Objective FP7-ICT-2009-5-257448/D-5.5

Future Networks

Project 257448

"SAIL – Scalable and Adaptable Internet Solutions"

D-5.5
(D.D.4) Integrated Prototype System for Selected Use Cases

Date of preparation: **2013-02-28**
Start date of Project: **2010-08-01**
Project Coordinator: **Thomas Edwall**
Ericsson AB

Revision: **1.0**
Duration: **2013-02-28**

Document Properties

Document Number:	D-5.5
Document Title:	**(D.D.4) Integrated Prototype System for Selected Use Cases**
Document Responsible:	Azimeh Sefidcon (EAB)
Document Editor:	Fetahi Wuhib(EAB)
Authors:	João Monteiro Soares (PTIN), Vinay Yadhav (EAB), Daniel Turull (KTH), Suksant Sae Lor (HP), Matthias Keller (UPB), Rolf Stadler (KTH), Paul Périé (Lyatiss),
Target Dissemination Level:	PU
Status of the Document:	Final
Version:	1.0

Production Properties:

Reviewers:	Benoit Tremblay (EAB)

Document History:

Revision	Date	Issued by	Description
1.0	2013-02-28	Fetahi Wuhib	First Complete Version

Disclaimer:

Abstract:

This deliverable summarizes the various prototypes that were developed in WPD and demonstrated at two public events.

Keywords:

Cloud Networking, CloNe, Cloud Computing, Internet, SAIL, Prototypes, Evaluation

Executive Summary

Prototyping and experimentation are key to bringing research ideas to the market. With this in mind, a considerable amount of effort has been expended WP-D to realizing, via prototypes, the various concepts developed in the work package. The effort has resulted in seven prototypes whereby five where shown at the FuNeMS 2012 event and two at the final public event organized by SAIL. Selected components of the prototypes have also been made available to the public via open source.

Contents

Executive Summary iii

1 Introduction 1

2 Summary of Prototyped Demonstrators 2
2.1 Dynamic Enterprise on the CloNe Integrated Testbed . . . 2
2.2 Dynamic Resource Management . . . 2
2.3 libNetvirt: a Network Virtualization Library . . . 4
2.4 Model-driven Resource Allocation for Elastic Video . . . 5
2.5 In-Network Datacenter for Bulk Data Transfer . . . 6
2.6 An Administrative Portal for a Distributed Cloud . . . 6
2.7 Elastic NetInf on the CloNe Testbed . . . 8

3 Conclusion 9

Appendix: Posters for Demonstrated Prototypes 10

List of Acronyms 27

Bibliography 28

1 Introduction

Prototyping and experimentation are key to bring research ideas to the market. The purpose of prototyping is to build an early model for evaluating the new concepts or enhancing the precision of details. The result of prototyping can serve as a base for specification of a real system to be built. In SAIL WP-D, a considerable amount of effort has been expended in prototyping the various concepts developed in the work package. Most of the prototypes have been developed around a large-scale testbed that spanned four countries.

These prototypes have been demonstrated at two public events. The first batch of demonstrations were done at the 'Future Network & Moble Summit' (FuNeMS 2012) in Berlin Germany, from 4-6 July 2012 [1]. During this event, a Cloud Networking (CloNe) integrated testbed showing dynamic creation of Flash Network Slice (FNS)s spanning several countries (Section 2.1), a resource management system supporting dynamic resource allocation (Section 2.2), a virtualization library for networks (Section 2.3), model-driven resource provisioning for elastic video (Section 2.4) and an efficient technique for transferring bulk-data (Section 2.5) where shown.

The second batch of demonstrations were done at the 'Future Media Distribution using Information Centric Networks' event in Stockholm Sweden, on February 13, 2013 [2]. During this event, two more prototypes were demonstrated: an elastic deployment of Network of Information (NetInf) on the CloNe testbed using an adaptive deployment toolkit (Section 2.7) and a portal for the CloNe infrastructure showing the administrator's view of the distributed cloud (Section 2.6).

The purpose of this document is to summarize the various prototypes by (1) providing a description and a short overview of the results and (2) including the posters displayed during the events.

2 Summary of Prototyped Demonstrators

This section provides a summary of the demonstrated prototypes. Further details of the prototypes are also available in the referred publications as well as the posters of the demonstrations that are also included in the appendix of this deliverable.

2.1 Dynamic Enterprise on the CloNe Integrated Testbed

The CloNe work package set about to provide integrated on-demand allocation of computing, storage and networking across data centres and operator networks in a seamless way. More specifically, the prototyping activity has focused on the integration of wide area network services with infrastructure as a service providing customers with the ability to create distributed infrastructures at once. The objective behind the CloNe integarted testbed was to demonstrate the feasibility of this concept. The implemented testbed consists of four data centers: one in the UK(based on OpenStack), one in Sweden(also based on OpenStack), one in France(based on OpenNebula) and one in Portugal(also based on OpenNebula). The data centers are interconnected by an emulated 'operator network' running an MPLS backbone, which allowed creating FNSs with Multi-Protocol Label Switching (MPLS) Virtual Private Network (VPN)s.

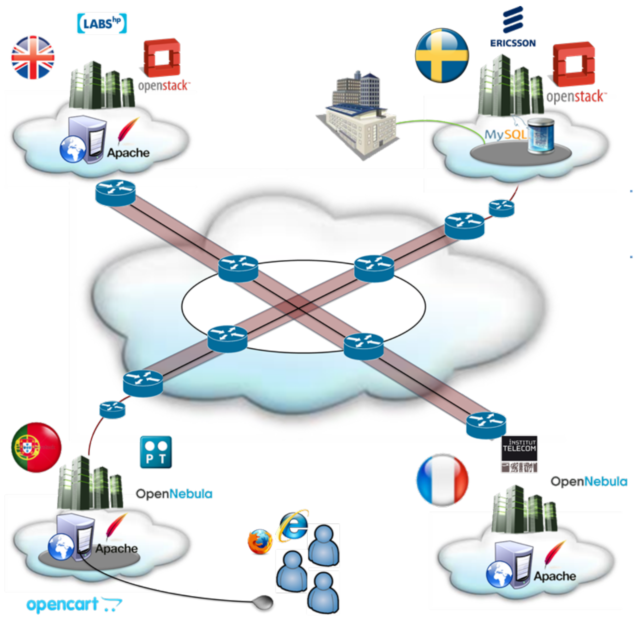
On this testbed, a customer's request for virtual infrastructure is specified with the Virtual eXecution Description Language (VXDL) description language (which may optionally be done with a Graphical User Interface (GUI)). CloudWeaver, which runs in the distributed infrastructure layer, decomposes this request into components to be provisioned by individual data centers/operator network domains. These requests are then forwarded to the data center domains as Open Cloud Compute Interface (OCCI) requests and to the operator network domains as Open Cloud Network Interface (OCNI) requests. A key protocol running in the Distributed Control Plane (DCP) of the CloNe architecture, the Link Negotiation Protocol (LNP), provides a mechanism for data center and operator network domains to negotiate parameters such as peering protocols, link bandwidth and encapsulation protocols, in order to provision the FNS. Figure 2.1b summarizes the various components and their interactions. Details of this testbed have been document in SAIL deliverable D.D.2 [3].

The functionalities implemented in the CloNe integrated testbed were demonstrated by a 'dynamic enterprise' use case scenario whereby a retailer of electronic appliances (the customer) wants to offer its products through a web shop (see Figure 2.1a). The web shop is to be integrated with the database used for its traditional shops. That database runs on the enterprise's own IT infrastructure in Sweden. The web front ends are to be provided from cloud data centers located in the targeted markets, in this case UK, Portugal and France.

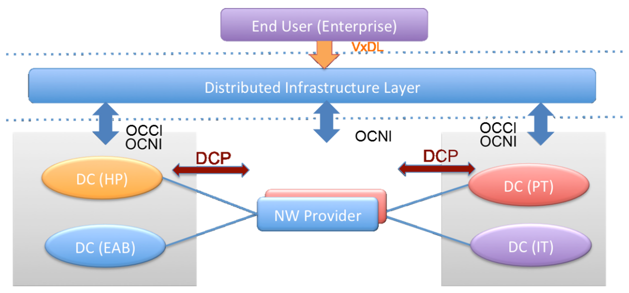
The prototype shows the feasibility of creating FNSs that span multiple administrative domains in a non-disruptive manner by building upon existing cloud and networking technologies.

2.2 Dynamic Resource Management

The service model for an Infrastructure-as-a-Service cloud includes the cloud service provider operating the physical infrastructure and the customer running its applications on the provisioned virtual infrastructure. Customers specify how their applications should be run through Service



(a) The use-case scenario and testbed



(b) Key components and their interaction

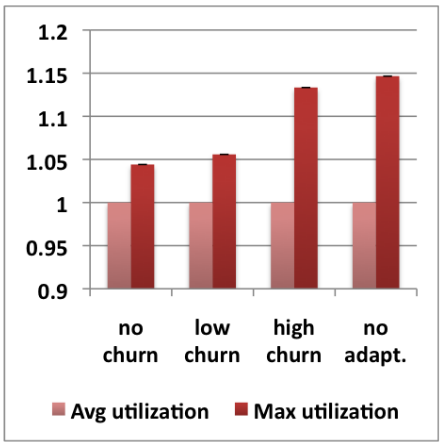
Figure 2.1: The integrated CloNe testbed

Level Agreement (SLA)s while the provider specifies how its physical resources should be allocated through a management objective. Examples of management objective include balanced load across servers and minimal energy consumption by the infrastructure. A key task of resource management is to allocate resources of the physical infrastructure such that the SLAs as well as the management objective are achieved.

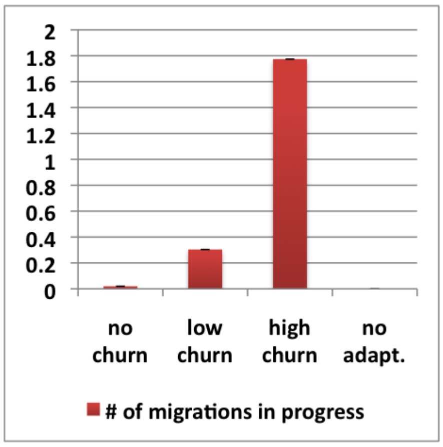
The CloNe management architecture [4] realizes resource management functionality through two management functions: a 'resource allocation' function that determines how resources are allocated to new requests for virtual infrastructure and a 'resource optimization' function that adapts an existing allocation of resources such that customer SLAs as well as the management objective are achieved at all times. The CloNe prototype includes generic realizations of these functions in the OpenStack cloud platform [5] and their instantiations for selected management objectives.

The implemented resource allocation function extends the OpenStack least-cost scheduler. The extensions include a set of filters and cost functions that instantiate the scheduler for balanced-load and minimized energy consumption objectives. The implemented resource optimization function is realized by a generic gossip-based protocol called Generic Resource Management Protocol (GRMP). Theoretical and simulation studies of the protocol have shown that GRMP can scale to well beyond 100,000 servers [6]. The prototype includes instantiations of the GRMP protocol that realize the two management objectives above.

The feasibility as well as the performance of the resource management system was evaluated on a testbed consisting of 9 servers running an average of 80 Virtual Machine (VM)s at any time. A load generator generates requests to start and stop VMs at different rates. We measured how well the management objectives are achieved and the cost of achieving those objectives. Figure 2.2 shows the results for the balanced load objective. The figure on left shows how well the load is balanced for different VM churn rates and when dynamic adaptation is disabled. The figure on right shows the associated cost. The figure shows that the load across the servers is well-balanced when there is little or no churn of VMs while the effectiveness degrades for high churn settings. In conjunction with the cost figure, this implies that optimization through dynamic adaptation is cost-effective only up to a certain rate of addition/removal of VMs in the system. The details of the evaluation result are available in [7].



(a) Effectiveness of resource allocation (normalized to avg. utilization)



(b) Cost of dynamic adaptation

Figure 2.2: Effectiveness and efficiency of dynamic resource allocation for balanced-load objective

2.3 libNetvirt: a Network Virtualization Library

libNetVirt: it is an Application Programming Interface (API) that creates a network abstraction layer to allow for the management of different network technologies in a programmable way. Currently, libNetVirt can control both OpenFlow and MPLS based networks. It uses single router abstraction to describe a network. The description includes the endpoints of the network, the constraints of specific paths as well as type of forwarding required for the underlying network. libNetVirt is composed of a common interface and a set of drivers (See Figure 2.3). Each driver implements the required configuration for a specific underlying technology. Both an OpenFlow driver and a L3 VPN driver are implemented as a proof of concept of this solution. libNetVirt offers two APIs in C and Python to operate the network. libNetVirt can be used in different ways:

1. Integrated with the network manager of a data center managed with OpenFlow. It allows the creation of different virtual networks between endpoints.
2. Directly integrated inside the network resource management system of the Network Operator to set up a MPLS network. When the network domain receives the request it calls libNetVirt with the python API.
3. libNetVirt has been integrated with pyOCNI and a specialized OCNI mixin for OpenFlow has been defined and implemented.
4. Set up the network automatically from a saved network description. libNetVirt uses an eXentisible Markup Language (XML) format to describe the network.

OpenFlow in libNetVirt permits a flexible and fast way to verify that a particular resource belongs to a specific FNS and to establish end to end path between two resources in a single domain. If multiple domains are present, OpenFlow can also be used to interconnect them. The use of OpenFlow in some parts of the SAIL test bed provided us with the following experiences:

- The use of a global view permits the use logically centralized algorithms to enforce isolation in the network.

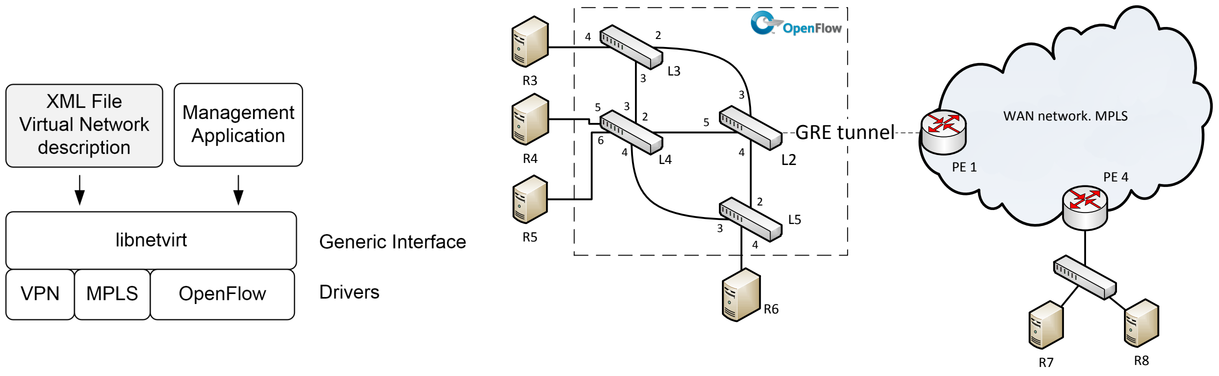


Figure 2.3: The architecture of libNetVirt (left) and the testbed it was evaluated on (right)

- Using OpenFlow for fault management permits quick reaction if a disruption in the network is created. The global view permits receiving notifications when a network element is down and reroute the affected traffic to another path.
- However, if the use of OpenFlow is reactive (the path is computed and installed when a flow arrives), a small delay is introduced in the first packet of the flow.
- OpenFlow controller can be tested in a simulated environment that approximates very well a real network environment. This permits the experimentation with the network for unexpected events without affecting the production network.

The prototype is shows the feasibly of managing different types of networks via the same API, in a similar manner to the popular libvirt API for platform virtualization.

2.4 Model-driven Resource Allocation for Elastic Video

Many applications deployed within clouds have dynamic needs in term of resources. Buzz effects, flash crowds and other types of gossip events may explain the volatility of their workload. Cloud operators may face tough challenges in conveying the proper level of QoS to their hosted applications when these applications exhibit dynamic workload profiles. They may then have crucial needs for efficient schemes to dynamically and rapidly allocate/release resources when an application needs it. Capturing these needs and translating them into action are real challenges for a Cloud Network controller.

In this demo, we demonstrate the VXDL language and CloudWeaver extensions to adequately model and enforce high elasticity within a cloud network so as to cope with the need of a hosted, very dynamic application. The purpose of this demo is twofold. First, we use our theoretical model [8, 9] to reproduce the workload dynamics of an-demand service that may be subject to buzz effects and flash-crowds types of events. This model exhibits buzz-free period (normal behavior) and buzz period (abnormal behavior) where the instantaneous workload of the e-service surges very sharply. Second, using this model as the workload generator, we use the VXDL language and CloudWeaver (Cloud Network Controller) extensions to dynamically adjust the level of provisioned resource that permits to adequately host the application. A new feature in this process stems from the use of large deviation principle that assists the decision making process for allocating or releasing resources.

The Use Case is centred on a Video on Demand (VoD) service. A server broadcasts video to a huge number of potential watchers. As the number of watchers varies, so does the workload on the VoD service (we assume no multicast transfer here).

This demo is deployed on the Grid'5000 platform. Agents acting as the VoD server or as potential viewers are deployed on geographically distant nodes. The popularity of a video is directly linked to the number of agents who have "seen" it in the past. On top of this, buzz (flash crowd) caused by an external event may occur and cause a sudden surge in the popularity of a video. In the demo, we are able to directly trigger buzz. Overall, the subsequent workload exhibit large and sometimes steep variations in its behavior.

At the VoD server, we collect measurements data and we use this information to better adjust the level of resource dedicated to this application. Of course, we consider a QoS objective but we also try to minimize the number of reconfigurations/reallocations in relation to CAPEX/OPEX. Our allocation scheme is mainly based on the large deviation properties. We rely on the virtual description language to dynamically provision and release resources to match the current need.

This prototype shows that our approach of model-driven resource allocation is effective in handling high-variability resource demands that are characteristic of video distribution systems.

2.5 In-Network Datacenter for Bulk Data Transfer

Emerging applications demand more efficient ways to handle the large amount of traffic they generate and send across sites. The current architecture solves the performance issue by caching the content at nodes on network edges. However, this is not suitable for dynamic behaviour of many services such as video streaming and bulk file transfers. Often data compression and load balancing are used to alleviate the situation. We attempted to explore an alternative way for caching data more efficiently; in particular, ones that are bulk in nature as it will significantly improve the network utilisation.

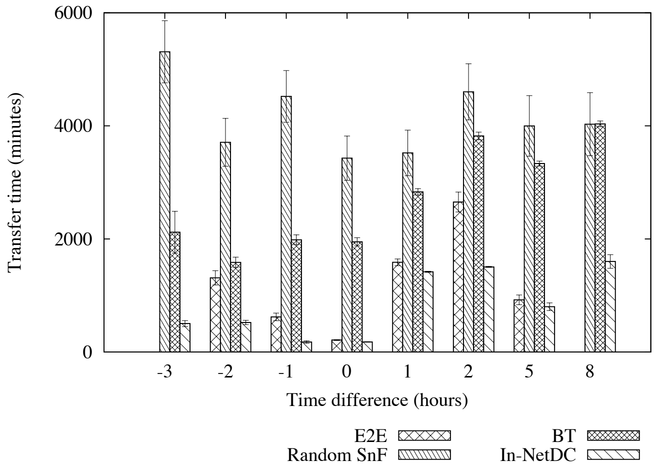
We proposed In-Network data center (In-NetDC) [10], which follows NetStitcher's store-and-forward principles [11] enabling content caching at intermediate nodes between the source and the destination where the actual consumption takes place. Note that, a simple solution such as end-to-end transfers is not possible due to typical rate-limiting policies enforced by network operators. In-NetDC differs from previous solutions as the caching nodes connect directly to the core network via high capacity links avoiding hops to the network edges where congestion level is higher than the core in typical scenarios.

To illustrate the performance of In-NetDC, a realistic topology that preserves the statistical properties of the traffic with regards to average delay, throughput and geographical constraints of a real network was used. We simulated the traffic using a simple non-stationary process with adhoc Gaussian as background traffic. The experiment was performed based on a network emulator in comparison with traditional end-to-end (E2E) transfer, BitTorrent and random store-and-forward mechanism. Furthermore, we performed real-world experiments using different chunk sizes of files transferred between data centres in the US and ones in the UK to determine their effects on transfer time.

From Figure 2.4, we can see that In-NetDC outperforms other existing solutions and the required transfer time between caching nodes in core networks significantly shorter than that of between edge and core for all chunk sizes. In addition to this performance advantage, our approach also provides on-demand deployment capabilities to create/terminate in-network resources of the required capacity at the required time through our OCCI implementation. Further details of the our approach and the evaluation results are available in [10].

2.6 An Administrative Portal for a Distributed Cloud

In order to complement the faithful vision of the CloNe service in a customer perspective, we have worked towards providing an organized and user-friendly view of distributed CloNe information



(a) Transfer time in function of time difference

Chunk size	US edge to US core (milliseconds)
1 MB	379.512 ± 21.465
10 MB	5911.873 ± 151.297
100 MB	14921.334 ± 2210.099

Chunk size	UK edge to UK core (milliseconds)
1 MB	483.12 ± 18.231
10 MB	6122.918 ± 308.35
100 MB	13421.012 ± 1126.671

Chunk size	UK core to US core (milliseconds)
1 MB	10.013 ± 2.12
10 MB	130.950 ± 1.248
100 MB	1122.129 ± 18.461

(b) Transfer times for In-Net DC

Figure 2.4: Performance of In-Net DC vs. other technologies

available in the system in an administrative perspective.

This perspective is reflected in a monitoring portal that provides a high level map of the domains that are part of the CloNe testbed and how these domains relate to each other. Moreover, important Wide Area Network (WAN) and data center information is provided along with virtual infrastructure and customer information:

- On the WAN side it is possible to see the type of services that each network operator can offer (L2 or L3), the amount of resources that can be used within that domain (e.g. number of FNSs per end-point, Service Provider Logical Link (SLL)s' available capacity), the FNSs that are deployed within that domain, etc. Depending on the SLA the portal agent has with the various administrative domains, the information available regarding a given domain can be much more detailed and could include, the status of provider edge equipments and inter-domain related information with respect to the LNP such as IP addresses used in the inter-domain Tenant Logical Link (TLL)s. (In the prototype, detailed information was available for the PTIN network domain.)

- On the data center side it is possible to see the amount of resources that can be used by the CloNe system, status of SLLs and the resources allocated within each data center.

- Virtual Infrastructure: it is possible to see the entire virtual infrastructure as well as where each individual part of it is allocated, i.e. in which domain.

- Customer: see the customer name, profile (in terms of Quality of Service (QoS) and Security) as well as which virtual infrastructure it has running on the CloNe system.

This prototype allowed us to assess the type and level of information required for a CloNe system broker to have with respect to the individual administrative domain, the virtual infrastructure, and the customer itself.

Moreover, it allowed us to understand how this information could be organized, made visible and understandable to a possible CloNe system administrator. In the end, it also became a framework that allows the demonstration of the concept, even from an administrative perspective.

2.7 Elastic NetInf on the CloNe Testbed

Virtualizing NetInf allows allocating resources where NetInf benefits most. In our demonstration, we showed a dynamic adaptation of NetInf's resources. In a typical use case, content with rising popularity results in (a) higher usage intensity at (b) different requester's locations. Today's NetInf replaces less popular content in caches by more popular content. This reduces the quality of access for such dropped content. In such a situation, increasing the cache capacity is desirable. We leverage the potential of cloud infrastructure's on-demand provisioning feature to swiftly provide NetInf with new resources at appropriate locations. This enables NetInf to handle content with rising popularity without reducing the access quality of the other content.

Remarkable points of the demonstration:

- Live presentation of interactively controlled load generators, of load measurements, of threshold decisions, of and virtual machine (VM) deployment.
- VMs are deployed on all testbed locations from all partners (EAB, HP, PTIN, IT). Load generators are operated from geographically different locations (UPB, KTH).
- Customized visualizations for the demonstration showing (1) The topological relationships and NetInf's virtual infrastructure and (2) Live plots for performance indicators and measured end-user's quality of service

Virtualized NetInf on top of CloNe realizes a deployment without the need of high upfront investments for exchanging hardware. Having cloud resource available today, our work enables a distributed deployment of NetInf in a very near future. This will shorten the time to evaluate NetInf in a worldwide deployment, which eases future migration steps. In addition CloNe offers network connectivity service and load-adaptive deployment. While the first enables NetInf to have guarantees for management communication, the latter one allocates that amount of resources at geographically distributed sites, which optimizes NetInf's performance.

CloNe's Application Deployment Toolkit (ADT) has two new facades: At first, its architecture can be customized to a wide range of application and even to NetInf [12]. Its steering concept introduces a new exchange of communication between infrastructure and application level. This enables optimization potential not utilized today. At second, the prototype shows a working, practical evaluation of the architecture across multiple layers: NetInf is running on NetInf's virtual infrastructure managed by ADT and deployed on the CloNe testbed. Both the architecture and the prototype lay the foundation of future research in topologically optimized application deployment.

3 Conclusion

The main ambition of the prototyping activity in CloNe had been to demonstrate the capabilities of an evolved Cloud Networking model. This model aims for a extensible architecture, which allows the deployment of complex applications over heterogeneous networks spread over multiple domains. The various concepts that were developed in the context of this model had been prototyped and demonstrated publicly at two separate events. We believe that these prototypes have fulfilled their goal of proving the feasibility of the model in general, and the various concepts in particular.

Beyond the public demonstration, various components of the demonstrated prototypes have been contributed to open source. This includes the OCNI specification which was part of the CloNe integrated testbed [13], the scheduler component of the dynamic resource management extension to OpenStack [14] and the libNetVirt implementation [15].

Appendix: Posters for Demonstrated Prototypes

Dynamic Enterprise on the CloNe Integrated Testbed

WpD : Interdomain Flash Network Slice Creation

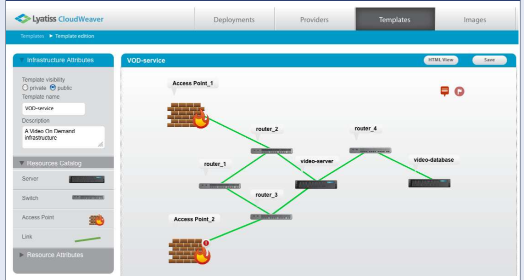
Hareesh Puthalath, Bob Melander, Vinay Yadhav, Enrique Fernández, Kalle Persson[1], João Soares, Márcio Melo[2], Houssem Medhioub, Marouen Mechtri[3], Dev Audsin, Luis M. Vaquero[4], Paul Perie[5]

Ericsson[1], PT Inovação[2], Institut Telecom[3], Hewlett-Packard[4], Lyatiss[5]

SAIL

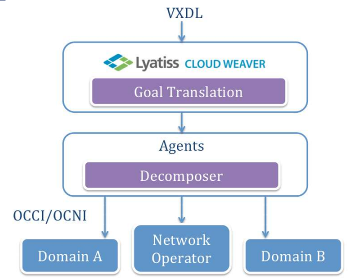
SEVENTH FRAMEWORK PROGRAMME

DEFINE



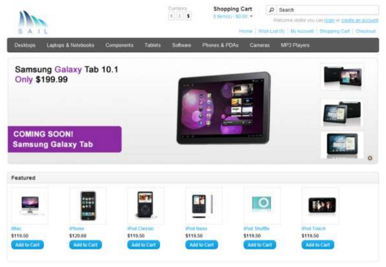
Graphical specification in VXDL language

DEPLOY



Decomposition of the Virtual Infrastructure

ACCESS



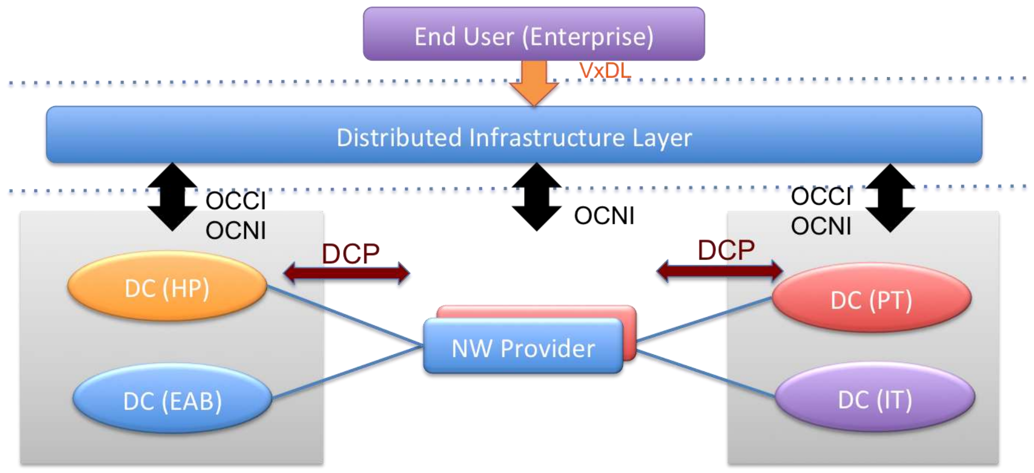
Deployed Webshop

SUMMARY

Starting an European-spanning Webshop Service in Minutes using Distributed Clouds and Flash Network Slices.

Webshop System Spec + Service Constraints/Goals ➔ Goal Translation & Constraint Resolution ➔ Virtual Infrastructure Description ➔ Virtual Resource Request using OCCI/OCNI to Networks & Clouds ➔ Distributed Control Plane to negotiate FNS protocols, properties & Instantiation ➔ Webshop service deployed!

INTERFACES



TECHNOLOGY HIGHLIGHTS

Inter-domain Interfaces:
OCNI, OCCI, DCP

Description Language:
VXDL

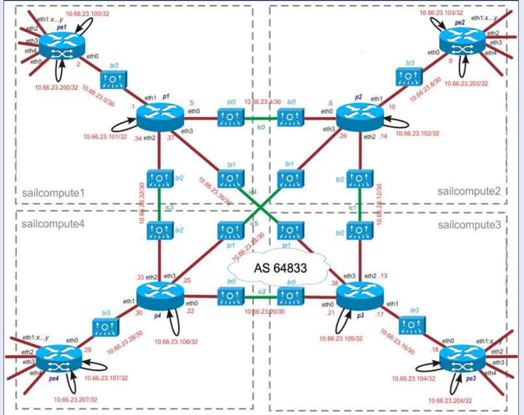
Interface Technology:
HTTP Rest, MQ

Cloud Fabric Controller:
Openstack, OpenNebula

VPN & Network Technology:
L3 MPLS, IPSec, GRE, OpenVSwitch

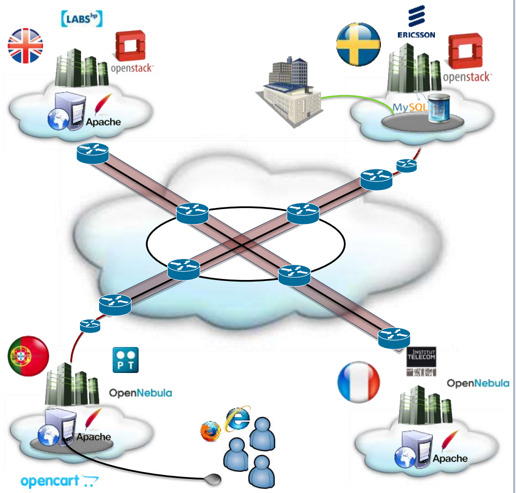
Network Virtualization:
Diverter, OpenFlow, VLAN

TEST BED



Real protocols, real implementation
No emulation or simulation

DEMONSTRATOR

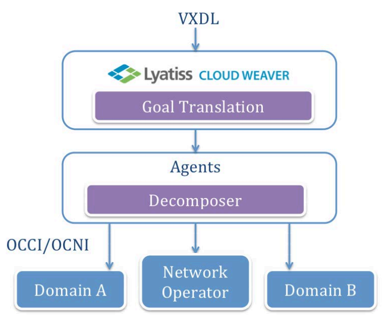


Interdomain Flash Network Slice Creation

SAIL Cloud Networking (CloNe)

User interaction model

ORCHESTRATOR

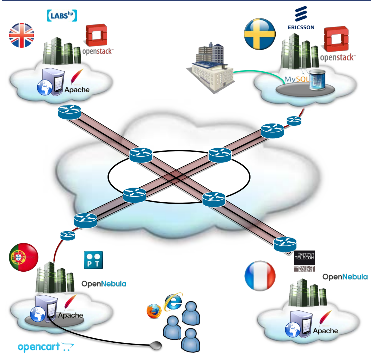


Decomposition of the Virtual Infrastructure

“A Flash Network Slice is an elastic network resource with customizable performance and isolation properties that can be allocated in a similar timescale as other basic cloud resources such as compute and storage.”

Technology

DEMONSTRATOR



Real protocols, real implementation
No emulation or simulation

Main project contributions

SCENARIO

Deploying a European-spanning Webshop Service in minutes using distributed clouds and Flash Network Slices.

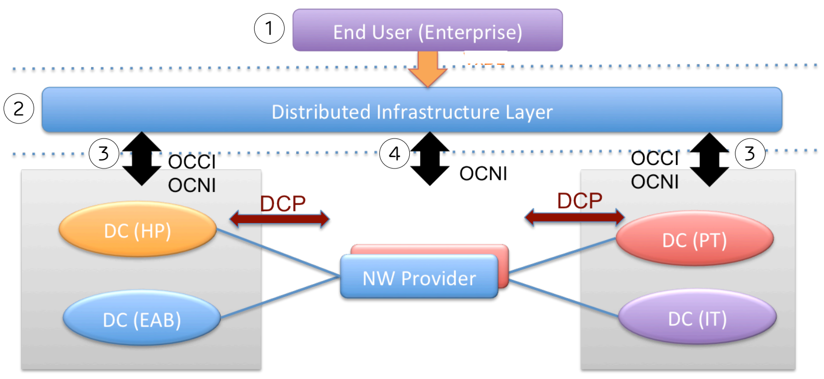
TECHNOLOGY HIGHLIGHTS

Inter-domain Interfaces:
OCNI, OCCI, DCP

Description Language:
VXDL

Interface Technology:
HTTP Rest, MQ

INTERFACES



TECHNOLOGY HIGHLIGHTS

Cloud Fabric Controller:
Openstack, OpenNebula

VPN & Network Technology:
L3 MPLS, IPSec, GRE, OpenVSwitch

Network Virtualization:
Diverter, OpenFlow, VLAN

WORKFLOW

1. User defines desired Virtual Infrastructure.
2. Goal translation and decomposition.
3. Deployment of infrastructure components.
4. Connection of the distributed infrastructure.

Hareesh Puthalath, Bob Melander, Vinay Yadhav, Enrique Fernández, Kalle Persson[1], João Soares, Márcio Melo[2], Houssem Medhioub, Marouen Mechtri[3], Dev Audsin, Luis M. Vaquero[4], Paul Perie[5]

Ericsson[1], PT Inovação[2], Institut Telecom[3], Hewlett-Packard[4], Lyatiss[5]

www.sail-project.eu

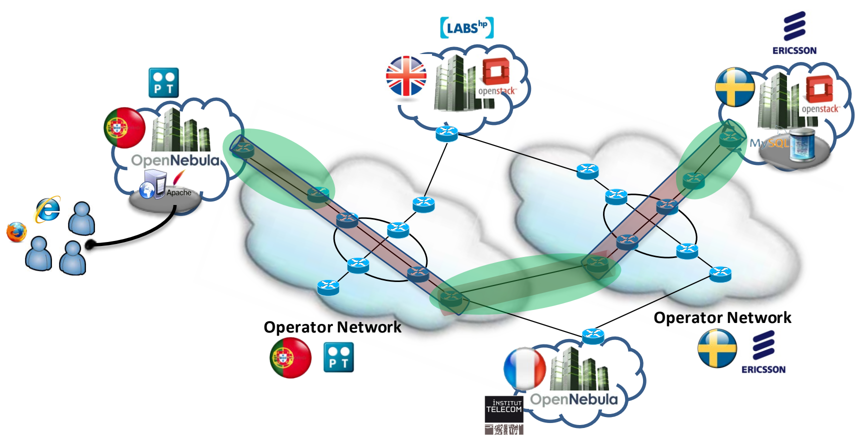
Link Negotiation Protocol

SAIL Cloud Networking (CloNe)

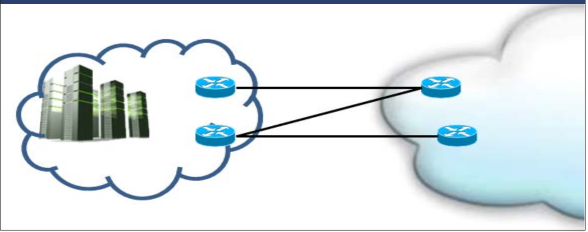
Summary

A multi-domain protocol, with support for multiple technologies, for creating virtual links belonging to a virtual infrastructure that span multiple domains.
The protocol is part of the Distributed Control Plane (DCP) of CloNe.

Use Case



Example DC-NO Setup



Highlights

Network Technologies

L3 MPLS VPN, VPLS , Openflow..

Protocols

OSPF, RIP, BGP, ...

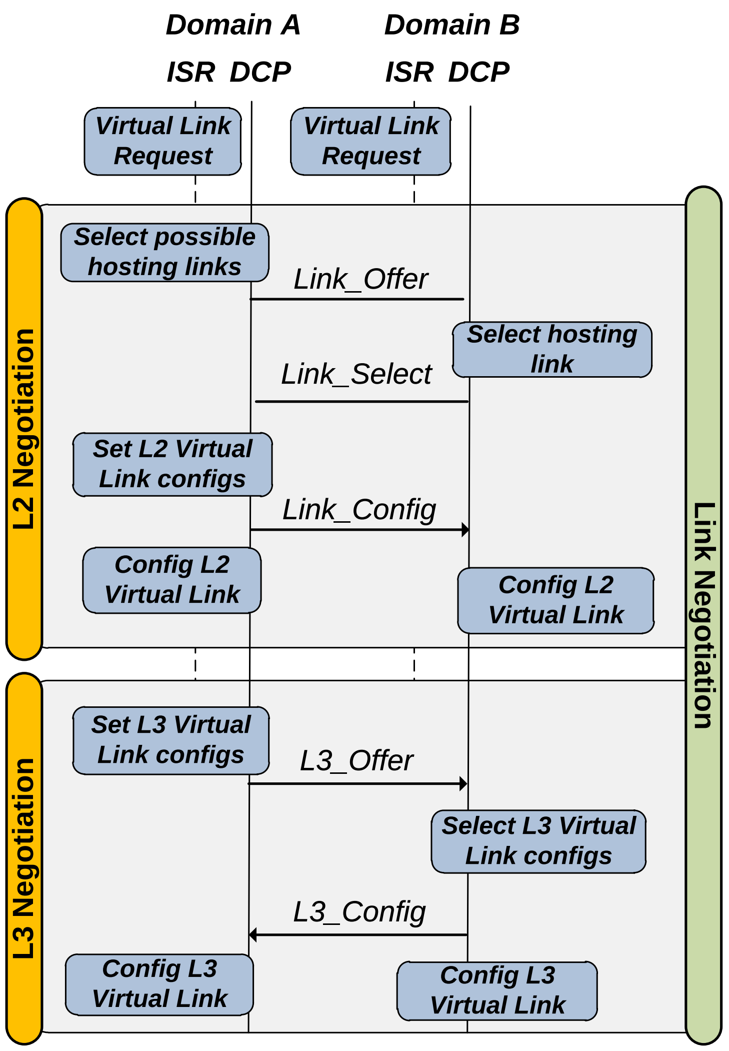
Encapsulation schemes

VLAN,GRE, IEEE 802.1ah ...

Message example

```
"infra_id": 1234,
"msg_type": "L3_CONFIG",
"sender": "dc.pt.ptinovacao",
"service_type": "L3",
"transaction_id": 161803399,
"virtual_link": {
    "TLL": {
        "L3_config": {
            "ip_dst": "10.0.0.4/30",
            "ip_src": "10.0.0.3/30",
            "routing_protocol": {
                "attributes": null,
                "type": "OSPFv2"
            },
            "routing_protocol_offer":
```

Message Exchange Diagram



Hareesh Puthalath, Bob Melander[1], João Soares, Márcio Melo[2]

Ericsson[1], PT Inovação[2]

www.sail-project.eu

Dynamic Resource Management

Dynamic Resource Management with Management Objectives

Background

The service model for an **Infrastructure-as-a-Service** cloud includes the **cloud service provider** operating the physical infrastructure and the **customer** running its applications on the provisioned virtual infrastructure. In this model, customers specify how their applications should be run through **SLAs** while the provider specifies how its physical resources should be allocated through a **management objective**. A key task of **resource management** is to allocate resources of the physical infrastructure such that the SLAs as well as the management objective are achieved.

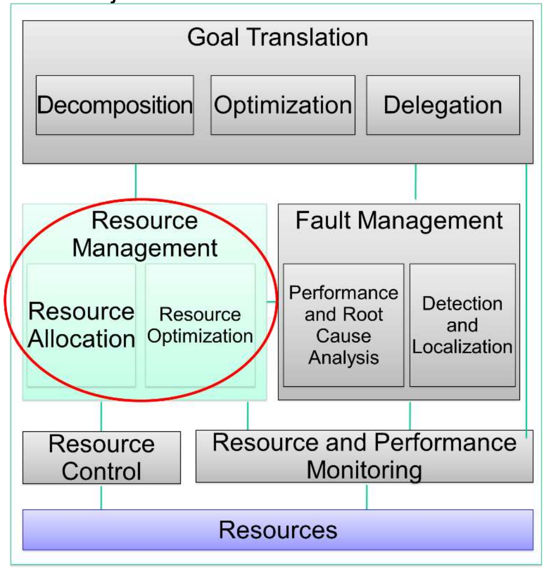
Management Objectives

The management objective of a service provider depends on factors such as its customers, their applications, its physical infrastructure and business strategy. Examples of management objective include:

- balanced load across servers
- minimal energy consumption by the infrastructure
- fair allocation of resources to customers
- service differentiation among different service classes

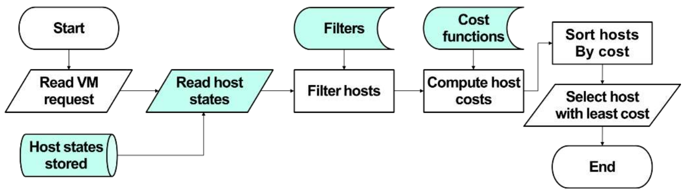
Resource Management Functions

The CloNe management architecture realizes **resource management** functionality through two management functions: a **resource allocation** function that determines how resources are allocated to new requests for virtual infrastructure and a **resource optimization** function that adapts an existing allocation of resources such that customer SLAs as well as the management objective are achieved at all times. The CloNe prototype includes generic realizations of these functions in the **OpenStack** cloud platform. It also includes instantiations for select management objectives.



Resource Allocation

The implemented resource allocation function extends the OpenStack **least-cost scheduler** (flowchart shown below). The extensions include a set of **filters** and **cost functions** that instantiate the scheduler for **balanced-load** and **minimized energy consumption** objectives.



Resource Optimization

The resource optimization function is realized by a generic **gossip-based** protocol called **GRMP** [1,3] (pseudocode shown below). Theoretical and simulation studies [2,3] show that GRMP can scale to well beyond 100,000 servers. The prototype includes instantiations of GRMP's three abstract methods that realize the two management objectives above.

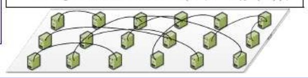
```
initialization
1: read $\omega, \gamma, \Omega, \Gamma, row_n(A)$;
2: $initInstance()$;
3: start passive and active threads;

active thread
1: for $r = 1$ to $r_{max}$ do
2:     $n' = choosePeer()$;
3:     $send(n', row_n(A))$;
4:     $row_{n'}(A) = receive(n')$;
5:     $updatePlacement(n', row_{n'}(A))$;
6:     sleep until end of round;
7: write $row_n(A)$;

passive thread
1: while $true$ do
2:     $row_{n'}(A) = receive(n')$;
3:     $send(n', row_n(A))$;
4:     $updatePlacement(n', row_{n'}(A))$;
```

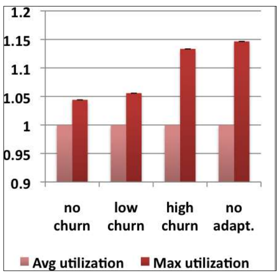
Three abstract methods:

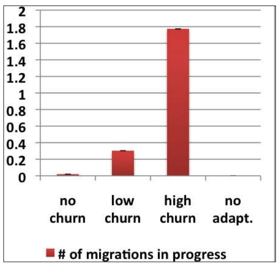
- $initInstance()$;
- $choosePeer()$;
- $updatePlacement(n', row_{n'}(A))$;



Evaluation Results

We evaluate the performance of the resource management system on a testbed consisting of 9 servers (24 cores, 64GB RAM each) running an average of 80 VMs at any given time. A load generator generates requests to start and stop VMs at different rates. For balanced load objective, the plot on top shows how well the load is balanced for different VM churn rates and when dynamic adaptation is disabled while the plot on the bottom shows the associated cost. An interesting observation is that optimization through dynamic adaptation is cost-effective only up to a certain churn rate.





[1] F. Wuhib, R. Stadler, H. Lindgren: "Dynamic Resource Allocation with Management Objectives: Implementation for an OpenStack Cloud," submitted for publication.

[2] F. Wuhib, R. Stadler, M. Spreitzer: "A Gossip Protocol for Dynamic Resource Management in Large Cloud Environments," *IEEE Transactions on Network and Service Management (TNSM)*, Vol. 9, No. 2, June 2012.

[3] R. Yanggratoke, F. Wuhib, R. Stadler: "Gossip-based Resource Allocation for Green Computing in Large Clouds," International Conference on Network and Service Management (CNSM), Paris, France, October 24-28, 2011.

Fetahi Wuhib[1], Rolf Stadler[1], Hans Lindgren[1] and Hareesh Puthalath[2]
[1]KTH Royal Institute of Technology, Stockholm, Sweden; [2]Ericsson Research, Stockholm, Sweden

libNetVirt: a Network Virtualization Library

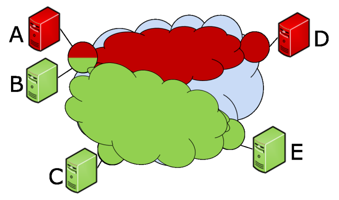
LibNetVirt: the Network Virtualization Library

Objectives

- Define and develop a control framework for defining and instantiating virtual networks (VNs)
- Common interface for multiple technologies and reusable
- Evaluate OpenFlow to provide Virtual Network Provisioning

What is libNetVirt?

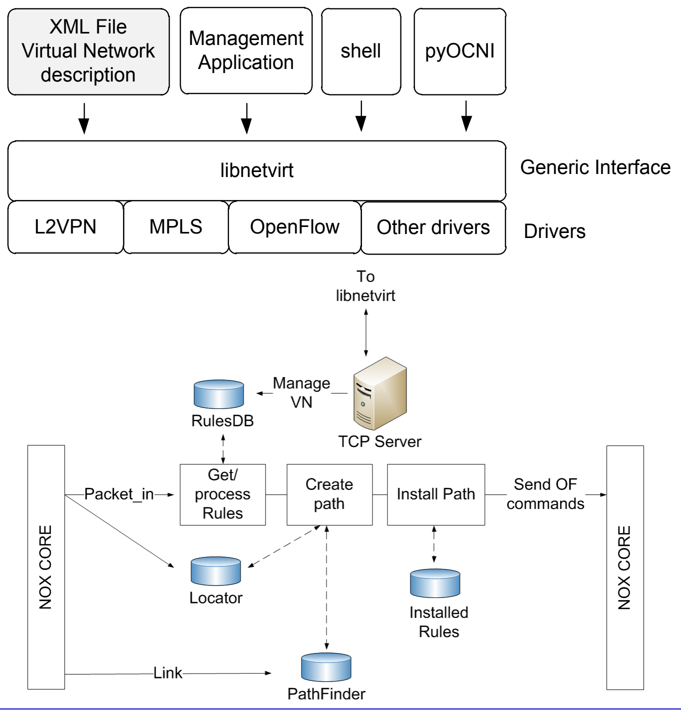
- C library with Python wrappers to operate Virtual Networks
- Single router abstraction
- Configuration in XML description
- Definition of endpoints
- Reusable and modular



Virtual Network description

```
<?xml version="1.0" encoding="UTF-8"?>
<description xmlns="http://www.sail-project.eu/fns">
  <fns name="demo" uuid="1">
    <endpoint uuid="21" >
      <swld>2</swld>
      <port>2</port>
      <vlan>10</vlan>
    </endpoint>
    <endpoint uuid="41">
      <swld>4</swld>
      <port>1</port>
      <vlan>20</vlan>
    </endpoint>
    <forwarding>L2</forwarding>
  </fns>
</description>
```

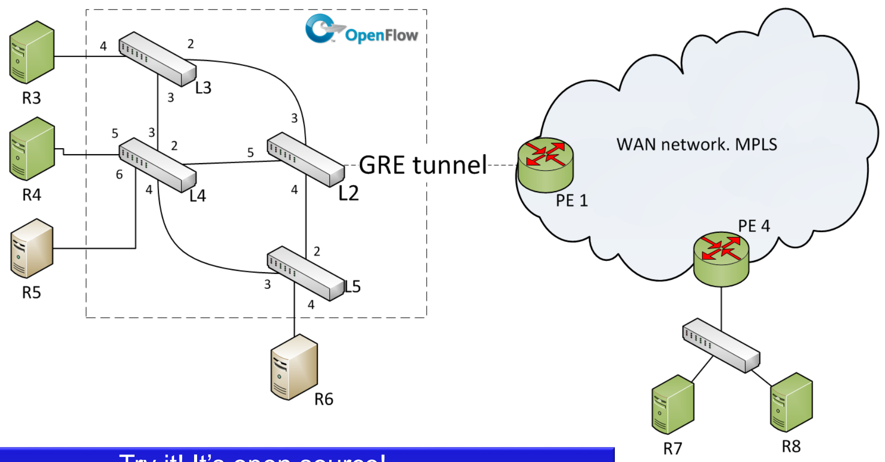
Architectural Design



Demo

- OpenFlow network with OpenVSwitch 1.4
- MPLS network with comercial network stack
- Hosts in different VLANs
- 2 management application:
  - Command Line Interface (CLI)
  - pyOCNI (from Institute Telecom)

1. Create 2 FNS (Green and Brown)
   - Creation of 2 OF network
   - Creation of a MPLS network
2. Add endpoint to a FNS
3. Remove endpoint from a FNS
4. Remove FNS



Try it! It's open source!
https://github.com/danieltt/libnetvirt

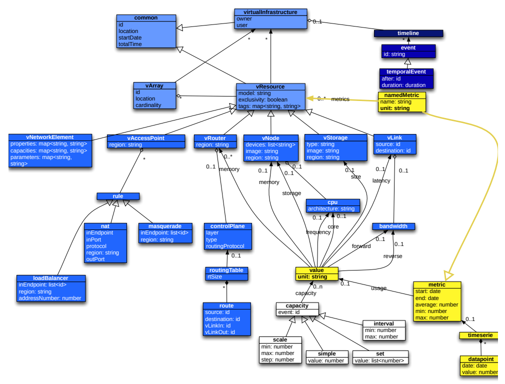
Daniel Turull, Markus Hidell, Peter Sjödin[1]
[1]KTH

Model-driven Resource Allocation for Elastic Video

Cloud Network Elasticity for VOD Services

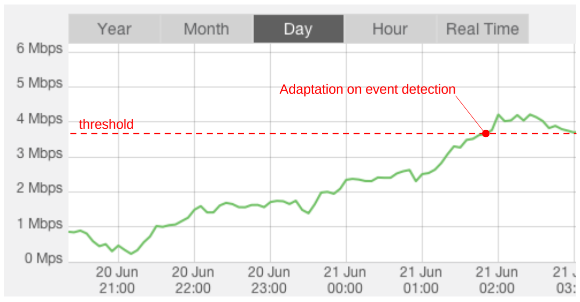
ELASTICITY ORCHESTRATION

CloudWeaver™ helps operators specify, provision and operate the Elastic Cloud Network for an assured video on demand service.



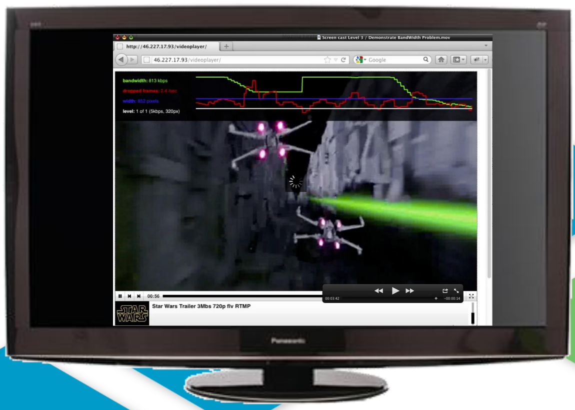
ELASTICITY MODELING

VXDL™, the Cloud modeling language, is extended with elasticity specification (goals, events, actions). The combination of events and actions represent the management goals for a given application. Events are used for identifying situations related with application's objectives (performance, cost-benefit execution, quality of experience). Actions are planned for reacting to an event.



EVENT DETECTION

Elasticity monitoring and alert management. Elasticity monitoring gives real-time information about the behavior of a virtual infrastructure's network hosting an application. Events are identified during the runtime enabling the automatic management (scaling up/down and resources reconfiguration/adaptation).



USE CASE

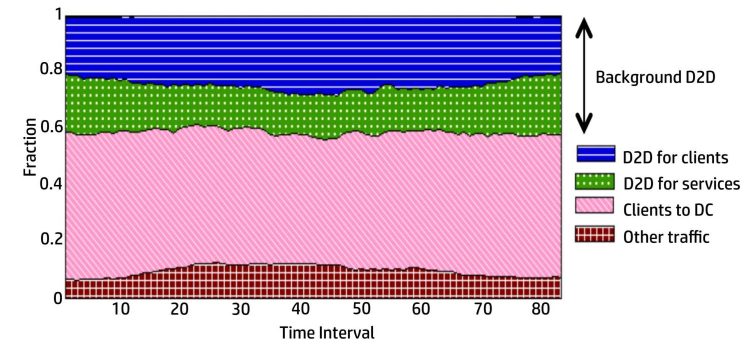
Service performance assurance for a video on demand service delivered in multi-locations.

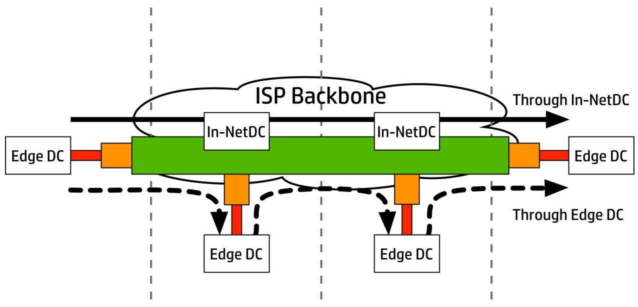
Lyatiss
www.lyatiss.com
contact@lyatiss.com

In-Network Datacenter for Bulk Data Transfer

Bulk Data Transfers Using In-Network Data Centres

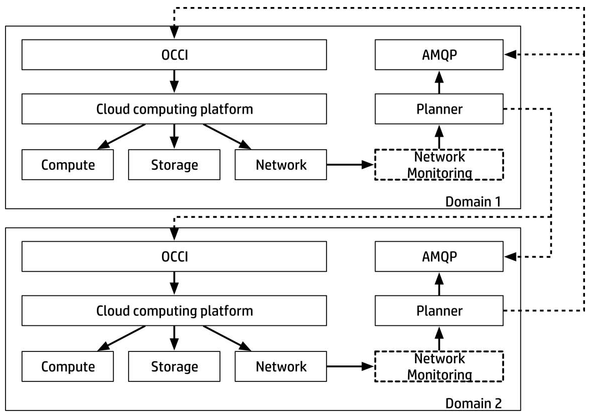
Motivation – recent data suggest that ~40% of inter-data centre traffic is background [1]. This type of traffic is usually transmitted in a store-and-forward manner making redundant hops to congested network edges.

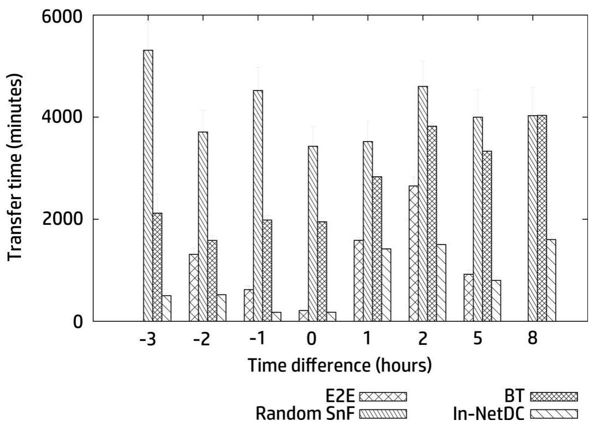


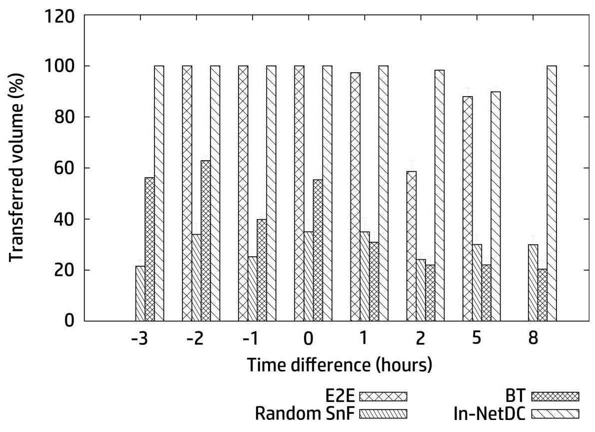


In-Network Data Centre (In-NetDC) – placing data centres in the core network enables dynamic deployment of in-network services and reduces the hops required to transfer files between data centres.

Implementation – an inferred topology of Exodus [2] was used for emulation. We used a realistic graph reduction method [3] to sample the network. In order to support the VM's deployment, we implemented an OCCI as a RESTful control interface. The CORE emulator [4] was used for network emulation and Advanced Message Queuing Protocol (AMQP) was used to aid inter In-NetDC nodes communication (e.g. acknowledgement of file chunk).







Results – In-NetDC performs better than Random Store-and-Forward (SnF), End-to-End (E2E) and BitTorrent (BT). It completes bulk file transfers timely in almost all scenarios, which is not achievable through other approaches.

[1] Y. Chen et al., "A first look at inter-data center traffic characteristics via Yahoo! datasets." IEEE INFOCOM'11.
[2] N. Spring et al., "Measuring ISP topologies with Rocketfuel." IEEE/ACM Transactions on Networking.
[3] L. Vaquero et al., "Sampling ISP Backbone Topologies." IEEE Communications Letters.
[4] J. Ahrenholz et al., "A real-time network emulator." IEEE MILCOM'08.

Luis M. Vaquero, Suksant Sae Lor, Paul Murray, Dev Audsin, Nick Wainwright
Cloud and Security Lab, Hewlett-Packard Labs

An Administrative Portal for a Distributed Cloud

CloNe Admin Perspective

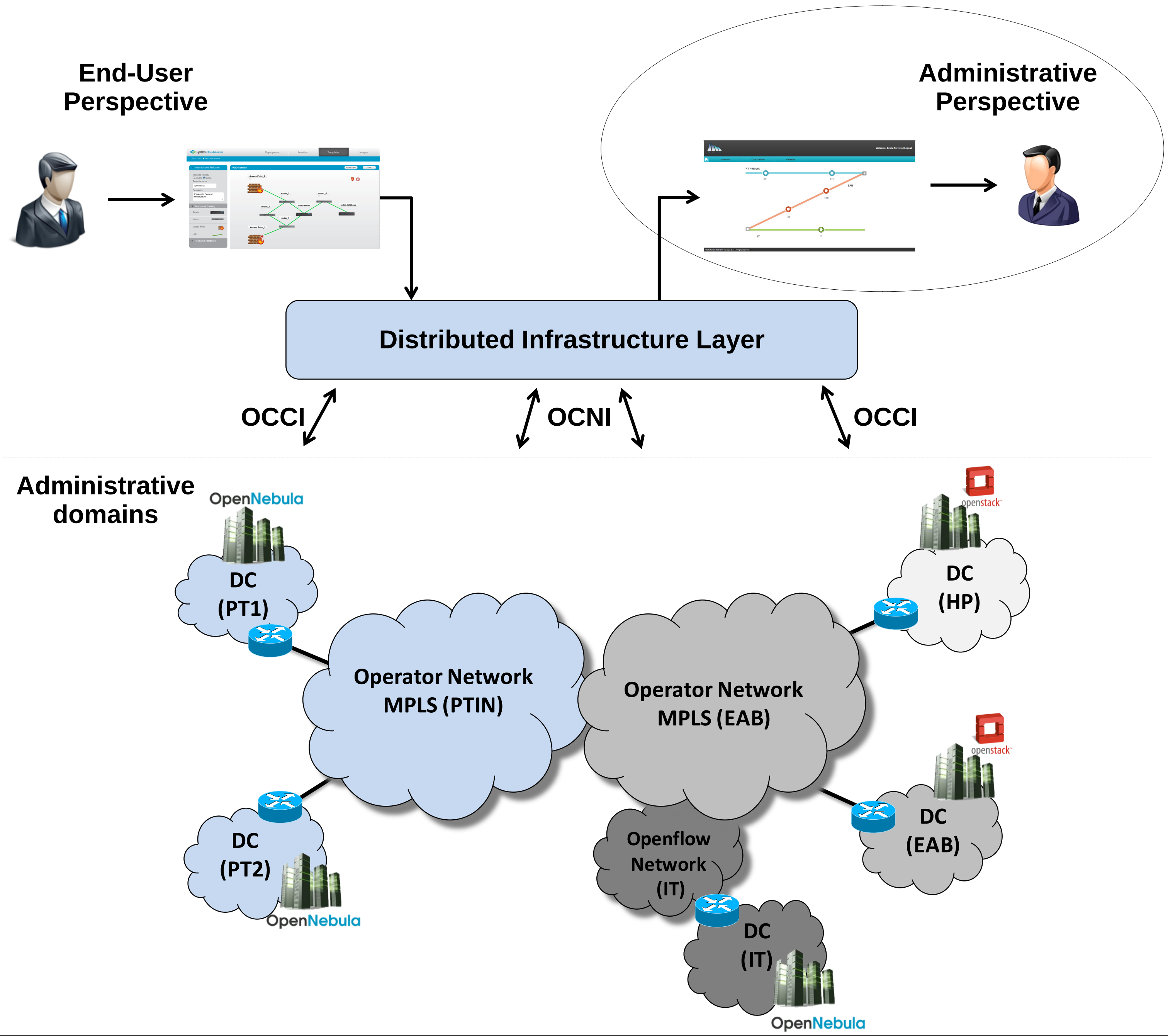
Monitoring Portal

Objective

To provide an organized and user-friendly view of distributed CloNe information available for brokers.
By gathering information from all involved domains without neglecting possible SLA restrictions.

Information

- High level Map of Domains
- Detailed Network Information
- Data Center Information
- Virtual Infrastructure Information
- Client Information



João Soares
joao-m-soares@ptinovacao.pt
Portugal Telecom Inovação

www.sail-project.eu

Elastic NetInf on the CloNe Testbed

Load-Adaptive Elastic NetInf Deployment

Fusing CloNe and NetInf

Introduction

The demonstration shows an elastic reconfiguration of NetInf's Virtual Infrastructure. A new cache will be deployed near users which are requesting content. CloNe's Application Deployment Toolkit (ADT) manages and controls the NetInf reconfiguration.

Target Scenario Wide area deployment of NetInf without the need of exchanging, expensive hardware. Use case of In-Network cloud resources

Load-Adaptive NetInf is tracking the current, node-local load situation (OPI)

Elastic New NetInf nodes are integrated into and phased out of the overall NetInf system while NetInf is working.

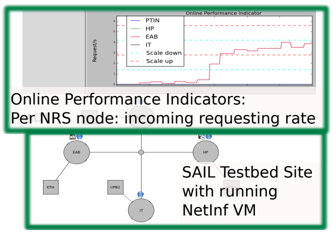
Flash Network Slice NetInf's internal communication, e.g., caching, uses allocated network resources between our testbed data centres.

The Fusion

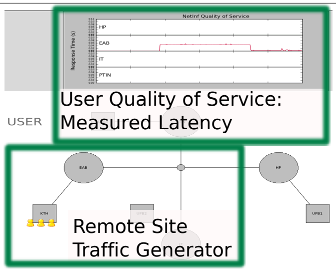
- ADT Framework plugin: A customized NetInf Decision Module (DM) (**NetInf logic in CloNe**)
- Online Performance Indicators (OPIs): Usage Reports of NetInf nodes are reported to DM
- Topology: ADT's Topology Discovery Module
- Decision Algorithm: Multi Commodity Flow Problem or Coverage Facility Location Problem
- ADT applies results and reconfigures Infrastructure and NetInf (**Elasticity**)
- Steering Daemon support NetInf to react on infrastructure events (**CloNe events for NetInf**)
- (⇒ Poster: Application Deployment Toolkit)

Visualizations

Application Deployment Toolkit


Online Performance Indicator
PTIN
HP
EAB
IT
Scale down
Scale up
Online Performance Indicators:
Per NRS node: incoming requesting rate
SAIL Testbed Site with running NetInf VM

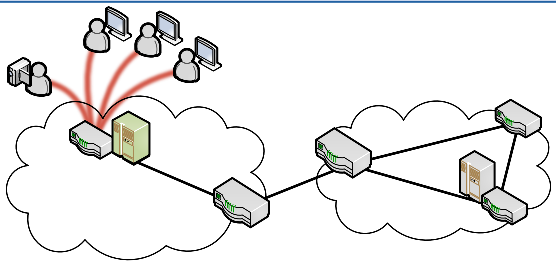
Traffic Generators

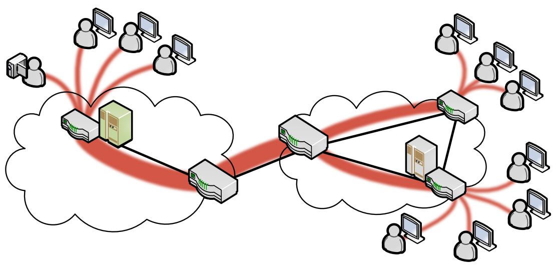


(⇒ Box: Architecture)

Scenario Storyline

1.) Alice shares a video over NetInf; it becomes popular...



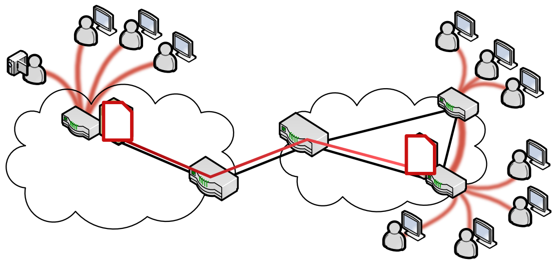


2.) ... and more popular: users downloading the content from a neighbouring network

(a) A lot of inter-provider traffic is caused

(b) **NetInf's quality** of service is reduced

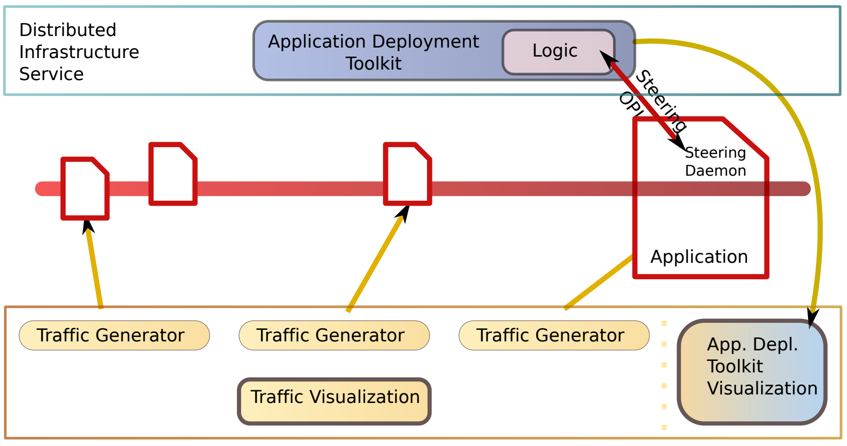
3.) A new cache is deployed close to users – decided by customize Decision Module for NetInf (⇒ Box: The Fusion)



Demo Architecture

Three systems realize the demonstration:

- **Blue**: CloNe's ADT bridging to NetInf
- **Red**: NetInf deployed in a Virtual Infrastructure
- **Yellow**: The Traffic Generators emulate accessing users and test load-adaption



University of Paderborn
Computer Networks Group
Prof. Dr. Holger Karl
`http://www.upb.de/cs/cn/`

Contact:
Matthias Keller
+49 5251 60-1754
mkeller@upb.de

SAIL
Objective:
FP7-ICT-2009-5-257448
`http://www.sail-project.eu/`

Application Deployment Toolkit

Matthias Keller, Manuel Peuster, Christoph Robbert

Introduction

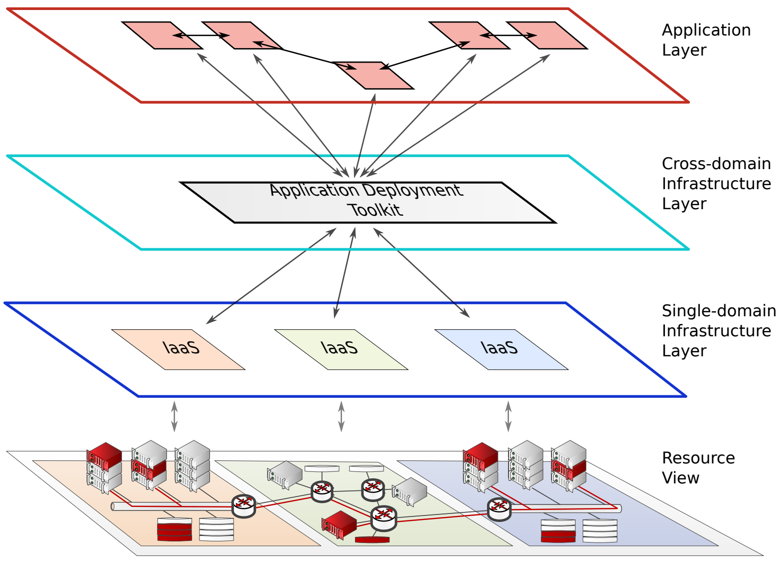
Scenario: In the near future: Many cloud infrastructure locations unveil a quality of service for large-scaled application not possible with today's deployment approaches.

Problem: How to **manage** such an large scaled, geographically distributed deployment?
How to **control** the resource adaption to improve service quality while minimizing costs?

Solution: The application deployment toolkit (ADT)...
...manages infrastructure changes and support application with notifications (**Elasticity**).
...controls and reconfigures the deployment – the **Decision Module** is used.
...is a framework for customized application needs (⇒ Box: Generic Framework)
...combines processing of infrastructure and application layer information (⇒ Item: Layer)

Layers: The right figure shows ADT on the cross domain layer utilizing the Infrastructure Provider. The red application entities are shown (a) above as components and (b) below as resources.

Layer Overview


Application Layer
Application Deployment Toolkit
Cross-domain Infrastructure Layer
IaaS
IaaS
IaaS
Single-domain Infrastructure Layer
Resource View

With ADT we enrich CloNe's capabilities by moving the application layer closer to CloNe layers.

WPD Architecture

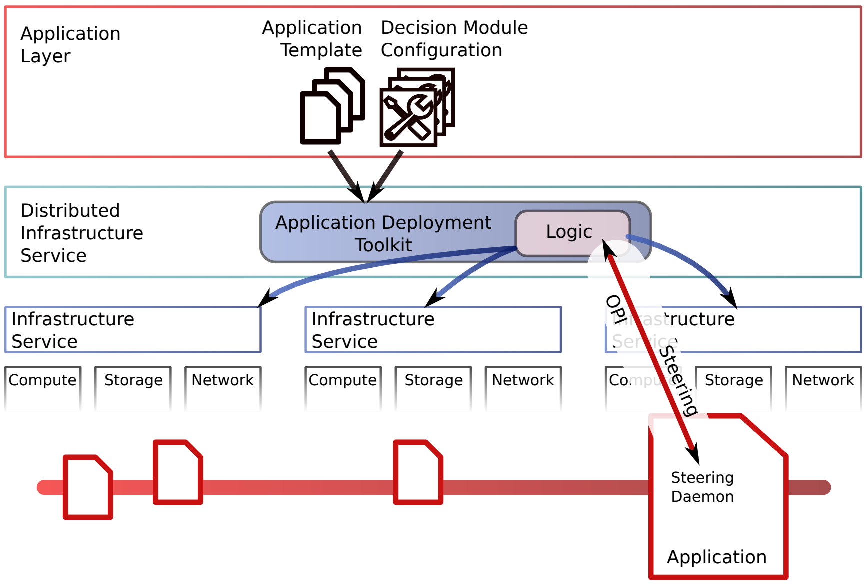


Figure: At the bottom of the architecture VMs are connected with a FNS.

Deployment: Application provider need to provide:
(a) An annotated graph of the application architecture – the **Application Template**
(b) A **Configuration** of the used algorithm (parameters, thresholds, etc.)
(c) VM images have a **Steering Daemon** preinstalled. This **bridge between Application and Infrastructure** enables elastic and adaptive deployment.

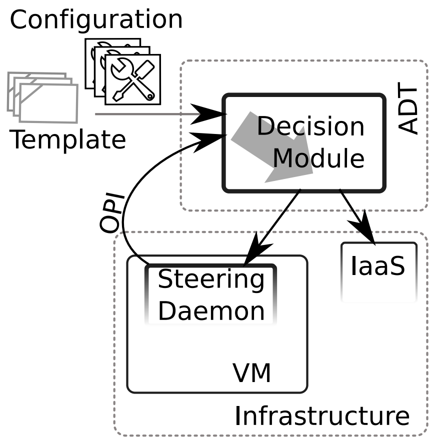
ADT: (a) realizes a more holistic management of a complex distributed applications
(b) enables plugged-in Decision Modules (DM) (⇒ Generic Framework)

Generic Framework

Generic: The framework supports ...
...plugin of customized Decision Modules (DMs)
...placeholder to communicate custom data

Customize: DM serves as a frame to plugin a wide range of algorithms and optimizations.
- The figure below shows the input data and output actions of a DM.
- ADT provides some interfaces, e.g., obtain topology informations, current application deployment, available Cloud sites, and Network services.

Placeholder: As the Decision Logic can be customized, it has to process custom input (OPI) and compute custom output.



University of Paderborn
Computer Networks Group
Prof. Dr. Holger Karl
http://www.upb.de/cs/cn/

Contact:
Matthias Keller
+49 5251 60-1754
mkeller@upb.de

SAIL
Objective:
FP7-ICT-2009-5-257448
http://www.sail-project.eu/

List of Acronyms

API	Application Programming Interface
CloNe	Cloud Networking
DCP	Distributed Control Plane
FNS	Flash Network Slice
GRMP	Generic Resource Management Protocol
GUI	Graphical User Interface
LNP	Link Negotiation Protocol
MPLS	Multi-Protocol Label Switching
NetInf	Network of Information
OCCI	Open Cloud Compute Interface
OCNI	Open Cloud Network Interface
QoS	Quality of Service
SLA	Service Level Agreement
SLL	Service Provider Logical Link
TLL	Tenant Logical Link
VM	Virtual Machine
VoD	Video on Demand
VXDL	Virtual eXecution Description Language
VPN	Virtual Private Network
WAN	Wide Area Network
XML	eXentisible Markup Language

Bibliography

[1] Future Network Mobile Summit 2012, Berlin, Germany. http://www.futurenetworksummit.eu/2012/.

[2] Future Media Distribution using Information Centric Networks 2013, Stockholm, Sweden. http://www.sail-project.eu/future-media-distribution-information-centric-networks/.

[3] Hareesh Puthalath et al. Description of Implemented Prototype. Deliverable FP7-ICT-2009-5-257448-SAIL/D.D.2, SAIL project, October 2012. Available online from http://www.sail-project.eu.

[4] Paul Murray et al. Cloud Networking Architecture Description. Deliverable FP7-ICT-2009-5-257448-SAIL/D.D.3, SAIL project, October 2012. Available online from http://www.sail-project.eu.

[5] Openstack website. http://www.openstack.org. Accessed: 31/05/2012.

[6] F. Wuhib, R. Stadler, and M. Spreitzer. A gossip protocol for dynamic resource management in large cloud environments. *Network and Service Management, IEEE Transactions on*, 2012.

[7] Fetahi Wuhib, Rolf Stadler, and Hans Lindgren. Dynamic resource allocation with management objectives — implementation for an openstack cloud. In *Network and Service Management (CNSM), 2012 8th International Conference on*, pages 309 –315, oct. 2012.

[8] Paul Murray et al. Cloud Networking Architecture Description. Deliverable FP7-ICT-2009-5-257448-SAIL/D.D.1, SAIL project, July 2011. Available online from http://www.sail-project.eu.

[9] Shubhabrata Roy, Thomas Begin, Patrick Loiseau, and Paulo Gonçalves. Un modèle de trafic adapté à la. *CoRR*, abs/1209.5158, 2012.

[10] S.S. Lor, L.M. Vaquero, and P. Murray. In-netdc: The cloud in core networks. *Communications Letters, IEEE*, 16(10):1703 –1706, october 2012.

[11] Nikolaos Laoutaris, Michael Sirivianos, Xiaoyuan Yang, and Pablo Rodriguez. Inter-datacenter bulk transfers with netstitcher. *SIGCOMM Comput. Commun. Rev.*, 41(4):74–85, August 2011.

[12] Benoit Tremblay et al. Final Harmonised SAIL Architecture. Deliverable FP7-ICT-2009-5-257448-SAIL/D.A.3, SAIL project, February 2013. Available online from http://www.sail-project.eu.

[13] pyOCNI - a Python implementation of an extended OCCI with a JSON serialization and a cloud networking extension. Online URL: http://occi-wg.org/2012/02/20/occi-pyocni/.

[14] Utilization based scheduling for openstack. https://wiki.openstack.org/wiki/UtilizationBasedSchedulingSpec. Accessed: 28/02/2013.

[15] libnetvirt repository. https://github.com/danieltt/libnetvirt. Accessed: 28/02/2013.