Orchestration in a real network: a case study

Fulvio Risso, Politecnico di Torino, Italy
Outline

• The multidomain orchestration ride @ POLITO/TIM
  • Steps 1...2...3...4...5

• What we learned

• What to do next?
OpenStack overview

Controller Node

- Generic object store (Swift)
- VM images manager (Glance)
- Authentication Server + user profiles (Keystone)
- Compute API (Nova)
- Network API (Neutron)

- Service-layer orchestrator (Heat)
- Dashboard (Horizon)

- Nova scheduler
- Modular layer 2 (ML2)
- OVS
- Linux-bridge
- ODL

- Nova compute agent
- libvirt

- VM
- Software switch (OVS)

Note: arrows show the most important communication paths among components

Network infrastructure
Step 1: Pure OpenStack

OpenStack is a solution for **data centers**

- No traffic steering
- VMs scheduled based on availability of compute resources
  - Cannot schedule VMs based on network constraints
  - Does not consider network topology/status
- Not able to control/get feedback from the network (overlay model, tunnels in the vSwitch)
Step 2: OpenStack + Network Controller

- No traffic steering
- VMs scheduled based on availability of compute resources
  - Cannot schedule VMs based on network constraints
  - Does not consider network topology/status
- Not able to control/get feedback from the network (overlay model, tunnels in the vSwitch)

- The network controller can do its best to implement the service requested by OpenStack
  - Better than nothing, but it looks like a “slave” of OpenStack
Step 2b: Deeply modified OS + NC

- Traffic steering
- VMs scheduled based on availability of joint network and compute resources
  - Capability to react to compute/network events and re-schedule the service

- Deep modifications across all the software stack
- Our “OpenStack+” working in TIM/POLITO, then abandoned

Main points modified
Step 3: An overarching Orchestrator

- The (extended) network controller can be “remotely controlled” to “interpret” the service coming from OpenStack
- Complex service logic can be implemented in the orchestrator, such as
  - Suggesting the proper VM scheduling (e.g., through availability zones) to OpenStack
  - Relocating the service based on network feedback (network-aware scheduling in the orchestrator)
- Now OpenStack becomes the “slave” of the orchestrator

- Rather complex
  - E.g., traffic steering has to be added by the Network Controller
Step 3: other problems arose

• OpenStack probably not appropriate to control the entire infrastructure of a telco
  • Probably OK for the central datacenter and the POP mini-datacenter
  • CPEs does not fit well in the picture
    • Either domestic (almost no compute capabilities) or business (some compute capabilities may be available) CPEs
    • The network infrastructure may need a different controller

• Telecom Italia experimental network (JOLnet) may be controller better by defining multiple domains
  • In order to understand the reason, let’s have a look at the JOLnet infrastructure
The JOLnet SDN infrastructure

Note: some links have been omitted for the sake of clarity.
Step 3: overarching orchestrator (in JOLnet)

**Orchestrator**

- OpenDayLight control adapter
- OpenStack control adapter

**Network controller**

**OpenStack controller**

**Core (Turin)**

**CPE (Turin)**

**Core (Pisa)**

**CPE (Pisa)**

**Note:** some links have been omitted for the sake of clarity.
Step 4: multi-domain orchestrator

Overarching orchestrator

OpenDayLight control adapter

OpenFlow Domain Orchestrator

Network controller

OpenStack control adapter

OpenStack Domain Orchestrator

OpenStack controller

Universal Node

Core (TurinU)

CPE (TurinU)

Core (Pisa)

CPE (Pisa)

Note: some links have been omitted for the sake of clarity.
Step 4b: where’s the Service Layer?

How can the Service Layer know which **port** of the CPE the user is connected to, and the user **MAC address**?
Step 5: bus-based architecture

1. Domain capabilities (e.g., support for VLANs, for GRE tunnels)
2. Domain resources (e.g., VLAN IDs 7-12 available, GRE available on IP address 10.1.1.1)
3. Domain external topology (e.g., direct Ethernet link with Domain 2)
4. Service layer information (e.g., user Green connected to port 1/0)

Message bus (pub/sub model with YANG/OpenConfig description)

Portion of the service graph associated to the selected domain

Universal Node (CPE) Domain Orchestrator

OpenFlow Domain Orchestrator

OpenStack Domain Orchestrator

Bittorrent client

Firewall

Internet

BitTorr. client

Firewall

Bridge

Internet

VLAN 7

VLAN 12

(1) Domain capabilities (e.g., support for VLANs, for GRE tunnels)
(2) Domain resources (e.g., VLAN IDs 7-12 available, GRE available on IP address 10.1.1.1)
(3) Domain external topology (e.g., direct Ethernet link with Domain 2)
(4) Service layer information (e.g., user Green connected to port 1/0)
What we learned

We’re hitting the top of the iceberg.

Orchestration is very hard.

Orchestration is not just optimized scheduling.

Orchestration is:
  scalability,
  multitenancy,
  security,
  isolation,
  multiple technological domains,
  multiple administrative domains,
  support for Internet of Things

Anything else?
What to do next

• How many orchestrators do we have to design and engineer?
  • One fits all (hence one winner and so many losers), or should we design domain-specific orchestrators?
    • OpenStack, network-only, ...

• Some possible technical actions
  • Define a **detailed** list of technical requirements?
  • Define a common **language** between orchestrators?

• More collaboration among the partners would be helpful
  • The orchestration space is so big!
Credits

• The people at TIM, who we work with
  • Special thanks to Antonio Manzalini, Mario Ullio, Vinicio Vercellone, Matteo D’Ambrosio, and many others
• The EU FP7 UNIFY project and all the people there
  • Universal Node was born there: http://github.com/netgroup-polito/un-orchestrator
• The crew at POLITO
  • ... for their effort in the FROG: http://github.com/netgroup-polito/frog4
  • Ivano Cerrato, Stefano Petrangeli, Roberto Bonafiglia, Sebastiano Miano, Gabriele Castellano, Francesco Benforte, Francesco Lucrezia, Mauricio Vasquez
    • Our past students, of course: Fabio Mignini, Alex Palesandro, Matteo Tiengo, Giacomo Ratta, Patrick Tomassone, Andrea Vida, Marco Migliore, Alessio Berrone, Sergio Nuccio... and many others
• The people at EIT, which are pushing hard for multi-domain orchestration
eitdigital.eu

Thanks for your attention!