2014-03-19

Openstack FR- Meetup

« BaGPipe » datacenter virtual networking, Neutron-driven & BGP-based

Thomas Morin – Orange Labs
@netpeeker

self

- Thomas Morin
- Network engineer & architect at Orange Labs since 2004
- working on core IP/MPLS networks and more recently on datacenters and their interconnections with IP/MPLS networks
  - … network engineering, architecture, IETF, routers lab/testing and software development for prototyping
- (opensource: using Linux a)s my primary desktop OS for 15+ years

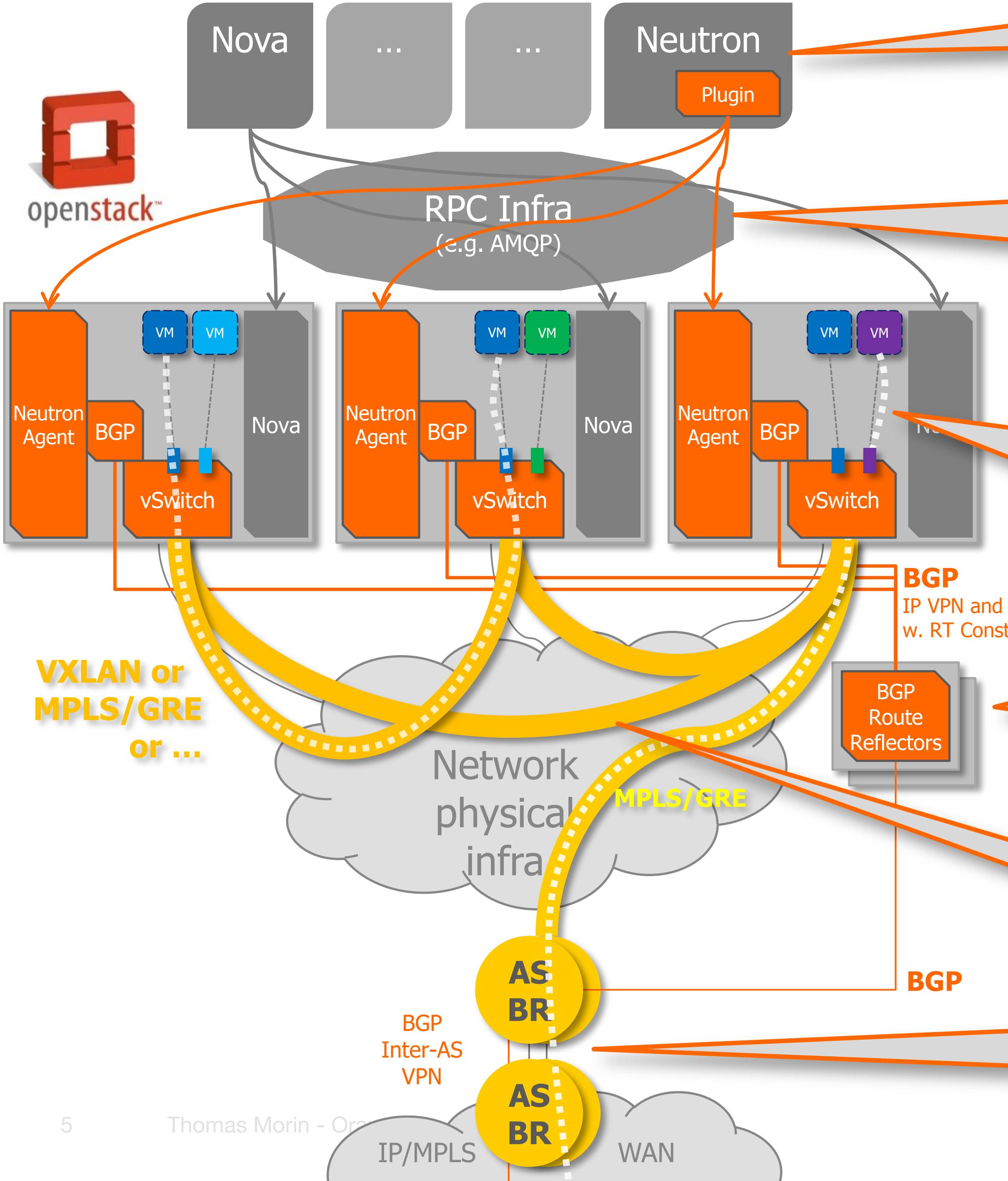
__init__

- Topic
  - datacenter network virtualization: automating the delivering of connectivity for large numbers virtual networks for virtual machines
  - and…
    - beyond virtual machines: external networks, bare-metal servers
    - more than connectivity
- Combination of two approaches:
  - Overlay networking driven directly by a Neutron plugin
  - Use BGP VPN extensions to interconnect...
    - with external IP VPNs through IP/MPLS router
    - with bare-metal servers or physical appliances through ToR switches
    - with other Openstack cells
- Discuss an architecture, illustrated based on our experience with a proof-of-concept implementation
- Discuss differences…
  - with SDN approaches relying on an external controller
  - with
- Discuss possible Neutron work in this direction

Quick intro on BGP VPN extensions

(from ietf.bgp.vpn import ip_vpn, e_vpn)

- (*not* about BGP for routing in the global Internet and *not* about your mainstream point-to-point SSL VPN)
- About providing IP or Ethernet connectivity between multiple sites, over a shared network operator infrastructure, and make it look like a private network infrastructure
  - typically for the enterprise market
- How ?
  - **encapsulating** to isolate the traffic of different virtual networks
    - in MPLS, MPLS-over-GRE, L2TP, VXLAN…
  - extensions to the BGP routing protocol are used to **announce VPN reachability**
    - “you can reach 1.2.3.0/24 for VPN X through router R using encapsulation Y”
    - “you can reach MAC de:ad:00:00:be:ef for VPN Z through R using encapsulation T”
    - + a publish/subscribe mode: “I want to receive information for VPN X,Y,Z…”
- For IP VPNs, the technology is very mature/proven
  - used since late 90’s
  - many operators deployments with millions of VPN sites and 100ks of virtual networks
  - solid standards (RFC4364 aka RFC2547)
- For Ethernet VPNs: more recent, but same architecture (draft-ietf-l2vpn-evpn)
- **These technologies are a very good fit for datacenter network virtualization**
  - (except we don’t want to play with the CLI and we don’t necessarily want it implemented in switches/routers)



Collapsing the "controller" into Neutron

Leveraging Openstack RPC infrastructure to trigger the instantiation of overlay endpoints.

The VM host server is a BGP VPN PE (E-VPN & IP-VPN). Lightweight stack, reduced to the minimum.

BGP Route Reflectors for a redundant & scalable control plane

Encapsulation: whatever works.

Seamless & distributed interconnection for extra-DC and inter-DC

“BaGPipe” an implementation of this architecture
(import bagpipe)

- We have implemented this architecture, with:
  - a Neutron plugin
    - (mostly doing BGP VPN identifier allocation – “Route Targets”)
  - a lightweight compute node BGP implementation
    - IP VPN and E-VPN, Route Target Constraints
    - no persistent configuration, no CLI, but a REST API (local to the compute node)
    - existing opensource components for the dataplane
      - OpenVSwich MPLS and Linux kernel’s native VXLAN
  - a local compute node agent
    - making the glue between the Neutron plugin and the BGP VPN implementation
- How complex ?
  - “on the shoulders of giants”… we reuse…
    - existing BGP Route Reflector implementations
    - ExaBGP Python BGP message encodings
    - opensource dataplanes: OVS and Linux kernel’s VXLAN
  - very few software dependencies, as close as possible to upstream
  - less than 6k lines written (Python), small team (total less than 2 man-year)
- Result…
  - standard-based intra-DC virtual networks (VXLAN encap)
  - seamless interconnects with external IP VPNs (MPLS/GRE encap)

Comparison with proposals centered around an external controller

- Why re-solve, for networking, problems that have to be solved for the rest of the Openstack infrastructure ?
  - High-Availability for APIs endoints
  - persistency challenges: DB distribution, synchronization and failover
  - controlling agents/vswitches on compute nodes
    - scaling challenge, AMQP, ZeroMQ…
  - deploying and configuring the components

  All this is already supposed to be done for Neutron and other Openstack components – why do it again and separately for the components of the "SDN" solution ?

- Scaling-out the "controller"
  - can re-use BGP distributed routing architecture
    - (some solutions do it today)
- Interconnecting with external MPLS-based VPNs
  - built-in when BGP is already used internally inside the DC
    - (again, some solutions do it today)
  - need to implement gateways when not
    - control plane GWs or even dataplane gateways
    - additional engineering/deployment work
    - potential scaling bottlenecks

Comparison with ML2 with L2 Population

- L2 Population approach
  - use Openstack RPC to distribute information about which server hosts MAC X/IP Y of virtual network Z
  - very lightweight compared to introducing BGP in the architecture, with or without an intermediate “SDN” controller
  - but probably limited to an AMQP domain and nova ports
    - what about bare-metal ? physical appliances ?
    - what about multi-DC, multi-cell ?
- Neutron-driven BGP-based virtual networks…
  - use Openstack RPC to create VPN instances on servers
  - use BGP to distribute information about which server hosts MAC X/IP Y of virtual network Z
  - not as lightweight as L2 POP, but:
    - offers a path to interconnect bare-metal servers or appliances through ToR switches implementing E-VPN
    - offers a path to interconnect with IP VPNs external to datacenters
      - important for many (cloud+network) operators
      - => fully distributed fashion: no dataplane or control plane bottleneck
    - (can offers a path for multi-cell Neutron)

BaGPipe current status and next steps

- current status: “proof-of-concept”-level in terms of usability
  - Neutron plugin written for Grizzly, not ported yet
  - patched Horizon to access API extension for external VPN connectivity
  - requires a recent Linux kernel for VXLAN (same as ML2 L2 Pop)
  - MPLS requires OVS patch under review (planned for inclusion in OVS 2.2)
  - MPLS/GRE with OVS requires a patch (work in progress)
- next steps…
  - opensource BaGPipe BGP component
  - reuse BaGPipe BGP component to implement the neutron-bgp-mpls-vpn blueprint in conjunction with ML2 plugin and neutron OVS agent
  - implement Neutron networks with BaGPipe, via an ML2 mechanism driver (rather than a full-blown Neutron plugin)
- other areas possibly worth looking at… (not exhaustive…)
  - implement a BaGPipe dataplane driver to use OpenContrail vrouter GRE/MPLS and VXLAN implementation (why not ?)
  - identify an opensource BGP implementation usable as BGP Route Reflector for VPN routes

Link with other Neutron work in progress…

- L3VPN connectivity (neutron-bgp-mpls-vpn)
- distributed router
- more modular OVS agent
- group-based policy API
- services insertion, chaining, and steering
- multinode support in Openstack functional integration testing
- ceilometer: what about network information ?
- interconnection between ports in different Neutron scopes

Wrap up…

- BaGPipe is a small project, don’t even try to compare with existing products
- We are trying to outline an approach that would bring the best of current proposals…
  - as lightweight as possible
  - keep BGP for interconnections
  - leverage Openstack RPC infra when relevant
  - avoid chokepoints by distributing
- Maybe others will want to explore this path as well
  - (complementary with other approaches on some aspects!)
- You’ll hear more about BaGPipe soon !

merci