An Architecture for Creating and Managing Virtual Networks

David Perez-Caparros*, Ishan Vaishnavi*‡, Stefan Schmid†, Ashiq Khan*
*DOCOMO Eurolabs, Munich, Germany
Email: {caparros, khan}@docomolab-euro.com, ishan.vaishnavi@gmail.com
†Telekom Innovation Labs (T-Labs) / TU Berlin, Germany
Email: stefan@net.t-labs.tu-berlin.de

***Abstract*—Reducing operational costs and increasing energy efficiency are critical for mobile operators. One accepted way to tackle these issues is to make use of virtualization to share the same physical infrastructure among several virtual networks. However, sharing physical infrastructure presents a number of challenges, such as security issues, conflicting hardware control, and fair resource sharing. In our earlier work, we analyzed these challenges and demonstrated how virtualized mobile operator networks can address them. However, to create and manage virtual mobile networks, an automated configuration system is required. This system must incorporate the diverse nature of technology, ownership, and geographic domains into an over-arching single virtual network. This paper discusses the main challenges in creating such an architecture. We then present our proposed architecture and describe how it meets these challenges. To conclude, we present our experiences in the real-world implementation of this architecture.**

I. Introduction

Mobile networks have become ubiquitous. This, together with unlimited data flat-rate contracts, has ensured an ever increasing access to varied Internet content over the mobile networks. On the one hand, this move has made mobile networks an integral part of our everyday lives; on the other hand, it has put a strain on the mobile operator resources that was previously unseen. One solution is to update the technology of the mobile networks to provide higher bandwidth to their users. This, though, is a very expensive affair. In such a situation, an alternative is to deploy an infrastructure where multiple generations of mobile network can coexist [1]. This network sharing solution presents many challenges, such as how to ensure traffic isolation, security, dynamic sharing, and monitoring. All of these challenges were presented in our earlier work [2]. Therein we also looked at network virtualization as a possible solution to address them. Creating and managing such virtual networks requires the existence of an automated network management system. We call this system the Network Configuration Platform (NCP). This paper presents the challenges in the design of such an automated architecture, the architecture itself, and our implementation of the NCP. The next section looks some related work followed by the challenges in designing the NCP in Section III. Thereafter, in Section IV we present our architecture of the NCP and address how this architecture helps in meeting those challenges. In Section V we present our implementation of this architecture as a proof of concept.

‡Work done while working at DOCOMO Eurolabs.

II. Related Work

Over the last five years significant research effort has focused on building virtual networks over the same physical infrastructure. The reader is referred to the recent survey [3] [4] and projects GENI, G-Lab and 4WARD for an introduction. GENI[5] is an NSF-funded research project that provides the academia and the industry with a geographically distributed virtual testbed for at-scale networking experimentation. G-Lab[6] is a Germany-wide research project that fosters experimentally driven research on future internet technologies. 4WARD[7] is an EU funded project including partners from industry and academia that focus on improving the management, coexistence, and inter-operability between several network architectures.

To the best of our knowledge, so far, there is no network architecture based on virtualization technologies that fully covers an end-to-end provisioning of virtualized network resources across heterogeneous domains.

III. Challenges

Three distinct business entities can be identified in the proposed architecture: the Physical Infrastructure Owner (PIO), the Virtual Network as a Service Provider (VNaaSP) and the Virtual Network Operator (VNO). The PIO is the entity that deploys and owns the physical infrastructure resources. The VNaaSP is the brokering entity that interacts with one or several PIOs to deploy virtual networks based on the requirements specified by the VNO, the VNaaS consumer. These virtual networks can extend across multiple PIOs. Note that PIOs are not necessarily separate business entities, they may also be separated administratively, geographically, or even by the underlying technology. VNaaSPs may also peer with other VNaaSPs to create virtual networks. The business entities we present are flexible. The same business entity may act as the VNaaSP and the PIO, or for that matter all three, as is the case with mobile operators.

The basic interaction between the business entities to create a virtual network starts with the NCP receiving a request for a virtual network from a VNO. This request specifies requirements as well as constraints that the VNO may impose on the virtual network. The NCP understands this request and finds out the most efficient way to create this virtual network over the physical resources it owns or has interfaces to. It must then issue requests to the PIO to create this network. In case the requested virtual network spans multiple PIOs, the request must be split accordingly into sub-requests to create a complete virtual network.

This generalized business model implies several challenges that need to be tackled. The following subsections look at each of these challenges in more detail.

A. Resource Description Language (RDL)

The RDL serves as a mean of communication between the VNO and the NCP: the requesting entity can specify the requirements of the virtual network in terms of parameters such as network topology, bandwidth, latency to the users, needed processing power and so on. The requirements may include virtualization technology and software / hardware requests. Due to the flexible nature and possible evolution in requirements and architectures, the RDL must be easily extensible. A less intuitive requirement affecting all actors in their economic benefits is the possibility to omit precise specification while at the same time describing real world scenarios. Not only may an RDL ignoring this constraint easily become too bulky to use, it is also likely to force service operators to focus on details they are not interested in. The imprecise nature of the RDL gives the NCP flexibility for a more efficient realization. It can also be used to communicate resource requirements across providers in case of inter-PIO virtual networks.

Many architectures require RDLs to describe and communicate network requirements, services, and embeddings. PlanetLab[8] uses the *Resource Specification* (RSpec) language to communicate network requirements between various actors. RSpec is specified under the *ProtoGENI* project and is a basic RDL tailored to describe virtual network slices. It supports the advertisement and request of resources. RSpec can be used to describe *nodes* and *links*, the interaction between them, and even the services hosted on the nodes. The *Open Cloud Computing Interface* (OCCI) was originally specified by the Open Grid Forum (OGF) as an API for services based on the infrastructure-as-a-service model. It has since undergone many extensions to serve services based on platform-as-a-service and system-as-a-service models. The Network Description Language (NDL)[9] models computer networks and it makes use of the Resource Description Framework (RDF) for this purpose. NDL has an special focus on optical networking, facilitating the provisioning of lightpaths across multiple domains.

There are many RDLs available that fulfill specific needs to describe virtual networks and services. The main challenge facing the RDL is that it needs to be globally standardized to facilitate communication across the various business entities.

B. Isolation

Network isolation refers to guaranteeing complete separation between the various virtual networks over the same physical network. This means that overuse of resources in one virtual network must not affect the service quality in the other. Complete isolation is difficult to achieve. In the case of virtual mobile networks, all their different parts need to be connected with no breaks in the isolation along the end-to-end virtual network. The different isolation techniques for virtualization of the various infrastructure domains of the mobile network may occur at different network layers. Therefore, when a packet crosses infrastructure boundaries it may need to be re-classified. This re-classifier needs to look at all packets, thereby, breaking isolation. Technologies which combine different network layers, such as GMPLS or OpenFlow, can help in achieving this isolation.

C. Cross-domain Embedding

An additional challenge that arises with embedding a mobile operator network is the need to embed the virtual network across multiple technology domains. This implies that the NCP must interact with the operation and management (O&M) planes of each of the different technical domains. It must be able to inquire about resource availability, add or delete virtual resources, configure and re-configure them, and monitor their performance.

The question of where to embed or realize which components of the virtual network is one of the most intensively discussed problems in the network virtualization community. Due to the computational complexity, many heuristics have been proposed on how to reconfigure networks [10] [11], to embed virtual networks with simulated annealing techniques [12], or compute good solutions in special graphs [13]. Many approaches in the literature embed virtual nodes and virtual links separately [14], which entails a loss of efficiency. Virtual network embeddings have also been studied for cross-provider settings [15] and also from a distributed computing point-of-view [16].

All these algorithms assume complete knowledge of the physical infrastructure. In our case, the PIOs may hide the exact topological information or want to optimize embedding based on different parameters from the NCP, rendering these algorithms useless. As we shall see later, our architecture design solves this issue by introducing a hierarchical design to the NCP, thereby, enabling all the previous research to still be used.

D. Monitoring

The performance of each virtual entity needs to be monitored after its creation by the NCP. The adherence to the service level agreements that the NCP has guaranteed to the service operator needs to be ensured. Furthermore, the results of this monitoring, such as bandwidth logs, need to be reported to the service operator. Lastly, monitoring needs to detect service failures and over-load and perform appropriate actions, like service migration, to rectify them. Besides the virtual network, the physical network, and its geographical environment need to be monitored. Failure of physical machines and or links should be dynamically rectified and hidden from the service operator and the users. If this is not possible, appropriate notifications to the operator must be issued. Furthermore, the geographical triggers need to be monitored for possible disaster scenarios. With the sudden increase in demand that accompanies a disaster, essential services, such as voice calls, SMS, and e-mails should be dynamically assigned more network resources and non-essential services, such as video streaming, restricted.

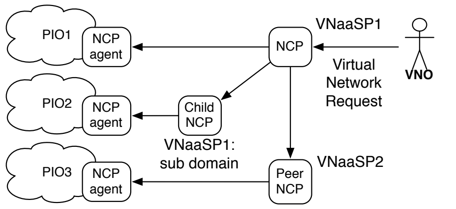


Fig. 1: Architecture: the big picture.

IV. Architecture: The Hierarchical NCP

Based on the challenges presented above, Figure 1 shows an overview architecture of the system showing the NCP as a black box. The main detail of the architecture is that it focuses by design on keeping NCP as a black-box that receives the virtual network resource specification and outputs split resource specifications for the underlying PIOs and peer NCPs. This black box approach implies the NCPs can be stacked on top, or next to each other as peers, to mirror multiple layers of business, technical and administrative entities each having its own NCP. At the lowest level, each PIO also has what we call an NCP agent. The job of this agent is to generate platform specific commands to create and manage the requested virtual resources.

This hierarchical structure solves the major challenges posed in the previous section. The NCP agents are owned by the PIOs and only expose high level information to their parent NCP, such as available computing power, ingress and egress nodes and the bandwidth/delay characteristics between them. Internally, the agent can calculate the most efficient embedding for its part of the sub-virtual network based on the parameters specified by the PIO. Furthermore, peering and stacking of NCPs reflects the varied business technical and administrative domains. Each such NCP abstracts the internal information of that domain and only exposes a common interface to the outside world.

A. *Inside the black box*

Figure 2 shows the inner architecture of the NCP. The NCP's job is to orchestrate the creation and management of end-to-end virtual networks and services for mobile operators. These networks must be comprised of various technical domains, typically: the access network, the core transport network and the service back-end. NCP must completely abstract the physical infrastructure to the service operators. This section explains the function of the various building blocks presented in our design of the NCP.

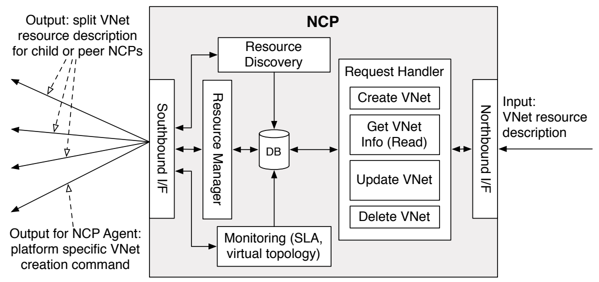

NCP
Output: split VNet resource description for child or peer NCPs
Output for NCP Agent: platform specific VNet creation command
Southbound I/F
Resource Discovery
Resource Manager
DB
Monitoring (SLA, virtual topology)
Request Handler
Create VNet
Get VNet Info (Read)
Update VNet
Delete VNet
Northbound I/F
Input: VNet resource description

Fig. 2: The architecture of the NCP.

The *northbound interface* gives access to the operations provided by the NCP to the VNO and peer NCPs. It checks the syntax of the request and performs the needed authentication. If there are no errors with the received request, it forwards it to the next functional block, the *request handler*. The request handler is responsible for interpreting the request and triggering the appropriate CRUD (Create,Read,Update,Delete) operation in the NCP. There are subtle differences between create and update requests. A virtual network *create request* may require through examination and multiple negotiations (with or without human intervention) on the plausibility and cost of its realization within the specified constraints provided by the VNO. Handling an existing virtual network (*update request*) is easier in the sense that smaller modifications which do not break the existing agreement may be possible. Note that the NCP only receives requests for creation, deletion, modification, the access credentials and the status of the virtual network (update operation). The operation of the services within the virtual network itself is outside the scope of the NCP as long as it does not require any resource modification.

The *resource discovery* component periodically fetch information coming from the NCP agents on available resources in the physical network and stores it in the database. This is to ensure that the *resource manager*, which implements the request splitting function and the resource allocation algorithm, always has up-to-date view of the physical network status. For example, in case of creating a new virtual network, the resource manager performs a preliminary check to see if there are enough resources in the physical network database to realize the request. If the request is plausible, it calculates the end-to-end embedding of the virtual network over the physical network based on service constraints. This may include federating the physical infrastructures across business and technological domains. The controller may also take actions based on monitoring reports it receives. This implements fault tolerance and service migration. The resource manager identifies the remaining resources and maps the requested virtual network according to the implemented resource allocation algorithm. If the downlink NCPs choose to hide physical information details from the manager then it must calculate a split of the virtual network based on the partial information about the physical infrastructure it has.

Once the resource manager has calculated the appropriate embedding it transfers the virtual network split request through the *southbound interface* to the corresponding NCP agent. The agent then sends instructions to the respective domain O&M system to realize the physical embedding of the requested virtual resources. Each domain's O&M system must be able to create *isolated* virtual resources and links within their domain. While the technology to do this has matured in the services domain, it is still in its nascence in the access network. The O&M interfaces must also report on the performance of each of the virtual resources back to the *monitoring* component of the NCP via the NCP agent.

The *monitoring* component is responsible for ensuring that the performance of the virtual network is within the specified parameters. This includes both the SLAs within the physical network as provided to the operator as well as the usage of the virtual network by the operator. As discussed in Section III, monitoring is a challenging subject in virtual networks and currently is an open research issue. The problem is both business and technological. On the business aspect the problem stems from the possible lack of trust between the physical network provider's measurements of performance and the service operator's view of performance. In such a case, the NCP owner may need to act as a reliable intermediary which both the physical infrastructure provider as well as the operator trust. On a technological level, the current architecture of virtual systems is not *predictable*. Various virtual machines may be scheduled by the hypervisor for execution at various intervals, thereby the instantaneous characteristics of the virtual network can vary from one instant to another. Long term statistics can however be reported to the NCP's monitoring component via the O&M of the physical network. The VNO may actively test connections for their performance through the NCP.

V. Prototype and Implementation

We have developed a prototype implementation of the envisioned NCP architecture, including the NCP itself and the NCP agents, as a proof of concept (see Figure 3). This prototype also includes a Graphical User Interface (GUI) for the VNO to define a virtual network request and to get access information to operate it, and another GUI to perform the NCP administration tasks.

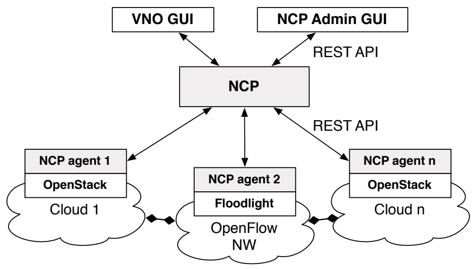


Fig. 3: Prototype overview.

The service provider defines the topology and properties of the nodes and links that form the new service using the provided GUI (see Figure 4). For example, a geographically distributed video streaming service request is shown in Figure 5. After finishing the description of the new virtual service, a *create request* is sent to NCP through the implemented RESTful API. This request is also accessible from the NCP administration GUI. In this GUI, the NCP administrator can access all the received requests and check how the NCP mapped them into the physical resources (see Figure 6).

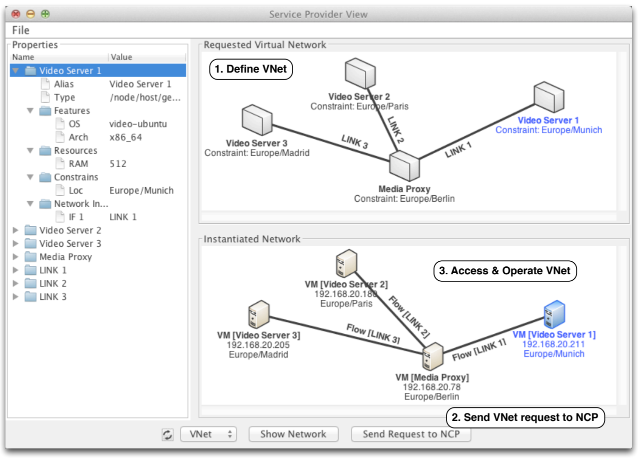
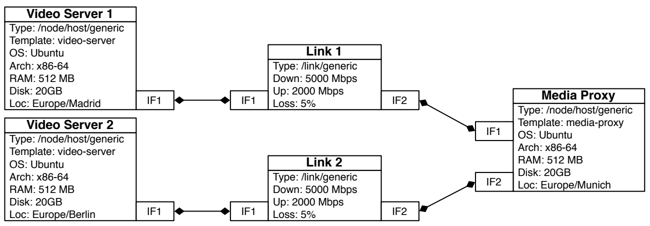


Fig. 4: VNO (e.g. Service provider) GUI.



Video Server 1
Type: /node/host/generic
Template: video-server
OS: Ubuntu
Arch: x86-64
RAM: 512 MB
Disk: 20GB
Loc: Europe/Madrid

Video Server 2
Type: /node/host/generic
Template: video-server
OS: Ubuntu
Arch: x86-64
RAM: 512 MB
Disk: 20GB
Loc: Europe/Berlin

Link 1
Type: /link/generic
Down: 5000 Mbps
Up: 2000 Mbps
Loss: 5%

Link 2
Type: /link/generic
Down: 5000 Mbps
Up: 2000 Mbps
Loss: 5%

Media Proxy
Type: /node/host/generic
Template: media-proxy
OS: Ubuntu
Arch: x86-64
RAM: 512 MB
Disk: 20GB
Loc: Europe/Munich

Fig. 5: Example of a video streaming service request.

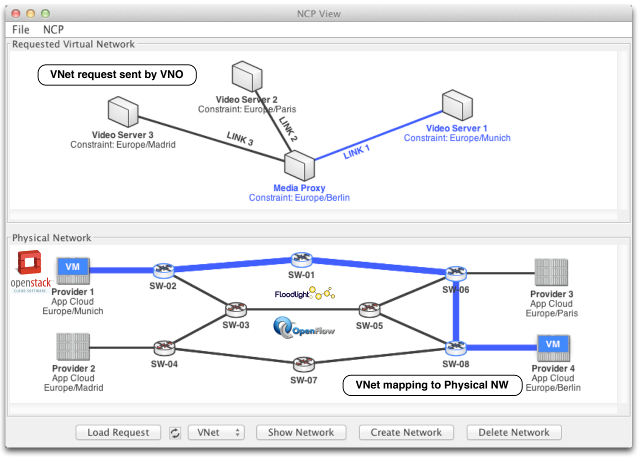


Fig. 6: NCP Administrator GUI.

Upon receiving the virtual service request description (in an RDL [17]), the NCP identifies the candidate PIO and splits the virtual network request into multiple domain specific subrequests. The splitting is based on partial information such as, ingress and egress node, available compute power and bandwidth / delay characteristics from the ingress to egress nodes, which is exposed by the PIO owned NCP agent to the NCP. The NCP takes also into account the constraints defined in the virtual network request (e.g. geographical location of the virtual nodes) when splitting the original request. It then negotiates with the PIOs regarding the necessary physical resources to embed the request. The virtual network provider also offers a management console access for the whole virtual network by relying on the management interfaces of the infrastructure provider.

In our architecture, each virtual network consists of a *data plane* and a *control plane*. The data plane is what one commonly refers to in the context of virtual networks. By default, a newly instantiated virtual network is just an empty set of resources that have to be configured. However, the NCP gives also the option to the VNO to provide certain initial configuration for the different nodes (such as start-up scripts, default users and passwords). For example, we provide basic templates for initial configuration of a virtual server and a media proxy. The video server is automatically configured to stream a video signal to an specified media proxy. While the media proxy is configured to listen for video streaming signals coming from one or several video servers and forward them to the video service clients. The control plane is also necessary during virtual network operation for specific virtual network management tasks. Every player maintains a control interface hosted on dedicated management nodes for "out-of-virtual network" access to the virtual network resources. End-users / end-systems can dynamically connect to a virtual network after successful authentication at the physical attachment point in their local physical infrastructure provider. The authentication occurs via an authentication channel, which may serve one or more virtual networks, provided by their local infrastructure provider. For this purpose, collaborating infrastructure providers may participate in a provisioning and management virtual network [18].

When the NCP agents finish creating the required virtual resources, the NCP gathers the needed access information in order to operate the new service and forwards it to the service provider. This information is shown in the service provider GUI. The service provider has also the option to ask the NCP to delete the virtual service.

In our testbed we have deployed four independent computing clouds interconnected by a transport network composed by eight OpenFlow enabled virtual switches. Each computing cloud is managed by OpenStack. An NCP agent interfaces with the API provided by the cloud operating system (OpenStack). It acts as a translator for the requests sent by NCP towards the cloud OS and for the responses sent by the cloud OS to the NCP. In the OpenFlow network, we use the Floodlight controller to configure forwarding tables of the OF-enabled virtual switches (Open vSwitch) and set up isolated flows in between virtual machines located in separate clouds. In the same way as before, an NCP agent interfaces with the API provided by the OF controller and translates the link requests coming from the NCP to the controller-specific commands and viceversa. The NCP agents make the NCP independent of the cloud or network management system.

VI. Conclusion

This paper described an architecture and prototype implementation that create customized virtual networks over a physical communication network. We presented our architecture and how some of the main challenges are addressed. We also presented our implementation as a proof of concept, with which we can create virtual services across geographically distributed clouds in a quick and easy way. In the future, we plan to expand our prototype implementation to emulate a real mobile network: real optical links, switches, and the access network. We also plan to use the prototype implementation to research and develop embedding algorithms more specific to mobile communication networks.

References

[1] A. Khan, W. Kellerer, K. Kozu, and M. Yabusaki, "Network sharing in the next mobile network: TCO reduction, management flexibility, and operational independence," *IEEE Commun. Mag.*, vol. 49, no. 10, pp. 134–142, 2011.

[2] S. Herker, I. Vaishnavi, A. Khan, and W. Kellerer, "Use cases and derived requirements for a reconfigurable mobile network," in *Proc. IEEE ICC*, 2012.

[3] M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Elsevier Computer Networks*, vol. 54, no. 5, 2010.

[4] A. Khan, A. Zugenmaier, D. Jurca, and W. Kellerer, "Network virtualization: a hypervisor for the Internet?" *IEEE Commun. Mag.*, vol. 50, no. 1, pp. 136 –143, january 2012.

[5] Global Environment for Network Innovations. Http://www.geni.net.

[6] G-Lab. Http://www.german-lab.de/.

[7] 4WARD. Http://www.4ward-project.eu/.

[8] PlanetLab. Http://www.planet-lab.org/.

[9] NDL. Http://www.sne.science.uva.nl/ndl.

[10] J. Fan and M. H. Ammar, "Dynamic topology configuration in service overlay networks: A study of reconfiguration policies," in *Proc. IEEE INFOCOM*, 2006.

[11] N. F. Butt, M. Chowdhury, and R. Boutaba, "Topology-awareness and reoptimization mechanism for virtual network embedding," in *Proc. IFIP/TC6 NETWORKING*, 2010.

[12] R. Ricci, C. Alfeld, and J. Lepreau, "A solver for the network testbed mapping problem," *ACM SIGCOMM CCR*, vol. 33, no. 2, 2003.

[13] J. Lu and J. Turner, "Efficient mapping of virtual networks onto a shared substrate," in *WUCSE-2006-35, Washington University*, 2006.

[14] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proc. ACM SIGCOMM VISA*, 2009.

[15] Y. Xin, I. Baldine, A. Mandal, C. Heermann, J. Chase, and A. Yumerefendi, "Embedding virtual topologies in networked clouds," in *Proc. CFI*, 2011.

[16] I. Houidi, W. Louati, and D. Zeghlache, "A distributed virtual network mapping algorithm," in *Proc. IEEE ICC*, 2008.

[17] G. Schaffrath, S. Schmid, I. Vaishnavi, A. Khan, and A. Feldmann, "A resource description language with vagueness support for multi-provider cloud networks," in *Proc. IEEE ICCCN*, 2012.

[18] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy, "Network virtualization architecture: Proposal and initial prototype," in *Proc. ACM SIGCOMM VISA*, 2009.