Future Generation Computer Systems 56 (2016) 64–76

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

SDN-aware federation of distributed data

Spiros Koulouzis[a,*], Adam S.Z. Belloum[a], Marian T. Bubak[a,b], Zhiming Zhao[a], Miroslav Živković[a], Cees T.A.M. de Laat[a]

[a] University of Amsterdam, Institute for Informatics, Amsterdam, The Netherlands
[b] Department of Computer Science, AGH University of Science and Technology, Krakow, Poland

HIGHLIGHTS

- Software defined networking (SDN) has created unprecedented opportunities for efficient data transfers.
- We present an observable and controllable network model to approach data transfers from multiple sources.
- We develop and implement two algorithms that take advantage of the programmability of the network.
- We introduce an architecture for transparent & adaptive data transfers and enable full exploitation of research infrastructures.
- We assess the improvement of data transfer rates resulting from SDN-enabling Distributed File Access Services (DFAS).

ARTICLE INFO

Article history:
Received 31 March 2015
Received in revised form
30 August 2015
Accepted 26 September 2015
Available online 23 October 2015

Keywords:
Data federations
Data transfers
Software defined networking (SDN)
Distributed file access service (DFAS)
Scientific workflows

ABSTRACT

The introduction of software defined networking (SDN) has created an opportunity for file access services to get a view of the underlying network and to further optimize large data transfers. This opportunity is still unexplored while the amount of data that needs to be transferred is growing. Data transfers are also becoming more frequent as a result of interdisciplinary collaborations and the nature of research infrastructures. To address the needs for larger and more frequent data transfers, we propose an approach which enables file access services to use SDN. We extend the file access services developed in our earlier work by including network resources in the provisioning for large data transfers. A novel SDN-aware file transfer mechanism is prototyped for improving the performance and reliability of large data transfers on research infrastructure equipped with programmable network switches. Our results show that I/O and data-intensive scientific workflows benefit from SDN-aware file access services.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Nowadays, data driven approaches in scientific research have increased the frequency and size of data transfers between geographically distributed locations. The emergence of the network programmability through software defined networking (SDN) has created unprecedented opportunities to enable network provisioning for I/O and data-intensive applications. The programmability of the network may improve data transfers and make them less sensitive to network congestion. However, the opportunity to program the network should not bring extra complexity to data transfers; it should be seamless and transparent to applications that should not be exposed to data access and data transfer specifics.

The size and frequency of data transfers is increasing because of the nature of scientific collaborations and commercial applications. Scientists increasingly access, exchange and share data from different data sources [1,2]. A key factor for this is the increase of interdisciplinary and international collaborations such as environmental sciences [3], brain modeling [4] and medical applications [5,6]. Commercial applications also need access to large volumes of data to provide business intelligence [7]. This data often reside in geographically distributed resources across multiple domains spread over several research infrastructures (RIs). Most of the applications supporting these collaborations treat the underlying network as a black box.

In this paper, we will mainly focus on data transfers between RIs. RIs play an increasingly important role in the advancement of knowledge and technology. They are a key instrument to bring together stakeholders with different backgrounds to look for solutions to many of the problems society is facing today. RIs offer unique research services to users from different countries, attract

* Corresponding author.
E-mail addresses: S.Koulouzis@uva.nl (S. Koulouzis), A.S.Z.Belloum@uva.nl (A.S.Z. Belloum), bubak@agh.edu.pl (M.T. Bubak), z.zhao@uva.nl (Z. Zhao), m.zivkovic@uva.nl (M. Živković), C.T.A.M.deLaat@uva.nl (C.T.A.M. de Laat).

http://dx.doi.org/10.1016/j.future.2015.09.032
0167-739X/© 2015 Elsevier B.V. All rights reserved.

young people to science, and help to shape scientific communities. RIs are hosted in research institutes and use a combination of high-performance computing (HPC), grid and cloud and are often interconnected with high-capacity networks. RIs are accessed by the scientific community through a layer of high level services that efficiently and transparently manage the execution of complex distributed CPU and data-intensive applications. Often these applications, modeled as workflows, are composed by loosely coupled tasks or jobs which communicate through file exchange [8,9]. This file exchange is facilitated by Distributed File Access Services (DFASs) which offer a single point of entry to discover and manage files and replication to improve file availability. Despite the effort to move computation to the data, there are cases where this is simply not possible. Technical constraints such as privacy and security issues, lack of computing power or the need for specialized hardware prevent computation from reaching the data [10,11]. Typically, scientific workflows need access to datasets which are processed to generate new datasets which have to be further processed by subsequent tasks to achieve a defined goal [12]. Therefore any DFAS which supports such an execution model needs to maintain strict consistency throughout its file access points. To enable use of off-the-shelf software and to hide the complexity of the network from the application developers and endusers some DFASs offer standardized protocols [13–15] which decouple the development of client software from the DFAS allowing for the implementation of clients that can present the DFAS as a POSIX file system through network transparency. Network transparency hides the way protocols transmit and receive data over the network, thus any operation that can be performed on a local file can also be performed on a remote file. To scale with increasing request load, Distributed File Access Services (DFASs) often employ redundant pools of servers.

Despite the availability of RIs, DFASs do not always take advantage of their capabilities. It is rarely the case when DFASs interact with network devices (switches, routers, etc.) to optimize data transfers, maintain quality of service (QoS). The main hurdle for interacting with network devices to optimize data transfers is the configuration of these devices each using different protocols and interfaces. This makes large data transfers between RIs difficult to facilitate and slows down the life cycle of scientific applications [16–18].

The configuration of network devices is challenging in traditional network architectures due to the design of these devises. Networks can be divided in two planes, the control plane and the data plane. The control plane is responsible for routing, it is where forwarding decisions are made. The data plane is where the actual data packets are moving based on the instructions from the control plane. In traditional IP networks the control plane and the data plane are inside the networking devices. When administrators need to adapt the network configuration (modifying the control plane) they need to separately configure all the network devices. This leads to complex and lengthy configuration steps that limit the flexibility of the network. This has forced DFASs to treat the network as a black box.

SDN is a new approach to computer networking which removes the control plane from network devices and places it into SDN controller, simplifying the network configuration. The communication between the control plane and the data plane is done through a standardized interface called "Southbound-API". To make network programmability more flexible, the control plane (SDN controller) also offers an interface called "Northbound-API". The overall architecture of a SDN is shown in Fig. 1. In SDN the data plane is also responsible for monitoring local information and gathering statistics [19,20].

OpenFlow [21] and Network Service Interface (NSI) [22] are two typical SDN examples. OpenFlow has gained a lot of attention. It is a standard for a Southbound-API and it describes the interaction between controllers and OpenFlow-compliant switches [23]. OpenFlow-compliant switches use flow tables to coordinate the data plane. These tables are updated by the SDN controller. Each flow (sequence of packets from a source host to a destination host) that passes through an OpenFlow-compliant switch should be matched with a flow entry. A flow entry is a rule that indicates where the switch should forward data packets. A flow entry also includes counters which collect statistics about flows by storing number of received packets, bytes, and duration of each flow. These statistics can be requested from the controller that sends OFStatisticsRequest messages in real time to the desired switches and returns the results via the Northbound-API. Hence, measurements that use the probes (i.e. packets) to infer the network properties are not necessary when employing SDN.

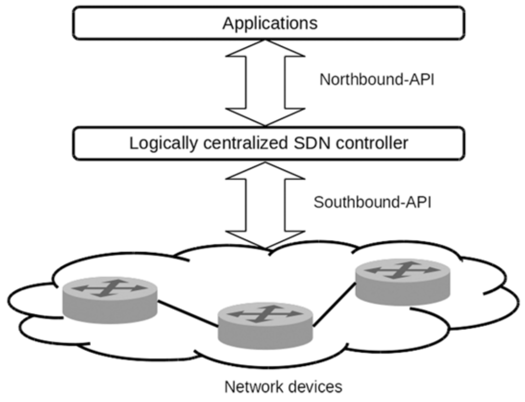


Fig. 1. A schematic software defined networking (SDN) architecture. The communication between the control plane and the data plane is done through the Southbound-API. The communication between the SDN controller and applications is done through the Northbound-API.

NSI [22] provides protocols for different domains to exchange topology information for inter-domain path selection. It also provides software services for selecting, reserving and provisioning network paths. In this work, we will use OpenFlow as our SDN protocol.

SDN technologies provide the opportunity to complement research infrastructures (RIs) with several new opportunities for supporting data and I/O-intensive workflows, for instance:

(1) customizing network connectivity between services,
(2) controlling the network QoS by controlling the data flows and,
(3) gathering information about the state of the network and help take decisions to optimize data flows or recover from error.

However, including SDN in DFASs is still unexplored.

The main objective of this work is to explore new opportunities offered by the network programmability. Such an insight into the network offers the ability to include network resources when provisioning for large data transfers. This will reduce the execution time of I/O and data-intensive scientific workflows that include scientific applications, which shall be referred to as *data consumers* or simply *consumers*.

The main contributions of this paper are:

- Investigation of the role of the SDN for file transfer services,
- A SDN-aware architecture for transparent and adaptive data transfers,
- An observable and controllable network model that describes data transfers from multiple sources,
- Two algorithms that take advantage of the programmability of the network,
- Assessment of data transfer improvement resulting from SDN-enabling DFASs.

This paper is organized as follows: In Section 2 we analyze related work on data transport and network resource provisioning. In Section 3 we present a formalism which will allow us to reason about the proposed solution. Section 4 describes our proposed architecture, its main components and its functionality. Section 5.1 describes the experimental setup we used to validate the proposed solution and Section 5 first describes a set of common scenarios that can result in the performance degradation of data transfers and later describes the execution of an I/O-intensive workflow. In Section 6 we present the performance results, while in Section 7, we discuss some relevant issues of this work and give an overview of future investigations.

2. Related work

Within the context of RIs there is a continuous effort to optimize data transfers and take into consideration the characteristics of the network and the storage resources. However, most of the approaches face problems of interoperability between network devices due to the lack of open standards. These approaches mostly focus on path reservations and for the most part ignore the network volatility. Moreover, most approaches are focused on specific frameworks or RIs. The increased popularity of SDN protocols has offered new opportunities for content delivery network (CDN)s that use SDN to optimize data flows and fully utilize all their network. CDNs however, base their large scale content availability to eventual consistency. This model of consistency although suitable for web content is not ideal for DFASs simply because applications need strict consistency. Even thought SDN protocols are becoming popular DFASs are not moving towards that direction.

StorNet is a resource provisioning and management system within RIs for data transfers [24]. StorNet relies on TeraPaths [25] to ensure end-to-end bandwidth reservations. Both StorNet and TeraPaths are systems developed for the ATLAS grid environment. Their purpose is to move data between ATLAS Tier-1 data centers [26]. They perform the data transfers using dedicated fractions of the available bandwidth. The TeraPaths approach is very similar to the SDN approach. It uses a set of Network Device Controller (NDC) modules to configure network devices within the domain of several RIs. The challenge here is to configure a number of heterogeneous network devices lacking a common configuration and control protocol. StorNet relies on advance reservations which have to specify the start and end time of the data transfers. Providing advance reservations can be difficult to achieve in an environment where multiple applications produce and consume data. The size of the data is often unknown before they are actually generated or consumed. The static aspect of these approaches makes them vulnerable to network traffic congestions or increasing load on the machines that host and transfer data.

A similar approach is presented in [27] which describes a mechanism for scheduling end-to-end network reservations for maintaining QoS. DFAS need to send reservation requests in advance before moving data. This approach requires the file access service to be tightly connected with the process of producing and consuming data. The reservation needs information about the data size, the time to start the transfer, the deadline for the completion of the transfer and the desired speed. Reservation techniques require from file access services to have a global view of the logic of an application. These techniques are difficult to implement in highly distributed and heterogeneous environments where tasks behave differently depending on the characteristics of the used infrastructure. This approach also faces problems regarding the configuration of heterogeneous network devices.

In [28] authors optimize transferring large data volumes between RIs for collaborative data analysis with scheduling multiple bandwidth reservation requests. As with the previous approaches, this one is concerned with bandwidth reservations which may be problematic for scientific workflows. The authors evaluated their approach by using simulations and therefore did not get into the specifics of configuring heterogeneous network devices.

Recently, SDN technologies attracted lots of attention for improving QoS and quality of experiences (QoE) in streaming video over the Internet. In [29] the authors use OpenFlow to get information about the state of the network and offer QoS for YouTube videos. The work presented in [30] introduces an SDN-based framework to achieve a network-wide QoE and fairness for video streaming in networks with limited resources. The authors of [31] propose a Network Control Plane for video streaming which maximizes users QoE and network utilization by reserving bandwidth on a per-flow basis. These approaches are developed for video streaming where data transfers involve files that in practice do not exceed 50 MB [32].These publications are concerned with CDNs that adopt an eventual consistency model which is not suited for scientific workflows.

In [33] the authors present Palantir, a system that abstracts network proximity for parallel/distributed computing frameworks using OpenFlow. Palantir's aim is to provide to parallel/distributed computing frameworks such as MapReduce a view of the underlying network to provide better job scheduling. This approach is one of few attempts to introduce SDN to RIs, but it is not oriented towards data transfers.

The work presented in [8] considers a peer-to-peer (P2P) approach to data sharing in scientific workflows running in the cloud. The authors evaluate their approach against traditional network file systems and conclude that although their approach performs better than NFS [34] which is a centralized file system, it does not perform as well as GlusterFS [35] which is a DFAS. According to the authors the reason why GlusterFS performs better than the P2P approach is because it is a fully distributed file system so it operates on a similar principle as P2P with files distributed across many nodes. Moreover, the proposed approach is not able to address issues of network congestion and optimal data flow routing. The authors take the traditional approach and consider the network as a black box.

3. Network model

There are three components involved in a file transfer: a source, a consumer and a network with a path from the source to the consumer. In the case of DFAS there are multiple sources to select from. Therefore, we model a network connecting several sites where a consumer requests data which can be downloaded from multiple data sources. We assume that the infrastructure is programmable, therefore the state of the network should be observable and controllable. At any given moment we can observe the number of bytes passing through each link and have a view of how many switches and devices are in the network. We also assume that we have a global view of the network topology and the path each data flow follows to reach its destination. The intention here is not to represent a model of a large network containing millions of consumers and sources, but rather a study a realistic usecase of a scientific workflow where dozens of consumers request data from dozens of sources through a network connecting several sites. Based on these observations the most suitable representation of this problem is the multiple source shortest path (MSSP) problem (see Fig. 2).

The network infrastructure is represented as a bidirectional weighted graph $G(V, E)$, where V is the set of all vertices in the network and E all the edges or links. This model considers a single consumer $cns \in V$ which is based on the assumption that every consumer request is processed by a separate thread.

Table 1
Description of the variables used in Eqs. (1)–(5).

Variable	Description
cns	The consumer that requests data
$D = \{d_1, d_2, \dots, d_n\} \subset V$	The set of data sources where data can be download from
$S = \{s_1, s_2, \dots, s_m\} \subset V$	The set of programmable switches
$P_{d_i,cns}$	A path from a data source d_i to the consumer cns
$c_{d_i,cns}$	The total cost of path $P_{d_i,cns}$
$c_{d_i,s_i}, c_{s_i,s_l}, c_{s_{cns},cns}$	The costs for traversing from data source d_i to switch si, from switch s_i to switch s_l and from switch s_l to a consumer cns
MTT_{s_i,s_j}	The minimum transfer time for moving a file of certain size from s_i to s_j
FS	The size of the file we want to move
L_{s_i,s_j}	The additional delay caused by third party traffic in the link connecting s_i and s_j
Bps_{s_i,s_j}	The maximum observed speed in bytes per second of the link connecting s_i and s_j
t_{s_i,s_j}	The average usage time of the link connecting s_i and s_j

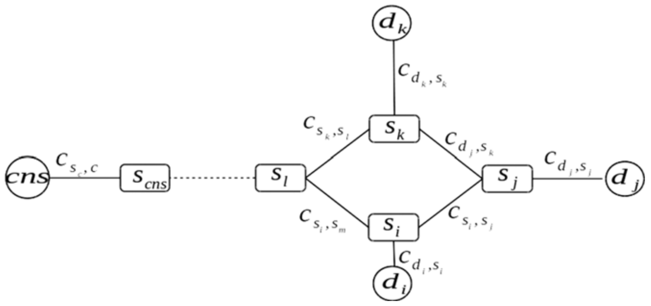


Fig. 2. Network model. Using information from the network, we are looking for the least cost path from a set of data sources $D = \{d_1, d_2, \dots, d_n\}$ to the consumer cns. The nodes named s_i, s_j, s_k, s_l, s_{cns} represent the switches of the network. The consumer cns is adjacent to the switch $s_{cns} \in S$ and each data source $d \in D$ is adjacent to exactly one switch from S.

The set of data sources is $D = \{d_1, d_2, \dots, d_n\} \subset V$ and the set of switches is $S = \{s_1, s_2, \dots, s_m\} \subset V$. S and D are disjoint sets. The number of vertices is hence $|V| = |D|+|S|+|cns| = n+m+1$. Table 1 summarizes the notations we will use in this paper.

In Fig. 2, a path from a data source d_i to cns is described as $P_{d_i,cns} = (d_i, s_i, s_l, s_cns, cns)$ and the total cost of path $P_{d_i,cns}$ is:

$$c_{d_i,cns} = c_{d_i,s_i} + \sum c_{s_i,s_l} + c_{s_{cns},cns}. \tag{1}$$

The MSSP problem is trying to find a data source $d_i \in D$ which is on a path $P_{d_i,cns}$ that minimizes the cost function in Eq. (1).

Because our goal is to speed up data transfers, the costs $c_{d_i,s_i}, c_{s_i,s_l}, c_{s_{cns},cns}$ in Eq. (1) represent features that impact data transfers such as bandwidth, latency and load. Therefore, the costs c_{s_i,s_l} assigned to the link between the switches s_i, s_j are defined as:

$$c_{s_i,s_j} = MTT_{s_i,s_j} + L_{s_i,s_j}. \tag{2}$$

The minimum transfer time, MTT_{s_i,s_j} is defined as the time it takes to move a file of size FS between switches s_i and s_j assuming exclusive access to the link between them. Therefore, MTT_{s_i,s_j} depends only on the bandwidth and latency of the link. L_{s_i,s_j} represents the additional delay caused by third party traffic traversing the link between switches s_i and s_j.

To calculate the minimum transfer time, MTT_{s_i,s_j}, it is necessary to obtain the maximum observed speed of a link. Therefore, we calculate the minimum transfer time MTT_{s_i,s_j} using the maximum observed speed Bps_{s_i,s_j}, divided by the file size FS. Bps_{s_i,s_j} is calculated by dividing the number of bytes in a flow by the duration of that flow. SDN provides us with the ability to retrieve from each switch the number of received packets, bytes, and the duration of each flow. We use these metrics to calculate MTT_{s_i,s_j}:

$$MTT_{s_i,s_j} = \frac{\max(Bps_{s_i,s_j})}{FS}. \tag{3}$$

In practice, it is challenging to obtain an accurate estimation of the time it takes to transfer a file. Issues like sudden traffic bursts and unexpected link failures make it particularly difficult to make transfer time estimations. Instead of trying to estimate the exact transfer time for a file, we calculate an additional delay for each link that depends on how often the link is used. The additional delay L_{s_i,s_j} between s_i, s_j is calculated by multiplying the average usage time t_{s_i,s_j} of the link by a factor b. Where b is a constant we use to control the impact of the link usage in Eq. (2). L_{s_i,s_j} is give by:

$$L_{s_i,s_j} = t_{s_i,s_j} \cdot b. \tag{4}$$

This approach is not intended to provide an accurate estimation of the added delay. Instead, it helps to provide a relative indication of the link load. We use this approach because it is impossible to obtain information about the bandwidth allocation policy of a link or the number of streams contained in a flow. In other words, it is not possible to know how many file transfers pass through a link at the same time or the bandwidth each transfer has been allocated.

In Eq. (4), usage time t_{s_i,s_j} is calculated by observing for how long flows occupy the link connecting s_i and s_j. t_{s_i,s_j} is calculated as an exponentially weighted moving average (EWMA) [36]. We use EWMA to obtain an average that smooths out short-term fluctuations and highlights longer-term trends. This way sudden traffic bursts or short-term link availability have no effect to the cost function in Eq. (2). The EWMA of t_{s_i,s_j} is given by:

$$t_{s_i,s_j,new} = \alpha \cdot t_{s_i,s_j,prev} + (1-\alpha) \cdot t_{s_i,s_j,curr}, \tag{5}$$

where $\alpha \in [0, 1]$ is the weighting factor, $t_{s_i,s_j,curr}$ is the current measurement, $t_{s_i,s_j,prev}$ the previous value and $t_{s_i,s_j,new}$ the newly calculated value. The $t_{s_i,s_j,new}$ is recalculated whenever there is a new flow on the link.

Algorithm 1 shows the process of updating the MTT_{s_i,s_j} and t_{s_i,s_j} metrics. As a first step the algorithm polls the SDN switches for active flows (data transfers) in the network (line 3). Then, for each flow in the network the algorithm calculates the sample maximum bandwidth which is necessary for Eq. (3) (lines 4–8). Finally, Algorithm 1 calculates the link usage as EWMA which is necessary for Eq. (4). This algorithm introduces little complexity ($O(N)$) as it only depends on the number of active flows and therefore scales linearly.

Algorithm 2 builds the graph $G(V, E)$ and calculates the cost for each edge as defined in Eq. (2). First, the algorithm starts by calculating the cost $c_{s_{cns},cns}$ of the link between consumer cns and its associated switch s_{cns}. Then it calculates the costs of the links on all the paths between the consumer switch and each data source. Once all the costs of the $G(V, E)$ are known, the algorithm calculates the shortest path between the consumer and all the available data sources. To find the shortest path we use the well known Dijkstra's algorithm, which has complexity $O(|E| + |V| + \log |V|)$ [37].

Algorithm 1 Updating *Bps* and *L* metrics for active flows

```
1: procedure UPDATEMETRICS
2:     while true do
3:         Gather all active flows from switches and store it in $F$
4:         for all flows $\in F$ do
5:             Calculate speed: $Bps_{new} = \frac{byteCount}{duration}$
6:             Retrieve the previous link speed $Bps_{old}$
7:             if $Bps_{new} \geq Bps_{old}$ then
8:                 Set link speed to $Bps_{new}$
9:             end if
10:            Calculate $MTT_{s_i,s_j}$ according to Equation 3.
11:            Retrieve the previous link average usage time $t_{prev}$
12:            Get the current link $t_{current}$
13:            Calculate link usage as EWMA according to Equation 5
14:            Calculate $L_{s_i,s_j}$ according to Equation 4
15:        end for
16:    end while
17: end procedure
```

Algorithm 2 Finding the least cost path from a set of data sources to a consumer and return the most optimal source

```
1: procedure GETLOWESTCOSTPATH($cns$,$D$)
2:     Calculate the cost $c_{c,s_{cns}}$ from the consumer $cns$ to its switch $s_{cns}$ using Equation 2
3:     for all data source $d_i \in D$ do
4:         Calculate the cost $c_{d_i,s_j}$ from data source $d_i$ to its switch $s_i$ using Equation 2
5:     end for
6:     for all switches $s_i \in S$ do
7:         Calculate the cost $c_{s_i,s_j}$ from switch $s_i$ to it switch $s_j$ using Equation 2
8:     end for
9:     Find the shortest path from all data sources $D$ to consumer $cns$
10:    Return the IP of data source $d_i$ which is part of the shortest path
11: end procedure
```

4. Integration of SDN and DFAS

The algorithms presented in Section 3 are implemented in an SDN-aware DFAS. To optimize file transfers, we propose an architecture which uses SDN to select the shortest path between a set of data sources and a consumer. The new service extends our early work: A DFAS that federates data storage named Large OBect Coud Dta storagE fedeRtion (LOBCDER) [38] and a network QoS-aware resource planning service named NEtwork aware Workflow QoS Planner (NEWQoSPlanner) [39,40].

4.1. Architecture requirements

The architecture of the new DFAS should provide transparent high performance and reliable data transfers. More specifically, the design should implement a standardized protocol for accessing datasets and offer strict consistency to consumers. This requirement allows the use of off-the-shelf client software which can offer a file system abstraction. This is important to maintain network transparency to files and allow applications to communicate through file exchange. The architecture should be able to use any type of data source whether this is grid or cloud storage. It should also have increased data availability and scale with increasing request load. The requesting consumer should be assigned to the best data source. If the selected data source is experiencing heavy load

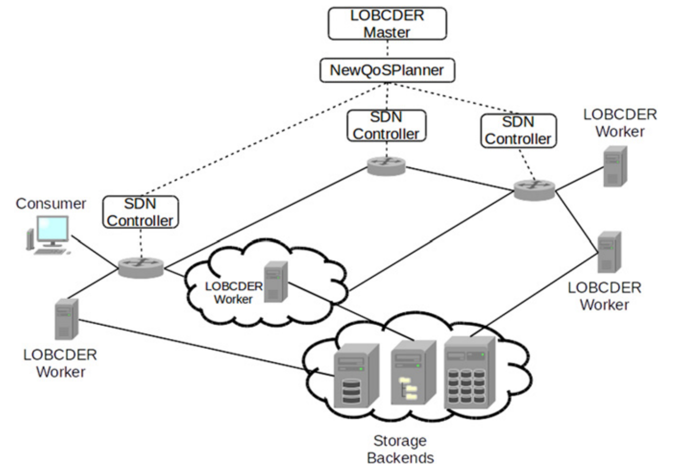


Fig. 3. Architecture of the SDN-Aware DFAS supporting multiple SDN domains. On the top level, the LOBCDER master communicates with the NEWQoSPlanner to get information for selecting the most optimal worker. The NEtwork aware Workflow QoS Planner (NEWQoSPlanner) aggregates information from all SDN controllers. For non-SDN domains the NEWQoSPlanner treats them as a black box, collecting information only on the input and out output of the domain. At transfer time the NEWQoSPlanner is used to optimize the data route.

the architecture should autonomously switch sources. The service should be able to dynamically adapt seamlessly to network congestions. Finally, there should be an abstraction layer between the service and the network devices to enable the network programmability.

We propose an SDN-aware DFAS architecture that consists of three main components: the LOBCDER master and workers, and the NEWQoSPlanner. The LOBCDER master has a WebDAV interface to offer clients a standardized data access protocol. Internally, the LOBCDER master keeps references of all data sources, where actual data are stored. The LOBCDER workers are stateless HTTP services that serve two purposes: 1) load-balance incoming requests and maintain high data availability and 2) translate various protocols used by the storage backend into standard HTTP to enable the use by WebDAV clients. The storage backends may be of any type, e.g. OpenStack-Swift [41], GridFTP [42], SFTP, etc. They may be located anywhere and do not require installation of any additional software to be used by the proposed architecture.

The NEWQoSPlanner aggregates information collected from the network, reserves paths, and controls the flows via SDN controllers. The proposed architecture relays on the existence of SDN controllers and programmable switches, something that is becoming more common in RIs. Fig. 3 shows the overall architecture; the LOBCDER workers are the data sources. The workers overlay a set of independent and heterogeneous storage backends.

4.2. LOBCDER

The LOBCDER [38] is a DFAS that provides access to distributed scientific data stored in various storage frameworks distributed across independent providers. It is a part of the Data and Compute Cloud Platform of the VPH-Share project [5]. LOBCDER is composed of two major components: a master and a number of workers. The workers take over file transfers while the master serves requests about the metadata of a file (like size, creation date, etc.) and replicates files to available storage frameworks owned by independent providers. The LOBCDER master implements a WebDAV interface that offers a standardized protocol. This way LOBCDER is able to provide a file system abstraction with strict consistency and, with the use of workers, offer improved availability. Both the master and workers can access any type of data source, whether this is grid or cloud storage, due to the use of a virtual file system (VFS) API. The VFS API offers unified access to a large number of storage resources such as Local File System, SFTP, WebDAV, GridFTP, SRM, LFC, iRODS, OpenStack-Swift, vCloud, AWS S3. LOBCDER workers are simple stateless servlets which allows them to be deployed on grid, cloud, or cluster servers. The number of LOBCDER workers can be increased or decreased following the number of incoming data transfer requests and therefore they elastically scale depending on the load.

4.3. NEWQoSPlanner

NEWQoSPlanner is an agent based system which bridges the application and the underlying network infrastructure. The NEWQoS-Planner selects network sources and invoke network services to reserve, allocate, provision and control the network resources. NEWQoSPlanner is able to provide cross-domain topology descriptions and information. The system originally developed in the context of the CineGrid [43] project to support collaborative video composition on large quantity of video material. The NEWQoS-Planner uses a Network Markup Language (NML) to describe the network topology and the connected devices. With the network description the NEWQoSPlanner resolves QoS constraints by searching for optimal combinations of network paths between sources and destinations. For controlling network services, it supports the NSI framework [44] and the OpenFlow standard.

In this work, we use the OpenFlow interface of the NEWQoS-Planner to obtain information from the SDN controllers and to monitor the performance and characteristics of each link in the network. This allows us to include Algorithms 1 and 2 into NEWQoSPlanner and assign to consumers the data source with the lowest cost and dynamically adapt to network congestions.

4.4. SDN-Aware DFAS

Algorithms 1 and 2 are implemented extending the NEWQoS-Planner. Algorithm 1 provides a view of the state of the network and Algorithm 1 uses this information to solve the MSSP problem. To communicate with LOBCDER, NEWQoSPlanner implements two additional interfaces. The first is used by the LOBCDER master to request a solution for the MSSP problem which allows for the selection of the most suitable worker. The second interface is used by the LOBCDER worker to request traffic re-routing in case a network congestion. We have also equipped the LOBCDER worker with the ability to monitor the performance of file transfers and detect any performance degradation. The interaction of the main architecture components is described below.

The NEWQoSPlanner constantly pools the SDN controllers for active flows and keeps a record of the *MTT* and *L* metrics computed in Algorithm 1. Additionally, it calculates the shortest paths based on the topology information which includes the location of the available workers and potential consumers with the use of Algorithm 1. As soon as LOBCDER master receives a request from a consumer to download a file, it requests from the NEWQoSPlanner the shortest path between the consumer and the available workers. The NEWQoSPlanner identifies the shortest path and returns to the master the IP of the worker that can be used to download the data. The LOBCDER master then sends a redirect message[1] back to the consumer which initiates the transfer. Finding the shortest path between the consumer and a set of workers may take a long time depending on the size of the network and number of workers. To ensure that the consumer gets a response within a reasonable amount of time, the LOBCDER master sets a deadline for getting the response from the NEWQoSPlanner. If the deadline expires and the LOBCDER master has no response, it assigns a worker based on a simple round-robin algorithm. Moreover, as soon as the transfer begins the assigned LOBCDER worker samples the speed of the transfer. The speed is calculated as a EWMA to smooth out any short term fluctuations. As soon as the speed drops below a certain threshold the worker sends to the NEWQoSPlanner a request to optimize the flow between the consumer and the LOBCDER worker. The NEWQoSPlanner looks for the shortest path between the consumer and worker using Algorithm 1 and directs the flow through the least cost path. To avoid suboptimal path selection, the worker sends periodical requests to the NEWQoSPlanner to search for possible paths with the lowest cost. The frequency of requests is decreased when the transfer nears its completion. Selecting a suboptimal path may occur if the NEWQoSPlanner selects a less used link (Eq. (4)) with less bandwidth (Eq. (3)) instead of a busy link with more bandwidth. However, if the link with more bandwidth becomes available during the transfer, the worker should be able to switch to link with more bandwidth. Moreover, if the worker detects a drop in performance and after sending requests to the NEWQoSPlanner observes no improvement, it assumes that either the worker link is busy, or the node hosting the worker is under high load. In the case of high load on the host resources it is preferable to monitor the transfer speed rather than CPU and memory load. CPU and memory usage of a virtual machine running on a cloud do not increase if the hosting node is under high load. If the worker detects a drop in performance and the consumer supports *byte serving*[2] the worker drops the connection forcing the consumer to resume the transfer from a different worker. In the case where the node hosting the worker is under heavy load the worker itself notifies the master about its state. This is done to avoid rescheduling of the same worker.

The way the three main components of the architecture (the LOBCDER master and worker, and NEWQoSPlanner) interact with each other can be seen in Fig. 4. They can be deployed on RIs with SDN-enabled switches and are invisible to consumers and therefore do not require any modifications to the software used by these consumers.

5. Experimental setup

In this section we provide a description of the test environment and the setup we used to run our experiments. In the first setup we validate our architecture with three scenarios that examine the role of the sources and the network. In the second setup we test our architecture with the use of an I/O-intensive workflow.

5.1. Virtual test environment

To assess the performance of the architecture we used a controlled test environment. This environment allows evaluation of the performance and functionality of the architecture against several scenarios. Using real live networks these scenarios might be particularly difficult to test. Moreover, a virtual test environment allows us to verify that the proposed system will perform as expected and generate reproducible results. It also gives us the chance to study the relationships between the components of the architecture in detail. We may control characteristics of the links specifying bandwidth and latency as well as introduce traffic in any part of the network. To conduct our experiments we used ExoGeni [45] a multi-domain infrastructure-as-a-service (NIaaS) federated testbed that provides a virtual laboratory for networking and distributed systems. ORCA (Open Resource Control Architecture), the ExoGENI control framework software, allows easy deployment of virtual topologies, composed by virtual machines, networking connections, and other elements.

[1] This is an HTTP Redirect message with a 302 return code.

[2] Consumer can request a range of bytes from a file.

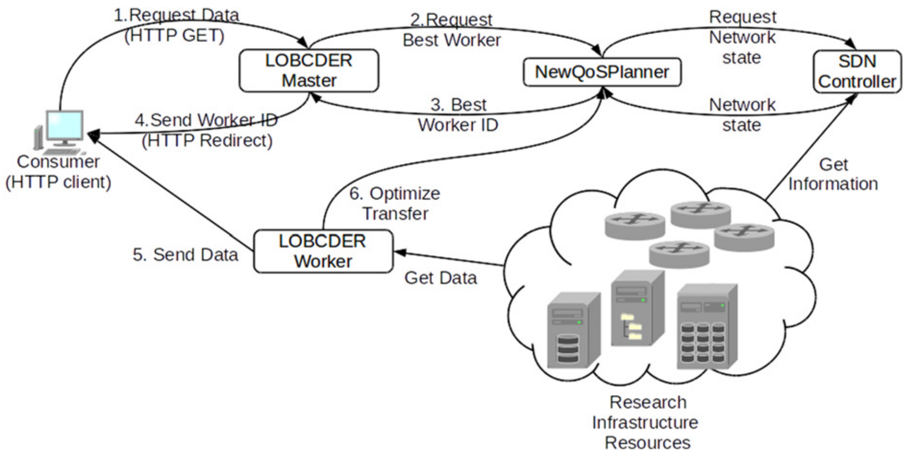


Fig. 4. Interaction of the main architecture components. When an HTTP consumer requests data Large OBect Coud Dta storagE fedeRtion (LOBCDER) provides the data transfer from heterogeneous storage backends, while the NEWQoSPlanner aggregates information about the state of the network and uses Algorithms 1 and 2 to select the optimal worker and path for the data.

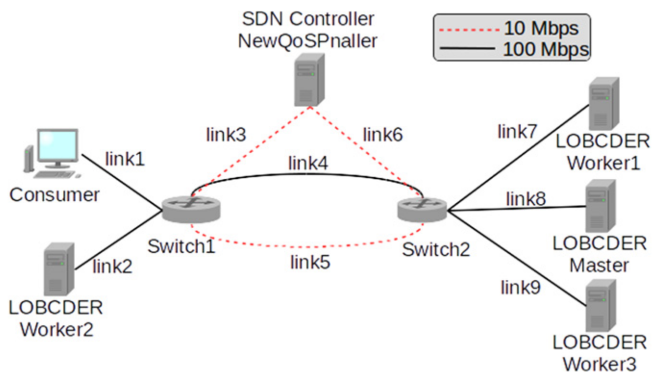


Fig. 5. Experimental topology for analysis of SDN-aware data transfer system. In this topology we are looking for the most optimal worker to transfer large files to the consumer and investigate the discovery of alternative routes between the worker and the consumer.

Table 2
Round trip time (RTT) from consumer to workers for the topology in Fig. 5. Worker 2, which is closer to the consumer, has lower RTT compared with the rest of the workers.

Node	Round Trip Time, ms
Worker 1	2.82 (via link 4)
Worker 1	3.11 (via link 5)
Worker 2	1.78
Worker 3	2.79 (via link 4)
Worker 3	2.74 (via link 5)

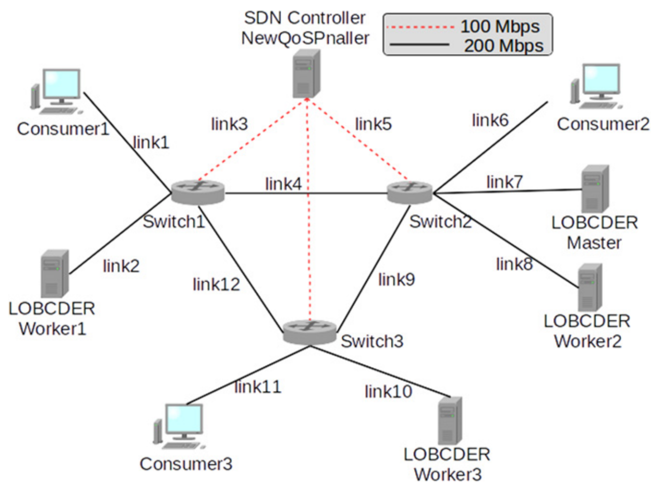


Fig. 6. Experimental topology for testing an I/O-intensive workflow with the Montage Astronomical Image Mosaic Engine (Montage). In this topology we use consumers 1, 2 and 3 to execute the tasks of the workflow shown in Fig. 7.

Our virtual test environment consists of the topologies shown in Figs. 5 and 6. The topology in Fig. 5 represents a common setup. It is often the case that two universities or RIs are connected with one or more direct links and share computational and storage resources to run experiments on the infrastructure as well as to execute large-scale, data-demanding scientific applications [46]. The topology in Fig. 6 represents a collaborative set-up between several RIs to execute a I/O-intensive workflow where tasks communicate through file exchange. Both testbeds were selected to validate all basic features of the elaborated approach, algorithms, and architecture. Within the ExoGENI framework, the testbed presented in Fig. 6 was the largest we could use in a controlled way. Building larger testbeds with more complex topologies requires usage of several sites which may go off-line for maintenance. In the future, when significantly larger testbed is available, additional experiments will be performed to validate the obtained results.

In these topologies all switches are Open vSwitch which are open-source software switches [47] and are connected to the SDN controller using OpenFlow. For the topology in Fig. 5, links 1, 2, 4, 7, 8 and 9 and have 100 Mbps bandwidth while links 3, 5, and 6 have 10 Mbps. All links except links 3 and 6 are used for data transfers. Links 3 and 6 are used to send and receive control messages between the switches and the controller. Latency plays an important role in performance; Table 2 shows the average round trip time from each node to the consumer.

For the topology in Fig. 6, links 3, 4 and 5 have 100 Mbps bandwidth while the rest — 200 Mbps bandwidth; links 3, 4 and 5 are used to send and receive control messages between the switches and the controller. Although ExoGENI may provide links with approximately 1 Gbps, we choose to use links of up to 200 Mbps to have a more reliable testbed, since requests in ExoGENI with large capacity links tend to fail. This choice will not affect the behavior of our architecture, since all nodes and methods use the same topologies and links.

In the experiment setup the consumer sends requests to download files to the LOBCDER master. LOBCDER master communicates with NEWQoSPlanner to get the optimal path between workers and the consumer and redirects requests to the appropriate worker. The workers stream data to the consumer from the backends and monitor the speed of the transfer. In this setup we assume that the workers are deployed on the backends themselves.

The workers use a simple caching mechanism where they evict the oldest file from their cache using a Least Recently Used (LRU) eviction policy. Therefore, since the consumer requests the same file it receives it directly from the worker. The NEWQoSPlanner collects information from SDN controllers (in this case the NEWQoSPlanner and SDN controller are on the same node) and provides the optimal path to the LOBCDER master. The two switches are registered on the SDN controller. The NEWQoSPlanner decides about the optimal routing and sets the routing rules on the switches via the SDN controller. Before the consumer makes any request the switches work in a standalone mode where they act as normal layer 2 switches. As soon as the consumer requests a file from LOBCDER, the master contacts the NEWQoSPlanner to set the most optimal path.

The virtual experimental topologies represent several separate networks, where we can control and collect information from. We assume that we can use resources on all networks to deploy LOBCDER workers. As stated earlier, this is in accordance with a real-world setup where several RIs are directly connected to each other and share available resources for experimenting on the infrastructure or to run data and I/O-intensive scientific applications. Therefore, with this common topologies we can investigate the selection of the most optimal worker, discover alternative routes between the worker and the consumer and assess the execution times of an I/O-intensive workflow.

5.2. Evaluation scenarios

As mentioned in Section 3, there are three components involved in a file transfer: a set of sources, a consumer and a network with a path from the source to the consumer. In this section we will provide three scenarios that examine the role of all three components and their impact of the performance of the data transfers. The following experiments have two objectives: validate the cost function used in Algorithm 2 (Scenario 1) and investigate problems that can appear at the edge of the network at the source nodes (scenario 2), or the network core (scenario 3).

We assume that transfers are performed on a shared infrastructure (multi-user, multi-application). As a consequence the network can become congested by additional traffic and the hosts can become overload by co-hosted applications. For each scenario we compare results with and without the use of the NEWQoSPlanner while the consumer requests to download a file from the LOBCDER master. The scenarios of each experiment are described as follows:

Scenario 1: Optimal worker Selection: The nodes hosting the LOBCDER workers 1 and 3 from which data is streamed to the consumer, also host applications generating additional traffic which reduce the available bandwidth. For this scenario we consider two cases: (1) one using the NEWQoSPlanner to choose the path and worker with less traffic and (2) one using a simple round-robin algorithm for selecting workers.

Scenario 2: Dynamic worker Selection: The LOBCDER master chooses the worker with the lowest traffic. During the transfer other applications on the nodes hosting LOBCDER workers 1 and 2 start I/O operations which reduces the performance of the data transfer. Here we consider two cases: (1) a static one where worker continues the transfer to the consumer on the overloaded link and (2) a dynamic case where the worker drops the connection if the speed is below a certain threshold, and the consumer makes a new request to the master to resume the transfers using another worker.

Scenario 3: Dynamic Traffic Routing: The LOBCDER master chooses the worker with less traffic. The node hosing worker 2 is not available due to downtime and during the transfer the selected LOBCDER worker detected a degradation of the network performance due to a congestion on the network. In the controlled experimental environment we simulate network congestion on link 4 and therefore reduce the available bandwidth (in this experiment we reduced the bandwidth to 10 kbps). This simulates link reliability issues (link starts to drop packets) and data sources failing. Similarly to the second scenario, we consider two cases: 1) the static one where the worker continues the transfer using the problematic link, 2) the dynamic case where the worker requests from the NWEQosPlanner to find an new optimal path to the consumer if the speed drops below a certain threshold.

5.3. I/O intensive workflow

To investigate the benefit of our approach on real I/O-intensive workflows we conducted a set of experiments using a Montage workflow. Montage is a toolkit for assembling Flexible Image Transport System (FITS) images into custom mosaics of the sky. It is used by astronomers to generate large images of the sky composed of smaller images. This application integrates multiple images taken from different parts of a galaxy to produce one image. Apart from the complex algorithm that ensures that the separate images will fit together while preserving some vital data, this workflow seen in Fig. 7, produces some intermediate images, that go on to further processing until they are composed into the final image [48].

The workflow seen in Fig. 7 creates a mosaic of the Pleiades star cluster using data from the Digitized Sky Survey (DSS2) [49]. All the input images required for the generation of the Pleiades star cluster are stored in the LOBCDER DFAS. All the tasks of the workflow communicate with each other through files exchange using LOBCDER. The Montage workflow generates over 2000 requests to LOBCDER of which approximately 1900 are operations on the data itself (upload, download and delete). The dataset used for this workflow is composed by 560 files totaling 4.1 GB.

To focus only on the file exchange between tasks and the effect of data transfers on the execution time of the workflow we run the workflow once and then used the trace of that execution. This trace only includes the file requests made my the workflow tasks. This way we eliminate the processing time from the workflow execution and only examine the data transfer time. Eliminating the processing time of the workflow makes it more challenging for LOBCDER since the amount of requests per second is increased. To run the Montage workflow we use the topology shown in Fig. 6. Consumers 1, 2 and 3 each execute one of the three strands of the workflow. After all strands are executed in parallel consumer 1 builds the final mosaic. All consumers have mounted LOBCDER and all input and intermediate data are accessed as a local files.

Similarly to Section 5.2 we consider two scenarios when running the Montage workflow. In the first we run the workflow on the topology without other applications generating network traffic. In the second the nodes hosting workers 1 and 3 also host applications that generate additional traffic which reduce the available bandwidth.

6. Performance evaluation

To evaluate the performance of the proposed architecture, we measured the speed at which the consumer downloads files of different sizes. The files we used scenario 1 and 2 described in Section 5.2 are 10 MB, 100 MB, and 1 GB. For the last two scenarios we used files of 100 MB, 1 GB and 5 GB. With the first two we investigate how we can increase file transfer performance by selecting the most optimal worker as well as the worker's ability to react to congestions on its link. Scenario 3 we look at the ability of the architecture to dynamically adapt and discover alternative routes

Table 3
Results for scenario 1. We measured the transfer speed using 10 MB, 100 MB and 1 GB files and compared it with a round-robin strategy and the use of SDN technologies that allows us to select the worker with the least amount of load on its link and resources.

File Size	NEWQoSPlanner		Round-Robin	
	Speed, MB/sec	Stdev, MB/sec	Speed, MB/sec	Stdev, MB/sec
10 MB	7.23	0.64	7.84	6.61
100 MB	10.77	0.56	9.08	3.36
1 GB	11.17	0.06	8.55	3.25

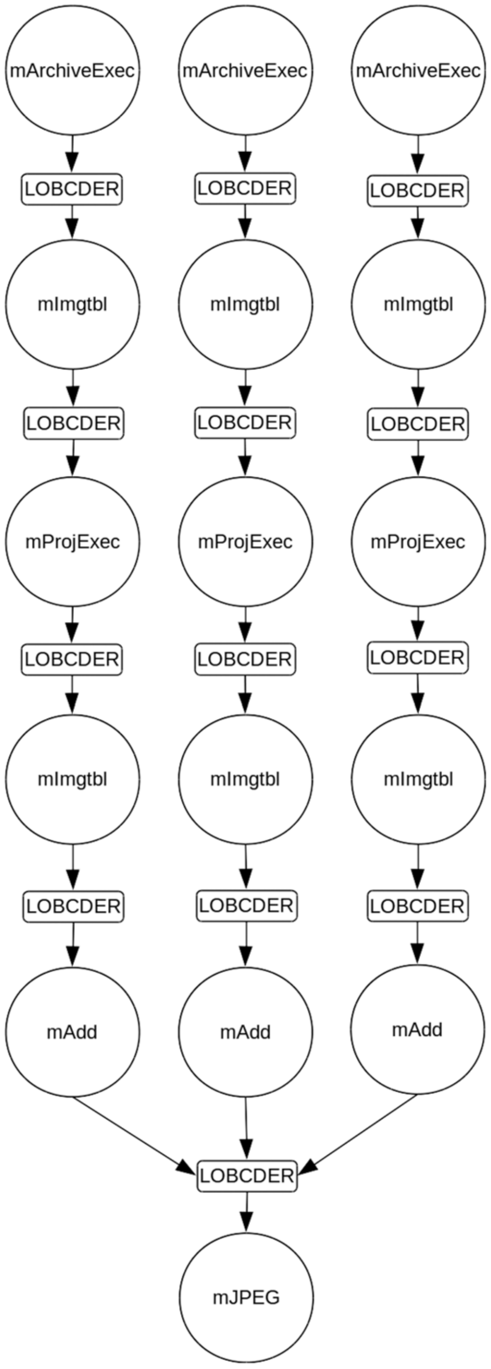


Fig. 7. Montage workflow for generating the Pleiades star cluster. This workflow generates over 2000 requests to LOBCDER of which approximately 1900 are operations on the physical data (upload, download and delete). The dataset used for this workflow is composed by 560 files totaling 4.1 GB.

between the worker and the consumer that avoid congestions and unreliable links.

6.1. Scenario 1: Optimal worker selection

As expected, it is clear from Table 3 that for small file transfers (up to 10 MB) the round-robin strategy is faster than the NEWQoS-Planner. This is mainly due to the overhead introduced by the procedures which try to optimize both the worker selection and the path to the consumer. However, for larger file transfers (over 100 MB), this overhead becomes negligible and the average speed of the transfers is very close to the link bandwidth. As mentioned in Section 5.1 for this scenario, all links have 100 Mbps (12 MB/sec) bandwidth and for file sizes over 100 MB our solution can utilize at least 89.7% of the link speed.

The round-robin strategy exhibits a higher variance because the selection process is not considering the characteristics of the workers or the link and each worker is selected according to a predefine order. If the algorithm selects workers 1 or 3 a significant drop in the transfer speed is registered. When selecting worker 2 the transfers are a lot faster. This is because worker 2 is connected to the consumer via a low-traffic path and has lower latency (see Table 2).

6.2. Scenario 2: Dynamic worker selection

In this scenario the worker monitors the quality of the transfer by calculating the EWMA (this is similar with Eq. (5) in Section 3) to smooth out short-term fluctuations. The sensitivity of the EWMA to short term variations is defined by a weighting factor $\alpha \in [0, 1]$. Smaller values of α make the EWMA very sensitive to network speed fluctuations while larger values of α (close to one) make EWMA less sensitive therefore the choice of α affects the stability of the worker. If α is too low the worker will be too sensitive to speed fluctuations and may unnecessarily close the connection. If α is too high the worker will not be able to react quickly to performance reductions. Fig. 8 shows the effect of α on the threshold used by the worker to decide when to drop the connection. In this Figure we monitor the performance of a transfer and measure the “real” speed. To simulate network congestion we introduce additional traffic after 220 s.

In Fig. 8a, with $\alpha = 0.5$ and the threshold is set to 90% of the maximum EWMA speed (each time we calculate the EWMA we compare if the newly calculated value is greater that the previously saved maximum EWMA speed). If the EWMA speed crosses the threshold the worker drops the connection. In this case the worker unnecessarily drops the connection after 36 s. It is only when $\alpha = 0.95$ that the worker behaves as expected and drop the connection only after the point where we have injected extra traffic to create network congestion (Fig. 8b).

Table 4 shows the results we obtain when testing scenario 2. The results show that the strategy of workers closing the connection does not always have a positive impact on the performance. For relatively small file transfers (100 MB) the overhead of finding an alternative worker, described in the second scenario is too high compared with the total transfer time. However, this strategy shows improvement for large data transfers. Fig. 9 shows the transfer speed measured at the consumer when downloading a 5 GB file.

Table 4
Results for the second scenario. We measured the transfer speed using 100 MB 1 GB and 5 GB files and compared it with two worker implementations: A worker that is not monitoring performance and a worker that monitors the transfer and reacts to performance drops.

File Size	Dynamic: Worker monitors performance		Static: Worker continues transfer on overloaded link	
	Speed, MB/sec	Stdev, MB/sec	Speed, MB/sec	Stdev, MB/sec
100 MB	2.44	0.29	7.07	1.53
1 GB	7.85	0.56	5.66	0.31
5 GB	10.20	0.28	5.94	0.56

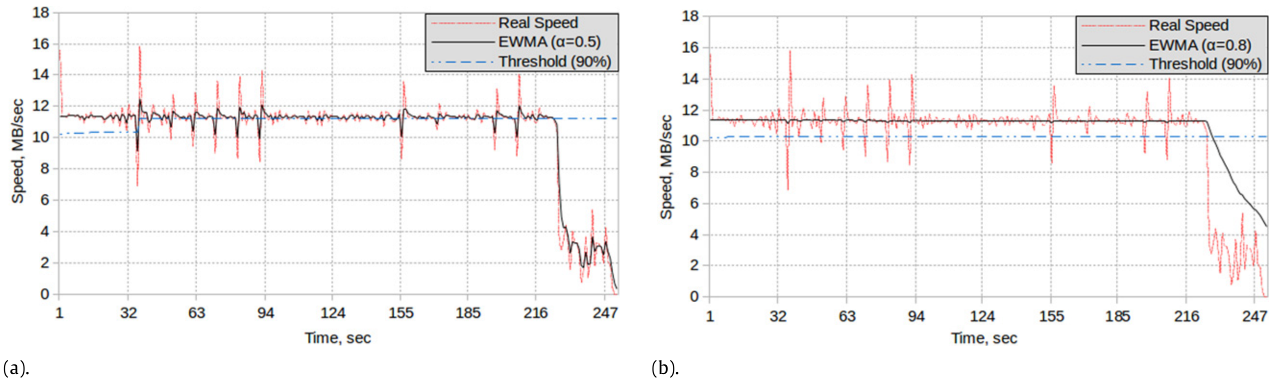


Fig. 8. $\alpha = 0.5$ and threshold 90% of the maximum exponentially weighted moving average (EWMA) speed.

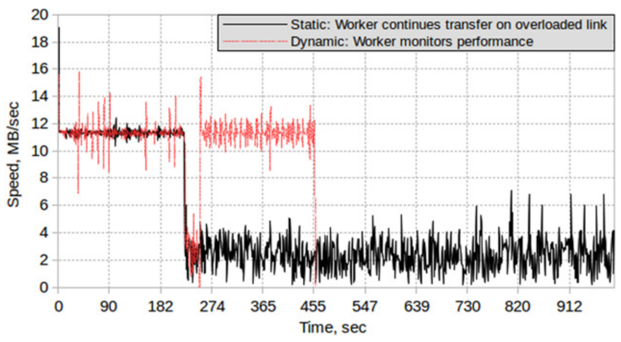


Fig. 9. Results for the second scenario. We measured speed from the consumer's perspective while downloading a 5 GB file. After approximately 200 s the performance drops dramatically because the node hosting the LOBCDER worker is performing some other I/O operation. In the static case the LOBCDER worker is not reacting while in the dynamic it drops the connection at approximately 220 s and the consumer resumes the transfer at approximately 250 s from an alternative LOBCDER worker.

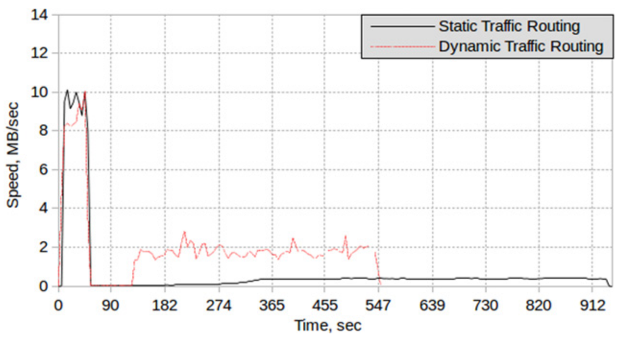


Fig. 10. Results for the third scenario. Measuring speed from the consumer's perspective while downloading a 1 GB file. After 55 s the performance of the transfer is reduced due to injected traffic on the chosen link. In the static routing case the transfer continues on the problematic link completing the transfer after 940 s. In the dynamic case, the LOBCDER worker sends an optimization request to the NEWQoSPlanner which reroutes the traffic from an alternative link complementing the transfer after 545 s.

After approximately 200 s the performance drops dramatically because the node hosting the LOBCDER worker is performing some other I/O operation. In the static case the LOBCDER worker is not reacting and the download is completed in approximately 990 s. Using the monitoring the LOBCDER worker drops the connection at approximately 220 s and the consumer resumes the transfer at approximately 250 s from an alternative LOBCDER worker complementing the download after 457 s.

6.3. Scenario 3: Dynamic traffic routing

Fig. 10 shows the results for the third scenario while downloading a 1 GB file. After 55 s the performance of the transfer is reduced. In the static routing case the transfer continues on the problematic link completing the transfer after 940 s. In the dynamic routing case the traffic is rerouted from an alternative link completing the transfer after 545 s. Therefore, flow optimization using NEWQoSPlanner significantly improves the file transfer which is completed approximately 40 % faster. The worker detects a drop of performance, sends a request to the NEWQoSPlanner to optimize the flow (which takes approximately 70 s). The NEWQoSPlanner provides an alternative route via link 5 which has a lower bandwidth (10 Mbps) but at this moment is faster than link 4 which has lower bandwidth (10 kbps) due to the injected traffic.

6.4. I/O intensive workflow

Figs. 11 and 12 show the execution times of each individual request to LOBCDER. In both graphs the *x*-axis represents the request number and the *y*-axis the time required to serve that request. Fig. 11 shows the results for running the Montage workflow without other applications generating network traffic. We can see that for the first part of the workflow our approach is significantly faster for the majority of the requests. For the rest of the execution the two methods have a similar performance with our approach performing slightly better. Our approach performs better on the first part of the execution because the NEWQoSPlanner selects the most optimal worker to server each consumer. For the rest

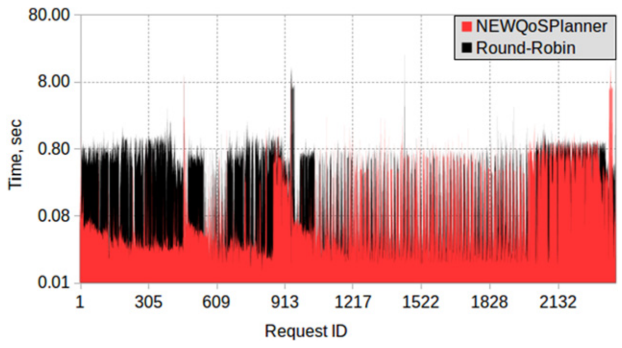


Fig. 11. Running workflow without additional traffic. For round-robin the execution time was 11.6 min and for SDN 9.5 min.

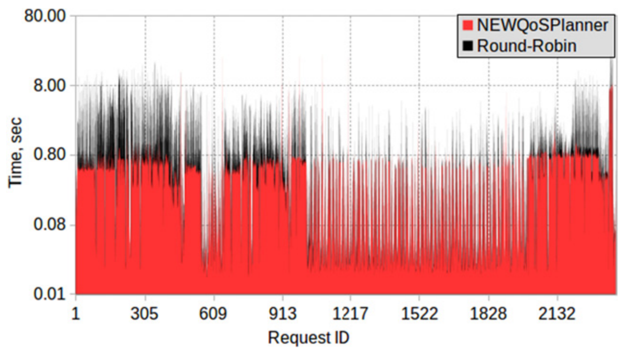


Fig. 12. Running workflow with traffic from 4 to 6. For round-robin the execution time was 32.1 min and for SDN 12.3 min.

of the execution because intermediate results are not replicated on all workers both approaches have a limited choice of workers. Nevertheless, with our approach the execution time was 9.5 min while with the round-robin approach 11.6 min.

Fig. 12 shows the results for running the Montage workflow while the nodes hosting workers 1 and 3 also host applications that generate additional traffic which reduce the available bandwidth. These results show that our approach serves the majority of the requests much faster that the round-robin approach. This is because NEWQoSPlanner is able to identify the least cost path using Algorithm 1 completing the workflow execution in 12.3 min while the round-robin approach completed the workflow execution in 32.1 min.

7. Conclusions and future work

In this paper, we demonstrated that exploiting SDN may improve performance of large data transfers and I/O-intensive workflows. The proposed approach is transparent for end users as they may locate, discover, and share datasets with off-the-shelf software like web browsers or WebDAV clients. We have presented how LOBCDER, which provides a standardized access protocol (WebDAV), may be integrated with the NEWQoSPlanner that uses SDN to optimize data transfers. The selection of optimal data sources introduces overhead and the degree to which the performance of a data transfer is affected by this overhead depends on the performance of the network, its size and number of available data sources. If the size of the data to be transferred is small with respect to the capacity of the link then searching for an optimal path will not improve performance. Similarly, if size of the network and number of resources to be considered is too large, the overhead of finding a solution to the MSSP problem will increase. Consequently, the solution presented here is intended for RIs with several consumers and sources and for transfer of large files. Avoiding the overhead for small file transfers is quite trivial and can be taken into consideration by modifying Eq. (4) with and an additional delaying factor. It is worth noticing that during data transfers, our solution transparently adapts to network traffic without any human intervention. It is achieved by taking advantage of SDN and rerouting traffic through less loaded paths and avoiding the influence of other applications using the network. This strategy has no overhead in initiating the data transfer.

With the proposed architecture, the traffic load is balanced between available resources. Introducing additional objectives to the cost function enables to take into consideration issues such as energy consumption and budget costs. More advanced SDN techniques should be used to address the challenge of discovering cross-domain network topologies and to provide information and control for the bandwidth allocation policy. As SDN technologies and approaches mature so will the reliability accuracy and specifications of these systems. Several publications such as [50] indicate that there is currently research done towards that direction. Such an approach would have a positive impact on big data disaster recovery [51,52].

Acknowledgments

We would like to thank Stavros Konstantaras for valuable discussions and Daniel F.C. Romao for his help in deploying the experimental topology on the ExoGENI testbed. We like to thank Victor Chang (Leeds Beckett University UK) and the reviewers for many valuable suggestions and recommendations. This work was partly funded by the VPH-Share[3] and COMMIT[4] projects. Zhiming Zhao acknowledges funding by ENVRIPLUS and SWITCH H2020 projects.

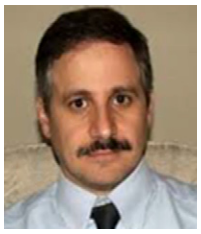
References

[1] P. Ayris, R.D.W. Group, et al. Leru roadmap for research data.
[2] A. Wöhrer, P. Brezany, I. Janciak, E. Mehofer, Modeling and optimizing large-scale data flows, Future Gener. Comput. Syst. 31 (0) (2014) 12–27. http://dx.doi.org/10.1016/j.future.2013.10.004. special Section: Advances in Computer Supported Collaboration: Systems and Technologies.
[3] J. Faghmous, A. Banerjee, S. Shekhar, M. Steinbach, V. Kumar, A. Ganguly, N. Samatova, Theory-guided data science for climate change, Computer 47 (11) (2014) 74–78. http://dx.doi.org/10.1109/MC.2014.335.
[4] The human brain project (hbp), https://www.humanbrainproject.eu/, [Online; accessed 4-December-2014] (2014).
[5] S. Benkner, C. Borckholder, M. Bubak, Y. Kaniovskyi, R. Knight, M. Koehler, S. Koulouzis, P. Nowakowski, S. Wood, A cloud-based framework for collaborative data management in the vph-share project, in: Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on, IEEE, 2013, pp. 1203–1210.
[6] D. Thilakanathan, S. Chen, S. Nepal, R. Calvo, L. Alem, A platform for secure monitoring and sharing of generic health data in the cloud, Future Gener. Comput. Syst. 35 (0) (2014) 102–113. http://dx.doi.org/10.1016/j.future.2013.09.011. special Section: Integration of Cloud Computing and Body Sensor Networks; Guest Editors: Giancarlo Fortino and Mukaddim Pathan.
[7] V. Chang, The business intelligence as a service in the cloud, Future Gener. Comput. Syst. 37 (0) (2014) 512–534. http://dx.doi.org/10.1016/j.future.2013.12.028. special Section: Innovative Methods and Algorithms for Advanced Data-Intensive Computing Special Section: Semantics, Intelligent processing and services for big data Special Section: Advances in Data-Intensive Modelling and Simulation Special Section: Hybrid Intelligence for Growing Internet and its Applications.
[8] R. Agarwal, G. Juve, E. Deelman, Peer-to-peer data sharing for scientific workflows on amazon ec2, in: High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion, 2012, pp. 82–89. http://dx.doi.org/10.1109/SC.Companion.2012.23.
[9] L. Wang, J. Tao, R. Ranjan, H. Marten, A. Streit, J. Chen, D. Chen, G-hadoop: Mapreduce across distributed data centers for data-intensive computing, Future Gener. Comput. Syst. 29 (3) (2013) 739–750. http://dx.doi.org/10.1016/j.future.2012.09.001. special Section: Recent Developments in High Performance Computing and Security.

[3] http://www.vph-share.eu/.
[4] http://www.commit-nl.nl/

[10] M.E. Vairavanathan, S. Al-Kiswany, A. Barros, L.B. Costa, H. Yang, G. Fedak, Z. Zhang, D.S. Katz, M. Wilde, A case for workflow-aware storage: An opportunity study using mosastore, Submitted to FGCS Journal.
[11] R. Tudoran, A. Costan, O. Nano, I. Santos, H. Soncu, G. Antoniu, Jetstream: Enabling high throughput live event streaming on multi-site clouds, Future Gener. Comput. Syst. (0) (2015) – http://dx.doi.org/10.1016/j.future.2015.01.016.
[12] A. Simonet, G. Fedak, M. Ripeanu, Active data: A programming model to manage data life cycle across heterogeneous systems and infrastructures, Future Gener. Comput. Syst. 53 (2015) 25–42. http://dx.doi.org/10.1016/j.future.2015.05.015.
[13] A. Sim, A. Shoshani, The storage resource manager interface specification, version 2.2, in: CERN, FNAL, JLAB, LBNL and RAL, Citeseer, 2007.
[14] T.I. Mandrichenko, W. Allcock, Gridftp v2 protocol description (May 2005).
[15] Cloud Data Management Interface, Tech. rep. (March 2010).
[16] J. Chen, A. Choudhary, S. Feldman, B. Hendrickson, C. Johnson, R. Mount, V. Sarkar, V. White, D. Williams, Synergistic Challenges in Data-intensive Science and Exascale Computing, DOE ASCAC Data Subcommittee Report, Department of Energy Office of Science.
[17] M. Kluge, S. Simms, T. William, R. Henschel, A. Georgi, C. Meyer, M.S. Mueller, C.A. Stewart, W. Wünsch, W.E. Nagel, Performance and quality of service of data and video movement over a 100 gbps testbed, Future Gener. Comput. Syst. 29 (1) (2013) 230–240. http://dx.doi.org/10.1016/j.future.2012.05.028. including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures..
[18] D. Kreutz, F. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proc. IEEE 103 (1) (2015) 14–76. http://dx.doi.org/10.1109/JPROC.2014.2371999.
[19] A. Hakiri, A. Gokhale, P. Berthou, D.C. Schmidt, T. Gayraud, Software-defined networking: Challenges and research opportunities for future internet, Comput. Netw. 75 (2014) 453–471. http://dx.doi.org/10.1016/j.comnet.2014.10.015. Part A (0).
[20] H. Farhady, H. Lee, A. Nakao, Software-defined networking: A survey, Comput. Netw. 81 (0) (2015) 79–95. http://dx.doi.org/10.1016/j.comnet.2015.02.014.
[21] M. Shirazipour, Y. Zhang, N. Beheshti, G. Lefebvre, M. Tatipamula, Openflow and multi-layer extensions: Overview and next steps, European Workshop on Software Defined Networking (EWSDN) 0 (2012) 13–17. http://doi.ieeecomputersociety.org/10.1109/EWSDN.2012.22.
[22] Network service interface, http://forge.ogf.org/sf/projects/nsi-wg/, [Online; accessed 1-July-2014] (2014).
[23] D. Kotani, K. Suzuki, H. Shimonishi, A design and implementation of openflow controller handling ip multicast with fast tree switching, in: 2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet, vol. 0, 2012, pp. 60–67. http://doi.ieeecomputersociety.org/10.1109/SAINT.2012.17.
[24] J. Gu, D. Katramatos, X. Liu, V. Natarajan, A. Shoshani, A. Sim, D. Yu, S. Bradley, S. McKee, Stornet: Co-scheduling of end-to-end bandwidth reservation on storage and network systems for high-performance data transfers, in: Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on, 2011, pp. 121–126. http://dx.doi.org/10.1109/INFCOMW.2011.5928792.
[25] B. Gibbard, D. Katramatos, D. Yu, Terapaths: End-to-end network path qos configuration using cross-domain reservation negotiation, in: Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on, 2006, pp. 1–9. http://dx.doi.org/10.1109/BROADNETS.2006.4374426.
[26] R. Jones, D. Barberis, The atlas computing model, J. Phys. Conf. Ser. 119 (7) (2008) 072020. http://stacks.iop.org/1742-6596/119/i=7/a=072020.
[27] S. Sharma, D. Katramatos, D. Yu, L. Shi, Design and implementation of an intelligent end-to-end network qos system, in: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '12, IEEE Computer Society Press, Los Alamitos, CA, USA, 2012, pp. 68:1–68:11. http://dl.acm.org/citation.cfm?id=2388996.2389089.
[28] L. Zuo, M. Zhu, Concurrent bandwidth reservation strategies for big data transfers in high-performance networks, Network and Service Management, IEEE Trans. PP Netw. Serv. Manag. (99) (2015) 1–1. http://dx.doi.org/10.1109/TNSM.2015.2430358.
[29] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, P. Tran-Gia, Sdn-based application-aware networking on the example of youtube video streaming, in: Software Defined Networks (EWSDN), 2013 Second European Workshop on, 2013, pp. 87–92. http://dx.doi.org/10.1109/EWSDN.2013.21.
[30] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, N. Race, Towards network-wide qoe fairness using openflow-assisted adaptive video streaming, in: Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking, FhMN '13, ACM, New York, NY, USA, 2013, pp. 15–20. http://dx.doi.org/10.1145/2491172.2491181.
[31] G. Cofano, L. De Cicco, S. Mascolo, A control architecture for massive adaptive video streaming delivery.
[32] X. Cheng, C. Dale, J. Liu, Statistics and social network of youtube videos, in: Quality of Service, 2008. IWQoS 2008. 16th International Workshop on, IEEE, 2008, pp. 229–238.
[33] Z. Yu, M. Li, X. Yang, X. Li, Palantir: Reseizing network proximity in large-scale distributed computing frameworks using sdn, in: Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on, 2014, pp. 440–447. http://dx.doi.org/10.1109/CLOUD.2014.66.
[34] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, B. Lyon, Design and Implementation or the Sun Network Filesystem (1985).
[35] R. Hat, Glusterfs, http://www.gluster.org/, [Online; accessed 19-June-2015] (2015).
[36] J.M. Lucas, M.S. Saccucci, R.V. Baxley Jr., W.H. Woodall, H.D. Maragh, F.W. Faltin, G.J. Hahn, W.T. Tucker, J.S. Hunter, J.F. MacGregor, T.J. Harris, Exponentially weighted moving average control schemes: Properties and enhancements, Technometrics 32 (1) (1990) 1–29. http://dx.doi.org/10.2307/1269835.
[37] M. Barbehenn, A note on the complexity of dijkstra's algorithm for graphs with weighted vertices, IEEE Trans. Comput. 47 (2) (1998) 263. http://dx.doi.org/10.1109/12.663776.
[38] S. Koulouzis, D. Vasyunin, R. Cushing, A. Belloum, M. Bubak, Cloud data federation for scientific applications, in: D. an Mey, M. Alexander, P. Bientinesi, M. Cannataro, C. Clauss, A. Costan, G. Kecskemeti, C. Morin, L. Ricci, J. Sahuquillo, M. Schulz, V. Scarano, S. Scott, J. Weidendorfer (Eds.), Euro-Par 2013: Parallel Processing Workshops, in: Lecture Notes in Computer Science, vol. 8374, Springer, Berlin Heidelberg, 2014, pp. 13–22. http://dx.doi.org/10.1007/978-3-642-54420-0_2.
[39] Z. Zhao, C. Dumitru, P. Grosso, C. de Laat, Network resource control for data intensive applications in heterogeneous infrastructures, in: Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International, 2012, pp. 2069–2076. http://dx.doi.org/10.1109/IPDPSW.2012.243.
[40] S. Konstantaras, A. Oprescu, Z. Zhao, PIRE ExoGENI–ENVRI preparation for Big Data science.
[41] Openstack-swift, https://wiki.openstack.org/wiki/Swift, [Online; accessed 18-Mar-2015] (2015).
[42] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Tuecke, Gridftp: Protocol extensions to ftp for the grid, Global Grid ForumGFD-RP 20.
[43] C. Dumitru, P. Grosso, C. de Laat, A user-centric execution environment for cinegrid workloads, Future Gener. Comput. Syst. 53 (0) (2015) 55–62. http://dx.doi.org/10.1016/j.future.2015.03.021.
[44] G. Roberts, T. Kudoh, I. Monga, J. Sobieski, J. Vollbrecht, Network Services Framework V1.0, Tech. Rep. GFD 173 (2010) http://www.gridforum.org/documents/GFD.173.pdf.
[45] I. Baldine, Y. Xin, A. Mandal, P. Ruth, C. Heerman, J. Chase, Exogeni: A multi-domain infrastructure-as-a-service testbed, in: Testbeds and Research Infrastructure, Development of Networks and Communities, Springer, 2012, pp. 97–113.
[46] R. Koning, P. Grosso, C. de Laat, Using ontologies for resource description in the cinegrid exchange, Future Gener. Comput. Syst. 27 (7) (2011) 960–965. http://dx.doi.org/10.1016/j.future.2010.11.027. cineGrid: Super high definition media over optical networks..
[47] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, S. Shenker, Extending networking into the virtualization layer, in: Hotnets, 2009.
[48] J.C. Jacob, D.S. Katz, G.B. Berriman, J. Good, A.C. Laity, E. Deelman, C. Kesselman, G. Singh, M.-H. Su, T.A. Prince, R. Williams, Montage: A grid portal and software toolkit for science-grade astronomical image mosaicking, Int. J. Comput. Sci. Eng.
[49] Eso online digitized sky survey, http://archive.eso.org/dss/dss, [Online; accessed 19-July-2015] (2015).
[50] F. Botelho, F. Valente Ramos, D. Kreutz, A. Bessani, On the feasibility of a consistent and fault-tolerant data store for sdns, in: Software Defined Networks (EWSDN), 2013 Second European Workshop on, 2013, pp. 38–43. http://dx.doi.org/10.1109/EWSDN.2013.13.
[51] V. Chang, Towards a big data system disaster recovery in a private cloud, Ad. Hoc. Netw. (2015) – http://dx.doi.org/http://dx.doi.org/10.1016/j.adhoc.2015.07.012http://www.sciencedirect.com/science/article/pii/S157087051500147X.
[52] S. Sengupta, K. Annervaz, Multi-site data distribution for disaster recovery — A planning framework, Future Gener. Comput. Syst. 41 (2014) 53–64. http://dx.doi.org/10.1016/j.future.2014.07.007. http://www.sciencedirect.com/science/article/pii/S0167739X1400140X.

Spiros Koulouzis, has B.Sc. degree in Electronic Computing Systems conferred September 2004 by Technical Educational Institute of Piraeus. He received an M.Sc. degree in Intelligent and Multi-Agent Systems conferred October 2006 by University of Westminster and M.Sc. in Grid Computing conferred March 2010 by University of Amsterdam. He is currently a Ph.D. candidate at the University of Amsterdam, and his research interests include distributed and parallel systems.

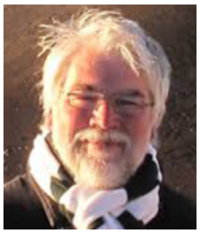


Adam Belloum, is an Assistant Professor at the computer science department of the University of Amsterdam. He received the M.Sc. and Ph.D. degrees from the Compiegne University of Technology, France.

Marian Bubak, has M.Sc. degree in Technical Physics and Ph.D. in Computer Science. He is an adjunct at the Department of Computer Science and Cyfronet, AGH University of Science and Technology, Krakow, Poland, and a Professor of Distributed System Engineering at the University of Amsterdam.

Zhiming Zhao obtained his PhD in computer science in 2004 from University of Amsterdam (UvA). He is currently a senior researcher in the System and Network Engineering group at UvA. He coordinates research effort on quality critical systems on programmable infrastructures in the context of European H2020 projects of SWTICH and ENVRIPLUS. His research interests include software defined networking, workflow management systems, multi agent system and big data research infrastructures.

Miroslav Zivkovic received the engineering degree in electronics and telecommunications from the Faculty of Electrical Engineering, University of Belgrade, Serbia, and his PhD degree from University of Twente, The Netherlands. From 1999 till 2008 he was with Bell Labs, Alcatel–Lucent, The Netherlands, where he contributed to the area of dynamic spectrum management for DSL systems, intelligent service platforms and security solutions for the next generation networking. In 2008 he joined TNO, The Netherlands, where he was mainly involved in performance analysis of service oriented architecture systems and composite web services. Since 2013 he is with System and Network Engineering (SNE) group, Institute for Informatics, University of Amsterdam (UvA), where he works on different aspects of Software—Defined Networks and data centre infrastructures.



Prof. Dr. Ir. Cees de Laat is chair of the System and Network Engineering research group at the University of Amsterdam. Research in his group includes optical/switched Internet for data-transport in TeraScale eScience applications, Semantic web to describe networks and associated resources, distributed cross organization Authorization architectures and Systems Security privacy of information in distributed environments. He serves as at Large member of the Board of Directors in Open Grid Forum and is acting co-chair of the Grid High Performance Networking Research Group (GHPN-RG), is chair of GridForum.nl and boardmember of ISOC.nl. He is co-founder and organizer of several of the past meetings of the Global Lambda Integrated Facility (GLIF) and founding member of CineGrid.org.