The Dynamic Placement of Virtual Network Functions

Stuart Clayman[1] Elisa Maini[2] Alex Galis[1] Antonio Manzalini[3]
Nicola Mazzocca[2]

[1]Dept. of Electronic Engineering, University College London, London

[2]Department of Electrical Engineering and Information Technology, University of Naples Federico II, Naples

[3]Telecom Italia Strategy - Future Centre, Turin

May 2014

Overview

- Introduction
- Motivation
- Architecture
- Validation
- Results
- Conclusions

Introduction

- Address the problem of managing highly dynamic network and service environments, where virtual nodes and virtual links are created and destroyed depending on traffic volumes, service requests, or high-level goals such as reduction in energy consumption.

Introduction

- Address the problem of managing highly dynamic network and service environments, where virtual nodes and virtual links are created and destroyed depending on traffic volumes, service requests, or high-level goals such as reduction in energy consumption.
- Emerging paradigms such as Software Defined Networks (SDN) and Network Function Virtualization (NfV) are concrete steps towards infrastructures where network functions and services will be executed as applications in ensembles of virtual machines hosted in pervasive standard hardware resources located across a network.

Introduction

- Address the problem of managing highly dynamic network and service environments, where virtual nodes and virtual links are created and destroyed depending on traffic volumes, service requests, or high-level goals such as reduction in energy consumption.
- Emerging paradigms such as Software Defined Networks (SDN) and Network Function Virtualization (NfV) are concrete steps towards infrastructures where network functions and services will be executed as applications in ensembles of virtual machines hosted in pervasive standard hardware resources located across a network.
- The focus here is the dynamic management and orchestration of virtual networks (VNs), with particular attention paid to placement of these virtual resources. The physical resources will be shared by multiple virtual networks which are independent of each other and are allocated to different customers of the network.

Introduction

- In order to manage these virtual elements and infrastructures there is a need to introduce high-level systems orchestration.

Introduction

- In order to manage these virtual elements and infrastructures there is a need to introduce high-level systems orchestration.
- Need an architecture for an orchestrater that ensures the automatic placement of the virtual nodes and the allocation of network services on them, supported by a monitoring system that collects and reports on the behaviour of the resources.

Introduction

- In order to manage these virtual elements and infrastructures there is a need to introduce high-level systems orchestration.
- Need an architecture for an orchestrater that ensures the automatic placement of the virtual nodes and the allocation of network services on them, supported by a monitoring system that collects and reports on the behaviour of the resources.
- The orchestrater manages the creation and removal of the virtual nodes, as well as configuring, monitoring, running and stopping software on them.

Introduction

- In order to manage these virtual elements and infrastructures there is a need to introduce high-level systems orchestration.
- Need an architecture for an orchestrater that ensures the automatic placement of the virtual nodes and the allocation of network services on them, supported by a monitoring system that collects and reports on the behaviour of the resources.
- The orchestrater manages the creation and removal of the virtual nodes, as well as configuring, monitoring, running and stopping software on them.
- As a proof of these concepts, a distributed orchestrater prototype has been designed, implemented and tested with the results of different placement algorithms presented.

Evolutionary Edge Networks

- Service Providers and Telecommunication Service Providers will utilize Software Defined Networks and Network Function Virtualisation to improve the quality of the applications for users, to increase business opportunities for both.

Evolutionary Edge Networks

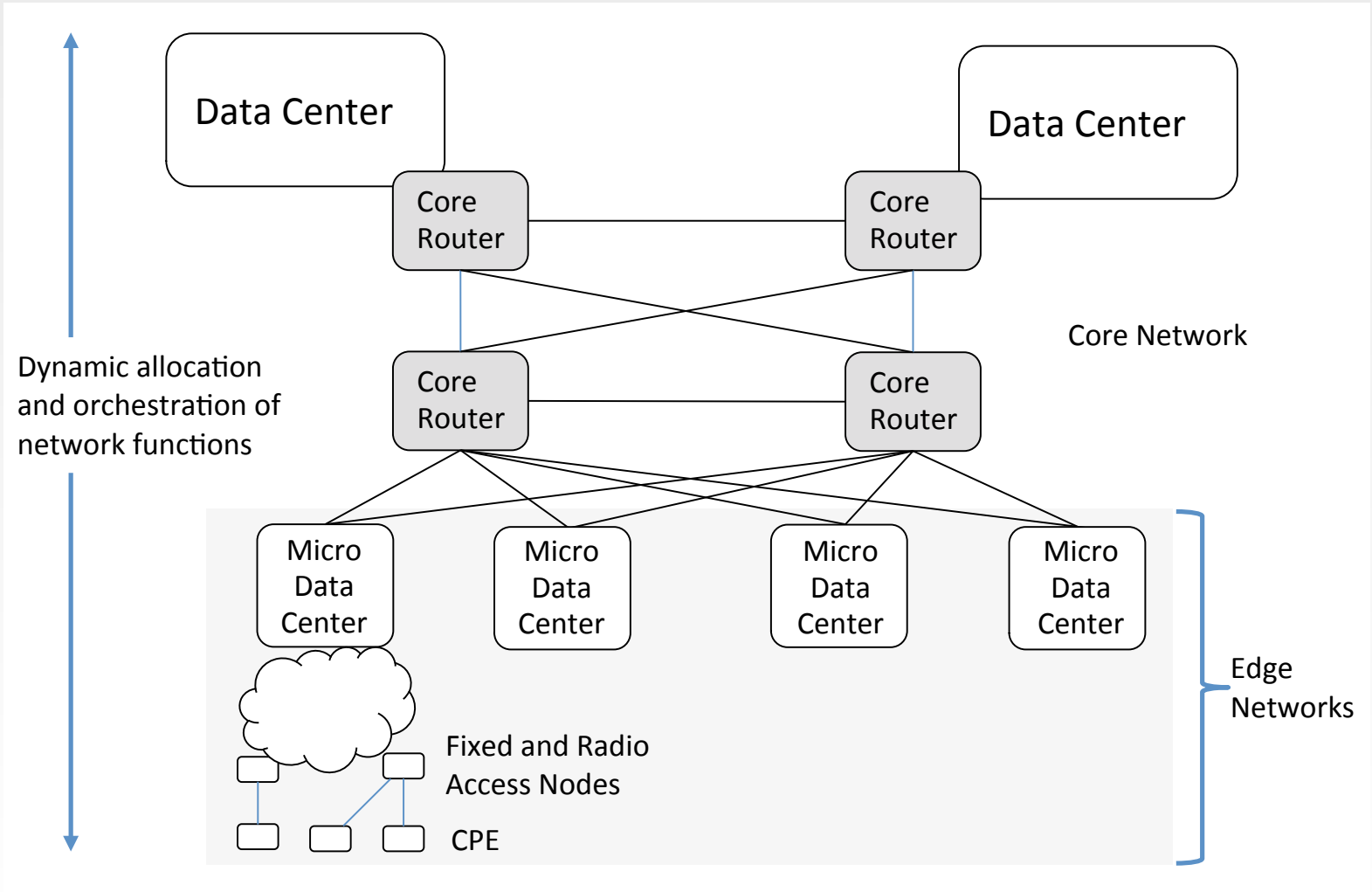
- Service Providers and Telecommunication Service Providers will utilize Software Defined Networks and Network Function Virtualisation to improve the quality of the applications for users, to increase business opportunities for both.
- The edges of the networks will be transformed into complex Micro Data Centers consisting of a number of diverse and autonomous, but inter-related nodes, devices, machines: this will create a "complexity" which has to be managed and controlled.

Evolutionary Edge Networks

- Service Providers and Telecommunication Service Providers will utilize Software Defined Networks and Network Function Virtualisation to improve the quality of the applications for users, to increase business opportunities for both.
- The edges of the networks will be transformed into complex Micro Data Centers consisting of a number of diverse and autonomous, but inter-related nodes, devices, machines: this will create a "complexity" which has to be managed and controlled.
- Each deployed virtual network is viewed as a *managed network service* by the management software. All managed network services are mapped to the available resources which allows users of the network service to access it, just like they would access a physical network service.

Evolutionary Edge Networks

Evolutionary view of the network.



Architecture

- From an architectural viewpoint, network functions and services can be defined as a number of software components with their accompanying context, together with configuration parameters.

Architecture

- From an architectural viewpoint, network functions and services can be defined as a number of software components with their accompanying context, together with configuration parameters.
- In general, the provisioning of a service involves the creation of an infrastructure, followed by the installation of all necessary software components into the infrastructure, and finally to configure and start those components.

Architecture

- From an architectural viewpoint, network functions and services can be defined as a number of software components with their accompanying context, together with configuration parameters.
- In general, the provisioning of a service involves the creation of an infrastructure, followed by the installation of all necessary software components into the infrastructure, and finally to configure and start those components.
- With SDN and NfV these processes can be simplified as the infrastructure provides a platform from which virtual machines can be run. SDNs can be directly manifested as virtual network topologies which need to be setup, have a managed lifecycle, and need to be shutdown - *all under software control*.

Architecture

The architecture for the Management and Orchestration has 4 main layers:

Architecture

The architecture for the Management and Orchestration has 4 main layers:

- an *Application Layer* which executes Management Applications that define the software components and network functions of a network service together with their configuration parameters.

Architecture

The architecture for the Management and Orchestration has 4 main layers:

- an *Application Layer* which executes Management Applications that define the software components and network functions of a network service together with their configuration parameters.
- an *Orchestration Layer* which does most of the management and orchestration and is in charge of managing the full lifecycle of the virtual routers in the network and the allocation of the applications running on the virtual nodes.

Architecture

The architecture for the Management and Orchestration has 4 main layers:

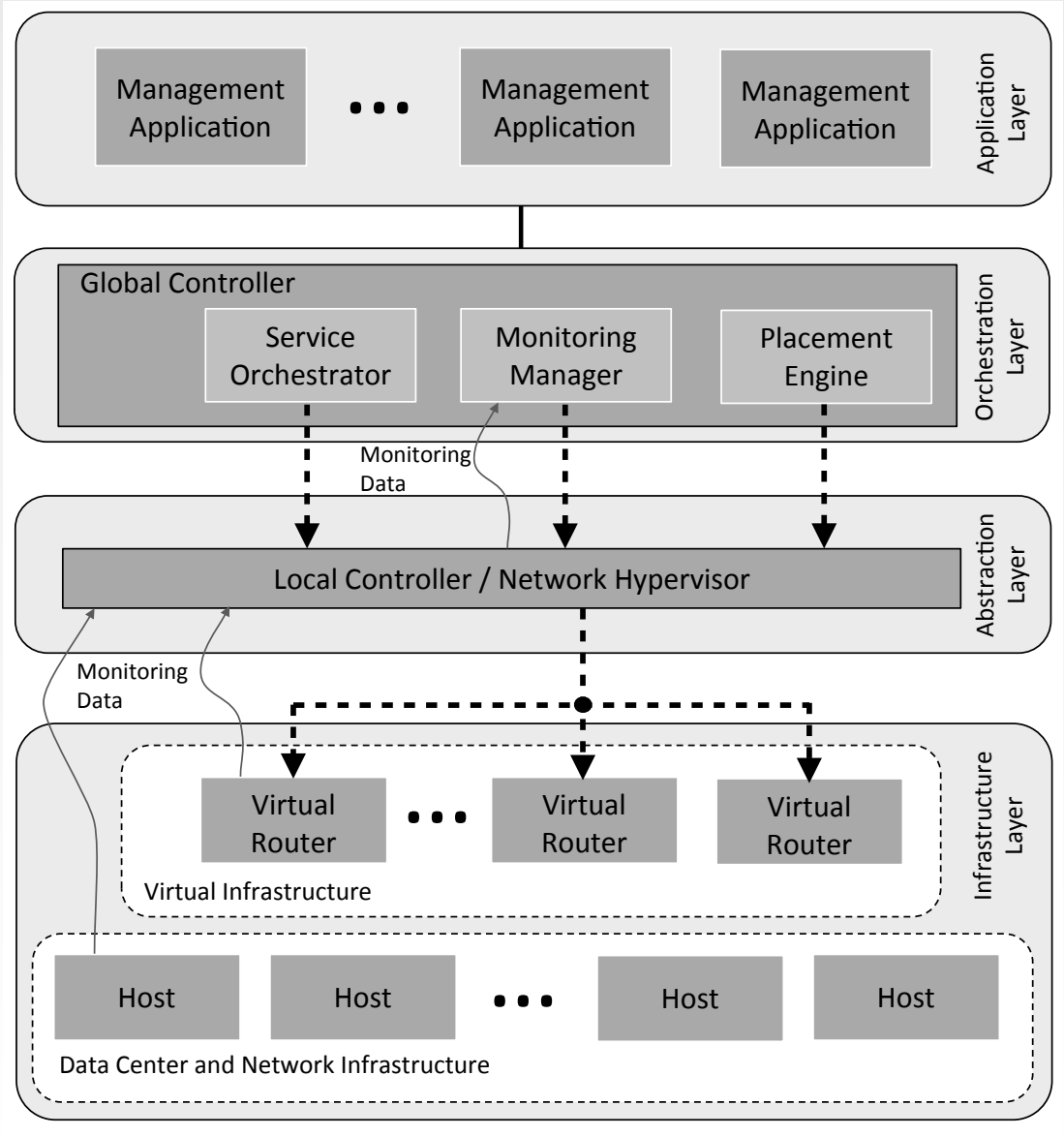
- an *Application Layer* which executes Management Applications that define the software components and network functions of a network service together with their configuration parameters.
- an *Orchestration Layer* which does most of the management and orchestration and is in charge of managing the full lifecycle of the virtual routers in the network and the allocation of the applications running on the virtual nodes.
- an *Abstraction Layer* contains a software element which presents an abstraction for starting, stopping, and configurating virtual elements.

Architecture

The architecture for the Management and Orchestration has 4 main layers:

- an *Application Layer* which executes Management Applications that define the software components and network functions of a network service together with their configuration parameters.
- an *Orchestration Layer* which does most of the management and orchestration and is in charge of managing the full lifecycle of the virtual routers in the network and the allocation of the applications running on the virtual nodes.
- an *Abstraction Layer* contains a software element which presents an abstraction for starting, stopping, and configurating virtual elements.
- the *Infrastructure Layer* contains both the *Virtual Infrastructure* which represents the virtual resources (i.e. the virtual routers) that make up the virtual networks, and the *Data Center Infrastructure* which are the physical resources that are the hosts running the VMs.

Overall System Architecture and Components



Placement Engine

The Service Orchestrator is the component in charge of performing the automatic deployment of the function/service as application running on the virtual routers.

Placement Engine

The Service Orchestrator is the component in charge of performing the automatic deployment of the function/service as application running on the virtual routers.

- The *Placement Engine* is the component in charge of performing the actual placement of the virtual routers according to the initial topology and the usage of the virtual network elements.

Placement Engine

The Service Orchestrator is the component in charge of performing the automatic deployment of the function/service as application running on the virtual routers.

- The *Placement Engine* is the component in charge of performing the actual placement of the virtual routers according to the initial topology and the usage of the virtual network elements.
- It is an important feature because when we configure a network, some of these parameters may change during the course of the system's operation and a reconfiguration may be required to maintain optimized collection of information.

Placement Engine

The Service Orchestrator is the component in charge of performing the automatic deployment of the function/service as application running on the virtual routers.

- The *Placement Engine* is the component in charge of performing the actual placement of the virtual routers according to the initial topology and the usage of the virtual network elements.
- It is an important feature because when we configure a network, some of these parameters may change during the course of the system's operation and a reconfiguration may be required to maintain optimized collection of information.
- The decision on the Placement Engine is encoded in an algorithm which can be rather simple, such as counting the number of virtual routers on a host, or it can be based on a set of constraints and policies that represent the network properties.

Monitoring Manager

- The *Monitoring Manager* is an important component of the architecture and its monitoring function is a vital part of a *full control loop* that goes from the Orchestrator, through a control path, to monitoring probes which collect and send data, back to the Orchestrator which makes decisions based on the data.
- Each virtual router has a probe to monitor the usage of the network resources (e.g. the state congestion of the links). The data provided by the probes is collected by the Monitoring Manager and used by the Orchestrator to create or remove the virtual routers according to the current state of the network.

Validation

- A working implementation of the architecture has been created. The experiments used the Very Lightweight Service Platform (VLSP) which has been implemented by UCL for the purpose of testing and evaluating various aspects of SDN and highly dynamic virtual environments.
- The VLSP testbed consists of a large number of software routers, each running inside JVMs. These virtual routers execute on a smaller number of physical machines. The virtual routers are logically independent software entities which cannot communicate with each other except via network interfaces.
- The testbed has been validated in previous work we have undertaken on virtual networks and highly dynamic networks

Validation

The testbed set up has three components:

i) the main component is the *Router* itself, which runs inside a JVM;

ii) a per-host controller, and

iii) a supervisor and experimental controller.

Validation

The testbed set up has three components:

i) the main component is the *Router* itself, which runs inside a JVM;

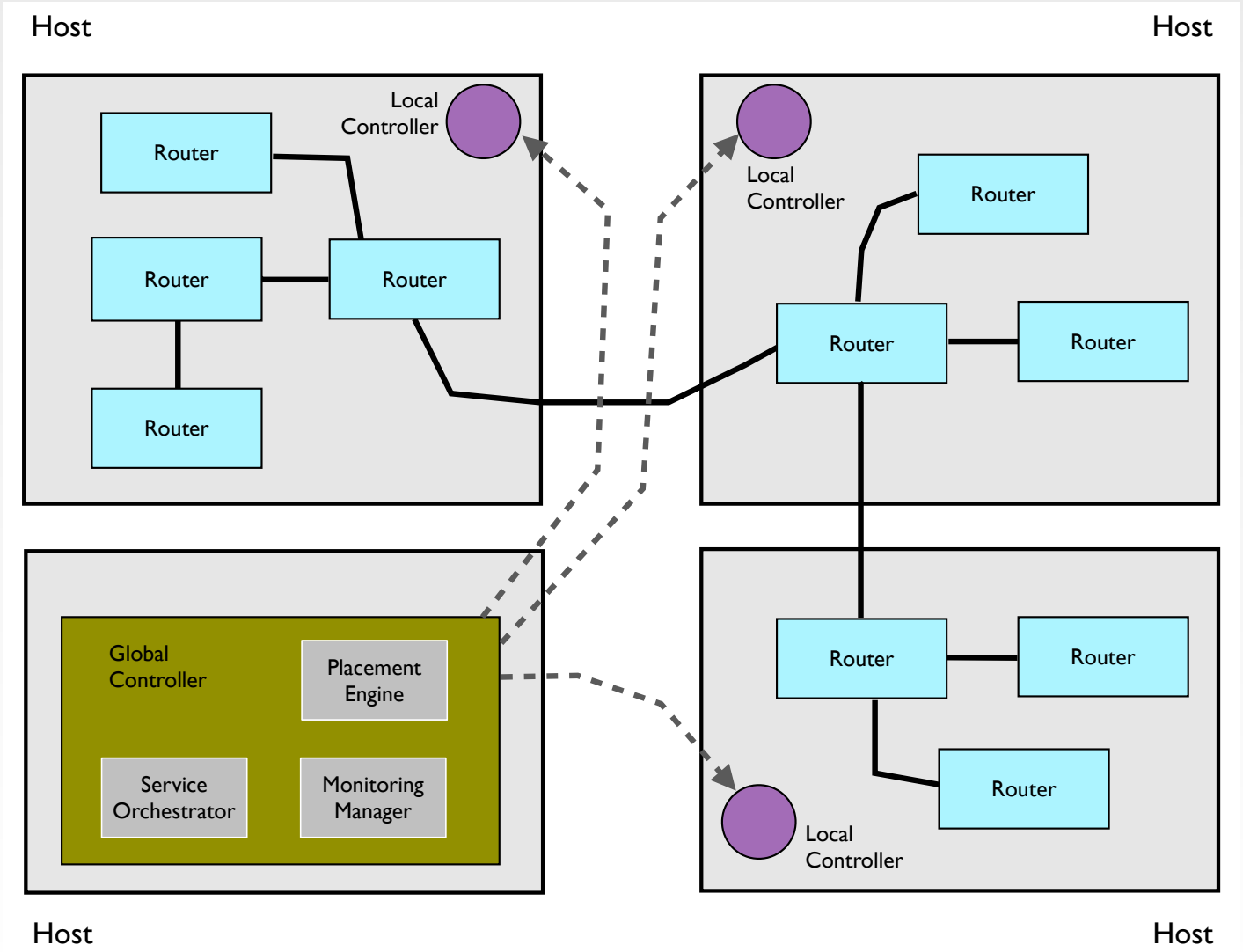
ii) a per-host controller, and

iii) a supervisor and experimental controller.

The routers are complemented by a lightweight per-host controller called the *Local Controller* which has the role of sending instructions to start up or shut down routers on the local machine and to inform routers to initiate or tear down connections with other routers.

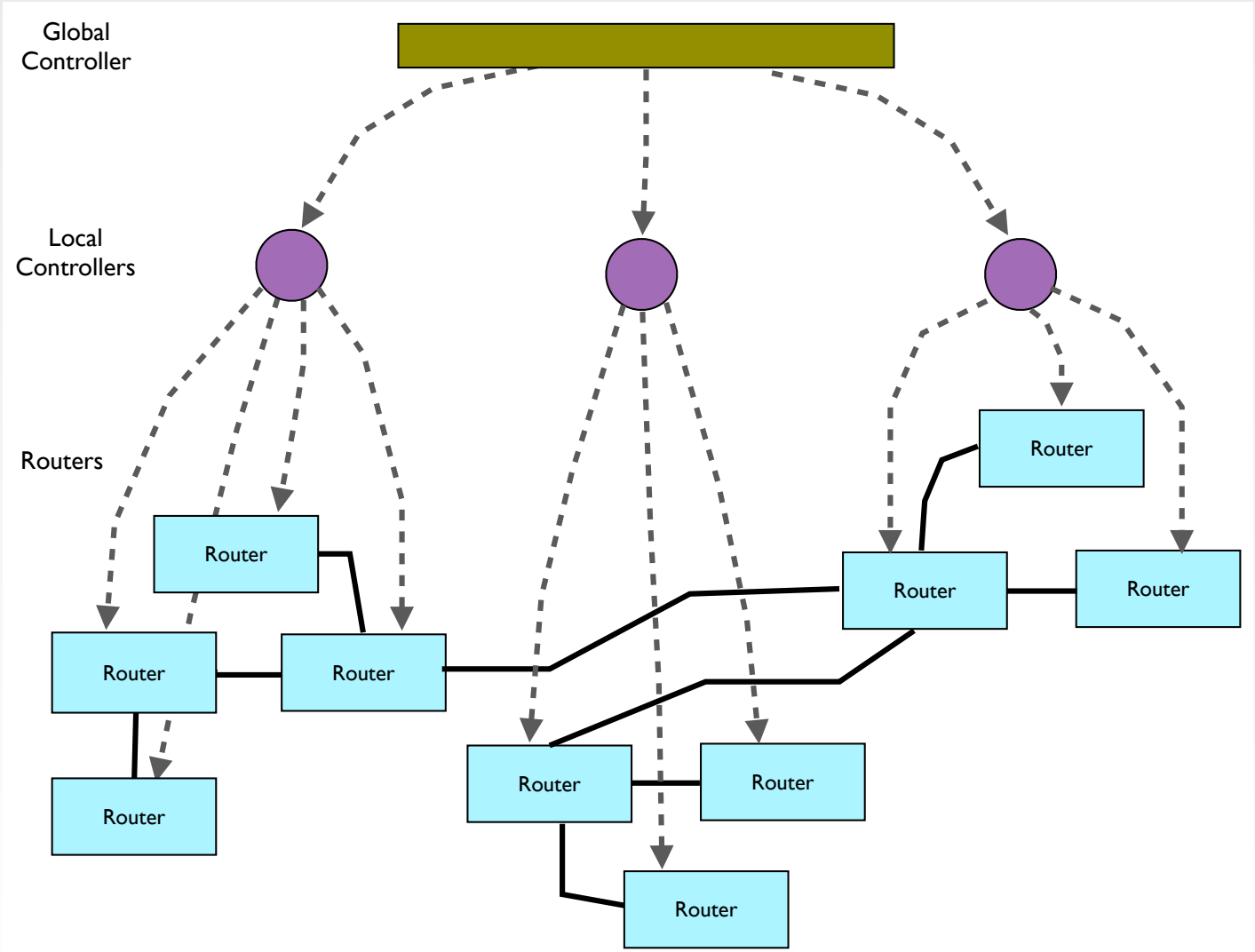
Validation

Mapping and allocation of elements to hosts



Validation

Control path to virtual routers and virtual links



Results

- A Placement Engine is the component that encapsulates an algorithm for choosing the *best* destination to place a new virtual router.

Results

- A Placement Engine is the component that encapsulates an algorithm for choosing the *best* destination to place a new virtual router.
- Placement Engine algorithms can be based on infrastructure metrics or on virtual network metrics. It is possible to write many different Placement Engines which rely on different metrics and algorithms.

Results

- A Placement Engine is the component that encapsulates an algorithm for choosing the *best* destination to place a new virtual router.
- Placement Engine algorithms can be based on infrastructure metrics or on virtual network metrics. It is possible to write many different Placement Engines which rely on different metrics and algorithms.
- The Placement Engine is a configurable module that can be changed as needed according to different placement strategies.

Results

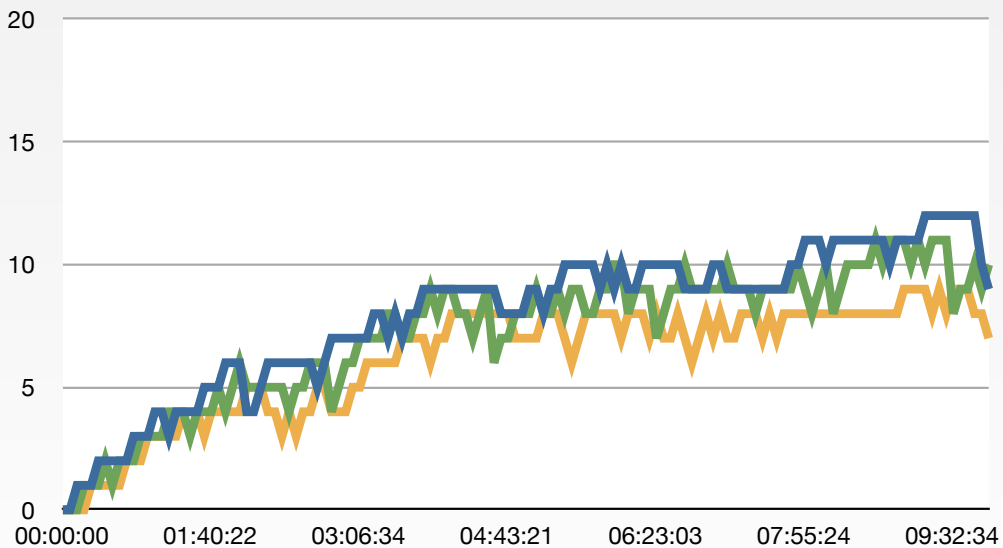
- A Placement Engine is the component that encapsulates an algorithm for choosing the *best* destination to place a new virtual router.
- Placement Engine algorithms can be based on infrastructure metrics or on virtual network metrics. It is possible to write many different Placement Engines which rely on different metrics and algorithms.
- The Placement Engine is a configurable module that can be changed as needed according to different placement strategies.
- The choice as to which Placement Engine to run is dependent on the high-level goals and policies that are set for the whole of the networked system.

Results

- A Placement Engine is the component that encapsulates an algorithm for choosing the *best* destination to place a new virtual router.
- Placement Engine algorithms can be based on infrastructure metrics or on virtual network metrics. It is possible to write many different Placement Engines which rely on different metrics and algorithms.
- The Placement Engine is a configurable module that can be changed as needed according to different placement strategies.
- The choice as to which Placement Engine to run is dependent on the high-level goals and policies that are set for the whole of the networked system.
- There may be a high-level goal which is "reduce energy consumption". To satisfy such a goal, the placement engine needs to have an algorithm which places virtual routers ensuring that the least number of physical resources are used. Conversely, a goal could be "balance the load across all physical nodes".

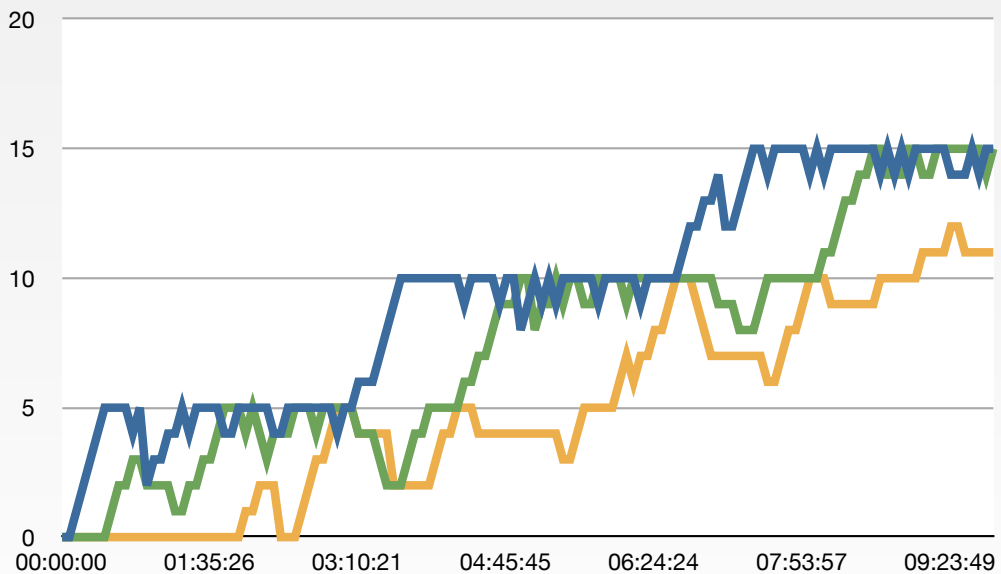
Placement Engine: Least Used

This collects the number of virtual routers allocated to each physical host and chooses the host that has the least number of virtual routers. If more than one host is at the minimum level, then a random host is chosen. This algorithm is a kind of load balancing algorithm as it tries to get a similar number of routers on each host.



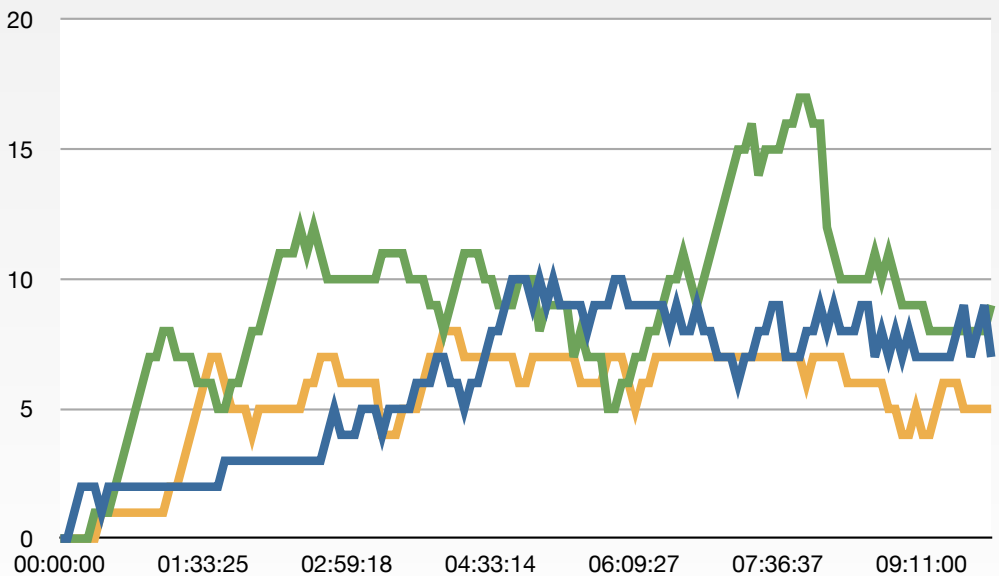
Placement Engine: N at a time: N = 5

Allocates N routers at a time into a single host. In this run, N = 5. If the number of routers is a factor of N, then the algorithm chooses are different host. When all the hosts are packed with a factor of N, then a random host is chosen.



Placement Engine: Least Busy

It tries to determine the host that is *least busy* in terms of virtual network traffic. It collects monitoring data from all the the virtual routers in the virtual network and calculates how much virtual traffic has been sent on each of the hosts. The host that has the lowest amount of traffic since the last placement decision is chosen as the host for the current placement.



Conclusions

- As a proof of concept, architectural elements have been designed and implemented and some experimental results using the VLSP testbed presented. We have shown how Placement Engines can start different virtual routers on different physical hosts depending on different factors, such as infrastructure metrics or virtual network metrics.

Conclusions

- As a proof of concept, architectural elements have been designed and implemented and some experimental results using the VLSP testbed presented. We have shown how Placement Engines can start different virtual routers on different physical hosts depending on different factors, such as infrastructure metrics or virtual network metrics.
- The results presented demonstrate that the different embedded algorithms in each of the Placement Engines give very different placement strategies for the virtual routers.

Conclusions

- As a proof of concept, architectural elements have been designed and implemented and some experimental results using the VLSP testbed presented. We have shown how Placement Engines can start different virtual routers on different physical hosts depending on different factors, such as infrastructure metrics or virtual network metrics.
- The results presented demonstrate that the different embedded algorithms in each of the Placement Engines give very different placement strategies for the virtual routers.
- These placement engines may be able to utilize or share some algorithmic elements from compute cloud placement algorithms.

Conclusions

- As a proof of concept, architectural elements have been designed and implemented and some experimental results using the VLSP testbed presented. We have shown how Placement Engines can start different virtual routers on different physical hosts depending on different factors, such as infrastructure metrics or virtual network metrics.
- The results presented demonstrate that the different embedded algorithms in each of the Placement Engines give very different placement strategies for the virtual routers.
- These placement engines may be able to utilize or share some algorithmic elements from compute cloud placement algorithms.
- It is expected that over time the placement algorithms for virtual routers will become more complex and factor-in metrics from both infrastructure and virtual resources, and also to consider placement across multiple data centers.