA

Every packet on an SA gets a unique sequence number

Synchronization problem if spread across cores

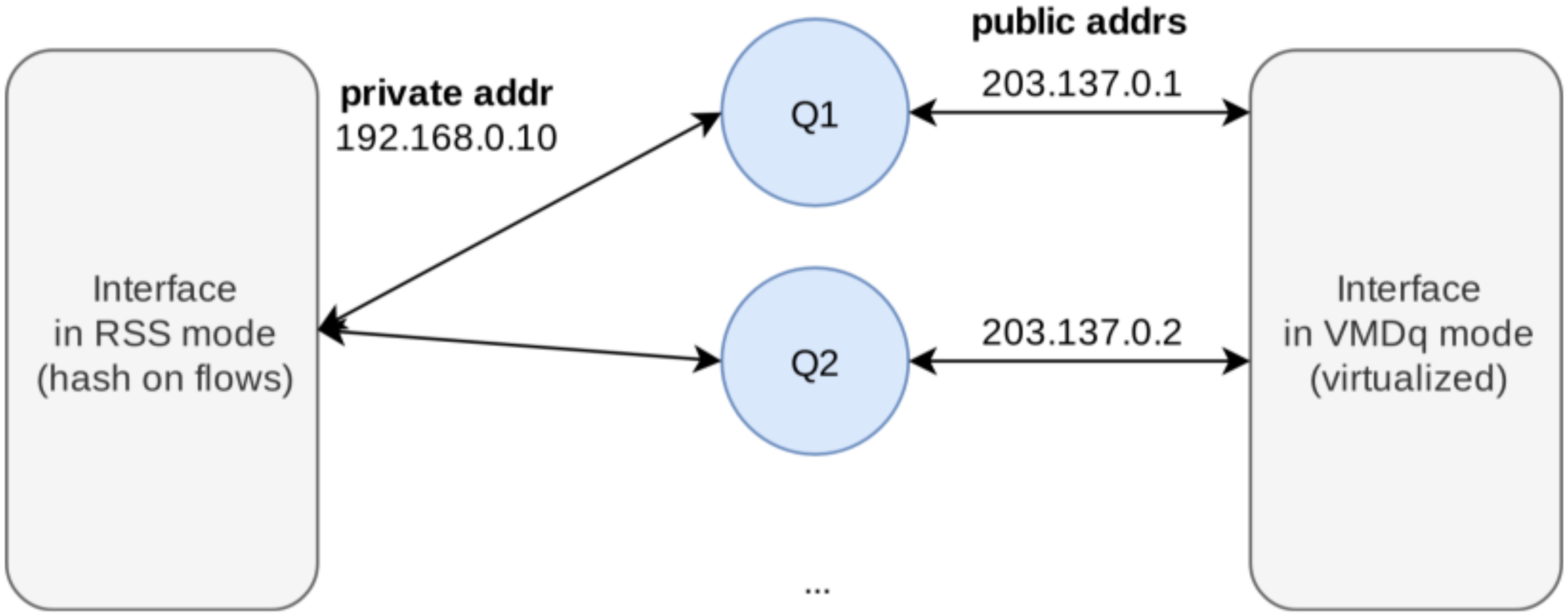
How?

Solution: scale out (multiple SAs per route)

RSS on private interface: distribute onto SAs

VMDq on public interface: aggregate SAs

How?



private addr
192.168.0.10
public addrs
203.137.0.1
203.137.0.2
Q1
Q2
...
Interface
in RSS mode
(hash on flows)
Interface
in VMDq mode
(virtualized)

Vita

IPsec ESP?

Standardised initially with 32 bit Sequence Numbers

(wait for it...)

IPsec ESP?

Extended 64 bit Sequence Numbers!

Did not update the header though...

IPsec ESP?

Extended 64 bit Sequence Numbers!

Did not update the header though...

Transmit lower half, guess the rest (really!)

IPsec ESP?

What if sender and receiver loose sync?

Resynchronize using tricky algorithm (really?)

Likely not relevant in real deployments...

AKE (authenticated key exchange)

Achilles heel

Want to cycle SAs often and without loosing packets (perfect forward secrecy)

AKE (authenticated key exchange)

Some options:

IKEv2 (interoperable)

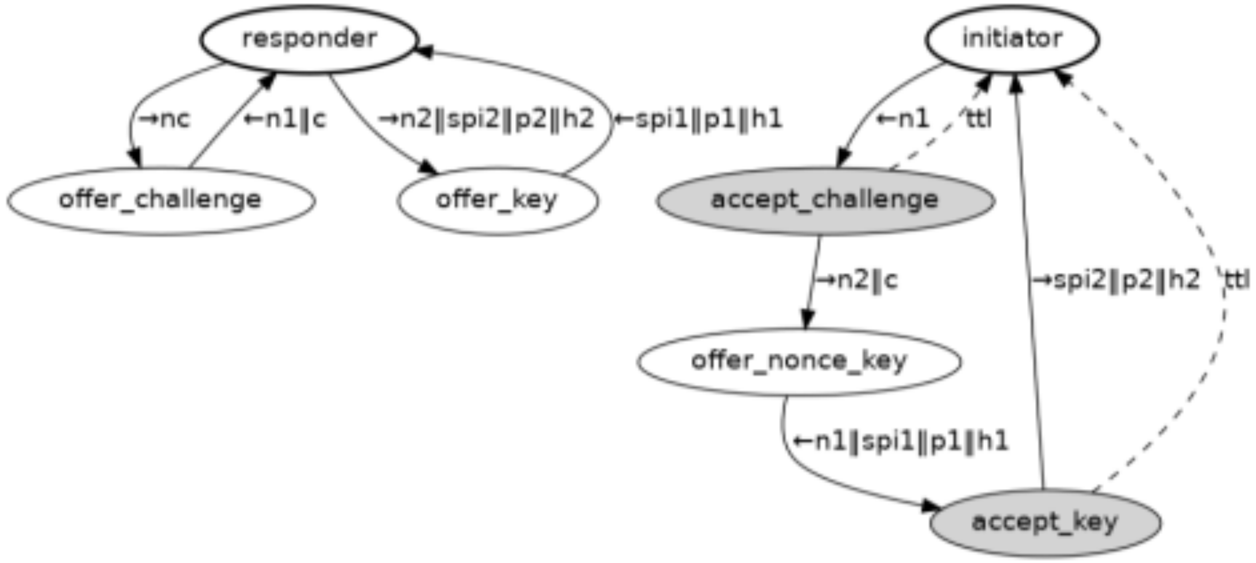
Noise (modern protocol framework)

Roll your own (no?!)

AKE (authenticated key exchange)

Roll your own (yes!)

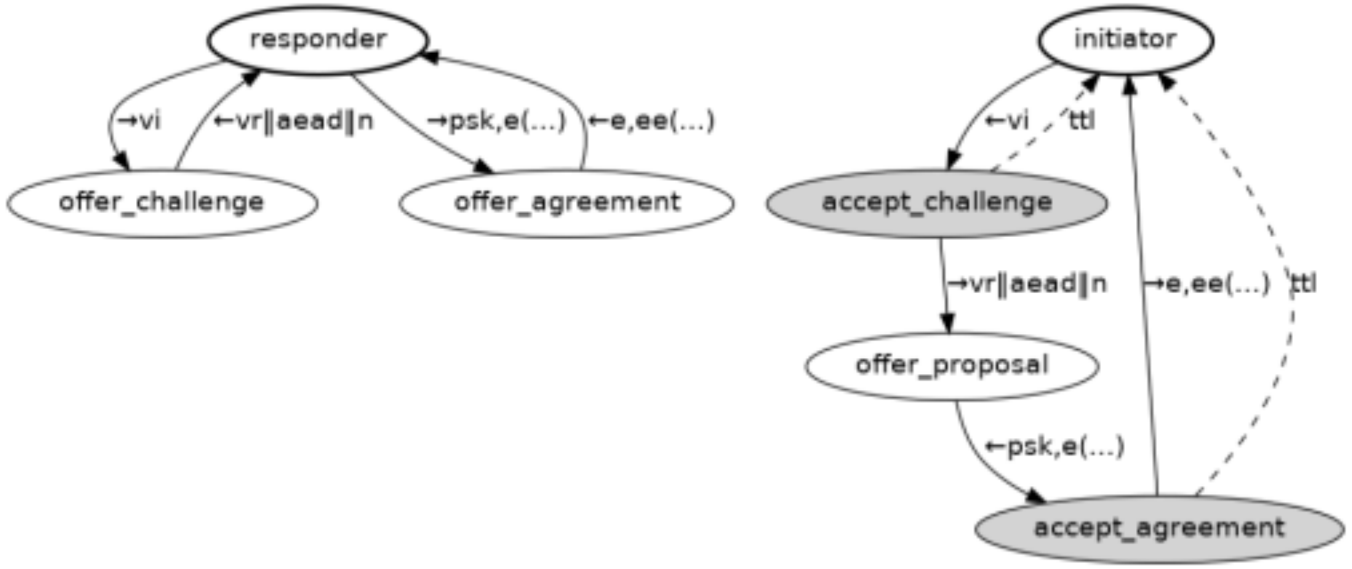
...hey, you can learn things :-)



AKE (authenticated key exchange)

Do the most simple thing that could possibly work

If it breaks, try the next least complex thing



Vita

AKE (authenticated key exchange)

But we actually explored all three possibilities:

SWITCH engineer Alexander Gall provides StrongSwan plugin+interop with Snabb

Vita’s current default AKE protocol is based on Noise!

Configuration & operation

Based on a YANG model

…includes runtime statistics

```
module vita-esp-gateway {
   ...
```

Configuration & operation

Query/update configuration via RPC

Query runtime statistics via RPC

```
$ snabb config get-state /gateway-state/private-interface
```

Configuration & operation

Friendly to operators!

Lots of stats for ICMP events, data- and control-plane errors etc…

Transparent traceroute (appears as two hops)

Testing

Continuous integration, performance, and unit testing

Interop testing with Linux ESP stack

Next step: fuzz all the things

Hardware support (NICs)

Intel 52899, i350 (niantic) 10GbE, 1GbE

Mellanox ConnectX 1-100GbE

Working on: Intel AVF, AF_XDP

Let’s encrypt some traffic

Medium-term goal: tunnel 100 Gbps line-rate at 60 byte packets on a generic x86 server using a fully open source software stack.

Thanks!

Get involved:
github.com/snabbco/snabb
github.com/inters/vita

Get support and consulting:
https://inters.co

inter—
stellar

Email me:
max@mr.gy

Gritty details on my blog:
https://mr.gy/blog