An Integrated SDN Architecture for Application Driven Networking

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

The amount of data to be handled by networks and associated resources across industry, universities and supercomputing centers for research, education, commerce and the internet business at large, grew significantly over the past few years. Furthermore, international collaboration became standard. Federation techniques implement a consolidated view on distributed data for end users. On the other hand, hybrid network architectures, multi-vendor environments and heterogeneous infrastructures steadily increase the complexity of data mining, computing and networking. Software Defined Networking (SDN) is a promising solution for the reduction of complexity: It opens the control layer allowing for direct programmability. Our target – first introduced in 2013 for geographically, dispersed datasets [1] – is to provide an automated arbitration layer between applications and network, thus reducing the operational complexity within heterogeneous environments. Furthermore, network resources can be distributed in a more efficient way by offering an open network interface for the application layer, which can be used to specify user requirements even beyond mere networking. Additionally, a central management provides a global view on the underlying infrastructure and enables the optimization of demands and available resources.

In this extended journal 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. Therefore, Section II gives an introduction into SDN, followed by SDN use cases in Section III. In Section IV some approaches providing automated configuration of network services are discussed and differentiated from our approach. Our architecture as well as an automated network configuration process arbitrating between the concurrent applications competing for network resources is described in Section V. Section VI describes a feasible migration process from existing network architectures to an application driven network architecture step by step. Finally, we present our results from a first prototype in a 400-Gigabit/s-Testbed in Section VII and summarize our approach in Section VIII.

II. Software Defined Networking

Today, network intelligence and state are inherent part of network devices and distributed among the entire infrastructure. Decisions can only be made based on local information, which often results in an inefficient distribution of global resources. Implementing network-wide policies can increase the efficiency of resource distribution, but therefore, all participating devices have to be configured. This results in long delays and implies additional expenditure and can also lead to inconsistencies, security breaches or non-compliance to regulations. In addition, large discrete sets of protocols are used to connect hosts over arbitrary distances, link speeds and topologies. Most of these protocols tend to be defined in isolation, without any abstraction layer. This leads to a more and more increasing complexity and as a result, to static networks in a dynamic IT environment.

Based on this heterogeneous, complex and static infrastructure, applications and network services with different requirements try to utilize the network at the same time. In most cases these resources are offered as a best effort service, which means they are distributed between the streams, depending

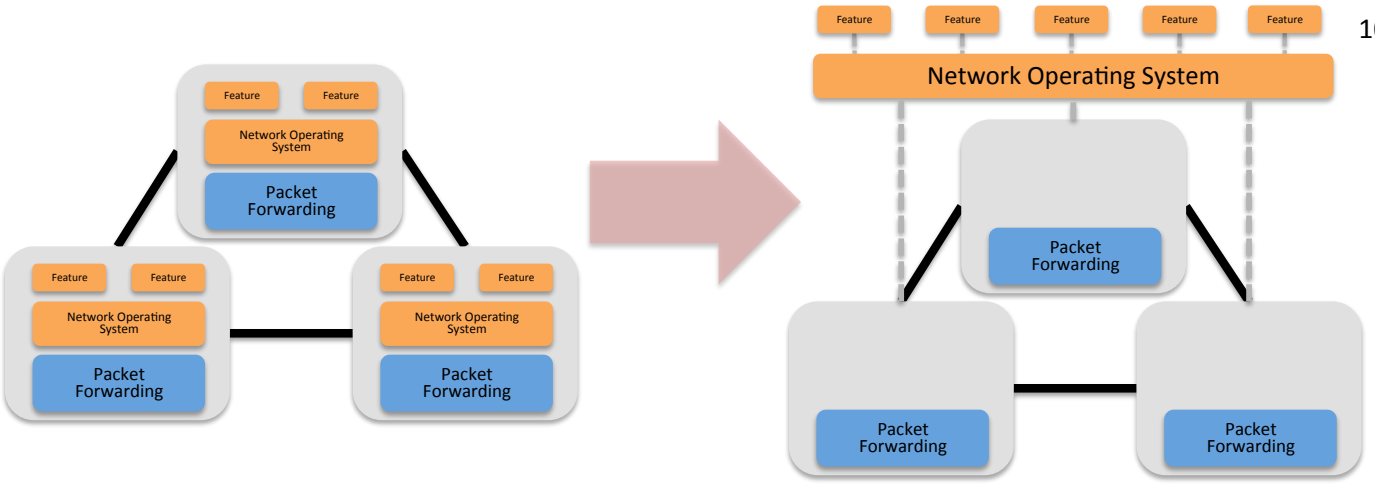

Figure 1: Conventional vs. SDN driven network, full separation of forwarding and control plane

on the current load. Because of missing information at the application layer, users do not know about the optimal point in time to start a data transfer. On the other hand, there is also no mechanism available which allows users to announce their requirements, so that the infrastructure can plan the upcoming traffic in advance. To enable an efficient resource management and reduce the complexity within networks, an open and programmable control layer is necessary, which interacts with users to manage their requirements, and with the underlying infrastructure, to map incoming requests to the available network devices.

Software Defined Networking (SDN) is an upcoming trend, promising the reduction of complexity: it opens the control layer and makes it directly programmable. SDN enabled networks provide an automated arbitration layer between applications and network and in consequence reduce the operational complexity within heterogeneous environments. Furthermore, network resources can be distributed in a more efficient way by offering an Open Network Interface for the Application Layer which can be used to specify user requirements. Additionally, a central management provides a global view on the underlying infrastructure and enables the optimization of demands and available resources. A comparison between the traditional network architecture and an SDN-enabled network is depicted in Figure 1.

III. Use Cases for SDN

In this section we introduce two applications, one from the climate community and one from high energy physics, both of which provide a use case for our architecture. So far if viewed separately. These two applications using the same SDN network simultaneously form a third use case.

Common buzz words of today's IT landscape are "Data Tsunami" or "Big Data". Whether countless small data packages have to be moved with minimal latency and highest safety, or humongous volumes of data have to be reliably moved from one place to another: The activities in data management rely on fast, broadband, reliable networks. At the moment, intercontinental network speeds are limited to 40 Gb/s [2]. The project *Advanced North Atlantic 100G Pilot* (ANA100G) tries to reach the 100 Gb/s barrier [3]. But practical experiences show, 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". Consequently, every "Big Data" application is a use case.

A practical example for these facts is the international *Coupled Model Inter-comparison Project* (CMIP) [4], which conducts sets of co-ordinated experiments with numerical climate models to compare them against each other. The project recently completed its 5th edition (CMIP5) [5], Such comparisons of climate models serve as 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 are complex numerical realisations of the physical, chemical, biological and other processes that play a role in the climate system. They regularly utilise to a high percentage high performance computers like those to be found in the TOP500 list [6]. 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. [7]). CMIP5 produced a sum of about 100 PB world-wide [8], produced in about 30 centres [9]. 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, e.g., see the results of the German C3-Grid initiative [10].

The climate modelling community agreed upon sub-setting the data to a volume of about 1.5 PB, containing only those

Table I: TIME TO TRANSPORT 1 TB AND 1 PB OF DATA FROM ONE CLIMATE CENTER TO ANOTHER [11]

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

Table II: CMIP6 CENTRES

Type	*EU*	*America*	*Asia*	*Australia*
PRODUCING	10	6	6	1
PROVIDING	2	3	3	1
DOWNLOADING	100s	10s	100s	10s

data most relevant for the comparison, and to make these data available in five centers world-wide for easier access and replication [4]. But, as experience shows, the process to do so took much more effort and time than expected, and stressed the scientific community considerably, leaving less time to creative scientific work that potentially would benefit the scientific value of the assessment report. Apart from a lot of hassle in the upper layers (co-ordination, federation, meta-data-systems, applications, formats) it was obvious that the network posed a crucial problem here. Not only is it error prone and unreliable, but just simply much too slow in many instances. 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 [3]. The fact that many network lines outward bound from centers seem rather underutilised does not contradict this observation: Burst-wise utilisation is commonplace, 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 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 (see Table II). More participating centres in Asia and South America will put higher demands on the network architecture in terms of geographical coverage and network quality. 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 script bundle called synchro-data [12] 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 cannot download from the same machine at the same time. Again: many networks seem to be underutilised, but not because the scientists do not 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 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 Computational Fluid Dynamics (CFD) and Computational Structural Mechanics (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.

A recurrent task here is the simulation of flow response to different Eigenmodes and phase angle combinations. An initial steady state CFD simulation of, e.g., a turbine runner blade passage is done to obtain boundary conditions for a subsequent CSM simulation, which results in the m Eigenmodes of the respective blade. Now for each Eigenmode a specific set of n relevant phase angles is identified. In the next step corresponding displacements are applied to the blade mesh resulting in $n \times m$ different CFD simulation setups that have to be solved. These simulations run for a fixed iteration count after which convergence analysis is performed. Based on the result of this analysis for each job a decision is made whether they need to run for further iterations or not.

Single simulation jobs hereby typically run on 32-64 cores and produce result files in the range of 100 MB written at the end of the simulation. Considering a real world setup with $n \times m = 300$ leads to a relatively moderate data volume of 30 GB. Ideally all jobs can be run at the same time, thus requiring $300 \times 64 = 19.200$ cores.

Therefore, the data replication mechanisms of the General Parallel File System (GPFS) are applied, to simultaneously replicate data to all clustered resources whenever write access to the filesystem occurs.

Now, looking at the ideal but none the less likely situation where 300 simulation jobs start at the same time and all writing their results within a time frame off 15 minutes quickly leads to peak bandwidth requirements up to 400 GB. In the case of less regularity in the workflow, where potentially all jobs are started at different points in time, write access might occur

over a time frame that corresponds to the duration of the overall workflow. In this case a much lower but sustained bandwidth can be observed.

To enable an application adapted virtual network configuration the respective applications need to have an interface communicating their requirements with regard to the available network resources.

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.

IV. Related work

Approaches to provide automated configuration of network services already exist for some time. Protocols for traffic engineering were specified to enable dedicated resource allocation to different applications. Most of these solutions were limited to a single carrier domain, in many cases to a single equipment vendor. Vertical interaction was achieved, i.e., communication between different layers, is well defined. Automation of configuration across domain borders was not considered. All control interaction had its origin in operator actions. The applications itself did not have any direct influence. In this section, we give an overview over the development starting with an information model and first attempts of implementation.

Within the ITU-T recommendation G.805 03/2000 [13] the authors abstract from the actual network elements. The recommendation describes a set of functional elements instead and the relationship between these elements. It is a generic multi layer information model, which is open to any kind of implementation. The notion of a layer in G.805 does not necessarily coincide with one layer of the OSI model. A layer can best be described as a set of all connection points of the same type, i.e., sources and sinks of data that can communicate without adaptation. Adaptation allows for communication between the layers. The information model developed in G.805 is the basis for any multi-layer interaction model of the future, the actual communication processes, both vertical and horizontal can be described based on this agnostic approach. Networks described in G.805 are connection oriented whereas the follow-up, ITU-T recommendation G.809 03/2003 [14], describes connectionless networks. However, both have in common an implementation agnostic information model. Communication between layers in both recommendations is enabled by adaptation functions.

Generalized Multi-Protocol Label Switching (GMPLS) [15] is a generalization of IP/MPLS for the connection oriented transport layer. It was originally defined to provide a control plane for Synchronous Digital Hierarchy (SDH), Optical Transport Hierarchy (OTH) and Wavelength-Division Multiplexing (WDM). The network elements are equipped with a GMPLS Routing Engine (GMRE) which made dynamic configuration and automated restoration possible. In a first iteration there was no interaction with layers above transport. With the definition of a northbound User Network Interface (UNI), communication with higher layers was enabled. In consequence, there were means to adapt transport bandwidth to the requirements of upper layers. Requirements from applications are not automatically considered. They still rely on operator intervention. Generalized MPLS is a method to do multi layer provisioning and traffic engineering, but it is normally restricted to one carrier and often to a single-vendor domain. Hence, it is sufficient for vertical integration but not for horizontal configuration purposes. Furthermore, flapping due to changing IP address spaces can be a problem. In the course of the VIOLA project [16] 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.

In 2008 a Multi-layer Network Model based on the ITU-T recommendation G.805 was published by Freek Dijkstra et al. [17], containing a proposal for a multiple layer control interaction model. Making use of the functional elements defined in G.805 it is possible to implement a multi layer data model. From GMPLS the technique of label switching was taken. The translation of client or application requirements still remains with the operator.

The common Network Information Service Schema Specification (cNIS) activities [18] 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.

The ITU-T recommendations G.805 and G.809 as well as the multi-layer network model from Dijkstra abstract 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, enabling fully automated network resource management. Furthermore, this interface should not only address network resources but should take into account other virtualized functions. Our integrated approach will address both, network and other resources as building blocks of a final functional graph.

V. Integrated SDN Architecture

To provide application-oriented network services, we suggest an architecture consisting of three layers, depicted in Figure 2. 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

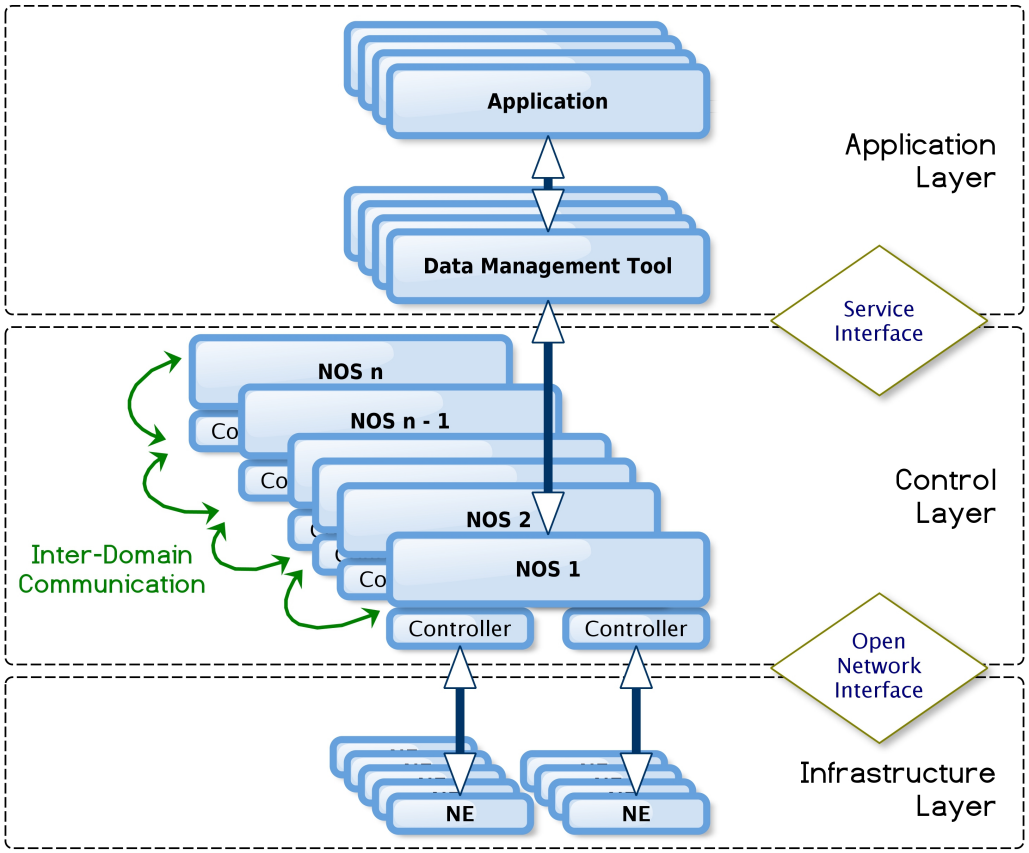

Figure 2: Integrated multi layer, multi domain SDN architecture

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.

1) Application Layer: The requirements of both applications and predefined services are not just network resources. We can think of storage, compute capacity and additional functionality related to the network. This of course makes the general model more complex to construct, but it takes strain from both, user and carrier. The main feature of this fully integrated model will be access to a building block repository [19]. Here we have infrastructure building blocks and functional building blocks. Infrastructure building blocks on the one hand, are network segments, covering different layers and different domains. Functional building blocks on the other hand, represent network services, e.g., encryption, compression or acceleration. An application link between Lawrence Livermore National Laboratory and German Data Centre for Climate Research involves for example multiple layers and multiple network domains. The application queries for the link and the extended SDN-enabled network protocol combines the required infrastructure building blocks to form a virtual network. Furthermore, the application requires additional services, like WAN acceleration, storage and a tool for ensuring data integrity. Depending on availability, the appropriate building blocks are added to the network graph. A real-time multi-site application like TV production involves multiple layers and possibly multiple domains. It requires a highly elastic network configuration, extremely low latency and high peak bandwidth. Data integrity must be guaranteed and synchronization must be provided. WAN acceleration would impose too much latency for a real time life production. Selected building blocks would again be network segments to form a virtual network as required, plus a tool to guarantee data integrity and synchronization functionality. Genome Sequencing to support surgeons requires high bandwidth for medium periods on short notice. While handling medical data a high level of privacy must be maintained. Again we have infrastructure building blocks in form of network segments

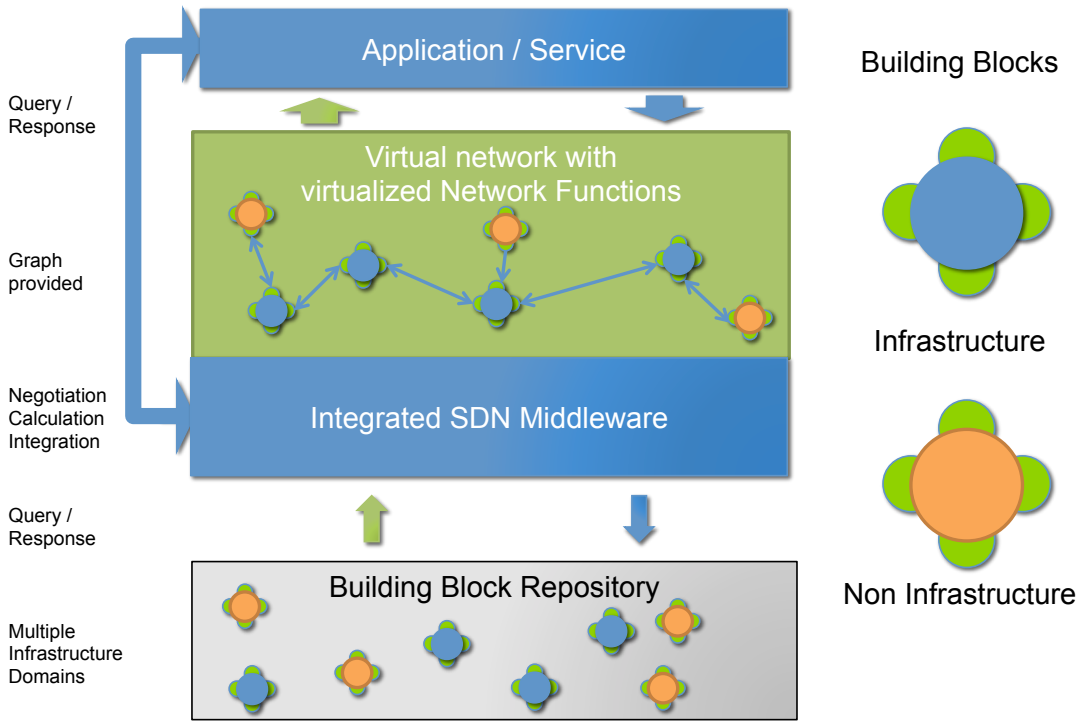

Figure 3: Functional model of an application-driven network architecture

plus functions to guarantee privacy, e.g., encryption and a Public-Key Infrastructure (PKI). The public Internet today is a major medium of social interaction and it is of utmost importance for an open society to guarantee equal rights and secure access to its resources. The provider would have to select suitable network segments to commission an Internet platform plus additional building blocks to guarantee privacy and security for the individual user.

Figure 3 shows our functional building block model to realize the work flow for the described application scenarios. Applications communicate their certain requirements towards the network-building stack. This in turn checks, if available resources satisfy these requirements and answers with a proposal of a combined graph.

2) Control Layer: Our control layer, depicted in Figure 4, 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 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.

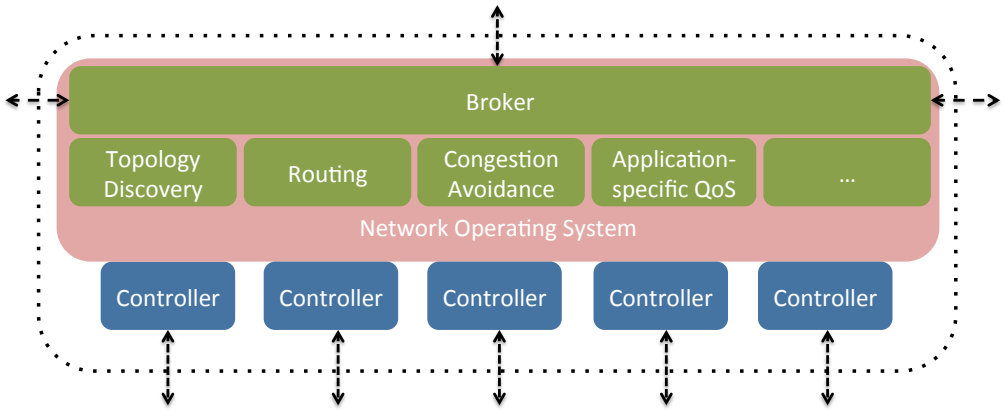

Figure 4: Single domain control layer overview

All described features are necessary to transfer the amounts of data described in Section V-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 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 5 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 [20], 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.
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 V-A requires information exchange. Therefore, interfaces

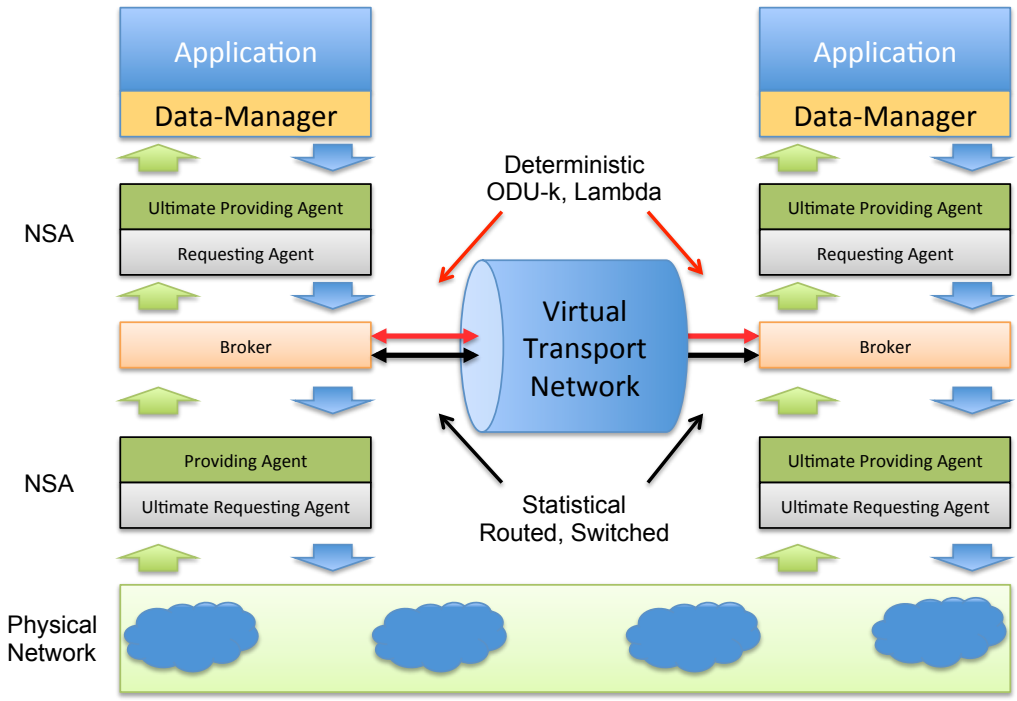

Figure 5: Deterministic and statistically multiplexed transport

Table III: 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

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 III. These requirements have to be mapped on the corresponding QoS properties – sustained bandwidth, latency 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 V-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.

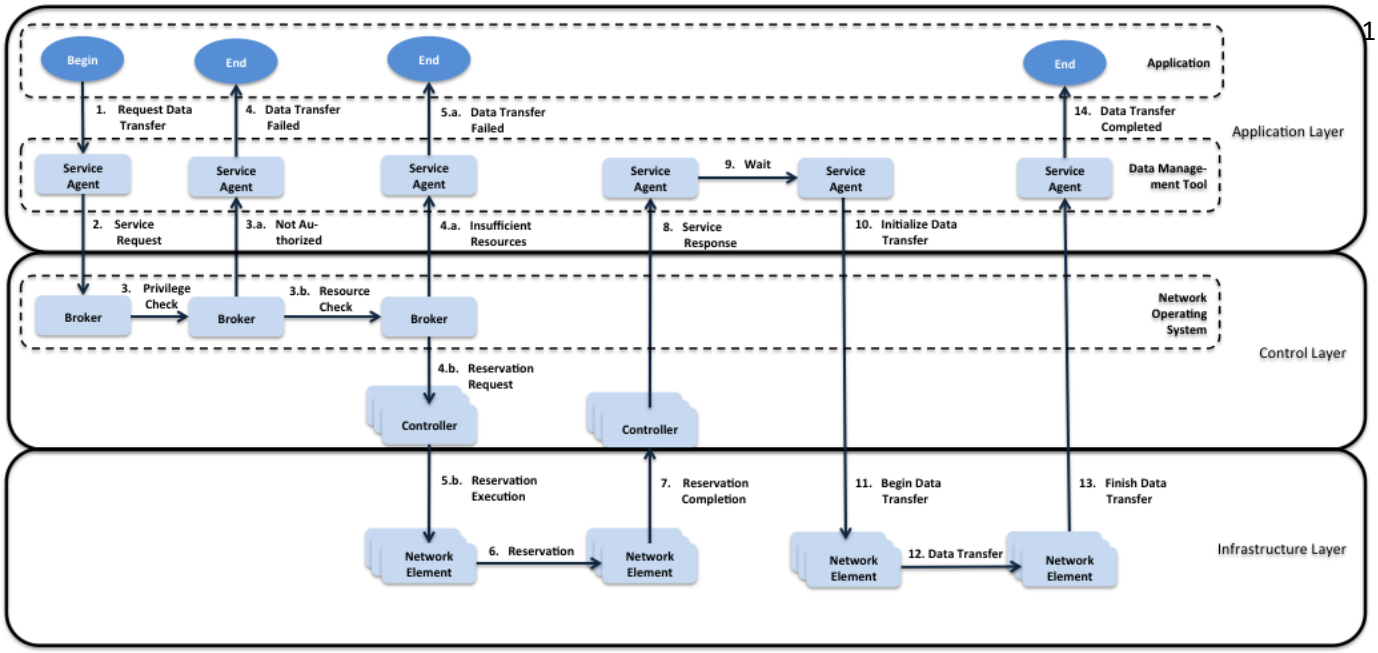

Figure 6: Integrated SDN communication process negotiating for network resources

This can be implemented by an initialization message during the startup of a network element and a link discovery protocol like LLDP [21]. 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 6 depicts the chronological sequence of a demand-oriented communication within the introduced SDN architecture. Thereby, the application communicates its requirements 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 V-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 6 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.

VI. Migration from Existing Architectures

Migration towards an application driven network configuration has to be done in a stepwise approach. In a first step, the network elements have to be enabled to support a common controller language. For network elements in use today, there will have to be a translation overlay. Controller-input at that stage will not be automated and it will be per domain or even per network element group.

In a second step, the Network Operating System (NOS) has to be defined for providing input to the controllers. First, only single domain interworking and bidirectional communication between domain NOS and domain controller will be supported. Based this horizontal integration of involved NOS can be implemented for ensuring interoperability between multiple domains.

The next milestone is to enable applications to communicate with the respective NOS. Thereby, the user has to know about the requirements and communicates them directly to the NOS. Later, a fully automated negotiation process can be initiated by the application.

In the final step, the NOS will be enabled to request and integrate required functional building blocks, additional to the requested network resources.

VII. Bandwidth-On-Demand Prototype

Global data traffic increases steadily by developments in Cloud Computing, Social Media and Big Data applications. According to projections, 2015 the core infrastructure of the Internet has to handle four times as much data than 2010 [22]. Therefore, extremely high bandwidth data networks are required, which was the reason for the two testbeds in Germany, described in the following.

That federated applications and services can profit from increasing the bandwidth was already proven within the 100-Gigabit-Testbed on a 60 km Wide Area Network between the *Center for Information Services and High Performance Computing* (ZIH) of the Technical University Dresden and the computing center of the *Technical University Bergakademie Freiberg*, started in 2010 [23].

A follow-up testbed which also introduced our first SDN prototype was presented on the ISC'13 [24], based on a 400-Gigabit/s-Demonstrator between the *Center for Information Services and High Performance Computing* (ZIH) in Dresden and the *Rechenzentrum Garching* (RZG). For the next generation 400-Gigabit/s-Ethernet technology not only the bandwidth but also the distance was increased. Therefore, the HPC centers in Dresden and Garching (Munich) were connected by a 640 km dark fiber, provided by Deutsche Telekom, combined with access and transport technology from Alcatel-Lucent. Cluster systems from Bull in Dresden and IBM in Garching were used to set up a General Parallel File System (GPFS) to give applications a consolidated view on distributed data. To achieve the necessary I/O throughput the cluster nodes contained high-speed PCIe RealSSD flash cards from EMC[2] with 3.2 GByte/s sequential read and 1.9 GByte/s sequential write performance. With three of these cards per server we achieved a theoretical peak read/write performance of 921.6/547.2 Gbit/s in total, which was sufficient to saturate the 400-Gigabit/s-link. The bandwidth to the WAN was ensured by 40-Gigabit/s-Ethernet cards from Mellanox, which were directly connected to the service router from Alcatel-Lucent. Additionally, the HPC clusters in Dresden and Garching had to be integrated into the testbed. Therefore, FDR InfiniBand cards from Mellanox were deployed. So the clusters on both sides were only used to direct the traffic from the location where the data was stored, to the computing nodes in the HPC centers. The entire architecture of the 400-Gigabit/s-Testbed is also shown in Figure 7.

Within the 400-Gbit/s-Testbed we have demonstrated the impact of different applications utilizing the same infrastructure, with and without interoperability between application and network infrastructure. On one hand, the turbine simulation, described in Section III, requires a parallel file system and distributed calculation. Therefore, low latency and high bandwidth elasticity are required to achieve a high performance application layer. On the other hand, the climate application scenario, also described in Section III, requires the transfer of huge geographically dispersed datasets for intercomparison. Consequently, a very high sustained bandwidth is required, to guarantee a reliable and predictable data transfer. To specify the different demands of these applications, we implemented a web-frontend, which forwarded incoming reservation requests to a centralized management system. This system evaluated incoming requests and instructed the controllers in Dresden and Garching to configure all participating network elements. Because there was no mechanism available to remove the

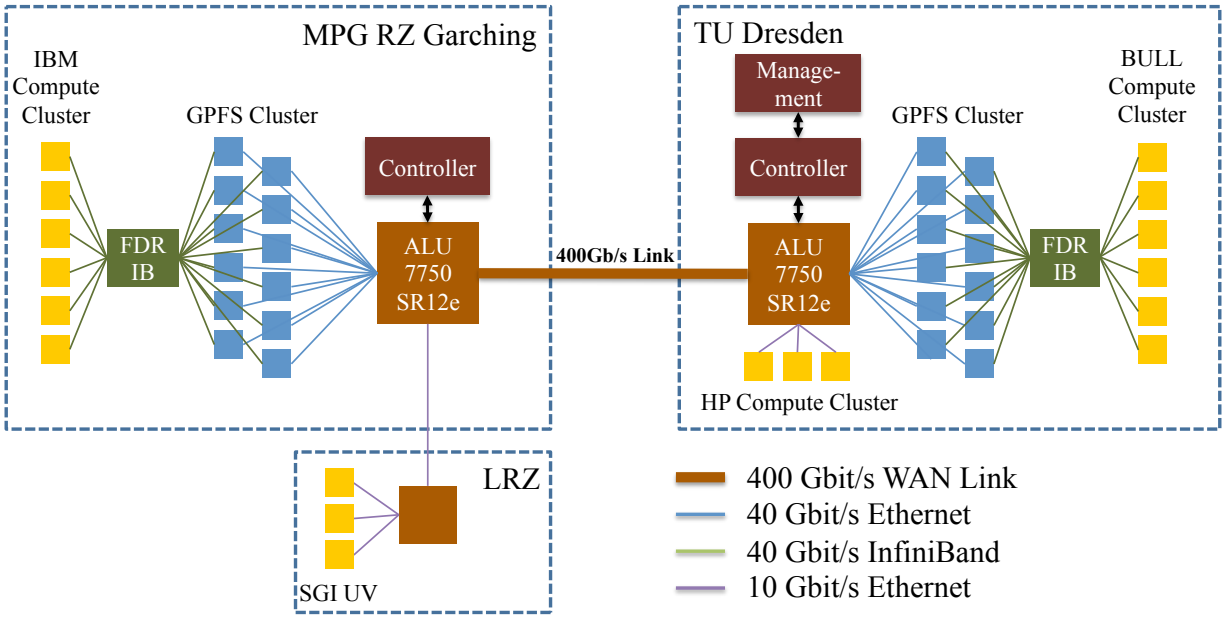

Figure 7: Overview 400-Gigabit/s-Testbed architecture

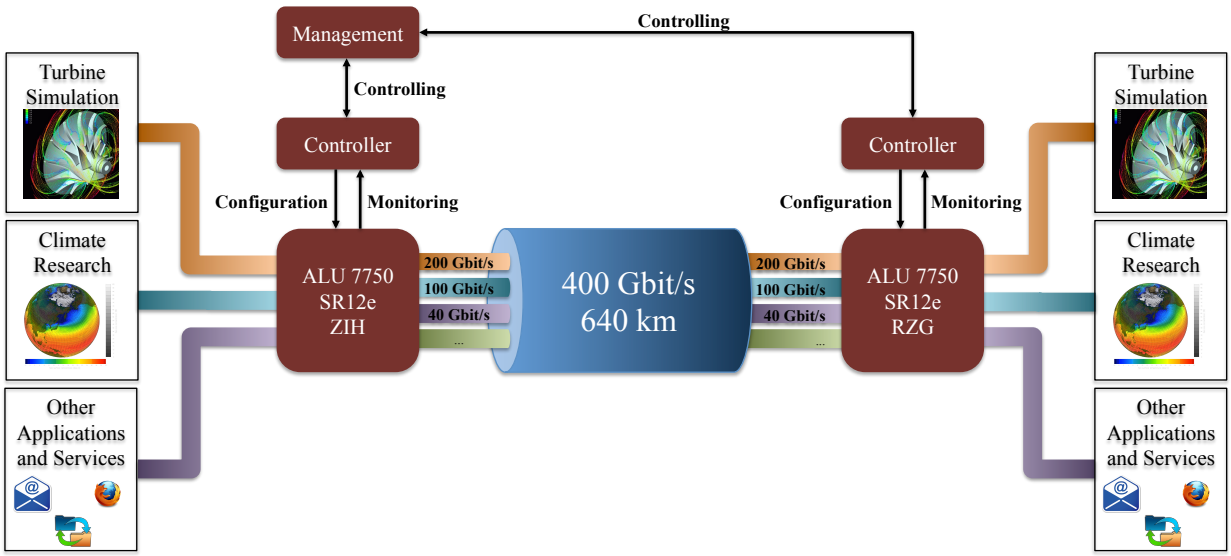

Figure 8: Setup of the Bandwidth-on-Demand Demonstrator

reservations automatically after expiration, the centralized management system also included a time table and scheduler for all reservations. Based on the entries in this time table, reservations were declined if conflicts occurred and removed after expiration. Figure 8 shows the entire setup of the Bandwidth-on-Demand demonstrator.

In this testbed, we were able to demonstrate that all applications can benefit from this new control layer, providing an interface between application and infrastructure layer. Especially the predictability of long term data transfers was increased tremendously, independent of the concurrent traffic on the link. Also, providing dedicated bandwidth for the turbine simulation increased the performance. Unfortunately, the latency could not be influenced, due to topology limitations. So, there is still potential for more optimization.

VIII. 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.

On the top the application layer represents the users view on this network architecture. A southbound Network Service

Agent (NSA) requesting resources from the underlying control layer. Communication between the NSAs is realized by the Open Service Interface (OSI). The providing NSA in turn is handing over the request to the Network Operating System (NOS), which links the network layer of its own domain with NOS's of other domains. The NOS has a centralized view on the network resources available and outstanding requests from applications, so it is able to arbitrate between them. Scalability and compatibility is enabled by using different Controllers, depending on the work load and the underlaying infrastructure. Thereby, our architecture supports end-to-end negotiation of network resources between multiple domains and without limitation to a specific protocol.

Additional to the architecture description we show up a feasible approach to migrate from existing traditional network architectures to an application driven network architecture. Furthermore, we were able to demonstrate the benefits of our approach for applications within a 400-Gigabit/s-demonstrator, connecting two High Performance Computing centers in Germany, by a prototypical implementation.

REFERENCES

[1] A. Georgi, R. Budich, Y. Meeres, R. Sperber, and H. Hérenger, "An Integrated SDN Architecture for Applications Relying on Huge, Geographically Dispersed Datasets," ser. INFOCOMP. IARIA, 2013, pp. 129–134.

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

[3] 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.

[4] "CLIVAR Exchanges - Special Issue: WCRP Coupled Model Intercomparison Project - Phase 5 - CMIP5," Project Report, May 2011. [Online]. Available: http://eprints.soton.ac.uk/194679/

[5] "CMIP5 Coupled Model Intercomparison Project," retrieved May 26th, 2014, from http://cmip-pcmdi.llnl.gov/cmip5.

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

[7] 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.

[8] "Cmip5Status," retrieved May 26th, 2014, from https://github.com/ESGF/esgf.github.io/wiki/Cmip5Status.

[9] "Modeling Groups and their Terms of Use," retrieved May 26th, 2014, from http://cmip-pcmdi.llnl.gov/cmip5/docs/CMIP5_modeling_groups.pdf.

[10] 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*, vol. 25, pp. 301–307, 2009, publication status: Published.

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

[12] J. Raciazek, "Synchro-data script bundle," Technical documentation, retrieved May 26th, 2014, from http://dods.ipsl.jussieu.fr/jripsl/synchro_data.

[13] "ITU-T Recommendation G.805: Generic functional architecture of transport networks," International Telecommunication Union, Tech. Rep., Mar. 2000. [Online]. Available: http://www.itu.int/rec/T-REC-G.805/en

[14] "ITU-T Recommendation G.809: Functional architecture of connectionless layer networks," Mar. 2003. [Online]. Available: http://www.itu.int/rec/T-REC-G.809/en

[15] E. Mannie, "Generalized Multi-Protocol Label Switching (GMPLS) Architecture," RFC 3945 (Proposed Standard), Internet Engineering Task Force, October 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3945.txt

[16] P. Kaufmann, *Gesamtdarstellung des VIOLA-Projektes (Vertically integrated optical testbed for large applications in DFN)*. DFN-Verein, 2007. [Online]. Available: http://books.google.de/books?id=9ZelPgAACAAJ

[17] F. Dijkstra, B. Andree, K. Koymans, J. van der Ham, P. Grosso, and C. de Laat, "A Multi-layer Network Model Based on ITU-T G.805," *Comput. Netw.*, vol. 52, no. 10, pp. 1927–1937, Jul. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2008.02.013

[18] M. Labedzki, C. Mazurek, A. Patil, and M. Wolski, "Common Network Information Service - modelling and interacting with a real life network," 2009.

[19] D. Schwerdel, D. Günther, M. R. Khondoker, B. Reuther, and P. Müller, "A building block interaction model for flexible future internet architectures," in *NGI*. IEEE, 2011, pp. 1–8.

[20] 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.

[21] "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)*, pp. 1–204, 2005.

[22] C. Inc., "Cisco Visual Networking Index: Forecast and Methodology, 2010–2015," Cisco Inc., Tech. Rep., June 2011.

[23] M. Kluge, S. C. Simms, T. William, R. Henschel, A. Georgi, C. Meyer, M. S. Müller, C. A. Stewart, W. Wünsch, and W. E. Nagel, "Performance and quality of service of data and video movement over a 100 gbps testbed," *Future Generation Comp. Syst.*, vol. 29, no. 1, pp. 230–240, 2013.

[24] R. Budich, A. Georgi, J. Müller and R. Sperber, "International Supercomputing Conference 2013," Leipzig, Jun. 2013.