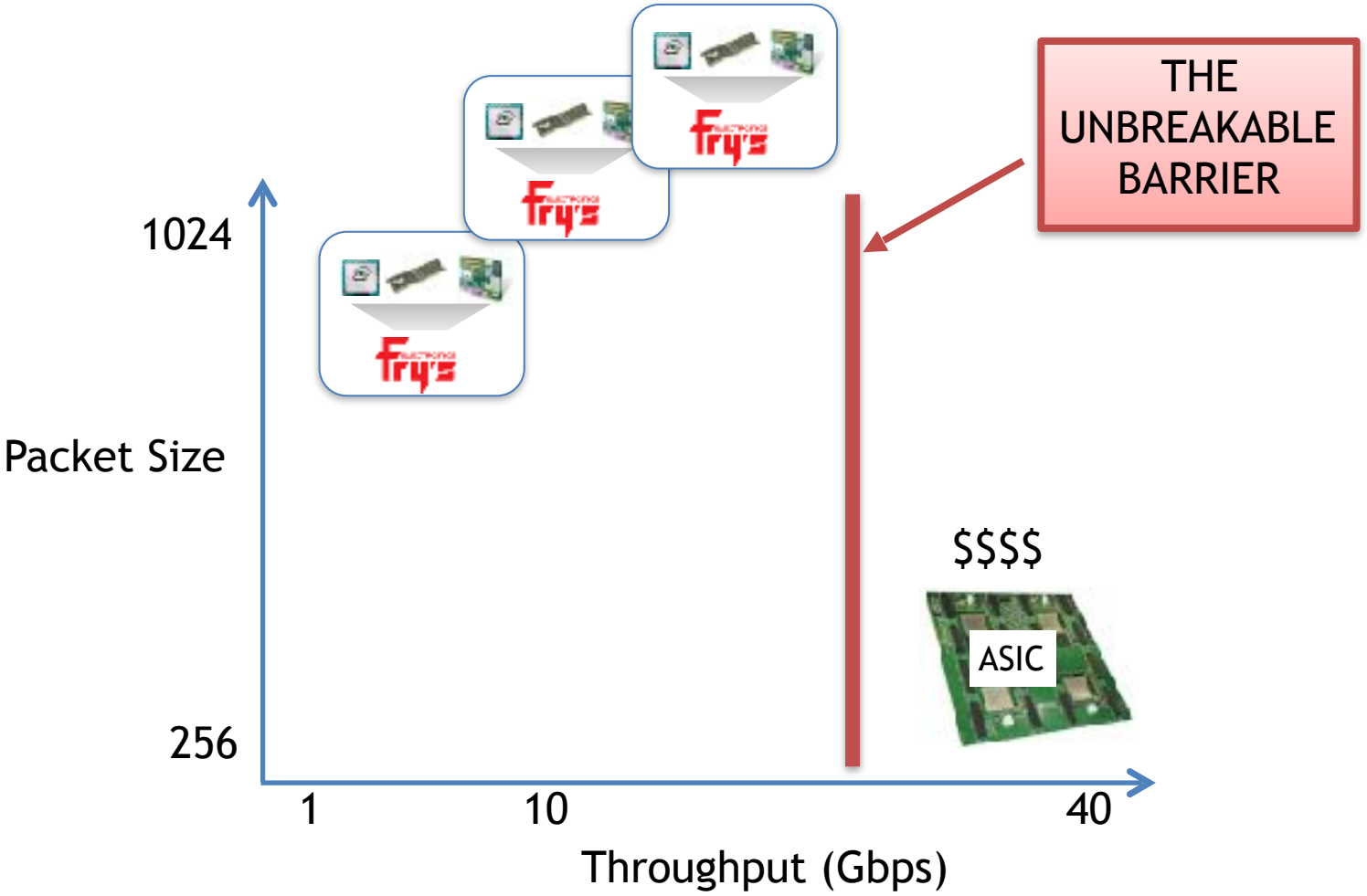
netgate

40 Gbps IPsec…
on Commodity Hardware

Jim Thompson
Founder & CTO
Netgate

REALITY BEFORE 2017

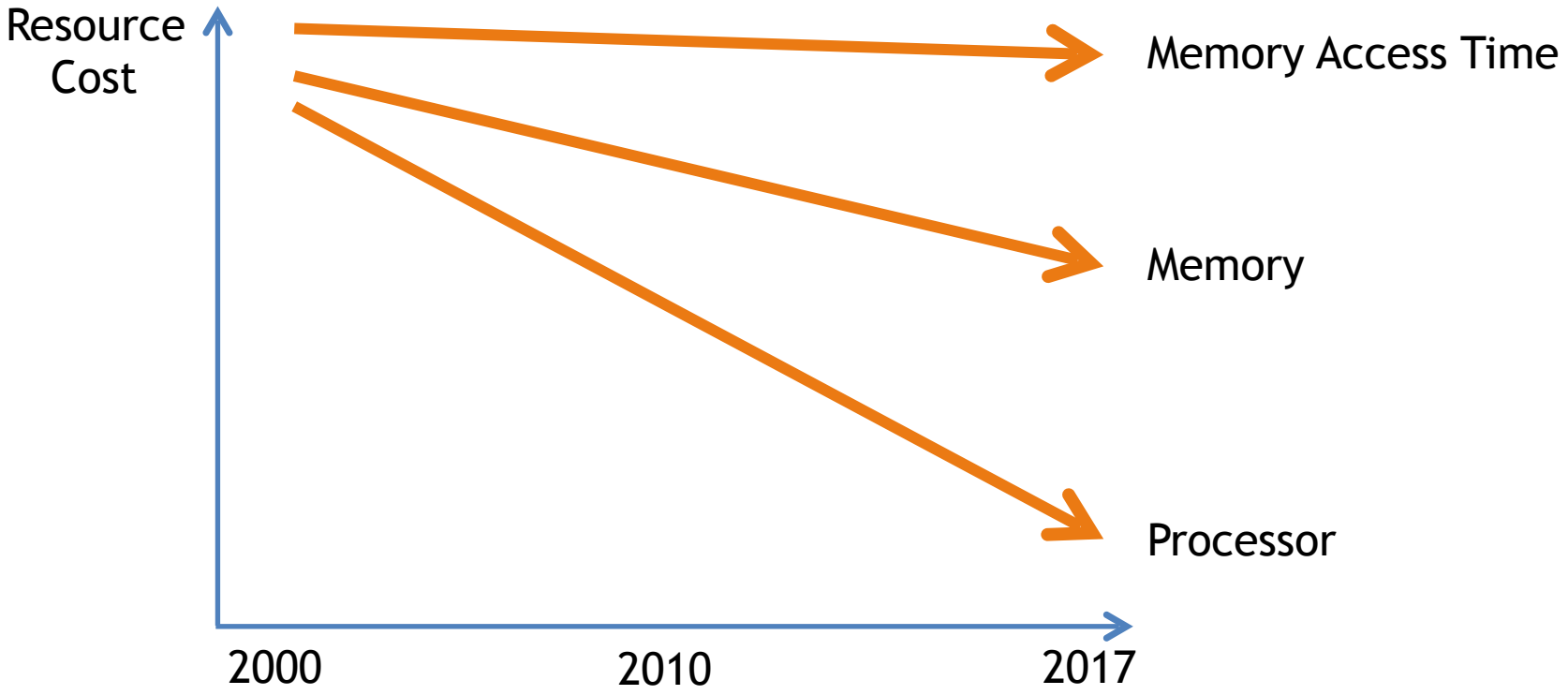
netgate



Want small packets and/or high(er) throughput?
Pay the Piper (Vendor)

Why?

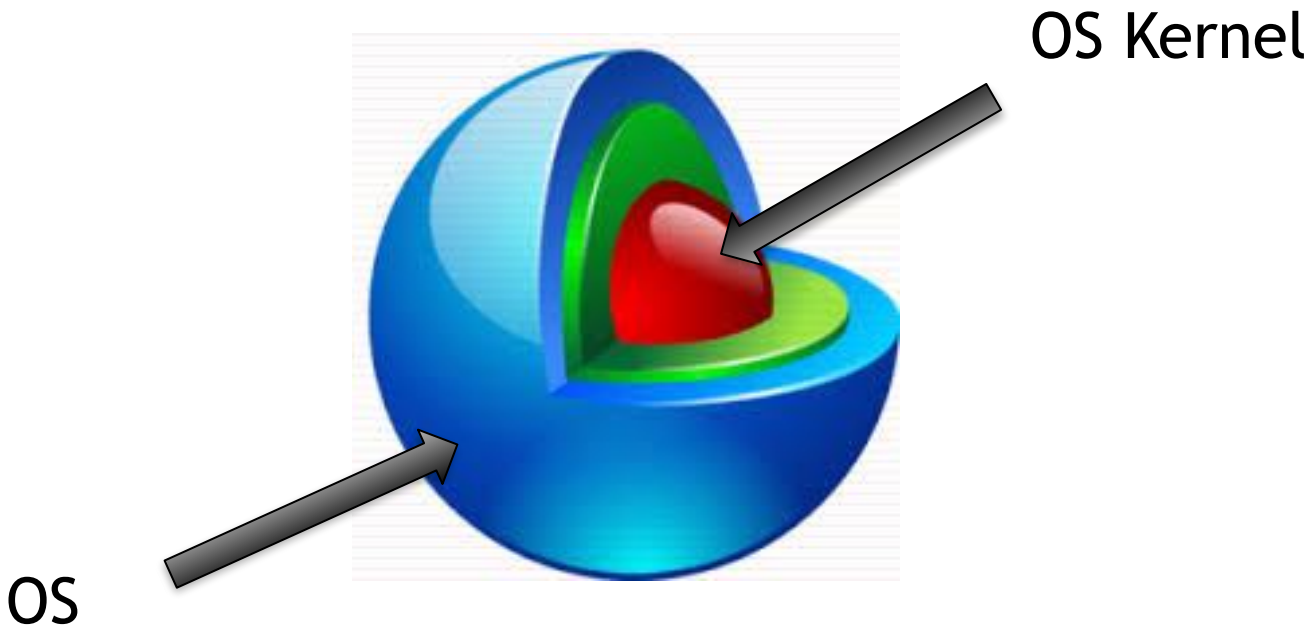
netgate



Memory access “resource cost” has remained relatively constant

Oh, and one more thing...

netgate



Kernel changes are expensive, painful, and far-reaching
Writing your own stack is difficult, and expensive

And then, a breakthrough…

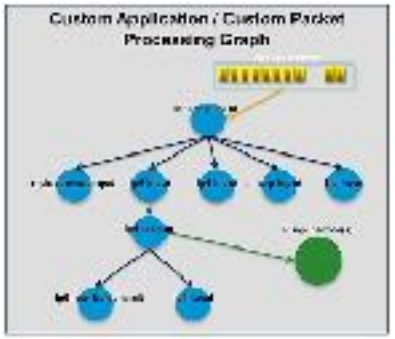
netgate

And $DEITY said, “Let there be VPP and DPDK.”

Uhhh…say what?

VPP & DPDK

netgate



Vector Packet Processing

- VPP takes a vector of packets from NIC
- Stuffs them into RAM
- Runs set of functions against packets
- Packets move as a group
- Instruction cache remains hot
- First packet penalty, 2nd packet is cheap
- Apache 2 licensed

DPDK
DATA PLANE DEVELOPMENT KIT

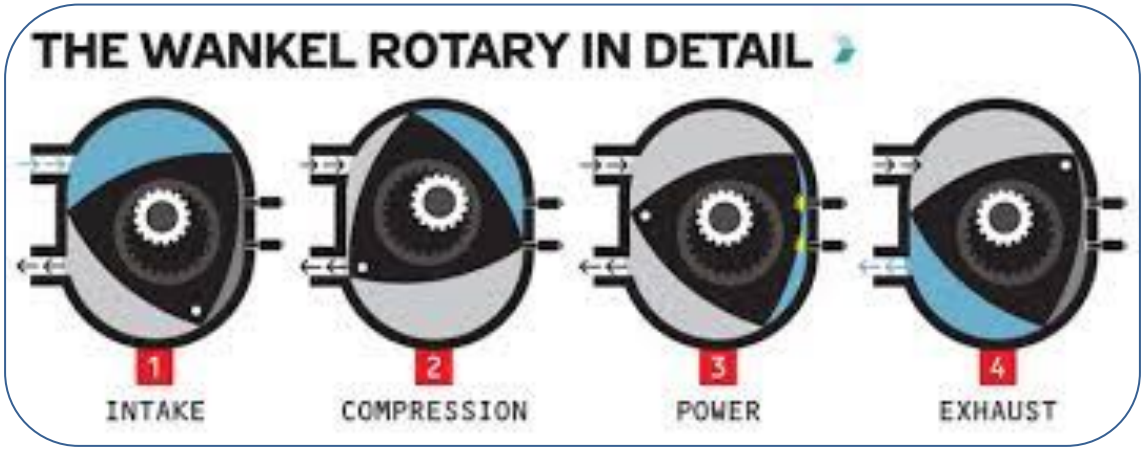
Data Plane Development Kit

- Set of data plane libraries and network interface controller drivers for fast packet processing
- Programming framework for Intel and ARM processors
- Scales from Intel Atom to Xeon and more
- BSD licensed

The wall just fell down…

Is this one of those eccentric ideas?

netgate



Answer: Nope. VPP was created and open sourced by...

CISCO

Why would they do such a thing?

netgate

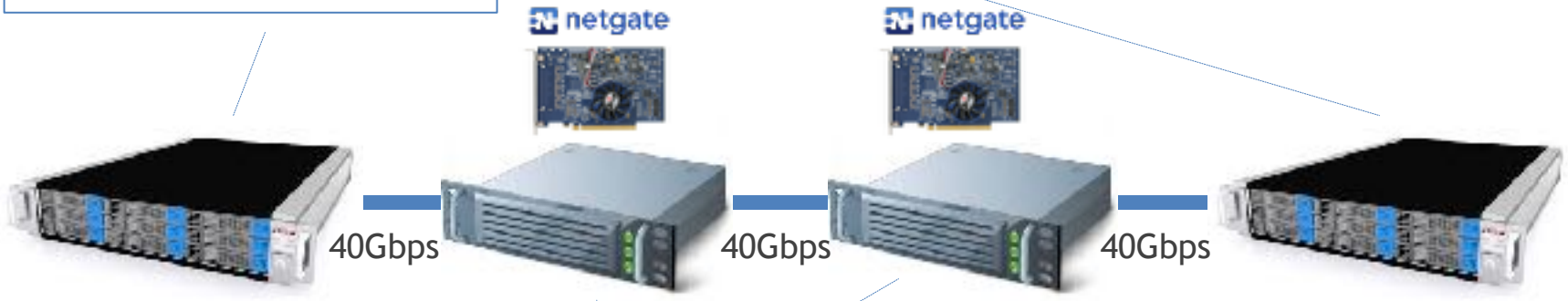
No choice. The world is changing fast.

Accept it, or accept your fate.

Test Harness

netgate

- Xeon E3-1275 v3
- 32GB RAM
- Intel XL710 Ethernet
- 1U server



- I7-6950X CPU, overclocked to 3.5GHz (10C)
- ASUS X99 board
- Water-cooled to avoid thermal throttling
- 32GB RAM
- XL710 40Gbps Ethernet
- CPIC (QAT) crypto offload card (Netgate)

Test Results

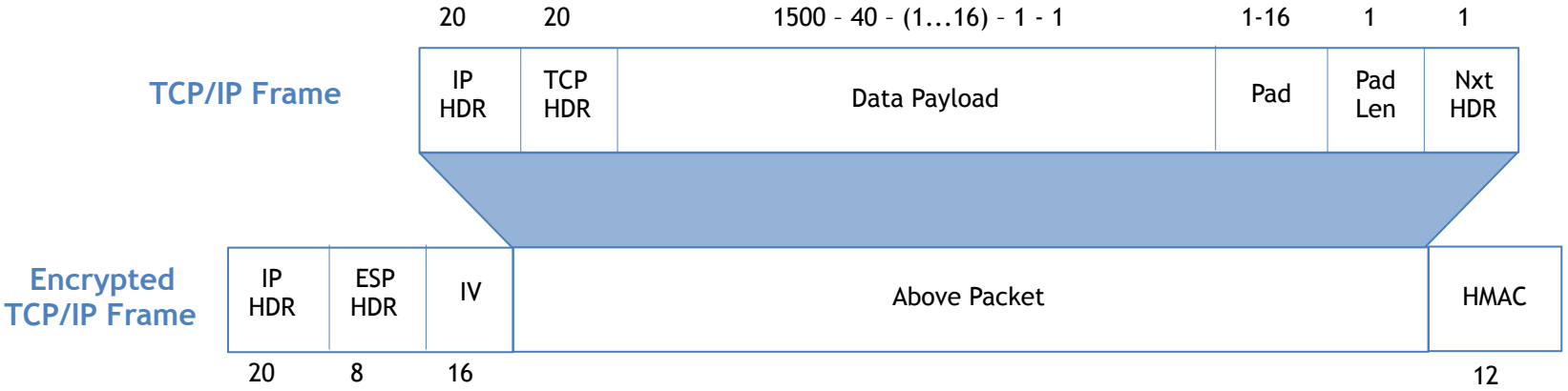
netgate

Context	Crypto processing	Crypto/AEAD algorithm	Integrity algorithm	# of SAs	Total # streams	Iperf3 TCP 1500	# of samples	pktgen UDP 64	pktgen UDP 512	pktgen UDP 1500
Kernel	AES-NI	AES-CBC-128	SHA1	1	1	2.09 Gbps	16	358 kpps	267 kpps	
Kernel	AES-NI	AES-CBC-128	SHA1	1	4	2.07 Gbps	16	360 kpps	266 kpps	
Kernel	AES-NI	AES-CBC-128	SHA1	8	8	10.85 Gbps	6	2.11 Mpps	1.71 Mpps	
Kernel	AES-NI	AES-GCM-128-16		1	1	5.06 Gbps	16	478 kpps	440 kpps	
Kernel	AES-NI	AES-GCM-128-16		1	4	5.06 Gbps	16	477 kpps	440 kpps	
Kernel	AES-NI	AES-GCM-128-16		8	8	25.25 Gbps	6	2.90 Mpps	2.46 Mpps	
Kernel	QAT	AES-CBC-128	SHA1	1	1	8.74 Gbps	16	575 kpps	579 kpps	
Kernel	QAT	AES-CBC-128	SHA1	1	4	8.74 Gbps	16	678 kpps	688 kpps	
Kernel	QAT	AES-CBC-128	SHA1	8	8	27.08 Gbps	6	2.44 Mpps	2.97 Mpps	
User	VPP native (OpenSSL 1.0.1)	AES-CBC-128	SHA1	1	1	2.03 Gbps	16	517 kpps	321 kpps	
User	VPP native (OpenSSL 1.0.1)	AES-CBC-128	SHA1	1	4	3.39 Gbps	16	860 kpps	523 kpps	
User	VPP native (OpenSSL 1.0.1)	AES-CBC-128	SHA1	8	8	9.45 Gbps	6	3.29 Mpps	2.25 Mpps	
User	VPP AESNI MB cryptodev	AES-CBC-128	SHA1	1	1	7.42 Gbps	6	2.38 Mpps	1.47 Mpps	
User	VPP AESNI MB cryptodev	AES-CBC-128	SHA1	1	4	8.28 Gbps	6	2.38 Mpps	1.47 Mpps	
User	VPP AESNI GCM cryptodev	AES-GCM-128-16		1	1	13.70 Gbps	6	3.27 Mpps	2.07 Mpps	
User	VPP AESNI GCM cryptodev	AES-GCM-128-16		1	4	13.93 Gbps	6	3.29 Mpps	2.07 Mpps	
User	VPP QAT cryptodev	AES-CBC-128	SHA1	1	1	32.68 Gbps	15	3.85 Mpps	3.70 Mpps	
User	VPP QAT cryptodev	AES-CBC-128	SHA1	1	4	35.72 Gbps	16	4.22 Mpps	4.15 Mpps	
User	VPP QAT cryptodev	AES-CBC-128	SHA1	8	8	36.32 Gbps	6	10.45 Mpps	7.47 Mpps	
User	VPP QAT cryptodev	AES-GCM-128-16		1	1	32.73 Gbps	6	3.86 Mpps	3.66 Mpps	
User	VPP QAT cryptodev	AES-GCM-128-16		1	4	32.99 Gbps	6	3.86 Mpps	3.66 Mpps	
Kernel	None	None		1				604 kpps	606 kpps	605 kpps
Kernel	None	None		4				2.33 Mpps	2.92 Mpps	2.91 Mpps
Kernel	None	None		8				5.16 Mpps	5.14 Mpps	3.28 Mpps
User	None	None		1				14.05 Mpps	8.84 Mpps	3.28 Mpps
User	None	None		4				32.23 Mpps	9.39 Mpps	3.28 Mpps
User	None	None		8				42.60 Mpps	9.38 Mpps	3.28 Mpps

Full Encryption

36 Gbps (which is not 40 Gbps...)

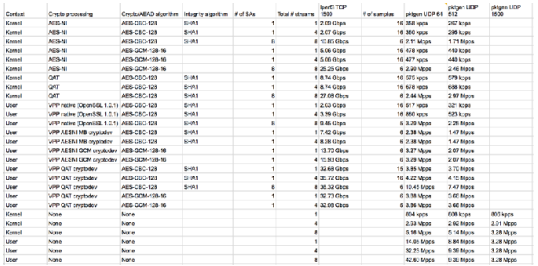
Where is the other 4 gbps?



- Original packet + padding has to fit in 1500 - 56 = 1444 bytes
- Length must be multiple of 16. 1444 mod 16 = 4, so 1440 is largest size for original packet + padding
- 40 bytes of original packet are headers, at least 2 bytes of padding added for pad length/next header
- So 1398 is max size of data that can be sent in the original packet without overflowing the 1500 bytes
- Therefore: ratio of payload to data transmitted = 1398/1538 = 0.90897
- Which equates to a maximum throughput of 36.36 Gbps (no tuning!)
- The highest result we saw was 36.3 Gbps

Ready for production?

netgate



Context	Crypto processing	Crypto/AEAD algorithm	Integrity algorithm	# of SAs	Total # streams	Ipsec: TCP 1500	# of samples	pktgen UDP 64	pktgen: UDP 512	pktgen UDP 1500
Kernel	AES-NI	AES-CBC-128	SHA1	1	1	2.03 Gbps	16	358 kpps	262 kpps	
Kernel	AES-NI	AES-CBC-128	SHA1	1	4	2.07 Gbps	16	360 kpps	266 kpps	
Kernel	AES-NI	AES-CBC-128	SHA1	8	8	10.85 Gbps	8	2.71 Mpps	1.71 Mpps	
Kernel	AES-NI	AES-GCM-128-16		1	1	5.06 Gbps	16	438 kpps	449 kpps	
Kernel	AES-NI	AES-GCM-128-16		1	4	5.66 Gbps	16	427 kpps	483 kpps	
Kernel	AES-NI	AES-GCM-128-16		8	8	26.25 Gbps	8	2.99 Mpps	2.46 Mpps	
Kernel	QAT	AES-CBC-128	SHA1	1	1	6.74 Gbps	16	571 kpps	579 kpps	
Kernel	QAT	AES-CBC-128	SHA1	1	4	8.74 Gbps	16	616 kpps	668 kpps	
Kernel	QAT	AES-CBC-128	SHA1	8	8	27.08 Gbps	8	2.44 Mpps	2.97 Mpps	
User	VPP native (OpenSSL 1.0.1)	AES-CBC-128	SHA1	1	1	2.59 Gbps	16	617 kpps	327 kpps	
User	VPP native (OpenSSL 1.0.1)	AES-CBC-128	SHA1	1	4	3.76 Gbps	16	850 kpps	525 kpps	
User	VPP native (OpenSSL 1.0.1)	AES-CBC-128	SHA1	8	8	9.46 Gbps	8	3.39 Mpps	2.26 Mpps	
User	VPP AESNI MB cryptodev	AES-CBC-128	SHA1	1	1	7.42 Gbps	8	2.38 Mpps	1.47 Mpps	
User	VPP AESNI MB cryptodev	AES-CBC-128	SHA1	1	4	8.38 Gbps	8	2.68 Mpps	1.47 Mpps	
User	VPP AESNI GCM cryptodev	AES-GCM-128-16		1	1	13.72 Gbps	8	3.27 Mpps	2.67 Mpps	
User	VPP AESNI GCM cryptodev	AES-GCM-128-16		1	4	10.93 Gbps	8	3.09 Mpps	2.67 Mpps	
User	VPP QAT cryptodev	AES-CBC-128	SHA1	1	1	34.68 Gbps	16	8.85 Mpps	3.76 Mpps	
User	VPP QAT cryptodev	AES-CBC-128	SHA1	1	4	35.72 Gbps	16	4.22 Mpps	4.16 Mpps	
User	VPP QAT cryptodev	AES-CBC-128	SHA1	8	8	36.24 Gbps	8	10.43 Mpps	7.47 Mpps	
User	VPP QAT cryptodev	AES-GCM-128-16		1	1	52.73 Gbps	8	3.38 Mpps	5.65 Mpps	
User	VPP QAT cryptodev	AES-GCM-128-16		1	4	32.68 Gbps	8	8.36 Mpps	3.66 Mpps	
Kernel	None	None			1			854 kpps	606 kpps	805 kpps
Kernel	None	None			4			2.33 Mpps	2.62 Mpps	2.01 Mpps
Kernel	None	None			8			5.18 Mpps	5.14 Mpps	3.28 Mpps
User	None	None			1			11.81 Mpps	8.84 Mpps	3.28 Mpps
User	None	None			4			32.51 Mpps	6.56 Mpps	3.28 Mpps
User	None	None			8			42.90 Mpps	9.26 Mpps	3.28 Mpps

VPP Lab testing

+

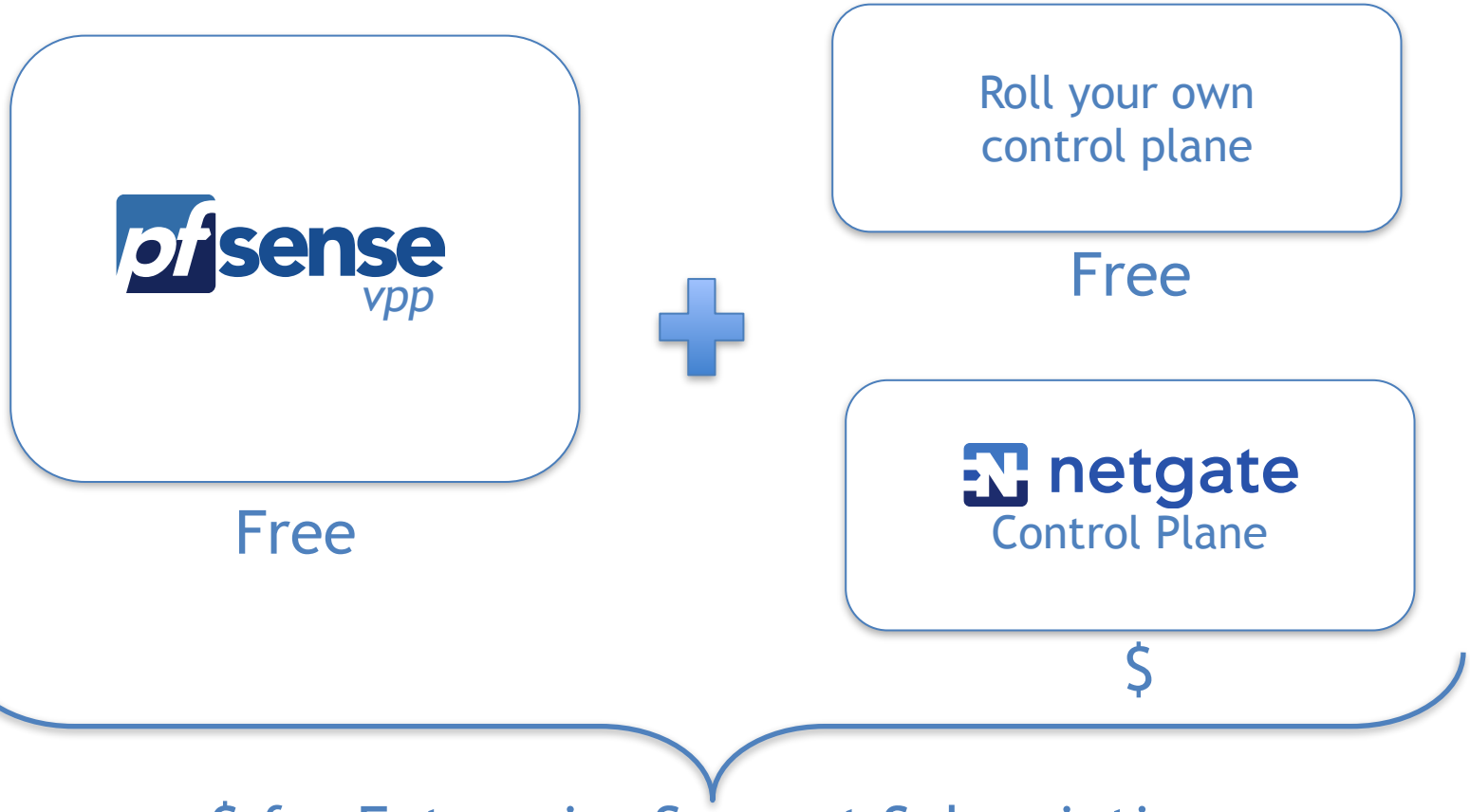
4Q17

NETCONF/RESTCONF/CLI control plane

Almost...

And it's all free & open source?

netgate



Programmers need to eat, too...

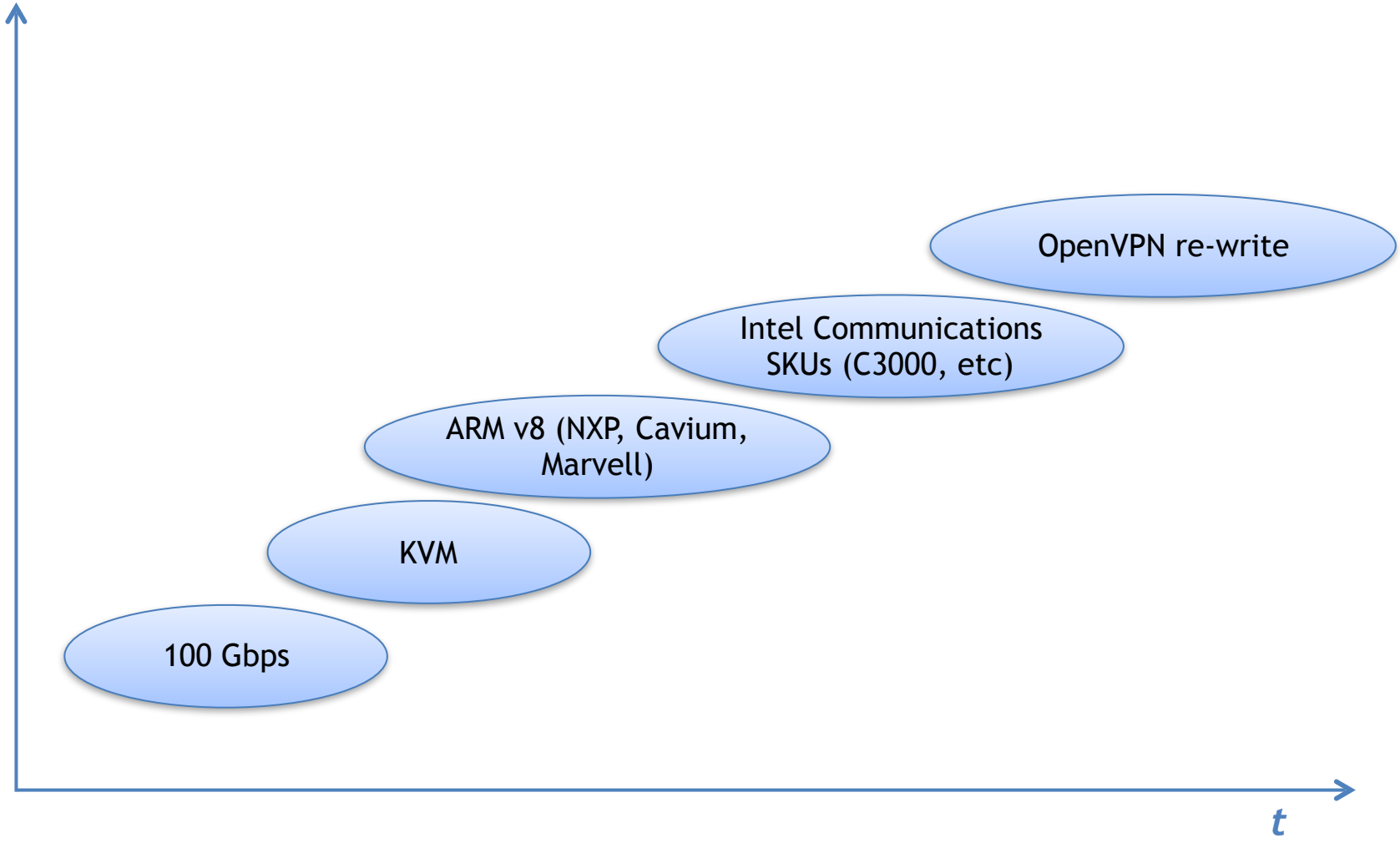
And why do I care?

netgate

The “middle man” cost is collapsed, yet again

What's Next?

netgate



References

- DPDK
  - http://www.dpdk.org
- VPP
  - https://wiki.fd.io/view/VP/What_is_VPP
- CPIC
  - https://www.netgate.com/products/cpic-8955.html
- CLIXON - NETCONF/RESTCONF/CLI
  - http://www.clicon.org
- “Building a Behemoth Router”
  - https://www.netgate.com/blog/building-a-behemoth-router.html

netgate

Thank You

Jim Thompson

Founder & CTO

Netgate

Story Line

netgate

1. Up until 2 years ago, 40 Gbps from off the shelf hardware was unthinkable
   - Processors vs. memory - Role Reversal
   - Packet Size – Big trouble comes in small package
2. Why unthinkable?
   - Processor cost
   - Moving data in and out of memory cost
   - Memory cost
3. But we have 40 Gbps boxes. How?
   - Specialized processors + offload processor
   - Memory has gotten much cheaper
   - Could not unhinge from specialty silicon because of kernel constraints
   - $$$ - High 5 figure box prices and up
4. What has changed?
   - Processors vs. memory – cost Role Reversal
   - But we needed innovation in the kernel for h/w to truly get there
   - But kernels are slow to change and for a good reason
     - It's a rip and replace game
5. So what innovation has made it possible?
   - VPP
   - What is that?
6. Is this a "Wankel engine"?
   - Nope. It came from Cisco
7. Why in the world would they let it go open?
   - Docker pressure
8. OK, and it's ready for prime time?
   - Yes and no.
   - Yes - look at this packet throughout data…
   - No - control plane sw still needed
9. And this is all open source?
   - Yes and no
   - There will be a version of pfSense that is a rocket – very soon
   - And you can of course build your own control plane with puppet, open stack et al
   - But's it really tricky
   - Ours will be easier to use and integrate, but it won't be free as it's our value add
   - And we'll have eprise service subscriptions to boot
10. Got it. Now, why do I care about all this?
    - The network infra game is changing fast
    - The middle box is gone
    - Security is shifting
    - Network infra price collapse is coming
    - Budget dollars will shift further to the edges
    - Network perimeter and zone fw and routing will not go away, but it will go hard to the hoop on high throughout for any biz app (think small packets) and ACLs will only increase
    - Open source is key ingredient in that movement
    - And in the end, OS can meet the challenge AND it will save beaucoup $$$ for e-prise, education and gov