An Architecture for Creating and Managing Virtual Networks

David Pérez Caparrós, Ishan Vaishnavi, Stefan Schmid, Ashiq Khan

Network Research Group
DOCOMO Euro-Labs
Munich, Germany
Outline

• Motivation

• Network Configuration Platform (NCP)
  – Architecture
  – Proof-of-Concept

• Conclusions & Future Work
Motivation

Requested Virtual network

Physical Infrastructure

v: Virtualized  DC: Datacenter  S-GW: Serving Gateway  P-GW: PDN Gateway
MME: Mobility Management Entity  HSS: Home Subscriber Server

vMME  vHSS  vP-GW  vS-GW
Motivation (II)

Requested Virtual network

Deployment blueprint

Physical Infrastructure

v: Virtualized  DC: Datacenter  S-GW: Serving Gateway  P-GW: PDN Gateway
MME: Mobility Management Entity  HSS: Home Subscriber Server

Copyright © 2013 DOCOMO Communications Laboratories Europe GmbH  Network Research Group
Motivation (III)

**Goal:** Automate the provisioning of end-to-end *virtual networks* composed by virtual network elements that have the *operational model of a virtual machine* (CREATE, START, STOP, DELETE...)
Network Configuration Platform (NCP)
Architecture

Requested Virtual network

vS-GW
vMME
vHSS
vP-GW

Physical Infrastructure

CMS
DC1

NMS
Transport NW

CMS
DC2

CMS
DC3

CMS: Cloud Management System
NMS: Network Management System
NCP: Network Configuration Platform

Copyright © 2013 DOCOMO Communications Laboratories Europe GmbH
Network Research Group
Network Configuration Platform (NCP) Architecture

Requested Virtual network

Multi-domain, multi-vendor orchestrator

Physical Infrastructure

CMS: Cloud Management System
NMS: Network Management System
NCP: Network Configuration Platform

Copyright © 2013 DOCOMO Communications Laboratories Europe GmbH
Network Research Group
Network Configuration Platform (NCP)
Architecture (II)

Request Handler
- Get VNet
- Create VNet
- Update VNet
- Delete VNet

Resource Discovery

Resource Manager

Monitoring (SLA, virtual topology)

Southbound I/F

Northbound I/F (API)

DB

NCP Agent
NMS

NCP Agent
CMS

DC1
DC2
DC3

vS-GW

vMME

Transport NW

vP-GW

vHSS

CMS: Cloud Management System
NMS: Network Management System
NCP: Network Configuration Platform
Outline

• Motivation

• Network Configuration Platform (NCP)
  – Architecture
  – Proof-of-Concept

• Conclusions & Future Work
Network Configuration Platform (NCP)
Proof-of-Concept

Service Request

Video Server 1
Paris

Media Proxy
Berlin

10 Mbps

Video Server 2
Madrid

Service Description following
Common Information Model / NDL

<table>
<thead>
<tr>
<th>Video Server 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type: /node/host/generic</td>
</tr>
<tr>
<td>Template: video-server</td>
</tr>
<tr>
<td>OS: Ubuntu</td>
</tr>
<tr>
<td>Arch: x86-64</td>
</tr>
<tr>
<td>RAM: 512 MB</td>
</tr>
<tr>
<td>Disk: 20GB</td>
</tr>
<tr>
<td>Loc: Europe/Paris</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Link 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type: /link/generic</td>
</tr>
<tr>
<td>Down: 10 Mbps</td>
</tr>
<tr>
<td>Up: 2 Mbps</td>
</tr>
<tr>
<td>Loss: 5%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Video Server 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type: /node/host/generic</td>
</tr>
<tr>
<td>Template: video-server</td>
</tr>
<tr>
<td>OS: Ubuntu</td>
</tr>
<tr>
<td>Arch: x86-64</td>
</tr>
<tr>
<td>RAM: 512 MB</td>
</tr>
<tr>
<td>Disk: 20GB</td>
</tr>
<tr>
<td>Loc: Europe/Madrid</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Link 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type: /link/generic</td>
</tr>
<tr>
<td>Down: 10 Mbps</td>
</tr>
<tr>
<td>Up: 2 Mbps</td>
</tr>
<tr>
<td>Loss: 5%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Media Proxy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type: /node/host/generic</td>
</tr>
<tr>
<td>Template: media-proxy</td>
</tr>
<tr>
<td>OS: Ubuntu</td>
</tr>
<tr>
<td>Arch: x86-64</td>
</tr>
<tr>
<td>RAM: 512 MB</td>
</tr>
<tr>
<td>Disk: 20GB</td>
</tr>
<tr>
<td>Loc: Europe/Munich</td>
</tr>
</tbody>
</table>

- Prototype uses NDL based on FleRD (developed in collaboration with TU Berlin)
- We need a standardized virtualization-aware Information Model / NDL

"alias":"Video Streaming Service",
"networkElements":{
  "id":1,
  "alias":"Video Server 1",
  "type":"/node/host/generic",
  "features":{
    "attribute":"arch",
    "value":"x86_64"
  }
},

Lightweight text-data interchange format (JSON, XML...)

Copyright © 2013 DOCOMO Communications Laboratories Europe GmbH
Network Research Group
Network Configuration Platform (NCP)
Proof-of-Concept (II)

Use Case: Geographically Distributed Video Streaming Service

-- Diagram --

Service Provider GUI

REST API

JSON

"alias":"Video Streaming Service",
"networkElements":{
  "id":1,
  "alias":"Video Server 1",
  "type":"/node/host/generic",
  "features":{
    "attribute":"arch",
    "value":"x86_64"
  }
}

---

Copyright © 2013 DOCOMO Communications Laboratories Europe GmbH

Network Research Group
Network Configuration Platform (NCP)
Proof-of-Concept (III)

Service Provider GUI
- Define end-to-end service
- Get access information

NCP Admin GUI
- View physical topology
- Check VNet mapping
Conclusions

• Summary
  – An architecture and its building blocks for orchestrating end-to-end services were proposed
  – A Proof-of-Concept of such an architecture was presented
  – Open research issues have been discussed

• Future Works
  – Standard based north- and southbound I/F design
  – Standard multi-layer, multi-vendor information model
  – Implement new resource allocation algorithm (see PIMRC’13 - Path Protection with Explicit Availability Constraints for Virtual Network Embedding)
Thank you!

David Pérez Caparrós
caparros@docomolab-euro.com
Network Configuration Platform (NCP)
Proof-of-Concept. REST API

• RESTful web API. CRUD (Create, Read, Update, Delete) operations
• Jersey 2.0 (Open Source implementation of JAX-RS)
• Some examples:
  – Create network:
    POST http://<NCP_IP>/NCP/api/v1/<USER_ID>/network
    Body: Network request in JSON or XML format
  – Get network information:
    GET http://<NCP_IP>/NCP/api/v1/<USER_ID>/network/<NETWORK_ID>
  – Modify network:
    PUT http://<NCP_IP>/NCP/api/v1/<USER_ID>/network/<NETWORK_ID>
    Body: Network request in JSON or XML format
  – Delete network:
    DELETE http://<NCP_IP>/NCP/api/v1/<USER_ID>/network/<NETWORK_ID>