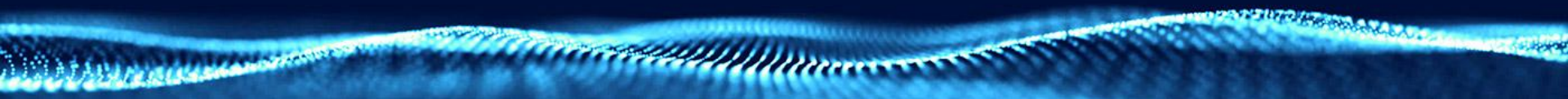
IP fabric rollout



Costas Drogos
<costasd@kentik.com>

kentik

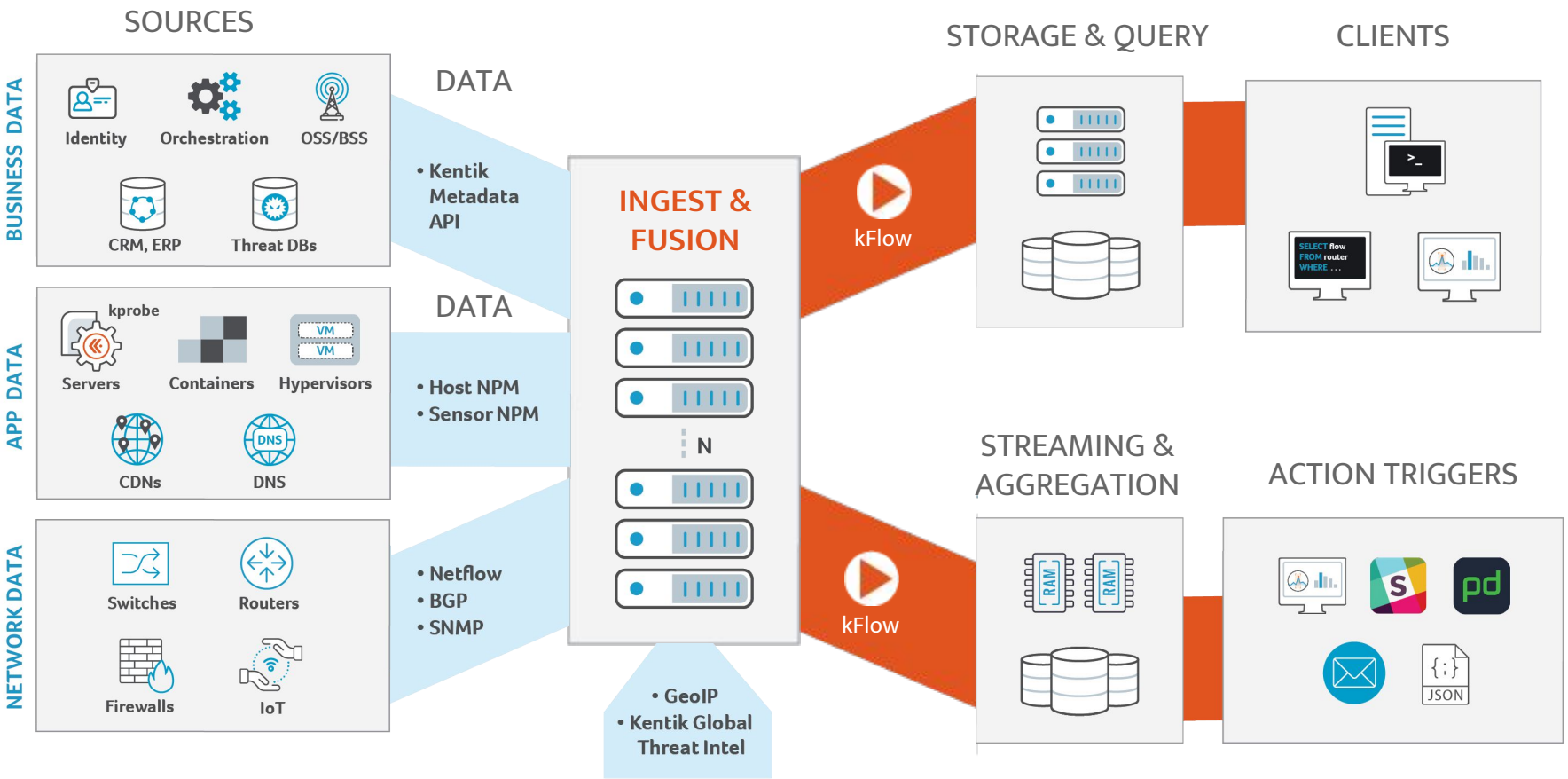
Note

- I’m not a full-time network engineer

- Will mostly use Junos terms
  - but everything here should apply on any other vendor if you replace the wording

- Feel free to interrupt at any time

Kentik Platform



Kentik is network sensitive

- UDP and TCP flow ingest
  - (packet loss sensitive)
- BGP ingest for flow enrichment
  - (stability, NOCs don't like flappy peerings)
- Hosts are DHCP'ed
  - (no DHCP, hosts become unreachable)
- Outgoing HTTP requests
  - (ie. actions or alerts to email/slack/pagerduty - need to be successful)
- Internal BGP must be stable - Hosts announce ingest IPs
  - (flaps mean ingest outages)
- Microservices, talking (mostly) over TCP/HTTP
  - (stable network or throughput will suffer)

Story

- Our traditional setup is a Virtual Chassis stack
- We got a new 10-rack cage
  - We need to be able to host more than one “cluster” of our product.
  - Each “cluster” may have different security requirements.
- We plan to run kubernetes at some point
- We try to build redundancy everywhere
- We like to tinker with new shiny things :)

IP fabric, leaf/spine ?

- RFC7938, “Use of BGP for Routing in Large-Scale Data Centers”
- Vendor-specific whitepapers, notes, terminology
- Other RFCs for more specific tech such as EVPN or VXLAN

- Different names, and implementations but (generally) the same idea:
  - Make L2 as small as possible: rack, server. That’s your “edge”
  - Use L3 routers beyond you L2’s designated “edge”
  - Use a L3 routing protocol such as BGP for dynamic routing
  - Optionally use some overlay over that L3 to have LANs across your “edges”

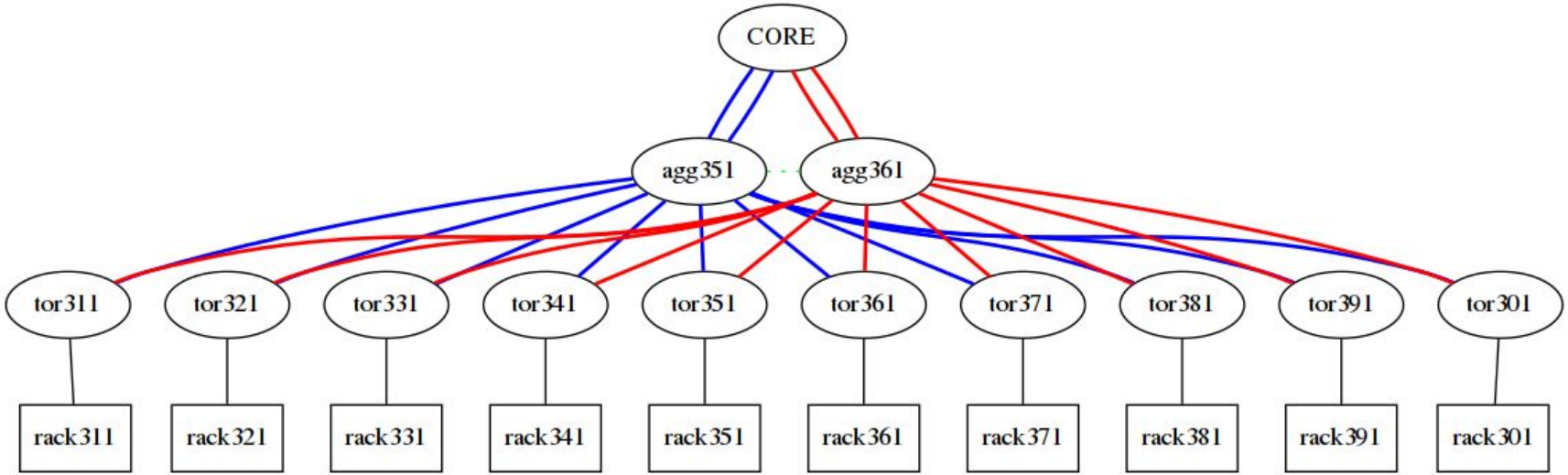
Why

- Containment: A misbehaving (not entirely down) switch doesn't affect the whole network
- Containment: Two different customers can be completely isolated
- Maintenance: Updates, debugging, config testing are easier
- No STP, no loops, no storms
- No network-wide floods
- No proprietary Virtual Chassis technologies and protocols
- Your network can now implement more internet-like features:
  - Routing policies and filters to influence traffic
  - MEDs, as-path prepends, localpref

Why Not

- At least two more devices are needed compared to a virtual chassis setup.
  - Each network device has a control plane now
- LAN is now local to {host, rack} - no flat vlans
  - Applications may need changes to function on an environment like that
  - Added complexity: if LAN architectures are needed across LAN boundaries, overlays must be used
- More devices, more cabling
- More internet resources needed (IP addresses, AS numbers)

Implementation: start with a drawing



Implementation: connected paths / redundancy

Where to add redundancy?

How will your design (and applications) behave with:

- A host going dark?
- A rack going dark?
- An aggregation switch going dark?

It’s not an all-or-nothing stack-goes-dark scenario anymore...

Implementation: L2

Where will your L2 boundaries be?

- Host
  - P-t-P /31, /127 down to each host
  - Each host is also a router, needs to have managed “routing” configuration
  - Flexible, a service may move anywhere (VMs, k8s etc)
- Rack
  - Allocate a subnet with a vlan and a default gw on the ToR (e.g. a /24, /64)
  - Not much change on how hosts view the network

Think: how servers will be provisioned - do you need DHCP? SLAAC? VRRP?

Think: Do you need public IPv4 addressing?

Implementation: ebgp/ibgp?

Pick what suits you better:

- eBGP
  - More ASNs and bgp configuration needed
  - Multipath can be used
  - Looks more WAN/DFZ/internet
- iBGP
  - Full mesh, or Route Reflectors (make sure they’re placed redundantly!)
  - Less traffic manipulation criteria
  - Less configuration

Implementation: Resources Needed

- Equipment, cabling, optics
  - Measure your bandwidth consumption on each topology level
    - Congestion will happen in near-to-full links, take that into consideration
  - 1 or 2 switch-routers per rack (is redundancy needed here?)
  - 2-4 aggregation switches depending on the architecture and redundancy planned

- Networks
  - IPv4 and IPv6 subnets sized to accommodate:
    - Loopbacks
    - P-t-P between network devices, and possibly also for hosts connected
    - Not all of them need to belong in the same, contiguous network, but it helps
    - Public networks?

Implementation: Resources Needed

- ASNs
  - Private AS numbers to use for each router
  - Go for 32-bit private ASNs!

- Human resources:
  - Think how this change will fit your and your team's existing tools and processes:
    - Configuration
    - Monitoring
    - Debugging

Implementation: v(x)lans, networks?

- Depends on whether you will have lans or terminate /31s on hosts
- Depends on whether you will have customers, segregation

We had to be backwards compatible:

- 1 private LAN per rack
  - With a default gateway and dhcp relay on the switch
- 1 public LAN per rack
  - Certain hosts do outgoing requests to customers' equipment

We plan to configure a vxlan to unite these public LANs at some point

Implementation: Automation

You now have more than one devices to manage

- But, they're easily groupable (e.g. leaf group or spine group)
- Prepare configs for first time provisioning
- Try to make policies as generic as possible to fit a whole group
  - Use communities, route-masks, filters to control flow inside the policy
- (Ab)Use every config reduction feature your vendor supports:
  - Templated/grouped configuration
  - allow bgp connections from ip range
  - Interface globs/masks/ranges
  - Routing instances

Implementation: Automation

- Consider embedding a bit of topology in your ASNs, IPv6 for quick debugging:
  - 2001:db8:100::24 for rack 100
  - 65232, 420000232 for rack 232
  - Vxlan IDs based on customer id
- Easy(-ier than a VC) to emulate with virtual routers
- Automate all the things!
  - If you don’t use any automation, it’s a good place to start
  - Git + programmatic way to push and rollback policies = success
  - No need to produce the whole router config on first try, build incrementally
  - Make your target to have policies synced everywhere

Was it worth it?

Yes:

- Debugging is simpler
- Maintenance is simpler
- Spawning a new rack is easier
- Flexibility with hosts networking

Things to consider:

- Without automation, you'll end up with different policies eventually
- More moving parts involved, need to monitor all of them
- Can be more expensive (short-term) to bootstrap

That’s all :)

Thanks!

Questions?