A case for an Intent-based control language in NFV environments

Pedro A. Aranda Gutiérrez

Universidad Carlos III de Madrid *paranda@it.uc3m.es*

July 19, 2018 - Montreal, QC, Canada

Introduction

What is NEMO

- Originally NEMO was NEtwork MOdelling language (more info in `http://nemo-project.net`)

Introduction

What is NEMO

- Originally NEMO was NEtwork MOdelling language (more info in `http://nemo-project.net`)
- Heavily adopted for the NFV use case since the last IETF in Berlin.

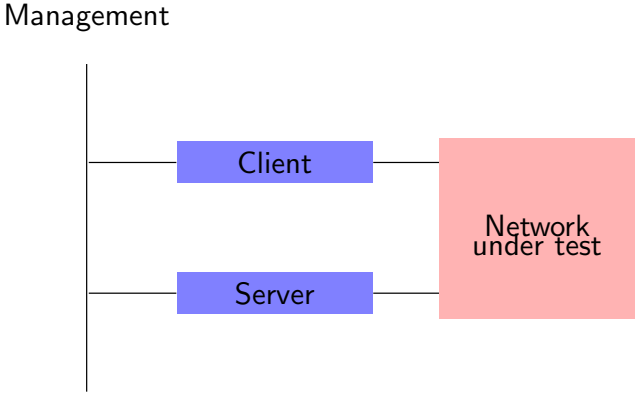
Introduction

What is NEMO

- Originally NEMO was NEtwork MOdelling language (more info in `http://nemo-project.net`)
- Heavily adopted for the NFV use case since the last IETF in Berlin.
- The beauty of it in the context of NFV:
  - NSD graphs consist of two elements: VNFCs and Links
  - VNFCs translate directly to NodeModels
  - And well, the networks connecting VNFCs translate directy into LinkModels

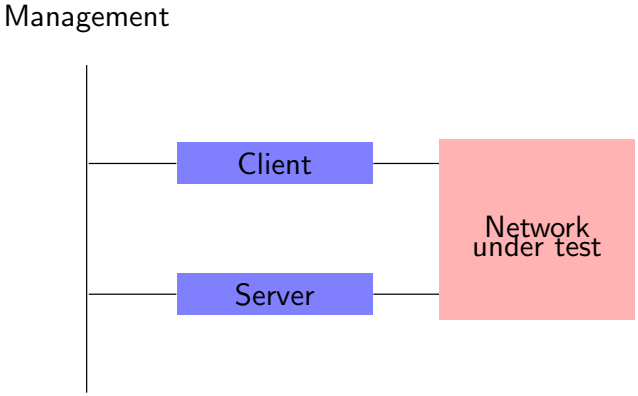
An example

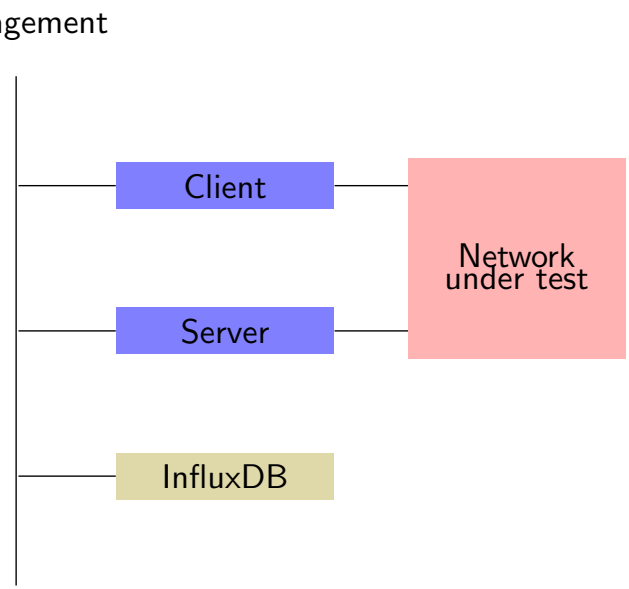
A NS to characterise networks in four different impersonations



An example

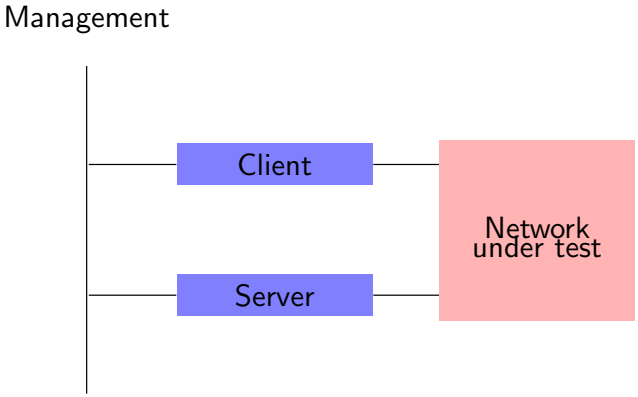
A NS to characterise networks in four different impersonations

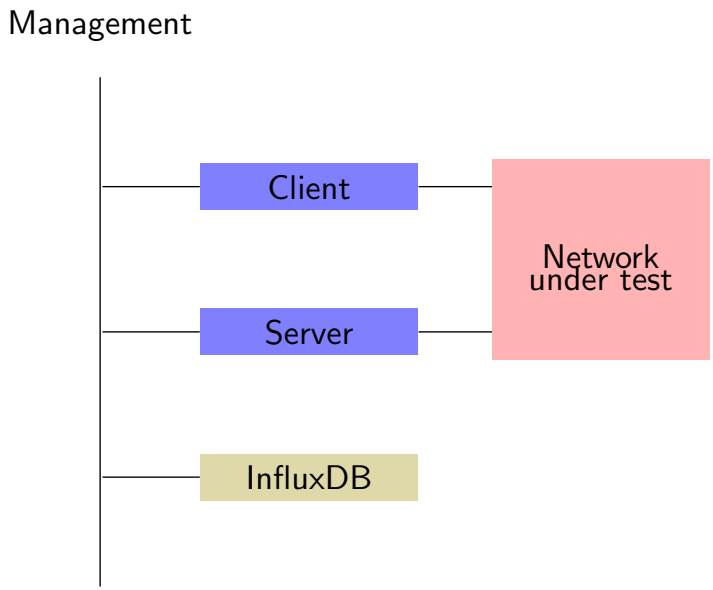


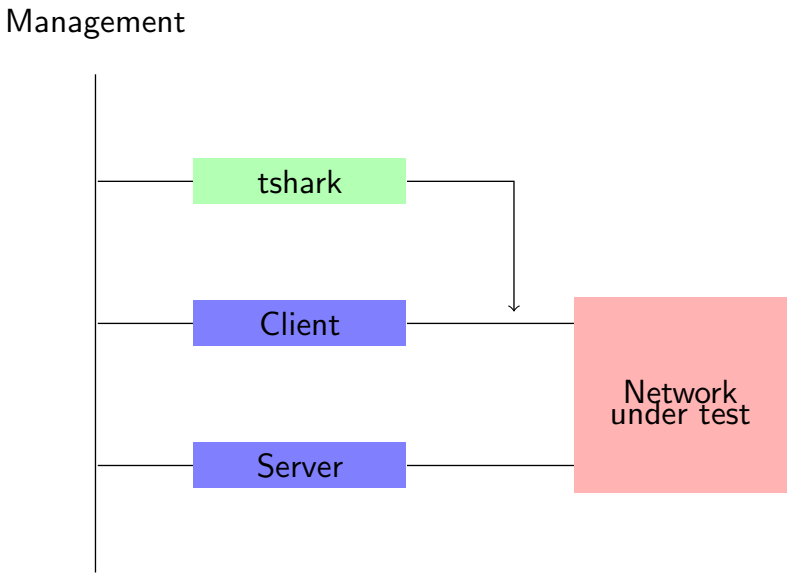


An example

A NS to characterise networks in four different impersonations

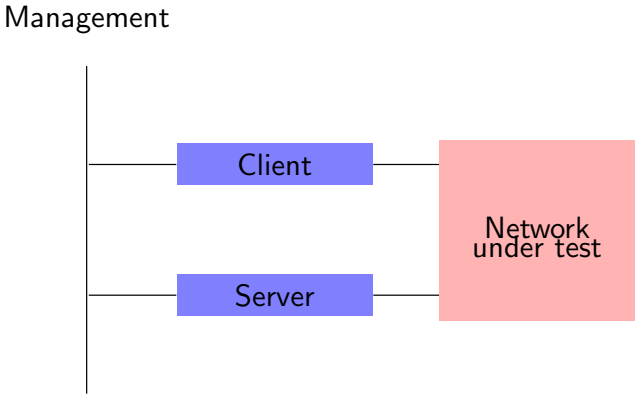


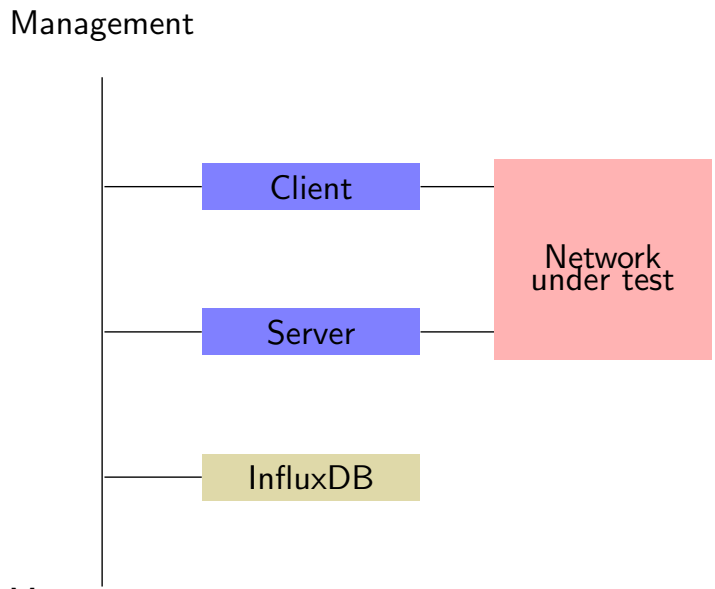


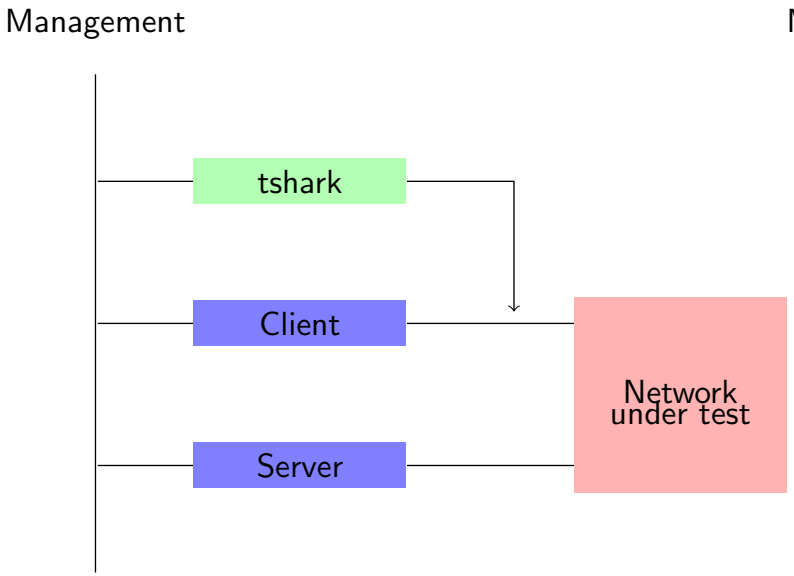


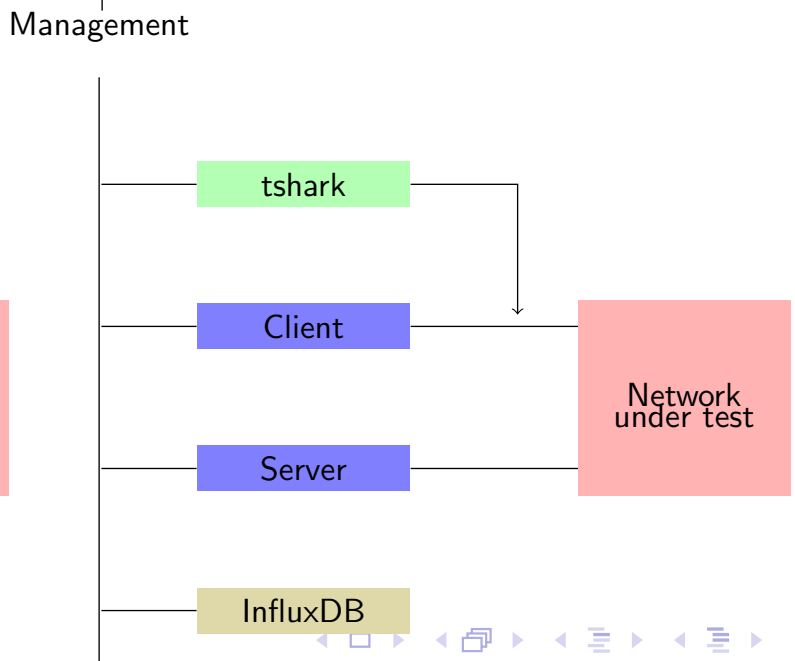
An example

A NS to characterise networks in four different impersonations









NEMO vs. YAML

Do we still write programs in Assembler?

- The 1970's vs today's approach

NEMO vs. YAML

Do we still write programs in Assembler?

- The 1970's vs today's approach

```
---
schema_version:  2
scenario:
  name:          lola-trafic
  description:   LoLa measurement
  vnfs:
    servers:
      vnf_name:  alpine-trafic
    clients:
      vnf_name:  alpine-trafic
    tshark:
      vnf_name:  alpine-trafic
    influxdb:
      vnf_name:  alpine-trafic
  networks:
    control:
      type:      bridge
      external:  true
      interfaces:
      - servers:  eth0
      - clients:  eth0
      - tshark:  eth0
      - influxdb:  eth0
    user-eq:
      type:      bridge
      external:  true
      interfaces:
      - clients:  eth1
      - sniffer:  eth1
```

NEMO vs. YAML

Do we still write programs in Assembler?

- The 1970's vs today's approach

```
---
schema_version:  2
scenario:
  name:          lola-trafic
  description:   LoLa measurement
  vnfs:
    servers:
      vnf_name:  alpine-trafic
    clients:
      vnf_name:  alpine-trafic
    tshark:
      vnf_name:  alpine-trafic
    influxdb:
      vnf_name:  alpine-trafic
  networks:
    control:
      type:      bridge
      external:  true
      interfaces:
      - servers:  eth0
      - clients:  eth0
      - tshark:  eth0
      - influxdb:  eth0
    user-eq:
      type:      bridge
      external:  true
      interfaces:
      - clients:  eth1
      - sniffer:  eth0
```

```
CREATE NodeModel alpine-trafic
    VNFD file:///<repo>/alpine-vnfc.yaml;
CREATE NodeModel lola-experiment;
    Node iperf-server Type lola-trafic;
    Node iper-client Type lola-trafic;
    Node tshark Type lola-trafic;
    Node influxdb Type lola-trafic;
    ConnectionPoint control;
    ConnectionPoint ue-net;
    ConnectionPoint prov-net;
    ConnectionPoint sniff;
    Connection ctl Type lan
        Endnodes iperf-server:control, iperf-client:control,
                 tshark:control, influxdb:control;
    Connection ue Type p2p Endnodes ue-net, iperf-client:measure
    Connection prov Type p2p Endnodes prov-net, iperf-server:mea
    Connection prov Type p2p Endnodes sniff, tshark:measure;
```

A goodie

Incremental development

```
CREATE NodeModel alpine-trafic
    VNFD file:///<repo>/alpine-vnfc.yaml;
CREATE NodeModel lola-experiment;
    Node iperf-server Type lola-trafic;
    Node iper-client Type lola-trafic;
    ConnectionPoint control;
    ConnectionPoint ue-net;
    ConnectionPoint prov-net;
    Connection ctl Type lan
        Endnodes iperf-server:control, iperf-client:control;
    Connection ue Type p2p Endnodes ue-net, iperf-client:measure;
    Connection prov Type p2p Endnodes prov-net, iperf-server:measure;
```

A goodie

Incremental development

```
CREATE NodeModel alpine-trafic
    VNFD file:///<repo>/alpine-vnfc.yaml;
CREATE NodeModel lola-experiment;
    Node iperf-server Type lola-trafic;
    Node iper-client Type lola-trafic;
    ConnectionPoint control;
    ConnectionPoint ue-net;
    ConnectionPoint prov-net;
    Connection ctl Type lan
        Endnodes iperf-server:control, iperf-client:control;
    Connection ue Type p2p Endnodes ue-net, iperf-client:measure;
    Connection prov Type p2p Endnodes prov-net, iperf-server:measure;
```

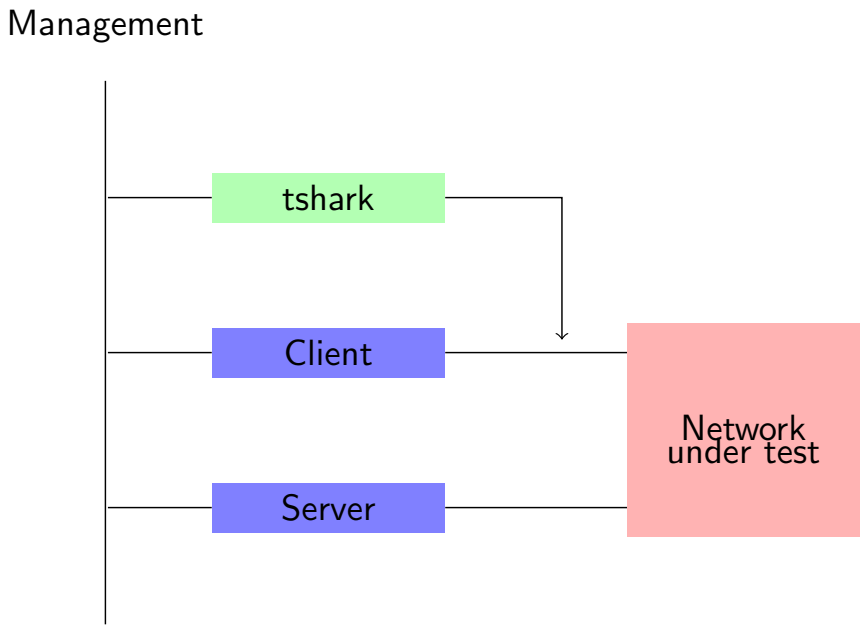
```
CREATE NodeModel lola-experiment;
    Node measure Type basic-lola;
    Node database Type lola-trafic;
    ConnectionPoint control;
    ConnectionPoint ue-net;
    ConnectionPoint prov-net;
    Connection ctl Type lan
        Endnodes measure:control, database:control;
    Connection ue Type p2p Endnodes ue-net, basic-lola:ue-net;
    Connection prov Type p2p Endnodes prov-net, basic-lola:ue-net;
```

Yet another goodie

- NEMO has *LinkModels* in addition to *NodeModels*

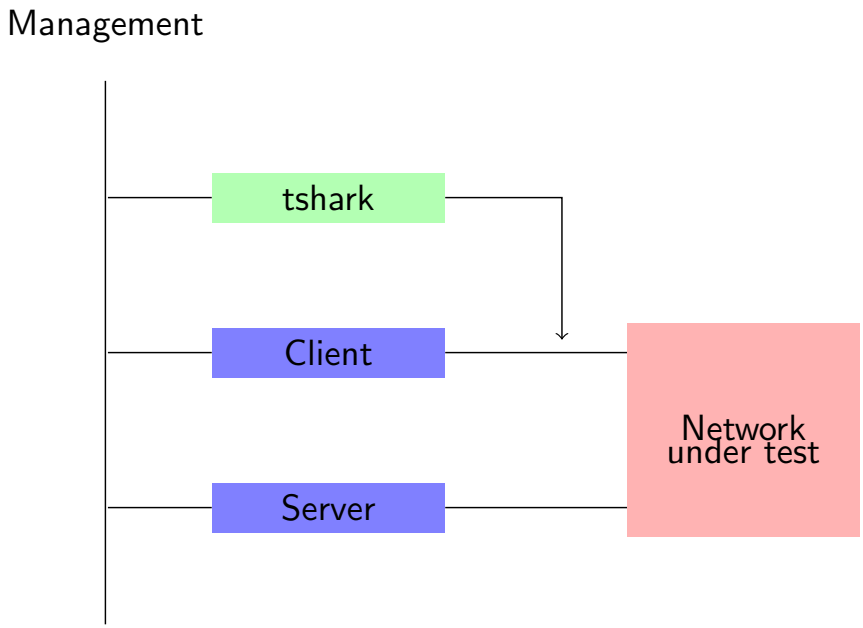
Yet another goodie

- NEMO has *LinkModels* in addition to *NodeModels*



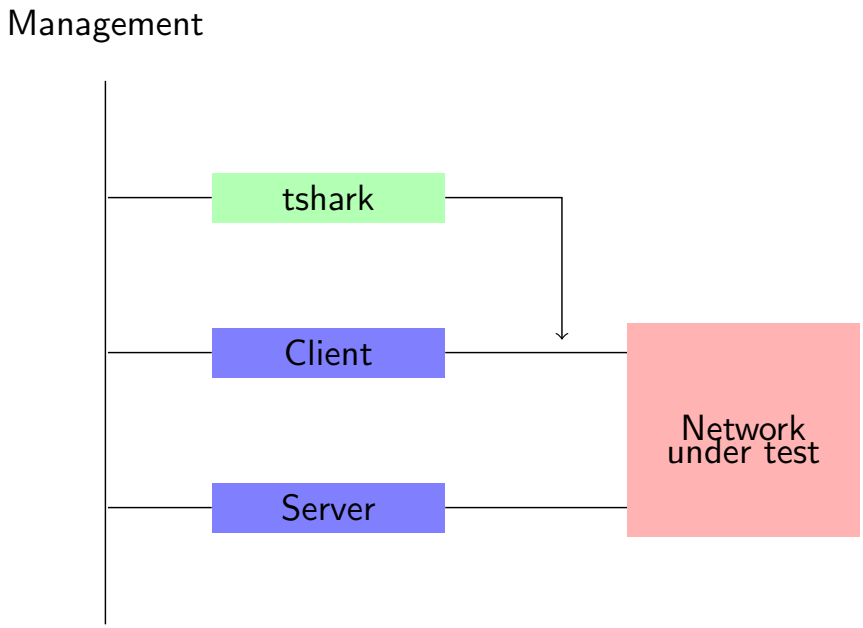
Yet another goodie

- NEMO has *LinkModels* in addition to *NodeModels*
- Once the underlying MANO infrastructure supports it, we can think of modelling the connection of the tshark VNFC as a Hub or as a TAP



Yet another goodie

- NEMO has *LinkModels* in addition to *NodeModels*
- Once the underlying MANO infrastructure supports it, we can think of modelling the connection of the tshark VNFC as a Hub or as a TAP
- And profit from the recent introduction of *TAP as a service* in OpenStack.



Final thoughts

Alla Breve

- Call it *Intent*, call it *High Level*, call it *Human Understable*

Final thoughts

Alla Breve

- Call it *Intent*, call it *High Level*, call it *Human Understable*
- The main thing is that we leave *machine readable* ASCII representations for *machines*

Final thoughts

Alla Breve

- Call it *Intent*, call it *High Level*, call it *Human Understable*
- The main thing is that we leave *machine readable* ASCII representations for *machines*
- Because, while you may grasp what is happening, odds are that you will get a *wrong* understanding

Final thoughts

Alla Breve

- Call it *Intent*, call it *High Level*, call it *Human Understable*
- The main thing is that we leave *machine readable* ASCII representations for *machines*
- Because, while you may grasp what is happening, odds are that you will get a *wrong* understanding
- And even when you understand what is going on in the YAML...

Final thoughts

Alla Breve

- Call it *Intent*, call it *High Level*, call it *Human Understable*
- The main thing is that we leave *machine readable* ASCII representations for *machines*
- Because, while you may grasp what is happening, odds are that you will get a *wrong* understanding
- And even when you understand what is going on in the YAML...
- How big was the effort, compared with what it would have taken you to understand the NEMO files

Final thoughts

Alla Breve

- Call it *Intent*, call it *High Level*, call it *Human Understable*
- The main thing is that we leave *machine readable* ASCII representations for *machines*
- Because, while you may grasp what is happening, odds are that you will get a *wrong* understanding
- And even when you understand what is going on in the YAML...
- How big was the effort, compared with what it would have taken you to understand the NEMO files
- And, BTW, how would you express the TAPaaS example in YAML?

Thank you for your attention

Pedro A. Aranda
mailto:paranda@it.uc3m.es