Enabling Cloud Connectivity Using SDN and NFV Technologies

Fariborz Derakhshan, Heidrun Grob-Lipski, Horst Roessler, Peter Schefczik, and Michael Soellner

Bell Labs Germany
Alcatel-Lucent

Abstract. Cloud environments play an important role for network and service providers. Cloud network providers require ubiquitous, broadband and minimum-delay connectivity from network providers. There are different realizations of cloud connectivity based on the Software Defined Networking (SDN) and the Network Function Virtualization (NFV) paradigm. In this paper we introduce a new concept based on the OConS architecture developed within the SAIL FP7 project. Our advanced connectivity concept focuses on interdomain connectivity.

Keywords: Interdomain path computation, cloud resource management, SDN, NFV, open connectivity, SAIL.

1 Introduction

Provisioning cloud services based on a platform of distributed data centers was the emerging approach of the past years (Amazon EC2 [1], Google Cloud Platform [2]). By means of virtualization of storage and processing resources, and more and more also network resources, the distributed cloud is able to provide infrastructure as a service for a wide range of applications. Two concepts support service-oriented (virtual) connectivity in the cloud: Software Defined Networking (SDN) and Network Function Virtualization (NFV). Based on these concepts we show how cloud connectivity can be realized in a multi-domain network of distributed clouds. The interdomain resource allocation described in this paper provides an open connectivity service to cope with the complexity of multi-domain networks, especially with regard to control, management and algorithmic elasticity.

This paper is organized as follows: In section 2 we give an overview on the current developments and problems concerning open connectivity concepts. In section 3 we handle the provisioning of resources for cloud connectivity. In section 4 we study the hierarchical and flat interdomain connection management. Section 5 concludes the paper and gives a short outlook to future work.

2 Current Developments and Problems

The concept of Software-defined Networking (SDN) was formally defined in 2009 by Martin Casado but originated as far back as 2007 on work from him, Nick

D. Pesch et al. (Eds.): MONAMI 2013, LNICST 125, pp. 245–258, 2013.
© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2013

McKeown and Scott Shenker. In 2011 SDN was taken up by IETF [3] and is nowadays brought forward by the Software-defined Networking Research Group (SDNRG) [4]. The second trend that is gaining more and more ground is called Network Function Virtualization (NFV) and is promoted by ETSI [5]. In the following we take a closer look at both technologies and clarify the relation between them and the current developments and the concerning problems.

SDN was originally conceived for a campus network usage and was later applied in data centers and is used today also between them. The SDN model is based on the split between the forwarding plane and the control plane. One goal of SDN is to allow applications to request services from the network which can automatically be deployed and monitored. Thus SDN is about bridging the gap between application and network.

In the past applications assumed bandwidth as free and abundant, but in todays networks bandwidth is a scarce resource and must be managed accordingly. Applications also could not impact the delay, availability and dynamicity of the network. SDN was conceived to allow applications to inform the network about their preferences such that it can configure the connectivity accordingly. On the other hand the network can also inform the applications about the changes of topology, bandwidth, delay etc. If the application is implemented to behave appropriately, an improved service is the result.

A prominent implementation that facilitates SDN is OpenFlow which was originally developed by the Stanford University. Today OpenFlow is being taken care of by the Open Networking Foundation [6]. However OpenFlow is not the only representative of an SDN implementation. With ForCES [7], General Switch Management Protocol (GSMP) [8], NetConf [9], and even the well known SNMP [10] there are more standard control technologies and interfaces that are able to configure the forwarding plane at least inside a single network node.

Another approach is the virtualization of network functions so that they can run on more or less general purpose hardware instead of dedicated hardware applied in telecommunication networks today. With the NFV approach operators have the possibility to implement their needed functions anywhere in the network. These functions include switching elements and routers, mobile network nodes, security equipment like firewalls and deep packet inspection appliances, as well as application layer solutions like session controllers, load balancers, content distribution networks, etc. Thereby NFV seeks to reuse existing virtualization mechanisms and is standardizing the interfaces between network elements. NFV is brought forward today by an ETSI ISG (Industry Specification Group).

It is to note that single network functions can also be virtualized and deployed without SDN. An SDN can also exist without NFV-based services. Thus, SDN and NFV are orthogonal technologies.

Network operators are not only interested to save equipment costs by applying NFV-based services on industry standard high volume servers, switches and storages, but also in the possibility to scale them arbitrarily. Furthermore operators need a management system that can orchestrate all the virtualized functions to offer optimized services. This latter goal was addressed in the Scalable and

Adaptive Internet Solutions (SAIL) Project [11] in the Open Connectivity Services (OConS) framework conceived to cope with the challenges posed by the Future Internet. OConS relieves the instantiation, launch and interconnection of functions by use of a specified orchestration procedure. With this OConS can even support challenging flash crowd situations as well as cloud networking and network of information scenarios [12].

SDN and NVF were conceived to enable an easy integration of new services and applications into the cloud. However there are some caveats with both technologies. The problem of SDN is that centralized SDN controllers are a bottleneck and represent a single point of failure. Moreover the performance of the software switch is an issue of concern, for example when it has to carry out complex rules or extensive header rewrites. The latter raises the need for expensive hardware counteracting the wish for a simple switch equipment. And, last but not least, for some applications where the rules in the switch are not preconfigurable additional delay must be taken into account when a logically central and physically distributed set of central controllers must be consulted for packets that a switch cannot handle on its own.

The problem with the NFV partly lies in the fact that the NFV software is more complex to build and harder to maintain due to its distribution onto several machines. Also the reliability of the service is not easily accomplished and depends on interconnectivity, link throughput and delay.

3 Provisioning Connectivity in the Cloud

In the age of social communities, web-based applications connect many social groups with many users via their shared and linked multi-media content like pictures, audio or video. New services built from combinations of such content sources can become popular in a very short time and need a dynamic and powerful infrastructure to be processed. In order to decouple the service growth from the available physical hardware basis, a cloud provider may be used to manage the virtual infrastructure.

The SAIL [11] project developed a model how a cloud provider that owns or contracts several distributed data centers can offer a system for service creation and deployment in a heterogeneous environment, i.e. dealing with several local cloud management systems, and interconnecting the local cloud domains [13].

Figure 1 gives a simplified view of the proposed interworking of cloud and network domains. At the cloud level the cloud provider will set up a distributed cloud manager to create and configure the storage and processing resources to be deployed as virtual machines (VMs, not shown in Figure 1) in the distributed data centers DCn.

Thinking about real implementations, the OpenStack [14] collection of open source technologies delivers a scalable cloud operating system. Initially the OpenStack project was announced in 2010 by Rackspace and NASA but today many players build on the OpenStack open source software initiative for building clouds. Over 150 companies gathered in the OpenStack collaboration and provide architectural input, contribute code, and/or integrate it into their business offerings.

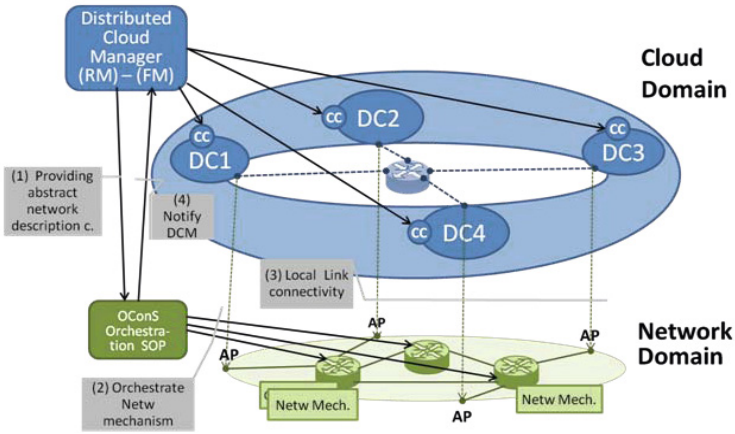

Fig. 1. Interworking of cloud and network domain functions

The OpenStack cloud operating system is able to control large pools of computing, storage, and networking resources. All these resources are managed through a so called dashboard which runs in a browser in a RESTful manner. The dashboard gives administrators and users the control needed to setup and monitor a complete service consisting of computing, storage and networking instances.

In order to set up an application network between the created VMs in the data centers, three steps are necessary as described in the approach of SAIL [15,16]:

- The first step is to transfer an application graph into a virtual network of communicating VMs.
- The second step is to connect the virtual VMs with the physical interfaces in the data centers which is typically done by support of a hypervisor respective the management tools that come with it (e.g. based on libvirt in an open source environment). This has to be managed in each data center by the local Cloud Controller (CC) in a coordinated way. To enhance this task, the SAIL project proposes to use the OCCI-based description technique OCNI [17], and provides the corresponding tools via the libNetVirt library [18,19].
- The third step is to connect the distributed data centers with dedicated managed networking resources, also across different network domains that guarantees a certain degree of end-to-end (i.e. DC to DC) QoS as required by the application network.

Typically the connectivity view between the distributed data centers at the cloud level is based on a "single router abstraction" for the network. The distributed cloud manager delegates the task of network and path deployment to the contracted network provider which first translates the single router network descriptions into potential subtasks to related network domains (and their

providers) and then orchestrates the appropriate managed network services. At the attachment points between data center and network as well as between network domains, the corresponding dynamic protocol parameters have to be exchanged through a link negotiation protocol [13].

Within this approach, the current paper discusses a proposal for the architecture, protocols and algorithms for resource and path management across multiple network domains offering interconnectivity services to cloud providers.

4 Interdomain Connection Management

The effort for connection management can significantly increase due to the dynamics of the network resources, and due to the universality of users' (cloud providers', end-users', applications', etc.) resource demands and preferences. To relieve the intradomain network controller in our architecture the management of domain-external resources is delegated to an entity called interdomain DCU (Domain Control Unit). The interdomain DCU treats each connected domain as a single resource with specific attributes.

In [20] we introduced a network architecture with a DCU in each domain separating the data plane from the control plane and comprising most of the control plane mechanisms (i.e. path computing) into one single entity. The DCU clients in each network element deliver topological and resource information to the local DCU which then abstracts from the details of domain-internal topology and resource information to a global information, e.g. domain load state, such that detailed domain-specific topology and resource information is hidden towards the interdomain DCU. Detailed information about topology (e.g. addresses of the network entities) and resources remains in the local DCU.

There are already diverse path computation mechanisms that can be applied for the interdomain resource allocation. In this paper we describe enhanced concepts that utilize additional resource information to improve the efficiency of the interdomain connectivity management.

Generally, the connection management can be divided into three parts: The first part comprises the gathering of current network topology and resource information. We focus on delivering abstract resource information from each domain to the interdomain DCU and storing them in its Traffic Engineering Database (TED) together with a timestamp. This TED also stores network topology, operator policies and predefined interdomain paths. In section 4.4 we introduce a novel mechanism called PSCEH (Publish Subscribe with Configurable Event Handling) to increase the efficiency and quality of the information exchange. Resource information can be for example the load of computational resources, current storage capacity, link bandwidth, and network topology. The second part consists of path computation and connectivity configuration performed by the interdomain DCU based on the available topology and resource information. In the third part controllers' decisions are enforced on the respective network entities in order to realize the connectivity.

Some aspects of information retrieval and path calculation follow the ideas of interdomain PCE (Path Computation Entity) procedures, e.g. [21]. Based on

the PCE concept the mechanisms described in this paper in sections 4.3 and 4.4 utilize additional resource information and interact tightly with the TED which further includes operator policies to improve the efficiency of the interdomain connectivity management.

4.1 Hierarchical Interdomain Connection Management

For hierarchical interdomain connection management the DCU of each domain registers at an interdomain DCU during network startup. The DCUs send a registration request to potential interdomain DCUs and receive an ACK from the responsible instance. Since the network structure is supposed to be quite static the mapping of the DCUs to the responsible interdomain DCU can be predefined by the network provider in order to reduce the discovery effort. During registration the operators deliver their policies to the interdomain DCU, which stores them in the TED together with the respective topology information.

Figure 2 shows the hierarchical interdomain connection management performed by the interdomain DCU to set up a path between the source domain S and the destination domain D.

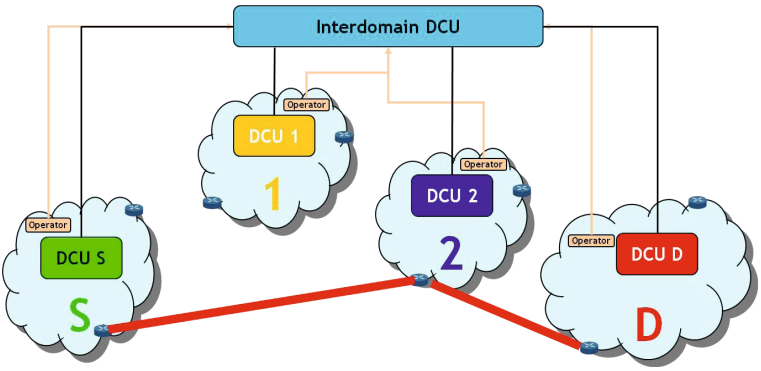

Fig. 2. Interdomain connection management by the interdomain DCU between a source domain S and a destination domain D via intermediate domains 1 and 2

When a request arrives at a local DCU, the DCU first checks whether the destination address of the request can be found in the topology information of the local TED. If the DCU cannot resolve the request, it forwards the request to its interdomain DCU. The interdomain DCU first sends a request to all of its DCUs in order to identify the domain containing the destination address. The DCUs then check whether the requested destination address is stored in their TEDs. The DCU which keeps the requested address in its TED responds to the interdomain DCU with an ACK. The information the local DCUs submit to the interdomain DCU is kept at minimum possible level to encapsulate the data only within elements where they are needed.

After the destination domain is identified, the interdomain DCU first retrieves the resource information from its TED to compute candidate paths from source to destination with respect to the constraints defined in the request. Constraints can be for example a minimum bandwidth, an application class like HD video, real-time video conferencing, user class, security level, etc.

If the resource information of some DCUs in the candidate paths is not up-to-date, i.e. the corresponding time stamp is too old, the interdomain DCU sends an explicit request to all concerned DCUs to retrieve the current resource state. Triggered by this request the respective DCUs perform a path computation within their domain to determine the current resource information. After receiving the resource state updates the interdomain DCU selects the optimum final solution among the candidate paths and informs the source DCU.

The hierarchical architecture however has two disadvantages. If the processing capacity of the interdomain DCU and the links towards the interdomain DCU are not dimensioned sufficiently (which can be very expensive) an overload situation might occur which in worst case can lead to an outage of the interdomain DCU and consequently of the complete interdomain connection management, if no duplicate interdomain DCU is available.

4.2 Peer-to-Peer Interdomain Connection Management

In order to cope with the disadvantages of the hierarchical interdomain connection management we define a flat architecture by transferring the interdomain DCU functionality into the DCUs. In this case unresolved resource requests are forwarded by the source DCU to all neighboring peer DCUs. For avoiding the flooding of the entire DCU network, it can be organized according to a spanning tree.

Like in the hierarchical architecture each DCU checks whether the requested destination address is stored in its TED. The DCU which keeps the destination address in its TED responds to the requesting DCU with an ACK. After the destination domain has been identified, the source DCU first retrieves the resource information from its TED to compute the candidate paths to the destination with respect to the constraints defined in the request.

If current resource information is needed, the source DCU triggers all DCUs of the candidate path to deliver current resource state updates. The source DCU then determines the optimum final solution. Figure 3 shows the flat peer-to-peer interconnection management architecture. In this example a path set up between a source domain S and a destination domain D is triggered by the DCU S.

In order to speed up the path computation process the source DCU can provoke each DCU on the candidate path to compute a local path and perform local flow establishment during the time when the source DCU itself does the same. This preconfiguration of the involved domains allows for immediate forwarding of the packets of the request.

The advantage of a flat architecture is the lack of a single point of failure, however its disadvantage is an increase of interdomain communication and more delay in responding to requests due to the absence of a central supervisor and

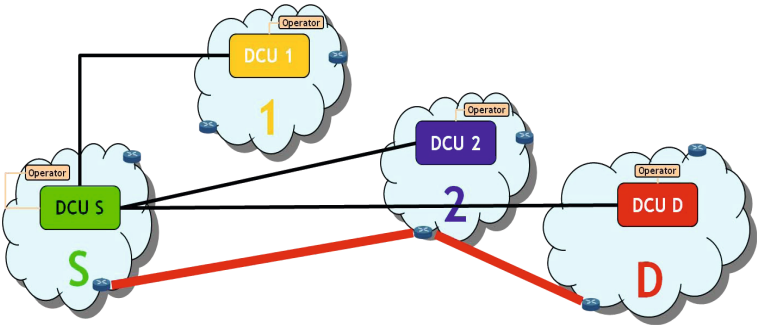

Fig. 3. Peer-to-peer interdomain connection management between a source domain S and a destination domain D via intermediate domains 1 and 2

database. Further, the transfer of the interdomain DCU intelligence into the DCUs makes the latter more complex and more expensive.

4.3 Multi-criteria Resource Management

For a rapid and adaptive path computation and for responding elastically to the requests of clients we introduce an algorithm for multi-criteria path computation to be applied by the interdomain DCU in a hierarchical architecture or by the source DCU in a flat architecture.

To cope with the complexity of the path computation the interdomain or the source DCU first considers the priorities of performance indicators (e.g. bandwidth, delay, etc.) as defined by the client through weights assigned to them. Given a set of n performance indicators with priorities $p_1, p_2, \cdots, p_n$ the DCU successively computes the paths satisfying the defined optimization criteria with respect to the corresponding indicators in decreasing order of importance. After sorting the indicators in descending order of their priorities the algorithm starts the computation of the optimum solutions $\{S_1\}$ with respect to the first indicator and a predefined default solution tolerance $t_{1,0}$. In case that no solution is found, the algorithm checks whether the acceptable tolerance limit is already reached or not. If not, it increases the tolerance and restarts the computation of solutions. Otherwise, if no more tolerance is acceptable, it returns an empty solution set $\{\emptyset\}$.

In case that a solution set $\{S_1\}$ is found, the algorithm continues the procedure with the next parameter only if it exists and has a non-negligible priority ($p_{i+1} \sim p_i$) that justifies the computation effort. In this case the algorithm searches the solutions $\{S_2\}$ within the previously found solution set $\{S_1\}$ that satisfied the criteria concerning the first indicator. If no solution is found the algorithm tries again to relax the solution constraints (increase the tolerance t_2) first with respect to the current less important indicator, and then successively, if still no solution is found, with respect to the previous more important indicators. In case that multiple solutions are found the final solution is randomly chosen among them. Figure 4 depicts a schematic flow chart of the algorithm.

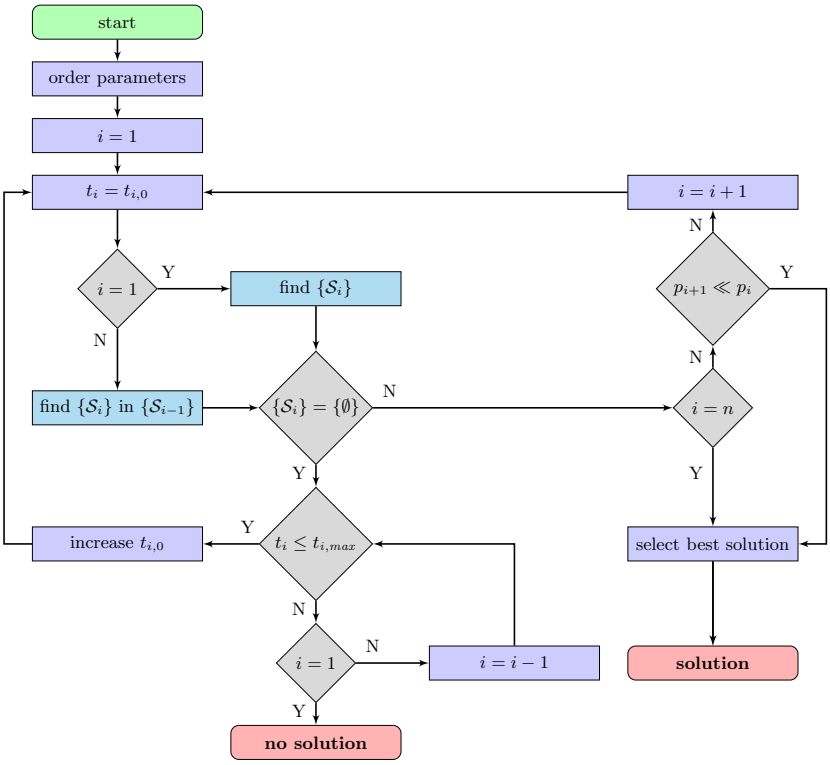

Fig. 4. Multi-criteria path computation starts with the performance indicator with the highest priority p_i and with a predefined initial solution tolerance $t_{i,0}$. It only proceeds to the next indicator if its priority p_{i+1} is non-negligible. In case of an empty solution set $\{\emptyset\}$ the algorithm tries to expand the tolerance interval of solutions and restarts the optimization. The process stops when an exit event occurs.

The priorities of indicators are predefined by the clients according their needs. The less the number of indicators and the larger the tolerances, the less the computational effort, but also the lower the quality of solutions. To increase the efficiency of the process, each solution can be tagged with a time stamp and stored in the TED such that a new computation of solutions is only started if the already available solution has become obsolete. The expiration conditions can be dynamically defined by the controller. However, a re-computation can also be triggered proactively to improve the quality of available solutions. The new solution can then be offered to the requester without its explicit request.

The described mechanism can be enhanced to estimate the resource demand of arriving requests in advance by applying statistical and probabilistic methods on the current network resource state.

4.4 Publish/Subscribe with Configurable Event Handling (PSCEH)

The algorithms described in section 4 need support from efficient and adjustable resource state reporting mechanisms. The manner of information exchange in distributed systems has a significant impact on the resource management efficiency. On the one hand, a low reporting frequency leads to a coarse knowledge of the state of the concerning resources which reduces the resource management performance. On the other hand, a high frequency of notifications leads to an increase of signaling overhead which again reduces the management performance.

There are diverse kinds of publish/subscribe mechanisms based on polling, pulling, pushing or advertising, e.g. [22]. For optimizing the information reporting we developed the PSCEH mechanism. It enhances the basic publish/subscribe paradigm by introducing provider-tailored publishing and consumer-oriented subscription, which represents the main advantage compared to already known information reporting mechanisms.

The PSCEH method defines a generic and flexible generalization of publish/subscribe techniques and exploits the advantages of them. It enables the involved resource management entities to customize their information exchange in order to minimize the computation effort and the required signaling, and thus enhances the resource management quality.

In the PSCEH process the subscribing DCU compiles a subscription profile containing its identity, the resource parameters of interest (bandwidth, delay, etc.) and the configuration parameters (events, event granularities, time hystereses, event thresholds, notification frequencies, etc.) for reporting. By sending this profile to the monitoring DCUs the subscribing DCU subscribes to the reports of them. The parameters in the profile may be defined as optional or mandatory. In case that an event as defined in the profile occurs, the concerning DCUs send a notification to the subscribing DCU.

Figure 5 shows an example of the PSCEH in a hierarchical environment with exchange of subscription and notification messages between interdomain DCU as subscriber and DCUs as publishers. If the interdomain DCU needs the resource information of the DCUs 1, 2 and D, it compiles a subscription profile as described above and sends it to them. Thus, whenever an event of interest occurs, the DCUs notify the subscribing interdomain DCU according to its subscription profile.

Figure 6 shows an example of the PSCEH in a flat peer-to-peer environment with exchange of subscription and notification messages between peer DCUs. In this case, if the DCU S needs the resource state information of DCUs 1, 2 and D, it compiles a subscription as above and sends it to them. The rest of the process is the same as described above.

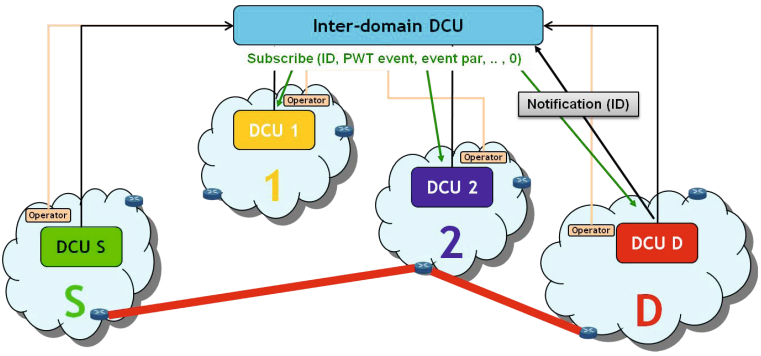

Fig. 5. PSCEH supporting hierarchical interdomain connection management

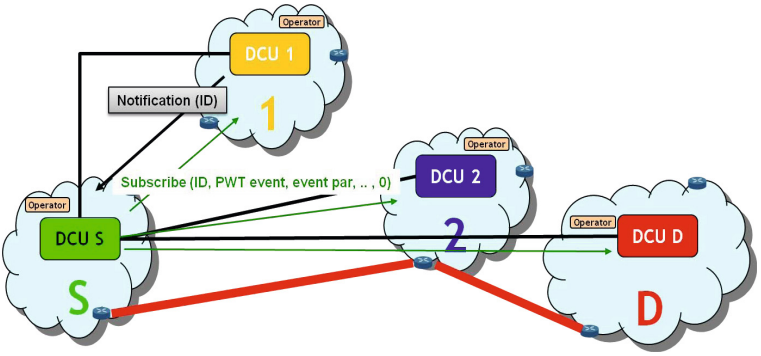

Fig. 6. PSCEH supporting peer-to-peer interdomain connection management

The subscribing DCU has the possibility to redefine the parameters of the reporting by updating the corresponding entries in the profile. This way, the PSCEH mechanism allows the DCUs to adapt not only the notification frequency but also the measurement parameters and the reporting events to their actual needs and privacy requirements.

Further, the PSCEH allows for collecting resource information prior to the arrival of resource requests such that the DCUs are relieved from time consuming message exchange. This depicts a non-trivial advantage in time-critical stress phases with short inter-arrival times of requests. By decoupling the information exchange from the time-critical decision phase the PSCEH enables the DCUs to apply more complex resource management and optimization algorithms with sophisticated and customized preprocessing of resource information.

For relieving the DCUs in the central architecture from interdomain path computation the interdomain DCU has to be protected by redundancy mechanisms to prevent any inoperability.

5 Conclusion and Future Work

In this paper we showed how cloud connectivity can be realized in a multi-domain network of distributed clouds. The SAIL open connectivity service (OConS) and the resource management algorithm of the DCU described in this paper support interdomain resource allocation that can deal with the complexity of forthcoming multi-domain networks. The introduced concepts are promising for dynamic service deployment in future cloud networks. Thereby OConS facilitates the instantiation, initiatian and interconnection of functions by a specified orchestration procedure.

A first proof-of-concept demonstration was shown at the FuNeMS 2012 in Berlin [20] with focus on the principles of load-dependent resource allocation between application, cloud management (CloNe) and network (OConS). Thereby a web-based service and management interface was used to demonstrate the control and management of the requested data paths. The DCU with its domain controller was built as an OpenFlow controller and the connectivity was provided using the OpenFlow protocol. To demonstrate elastic networking between cloud nodes (data centers) and network domains we used a distributed video processing application [23].

In the future we intend to take the NFV and "networked cloud" paradigm a step further by applying it to the mobility algorithms in a 4G mobile access network. Virtualization activities for the wireless access system are already studied, e.g. in the NGMN CRAN activities [24] or discussed in [25]. Additionally, mechanisms described in this paper can be considered in order to allow a better resource usage of processing resources in the wireless network. For a centralized architecture a subscription mechanism for the information exchange described in section 4.4 can be applied whereas for a decentralized architecture an information retrieval triggered by broadcast messages is more advantageous. Furthermore, it will enable a more efficient (physical processing) resource utilization allowing for either energy saving gains or additional processing headroom for next-generation features in wireless networks as already envisaged in the LTE-Advanced specifications.

Acknowledgments. This work has been partially funded by the European Commission under grant FP7-ICT-2009-5-257448-SAIL. We would like to thank our colleagues in the SAIL project for the many fruitful discussions.

Abbreviations

ACK	Acknowledgement
D	Destination
DC	Data Center
DCU	Domain Control Unit
I-DCU	Interdomain DCU
NFV	Network Function Virtualization
OCCI	Open Cloud Computing Interface
OCNI	Open Cloud Networking Interface
OConS	Open Connectivity Services
PCE	Path Computation Entity
PSCEH	Publish Subscribe with Configurable Event Handling
S	Source
SAIL	Scalable and Adaptive Internet Solutions
SDN	Software-Defined Networking
TED	Traffic Engineering Database
VM	Virtual Machine

References

1. Amazon Elastic Computer Cloud (Amazon EC2), `aws.amazon.com/ec2`
2. Google Cloud Platform, `cloud.google.com`
3. Bird of a feather session on Software-defined Networking (SDN), IETF-82, Taipei (November 17, 2011), `tools.ietf.org/agenda/82/sdn.html`
4. Software-defined Networking Research Group (SDNRG) of the IRTF, `trac.tools.ietf.org/group/irtf/trac/wiki/sdnrg`
5. Network Functions Virtualization Introductory White Paper. SDN and OpenFlow World Congress, Darmstadt, Germany October 22-24 (2012), `portal.etsi.org/NFV/NFV_White_Paper.pdf`
6. Open Networking Foundation, `http://www.opennetworking.org`
7. Doria, A., Hadi Salim, J., Haas, R., Khosravi, H., Wang, W., Dong, L., Gopal, R., Halpern, J.: "Forwarding and Control Element Separation (ForCES) Protocol Specification", IETF RFC 5810 (March 2010)
8. Doria, A., Hellstrand, F., Sundell, K., Worster, T.: General Switch Management Protocol (GSMP). V3, IETF RFC 3292 (June 2002)
9. Enns, R., Bjorklund, M., Schoenwaelder, J., Bierman, A.: Network Configuration Protocol (NETCONF). IETF RFC 6241 (June 2011)
10. Case, J., Harrington, D., Presuhn, R., Wijnen, B.: Message Processing and Dispatching for the Simple Network Management Protocol (SNMP). IETF RFC 3412 (December 2002)
11. The SAIL Consortium, `http://www.sail-project.eu`
12. Ferreira, L.S., et al.: Open Connectivity Services for the Future Internet. In: IEEE Wireless Communications and Networking Conference (WCNC2013), Shanghai (April 2013)

13. SAIL, Refined CloNe Architecture, Deliverable FP7-ICT-2009-5-257448-SAIL/D.D.3, SAIL project (October 2012), www.sail-project.eu
14. OpenStack Cloud Software, www.openstack.org
15. SAIL, Applications for Connectivity Services and Evaluation, Deliverable FP7-ICT-2009-5-257448-SAIL/D.C.4, SAIL project (February 2013), http://www.sail-project.eu
16. Puthalath, H., Soares, J., Melander, B., Sefidcon, A., Carapinha, J., Melo, M.: Negotiating On-demand Connectivity Between Clouds and Wide Area Networks. In: IEEE CloudNet 2012, Paris, France (November 2012)
17. PyONCI, github.com/danieltt/PyOCNI
18. libNetVirt, github.com/danieltt/libnetvirt
19. Turull, D., Hidell, M., Sjdin, P.: Using libNetVirt to Control the Virtual Network. IEEE CloudNet 2012, Paris, France (November 2012)
20. Derakhshan, F., Grob-Lipski, H., Roessler, H., Schefczik, P., Soellner, M.: On Converged Multi-domain Management of Connectivity in Heterogeneous Networks. In: Future Networks & Mobile Summit 2012 Berlin, Germany, July 04-06 (2012)
21. Vasseur, J.P., Zhang, R., Bitar, N., Le Roux, J.L.: Backward-recursive PCE-based Computation of Shortest Constrained Inter-domain Traffic Engineering Label Switched Paths. IETF RFC 5441 (April 2009)
22. Esposito, C.: A Tutorial on Reliability in Publish/Subscribe Services. In: DEBS 2012, Berlin, Germany (2012)
23. SAIL, Demonstrator for Connectivity Services. Deliverable FP7-ICT-2009-5-257448-SAIL/D.C.5, SAIL project (February 2013), http://www.sail-project.eu
24. NGMN Alliance, Projects, www.ngmn.org/de/workprogramme/wpoverview.html
25. Haberland, B., Derakhshan, F., Grob-Lipski, H., Klotsche, R., Rehm, W., Schefczik, P., Soellner, M.: Radio Base Stations in the Cloud. Bell Labs Technical Journal 18(1), 129–152 (2013)