An Integrated SDN Architecture for Applications Relying on Huge, Geographically Dispersed Datasets

Andy Georgi
Technische Universität Dresden
E-Mail: Andy.Georgi@tu-dresden.de

Reinhard G. Budich
Max Planck Institute for Meteorology
E-Mail: Reinhard.Budich@zmaw.de

Yvonne Meeres
Max Planck Institute for Meteorology
E-Mail: Yvonne.Meeres@zmaw.de

Rolf Sperber
Embrace HPC-Network Consulting
E-Mail: Rolf.Sperber@embrace-net.de

Hubert Hérenger
T-Systems Solutions for Research GmbH
E-Mail: Hubert.Herenger@t-systems-sfr.com

***Abstract*—The target of our effort is the definition of a dynamic network architecture meeting the requirements of applications competing for reliable high performance network resources. These applications have different requirements regarding reliability, bandwidth, latency, predictability, quality, reliable lead time and allocatability. At a designated instance in time a virtual network has to be defined automatically for a limited period of time, based on an existing physical network infrastructure, which implements the requirements of an application. We suggest an *integrated Software Defined Network* (SDN) architecture providing highly customizable functionalities required for efficient data transfer. It consists of a service interface towards the application and an open network interface towards the physical infrastructure. Control and forwarding plane are separated for better scalability. This type of architecture allows to negotiate the reservation of network resources involving multiple applications with different requirement profiles within multi-domain environments.**

Keywords – Software Defined Networking, Huge Data, Network Architecture

I. INTRODUCTION

In fields like climate research, astronomy and high energy physics, international collaboration is standard today. Consequently, mass data have to be transported between compute and storage sites. Federation techniques consolidate the view on dispersed datasets, however they do not transport them. Mass data applications relying on multi-site datasets require networks with highest possible bandwidth and additional features, such as low latency and low latency variation at a dedicated instance of time for a period of time.

Applications taking advantage of distributed computing rely on highest possible bandwidth and lowest possible delay and delay variation for short periods of time. Both types of applications should be able to utilize the same network without mutual interference. Current developments like the Open Flow standard [17] and the Network Service Interface (NSI) [21] suite are a step towards fully automated virtual network provisioning. The NSI activities bridge the gap between application and network while OpenFlow standardizes the communication between controller and network element.

In this paper, we introduce an integrated multi-domain SDN architecture, in which network control is decoupled from forwarding and directly programmable by open interfaces between different layers. In Section II some approaches targeting similar objectives are evaluated. Our architecture as well as an automated network configuration process arbitrating between the concurrent applications competing for network resources is described in Section III. Finally, we summarize our approach in Section IV.

II. RELATED WORK

Approaches to provide application-oriented sustained network services on demand already exist. Also, protocols for traffic engineering and optimization were specified. For both, some examples are given in this section, together with the reason they don't gain acceptance.

A. Network service architectures

ITU-T G. 805 03/2000 [1] describes a technology independent functional architecture of a transport network. At the time the standard was written it provided a set of functional architecture recommendations for the prevailing network transport technologies. However, because of the agnostic character of G.805 it is valid for all types of current transport technology and future implementations to come. Multi layer networking and multi layer control plane interworking are not considered.

GMPLS [16] is a generalization of IP/MPLS for layer 1 transport services. It was initially defined to enable dynamic restoration in transport networks. In the course of the VIOLA project [14] UNI-Client and UNI-Server interworking was implemented between the transport layer and an IP/MPLS layer. This way bandwidth requirement from the IP/MPLS layer could be communicated to the transport layer.

The common Network Information Service Schema Specification (cNIS) activities [15] by Geant2 community are targeted at supplying domain related network information to the application layer regardless of the network layers present in the respective domains. Inter domain exchange of information is part of the service. The cNIS activities are vital for the NSI definition.

G.805 abstracts from hardware related description of transport networks, GMPLS defines cross network layer interworking and cNIS finally makes the networking layer transparent for the application layer. The missing link is a control interface between the application and the network.

B. SDN architectures

We also evaluated other SDN approaches like e.g. [5], [6], [7], [8] or [20]. Thereby we recognized three main deficiencies: 1) Single-domain solutions only ([5], [8])

 ISBN: 978-1-61208-310-0

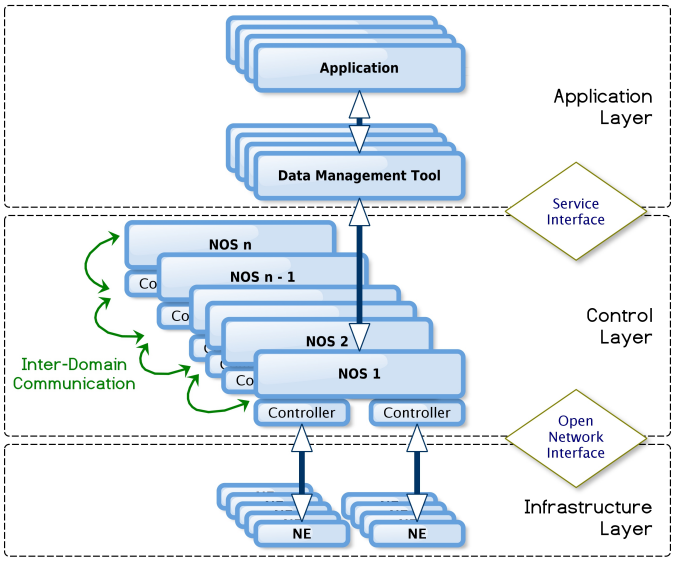

Fig. 1: Integrated SDN architecture overview

2) Based on a specific protocol like OpenFlow ([6], [8])
3) Missing consideration of the application layer ([7], [20]).
Hence, our architecture described in the following section will support end-to-end negotiation of network resources, depending on the requirements of the application. Thereby, applications are implemented by the end user and not equatable to network functions. Communication partners can be located within different domains and the infrastructure is not limited to a single protocol.

III. Integrated SDN Architecture

To provide application-oriented network services, we suggest an architecture consisting of three layers, depicted in Figure 1. The infrastructure layer defined by the network providers and hardware vendors is usually characterized by a vast heterogeneity. It lays at the bottom of our architecture. The control layer in the middle abstracts from the infrastructure layer and prevents direct user access to the hardware. At the top level sits the application layer representing the users view on this network architecture.

Interoperability between these layers enables an application- and user-oriented network infrastructure. To achieve this, additional communication protocols have to be specified providing the required functionality. Therefore, we introduce an Open Network Interface (ONI) between infrastructure and control layer as well as a Service Interface (SI) with an appropriate connection service protocol between control and application layer.

In the following, we describe the layers and the intermediate communication protocols in more detail and give an example of a communication process within this architecture.

A. Layer description

In the following sections we describe the three layers of the introduced architecture. The application layer representing the users view, the control layer as intermediary between application and network hardware, and the infrastructure layer, consisting of the network elements and their interconnects.

TABLE I: Time to Transport 1 TB and 1 PB of Data from one Climate Center to Another [9]

Transfer Rate	*Time to Transport Data of Size* 1 TB	1 PB
10 Mbps	9.7 days	27.20 years
50 Mbps	1.94 days	5.44 years
100 Mbps	23.3 hours	2.72 years
1 Gbps	2.28 hours	97.1 days
10 Gbps	13.65 minutes	9.7 days
100 Gbps	81.9 seconds	23.3 hours

1) Application Layer: The activities in data management rely on fast, broadband, reliable networks. At the moment, intercontinental network speeds are limited to 40 Gb/s [4]. The project *Advanced North Atlantic 100G Pilot* (ANA100G) tries to reach the 100 Gb/s barrier [10] On the other hand, practical experience shows, that the opportunistic networks (WANs) available today do not offer enough reliability, predictability and speed per cost necessary for the applications from the "Data Tsunami".

The international *Coupled Model Inter-comparison Project* (CMIP) [3] conducts sets of co-ordinated experiments with numerical climate models to compare them against each other. This comparison of climate models serves as the basis for the *Assessment Reports*, on which the Nobel Laureate *International Panel on Climate Change* (IPCC) bases its recommendations for policy makers.

Numerical Climate Models regularly utilise to a high percentage high performance computers like those to be found in the TOP500 list [22]. They also swamp these computers with data volumes at the bleeding edge of the most current technologies available (for an overview of some of the problems see e.g. [13]). As the name of the project suggests (Coupled Model Inter-*comparison* Project), these data need to be compared. Since the models and their data are situated at different places, they have to be transported. Or the applications that compare the data have to be available near to the data – unfortunately practical experience shows that it is much easier to organise data near to applications than applications near to data, see e.g. the results of the German C3-Grid initiative [11].

The climate modeling community agreed upon sub-setting the data to a volume of about 1,5 PB, containing only those data most relevant for the comparison, and to make these data available in five centers world-wide for easier access and replication [3]. ESnet says: "The fastest we could hope to move only 1 PB of data from PCMDI to one of the RCA data centers is essentially one day at 100 Gbps, whereas with a peak of 10 Gbps, it would take almost 1.5 weeks." An estimation of the speeds following the ESnet can be found in Table I, whereas the last row – the 100 Gbps – are not reached yet, but only addressed in the ANA100G project [10]. The fact that many network lines outward bound from centers seem rather underutilised does not contradict this observation: Burstwise utilisation is common-place, people try to get their job done, but the unreliability of the connections and the fact that the slowest part of the complete connection limits the transfer speed, make life of the users difficult: Maintaining constantly high data transfer speeds is near to impossible today.

If we interpret current negotiations about CMIP6, the future

 ISBN: 978-1-61208-310-0

edition of CMIP, correctly, it can be expected that the data volumes will be 1 to 2 orders of magnitude higher than in CMIP5, with the intercontinental network speeds staying about the same. With respect to the architecture of the application layer it can be expected that the available system (ESGF) will be stabilized and possibly extended by a federated file system. The situation for the CMIP5 data: Until now the scientists have to search through a data jungle by clicking through web portals, looking at folders on different servers or using scripts. With neither of these methods all data can be accessed. E.g. the scriptbundle called synchro-data [19] can access only the data available with the new ESGF login method, not with the old one, and requires a special port which is then locked. Two users at one time can't download from the same machine at the same time. Again: many networks seem to be underutilised, but not because the scientists don't need their data, but because retrieving them is intransparent. The scientists want to know how to get the data and how long the transfer takes.

A data management tool hides this data access complexity. It informs about the expected time constraints and offers reserving mechanisms. In contrast to the current tools it offers a feedback about the transfer – a very essential but missing feature. This new functionality is offered by lower SDN layers to the application layer.

A totally different, but also very demanding application for the network is state-of-the-art turbine development as it is performed at DLR (*German Aerospace Centre*). It requires a multitude of different process chains to be completed. Such process chains typically consist of different simulation tools such as CFD and CSM solvers, which are executed in a specific collaborative order. The data is needed "just in time": At DLR different clusters of different sizes and configurations are available at geographically distributed locations. Optimal resource usage implies high flexibility in where to run jobs. In order to avoid necessity of moving data to a selected resource in order to be able to run a job it is desirable to provide reliable and fast access to all data from all different resources and locations. This is not "Big Data" application but it urges the network to be prioritized.

Thus we have the climate application which needs high bandwidth for a long time and can manage interruptions, and the turbine application which needs prioritization to receive the data as fast as possible. These two applications do not cumber each other and are a good example for challenging our SDN architecture.

2) Control Layer: Our control layer, depicted in Figure 2, consists of two main components – the Network Operating System (NOS) on the application side and one or more Controllers on the side of the infrastructure. The communication between the upper and lower layer is realized by a north- and southbound API. Additionally the NOS module within the control layer needs an interface for inter-domain communication to enable multi-domain interoperability.

The NOS we introduce operates similar to typical operating systems. Within its domain it interacts as an intermediary between applications and network hardware, to avoid direct access to the network hardware and to hide unnecessary information. This increases the security on the one hand and enables the possibility to virtualize the network on the other hand. Therefore, the integrated Broker compares the requirements – transmitted through the northbound API – with

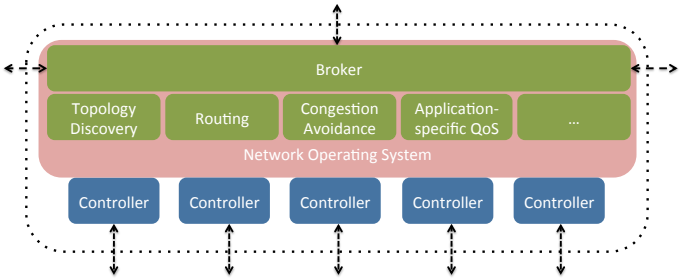

Fig. 2: Single domain control layer overview

the available network resources – which can be requested through the southbound API – and instructs the reservation if available resources meet the requirements. Negotiation between application layer and network layer should be possible. The requesting application receives only a partial graph, which can be a direct link with the corresponding characteristics between ingress and egress at the end.

Besides the virtualization our approach also takes traffic engineering into account. Link state information can be updated periodically or requested on demand via the southbound API. This way, weighted graphs are composed for the entire domain, in which the weights can represent any link parameter – like bandwidth, latency, utilization or costs – or any combination of them. Based on these graphs the route is optimized w.r.t. the requirements of the applications. Link parameters should not change during transmission. However, if a change is inevitable, the network resources dedicated to a certain application should be adapted within feasible bounds.

Real time communication is another feature which can be implemented within this network architecture. Especially with respect to large data volumes the transfer completion time is often more important than the entire transfer time. Knowing this point in time allows more efficient resource planning which can result in reduction of costs. This functionality is enabled by allowing reservations of network resources for specific periods of time. The reservations for specific flows are managed by the Broker, which has a global view on the entire domain. Thereby, overcommitment can be avoided and start and end points of the data transfer can be guaranteed.

All described features are necessary to transfer the amounts of data described in Section III-A1 efficiently through a shared multi-user network. Many different algorithms and other features exist, which can be realized by this architecture. So, we encapsulate these functionalities in different modules, which can be added, replaced or removed during runtime without interruption, similar to loadable kernel modules. Thereby, every domain can optimize its control layer for its requirements as long as the interoperability is still guaranteed.

To enable the described functionality between multiple domains, an inter-domain communication is required. Those incoming and outgoing requests are also handled by the Broker and will be processed similar to intra-domain requests. Therefore, external and internal request messages may only differ in the source tag which determines the security group classification and the consequential permissions of the service requestor.

Beside the loadable kernel modules and the Broker we

 ISBN: 978-1-61208-310-0

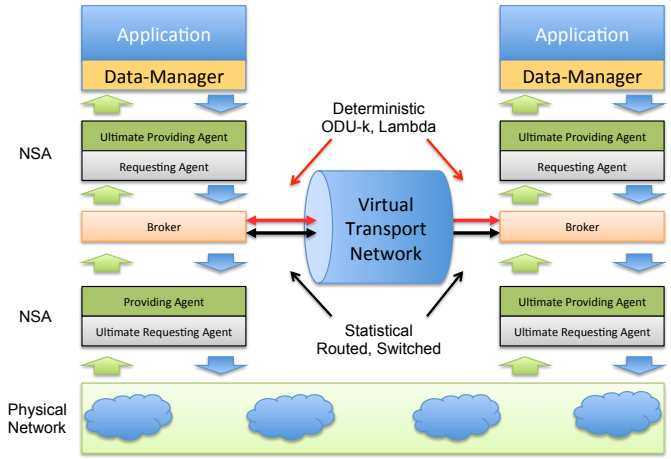

Fig. 3: Deterministic and statistically multiplexed transport

suggest to encapsulate the Controller as executive unit. Controllers implement the interface to the network infrastructure and perform requests or reservations, instructed by the Broker. Since this can result in a bottleneck depending on the number of requests and the domain size, we recommend to use more than one Controller. Thereby the separation from the NOS enables scalability by varying the number of Controllers depending on the network size and work load. An additional aspect which motivates for separation of NOS and Controller are the heterogeneous interfaces we expect to be provided by the network hardware vendors. To enable compatibility, at least one Controller for every network interface implementation has to be provided. The integration is mainly realized by the hardware vendors, similar to hardware drivers in conventional operating systems. Hence the encapsulation of the Controllers guarantees scalability and interoperability in our proposed architecture.

In summary the control layer we introduce provides required functionalities – like network virtualization, adaptive routing or real time communication – for the application layer, to enable an efficient transfer of big data volumes. The layer is highly customizable by integrating the functionality within loadable kernel modules which can be added, substituted or removed on demand. Additionally we took into account scalability and compatibility of an heterogenous infrastructure by encapsulating the Controllers as executive units.

3) Network Layer: In data networks there is a hierarchy of deterministic transport and statistical multiplexing. Deterministic transport can be utilized for client-to-client communication and as a transport layer for, e.g., routed services. The Broker instance shown in Figure 3 arbitrates between the requirements of multiple applications and available network services. Based on requirements communicated by the Network Service Agent (NSA), it will decide if the requested capacity will be provided on a deterministic or routed path.

Multi domain networks suffer from a lack of homogeneity. This in turn requires abstraction that allows for a unified network description language. The Network Description Language (NDL), introduced in [12], is a modular set of schemata. The topology schema describes devices and interactions between them on a single layer. The layer schema takes into account the existence of multiple layers and interactions between these layers. Capabilities of network devices are described in the capability schema and domain schemata have to deal with different domains and in consequence with administrative entities and services linked to these entities. Finally the physical schema describes the physical aspects of network elements. This set of schemata defines the ontology of network functionality.

TABLE II: SERVICE INTERFACE PRIMITIVES

Primitive	*Description*
RESERVE	The **requesting agent** (RA) requests the **providing agent** (PA) to reserve network resources
PROVISION	The RA requests the PA to provision network resources according to the previous reserve request. Depending on actually available resources the provision request may differ from the reserve request.
RELEASE	The RA requests the PA to de-provision resources without removing the reservation
ACTIVATE	The RA requests the PA to activate provisioned resources
TERMINATE	The RA request the PA to release provisioned resources and terminate the reservation
FORCED END	PA notifies RA that a reservation has been terminated
QUERY	Can be used as a status polling mechanism between RA and PA

Since most applications rely on resources from different domains, information about services and capabilities of these domains will have to be interpreted and coordinated. An application and its related data management is attached to a single domain. All information from external domains should be gathered here and communicated to the data manager to enable negotiation.

B. Communication Interfaces

Interoperability between the layers introduced in Section III-A requires information exchange. Therefore, interfaces have to be defined, which enable communication in both directions. To achieve compatibility, open interface standards are preferred. The following sections describe general requirements for service and network interfaces.

1) Open Service Interface (OSI): The northbound interface of the control layer communicates with the application layer, the southbound interface with the network layer. Since there is a multitude of network domains, horizontal communication is mandatory to enable federated network services based on a virtual multi domain network. Therefore both, application and control layer, implement embedded Network Service Agents (NSA) which are connected by a service interface. The application NSA is called requesting, the control layer NSA providing agent. Multiple services can be handled by a single NSA, in fact, as many as there are available on the end to end infrastructure. The requesting agent communicates only with the local NOS, information from other network domains is gathered and provided by the remote home domain NOS.

Because the NSA has no authority about local or remote resources, any kind of resource management is realized by the NOS in conjunction with the controller. Flexibility regarding to the introduction of new network services is enabled by the modularity of the OSI and NSA concept.

The OSI connection protocol communicates requirements to the providing agent and consists of 6 primitives, listed in Table II. These requirements have to be mapped on the corresponding QoS properties – sustained bandwidth, latency

 ISBN: 978-1-61208-310-0

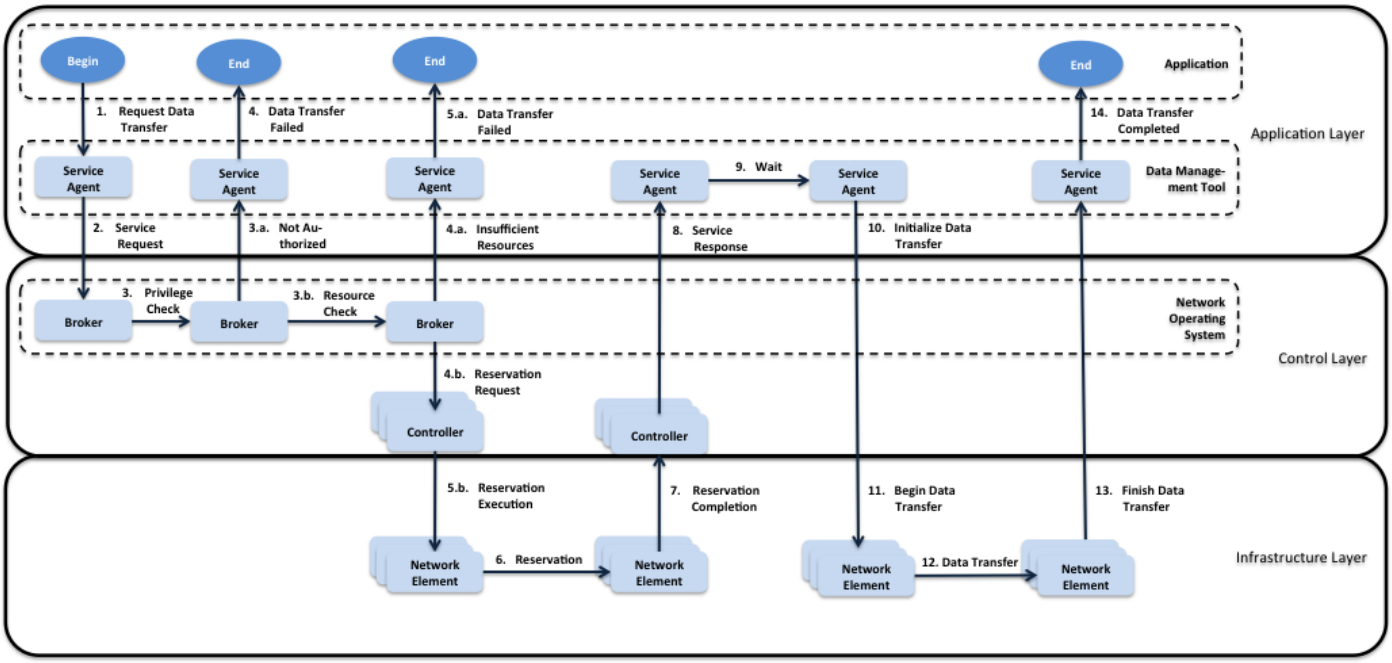

Fig. 4: Communication process within an integrated SDN architecture

and maximum latency variation. Furthermore the dedicated instance of time a certain transmission should start is communicated. The providing agent either answers with a complete confirmation or starts negotiating with the requesting agent. Once a service is confirmed there will be no further negotiations or limitations.

2) Open Network Interface (ONI): Current network elements implement control and forwarding plane on the same closed platform. Decoupling this control functionalities from the infrastructure, requires a protocol to exchange information between these two layers. This section describes the functions, required to implement the features described in Section III-A2.

An efficient placement of data flows requires a global view on the underlaying infrastructure. Therefore the position of all network elements within a domain and their connection between themselves has to be announced to the control layer. This can be implemented by an initialization message during the startup of a network element and a link discovery protocol like LLDP [2]. Once a network element and its connections is known, state changes are noticed implicitly as soon as a data flow can not be routed anymore. In this case the link is removed from the topology graph until the node is back and sends the initialization message.

Once the underlaying network topology is identified, link characteristics like bandwidth, latency or cost have to be communicated to the control layer during the initialization phase. These mostly static properties are stored together with source and destination of a link and are used to build a weighted graph for data transfers if requested.

Next to these properties, there are more varying link state informations like utilization, message rate or number of flows. Updating these values on every change would cause an immense overhead. Therefore these informations are requested periodically by the Controller and only reported to the NOS as soon as values exceed predefined thresholds. The controller can request these informations explicitly by a message, or implicitly as soon as a data transfer is completed.

Additionally to the upward directed information flow the ONI has to implement the reservation requests from the Control Layer to the network elements. These reservation requests can be combined with a period of time during which they are valid. If the reservation observance can be handled by the network elements only, the requests have to be transmitted. If not, the control layer has to add the reservation at the beginning and remove it at the end. This causes more overhead, but leaves the control function within the dedicated layer.

As described, the main objective of the ONI is to provide information about the infrastructure for the control layer to enable efficient traffic engineering. To reduce the emerging overhead, information should be updated implicitly on occurring events which already require a communication. Additionally the executive commands instructed by the control layer have to be transmitted to the network elements by the ONI.

C. Communication process

Specified requirements for data transfers can not be satisfied in any case, e.g., if the request exceeds available capacities. To ensure a data transfer anyway and independent from the current utilization, we recommend to partition the available capacity. One part for best-effort transfers, the other one for optimized SDN communications. This way, rejected data transfer requests can use conventional communication protocols. Also for small data sets the best-effort transfer might be the better path. Since the conventional best-effort communication is known, we confine the description in this section to the optimized SDN communication.

Figure 4 depicts the chronological sequence of a demand-oriented communication within the introduced SDN architecture. Thereby the application communicates its requirements

 ISBN: 978-1-61208-310-0

to the data management tool first. The integrated service agent determines the network services which are required to satisfy the request. Subsequently the agent can apply for these services by forwarding the request to the Broker of the Control Layer.

As described in section III-A2 the Broker verifies incoming requests. If the requestor is not authorized to use these services or if there are not enough resources to fulfill the request, the transfer fails and the application is informed by the service agent. At this point, a new request with different requirements can be initiated. This process can be repeated until both sides accept the conditions. The negotiation phase can also be implemented transparent to applications within the data management tool. This way the application defines tolerable ranges for the requested network parameters instead of single values. If both sides can not agree on a parameter set, the application has to transfer the data by using the conventional best-effort path.

If the request is valid the Broker initializes the reservation process and instructs all required Controllers to distribute the reservation to all participating network elements. Once all reservation confirmations arrived at the Controller, the Service Agent can be informed about the conditions of the requested transfer. At the communicated start point the data transfer can be initialized and accomplished. From the application's point of view, the following transfer does not differ from the conventional communication process, except that the infrastructure behaves like negotiated in the initialization phase.

As Figure 4 and the description of the communication process show, the overhead increases due to the initialization phase. Therefore the optimized data path is only recommended for elephant flows, where the transfer time is much higher than the startup time. In this case the overhead to define an optimized environment is worthwhile. However, small flows may still use the conventional data path.

IV. Conclusion

Our integrated SDN architecture enables concurrent applications competing for network resources, to define virtual networks that satisfy their respective requirements providing efficient network usage and reliable data transfers. We introduced the elements necessary for an end-to-end negotiation of network resources between multiple domains and without any limitation to specific protocols. The authors of this paper already introduced a first SDN prototype on the ISC'13 [18], based on a 400 Gbits demonstrator between the *Center for Information Services and High Performance Computing* (ZIH) in Dresden and the *Rechenzentrum Garching* (RZG).

References

[1] ITU-T Recommendation G.805: Generic functional architecture of transport networks. Technical report, International Telecommunication Union, Mar. 2000.

[2] IEEE Standard for Local and Metropolitan Area Networks– Station and Media Access Control Connectivity Discovery. *IEEE Std 802.1AB-2009 (Revision of IEEE Std 802.1AB-2005)*, pages 1–204, 2005.

[3] CLIVAR Exchanges - Special Issue: WCRP Coupled Model Intercomparison Project - Phase 5 - CMIP5. Project report, May 2011.

[4] Alcatel-Lucent. Alcatel-lucent upgrades cable system linking japan and california, January 21 2013. Retrieved April 26th, 2013, from http://www3.alcatel-lucent.com.

[5] Big Switch Networks. The Open SDN Architecture. Technical report, Big Switch Networks, Inc., 2012.

[6] N. Blefari-Melazzi, A. Detti, G. Morabito, S. Salsano, and L. Veltri. Information centric networking over sdn and openflow: Architectural aspects and experiments on the ofelia testbed. *CoRR*, abs/1301.5933, 2013.

[7] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian. Fabric: a retrospective on evolving sdn. In *Proceedings of the first workshop on Hot topics in software defined networks*, HotSDN '12, pages 85–90, New York, NY, USA, 2012. ACM.

[8] O. M. E. Committee. Software-Defined Networking: The New Norm for Networks. Technical report, Open Networking Foundation, April 2012.

[9] Energy Sciences Network ESnet. BER Science Network Requirements; Report of the Biological and Environmental Research. LBNL report LBNL-4089E, Network Requirements Workshop, April 29-30 2010.

[10] Energy Sciences Network ESnet. ESnet Partners with North American, European Research Networks in Pilot to Create First 100 Gbps Research Link Across Atlantic, April 24 2013. Retrieved April 26th, 2013, from http://esnetupdates.wordpress.com.

[11] C. Grimme and A. Papaspyrou. Cooperative negotiation and scheduling of scientific workflows in the collaborative climate community data and processing grid. *Future Generation Computer Systems*, 25:301–307, 2009. Publication status: Published.

[12] P. Grosso, A. Brown, A. Cedeyn, F. Dijkstra, J. van der Ham, A. Patil, P. Primet, M. Swany, and J. Zurawski. Network topology descriptions in hybrid networks, March 2010.

[13] N. Hemsoth. 20 lessons enterprise cios can learn from supercomputing. *datanami*, November 2012. http://www.datanami.com/datanami/2012-11-12/20_lessons_enterprise_big_data_buffs_can_learn_from_supercomputing.html.

[14] P. Kaufmann. *Gesamtdarstellung des VIOLA-Projektes (Vertically integrated optical testbed for large applications in DFN)*. DFN-Verein, 2007.

[15] M. Labedzki, C. Mazurek, A. Patil, and M. Wolski. common network information service - modelling and interacting with a real life network, 2009.

[16] E. Mannie. Generalized Multi-Protocol Label Switching (GMPLS) Architecture. RFC 3945 (Proposed Standard), October 2004.

[17] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Mar. 2008.

[18] R. Budich, A. Georgi, J. Müller and R. Sperber. International Supercomputing Conference 2013, June 2013.

[19] J. Raciazek. *synchro-data script bundle*. Technical documentation. http://dods.ipsl.jussieu.fr/jripsl/synchro_data.

[20] B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, and S. Shenker. Software-defined internet architecture: decoupling architecture from infrastructure. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, HotNets-XI, pages 43–48, New York, NY, USA, 2012. ACM.

[21] G. Roberts, T. Kudoh, I. Monga, J. Sobieski, and J. MacAuley. NSI Connection Service Protocol v1.1, 2012.

[22] Top500. Top 500 Supercomputer Sites. http://www.top500.org/, 2013.

 ISBN: 978-1-61208-310-0