Introduction of the SONATA project / Distributed VNF State Management

Manuel Peuster, Sevil Mehraghdam, Holger Karl

27.10.2015
Agenda

• Marketing: The 5G-PPP project SONATA

• VNF state management
  • Problem
  • Existing approaches
  • UPB’s Approach (ongoing work)
  • Future work
5G-PPP project “SONATA”

Agile Service Development and Orchestration in 5G Virtualized Networks
DevOps for NFV

DevOps

Plan

Test

Develop

Deploy

Monitor

Operate
Intended Outcome

1. An open source **service platform** with a modular **orchestration** framework that will help network operators optimize resource utilization, increase automation, reduce OpEx and facilitate their NFV transition.

2. A **programming model and SDK** that will empower service providers, network equipment vendors and SMEs to develop services based on new or existing network functions, opening the market and reducing time to delivery.

3. A set of **DevOps tools and methodology** that will help connect these stakeholders to a modern, agile workflow that supports the rapid development cycles of software-driven networks, as well as the inter-organizational challenges between them.
Project and Partners

- Start: 01/07/2015
- Duration: 30 Months

- Large consortium (15 organizations)
  - 3 Telecom Operators
  - 4 Vendors / IT System Integrators
  - 5 Research centers / universities
  - 3 SMEs
Upcoming

• Q4 2015: Requirements and 1st Architecture
• Summer 2016: First Prototype Integrated Release

Contact

• visit us at: www.sonata-nfv.eu
• Twitter: @sonataNFV
VNF State Management
VNFs and elasticity

Problem: VNFs are often stateful!

→ We can move flows between replicas but what about the corresponding state?
What to do?

• We want to have a “global view” across all instances
  • Think of “IDS match counters”
• ETSI reference architecture
  • Use a “central state repository”

What exactly is this? And how does it work?
Exploit typical per-flow state structure

Figure: Typical middlebox state structure [1], [3]

- Flow state identified by 5-tuple (sIP, sPort, dIP, dPort, protocol)
Example: OpenNF

- **State Manager**
  - `move`
  - `get + del`
  - `{state}`

- **Flow Manager**
  - `put {state}`

- **Control Application**

Existing work

- **OpenNF**
  - Central controller with state manager
  - Northbound API to control app.: move, copy, share
  - Southbound API to VNFs: put, get, del
    - State “chunks” (serialized data structures)
  - Loss-free, order-preserving
  - Improved versions with direct peering

- **Split/Merge**
  - Central control
  - Shared lib offering memory allocation API
  - State directly shared btw. Shared libs.
  - No global view

---


UPB’s approach (ongoing work)

- Get rid of central state-aware control application
- Build a fully distributed system
- No dependencies to orchestration layer
- “React” on SDN controlled flow moves

ETSI reference architecture
Ideas

• Shared library holding local state of VNF instance
  • Fast local access
• Operation to receive a global view on the state of all replicated VNF instances
  • Assumption: Global view updates needed less frequently, or when flows are moved between instances
  • Attention: Eventual consistency model

• Simple API
  • set(key, state)
  • get(key) : state
  • del(key)

  local

  • get_global(key, red_func_ptr) : state
  • red_func(list<state>) : state

• Flow state keys are typically 5-tuples (sIP, dIP, sPrt, dPrt, prot) with wildcards
API Usage Example

VNF 1

set("FlowA", v)
get("FlowA")

VNF 2

get_global("Pcount", sum())
set("Pcount", v)
set("Pcount", v)

FlowA
FlowB
Prototype: libestate

- LIBrary for Elastic SATATE

- Shared library implemented in C++
  - Offers standard C interface
  - Python wrapper available

- Communication
  - ZeroMQ
  - Pub/sub pattern for get_global request
  - Direct ZMQ message for replies
Prototype: libestate

- **Experiment setup**
  - “dummy IDS” middlebox
  - Mininet/Maxinet environment
  - Comparison: baseline (local only), centraldb, libestate
Ongoing / future work

- **Ongoing**
  - Integration with “real” middlebox project: PRADS asset monitor
  - Direct comparison to OpenNF with PRADS implementation

- **Future**
  - Add SDN handover procedure to support state move
    - Goal: loss-free moves like OpenNF
  - In detail performance evaluation
    - Different e.g. loss/delay conditions in management network, packet drop behavior
  - Study effect of weak consistency model on “real” middlebox software
Thanks.