# RADII: Bridging the Divide Between Data and Infrastructure Management to Suport Data-Driven Collaborations

A RENCI Technical Report

TR-16-01

**Claris Castillo & Charles Schmitt**
Renaissance Computing Institute (RENCI)
University of North Carolina at Chapel Hill
claris@renci.org | cschmitt@renci.org

**Fan Jiang**
Department of Computer Science
University of North Carolina at Chapel Hill
dcvan@csunc.edu

renci

www.renci.org

# *RADII* : Bridging the Divide between Data and Infrastructure Management to Support Data-Driven Collaborations

Fan Jiang
*Department of Computer Science*
*University of North Carolina-CH*
*Chapel Hill, North Carolina, US*
*dcvan@cs.unc.edu*

Claris Castillo, Charles Schmitt
*Renaissance Computing Institute*
*University of North Carolina-CH*
*Chapel Hill, North Carolina, US*
*claris@renci.org, cschmitt@renci.org*

***Abstract*—We have witnessed a dramatic increase in national cyberinfrastructure resources to support data-driven research. Orchestrating these resources to enable the creation of *collaborative infrastructure* capable of supporting data intensive activities is challenging. In this work we present RADII, a novel architecture and system that enables the provisioning and configuration of *collaborative infrastructure* by orchestrating data and infrastructure management in an integrated manner. We also introduce a cross-layer data annotation mechanism that together with Software-Defined-Networking (SDN) support allows the embedding of user-defined performance policies into file metadata which are translated into executable optimal network plans. We have deployed RADII on a worldwide production testbed and demonstrated through experimentation that RADII can improve network throughput of data transfers by 28.2% as compared to conventional approaches.**

***Keywords*-data-centric infrastructure; cyberinfrastructure; ExoGENI; iRODS; SDN**

## I. Introduction

We have recently witnessed a dramatic increase in national data services (NCBI [26]) along with national funded initiatives (NSF BDHUB [19]) to foster the development and sustainability of data-centric cyber communities. Complementing this trend are the large number of compute cyberinfrastructure efforts that seek to enable the provisioning of virtual compute infrastructure tailored to application needs (GENI [13]), compute platforms to run scientific workloads (OSG [28]), open-Cloud platforms (Cloudlab [30], Chameleon [5]) to enable the deployment of private Clouds within public Clouds, high-performance-compute (HPC) systems for supporting medium to large scale jobs (COMET [10]), among others. All these efforts have the potential of allowing scientists to run data-intensive workloads at scales not possible before.

The main goal of such cyberinfrastructure resources is to enable communities that support new models of data-driven research and education [11]. To achieve this we need to orchestrate the integration of these advanced technologies, data and scientists into unified *collaborative infrastructures* that scientists can use efficiently, with ease and with reasonable levels of performance. In this work, a *collaborative infrastructure* consists of durable compute and storage resources potentially spanning across multiple institutions and sites to support all aspects of a data-driven collaboration. As such, it must support a broad range of data activities tailored to specific needs, from transferring input data sets hosted in national data services, to analyzing intermediate results in national high performance computing (HPC) machines and sharing results in distributed data-sharing infrastructure. We face multiple challenges to realize this vision. Firstly, lack of dedicated network infrastructure between distributed cyberinfrastructure resources constraint scientists to transfer files over commodity Internet – a practice that results in long transfer times and hinders scientific productivity. More sophisticated approaches based on regional network providers, e.g., Internet2 [20] are possible but require the involvement of IT staff and are time consuming. More importantly, these approaches are fairly static and therefore fail to adapt to the dynamic nature of scientific collaborations. Consider the use case where network connectivity is required between local resources at a new collaborating institution and an existing *collaborative infrastructure*; this effort would demand complex logistics between the IT staff. Secondly, data management technologies that enable the sharing of data across sites and institutions, such as integrated Rule Oriented Data System (iRODS) [29] and Globus [14], are oblivious to the physical infrastructure underneath. This limits the level of control that scientists have on such environments and the optimization opportunities in the overall system. Such a design decision is typically made to hide the complexity of the underlying systems, however, it also limits, as an example, the scientists' ability to control the placement of data and the support for network-aware data replication.

We believe that the enabling of truly data-centric collaborations in the scientific community requires the design and development of novel data-driven software infrastructure, where the networking, storage, data and computing systems function together as a unified infrastructure [32]. We note that whereas there exists software solutions to manage and control data (access and storage), such as iRODS, and the network, such as Software-Defined-Networking (SDN), the boundaries that currently exist between these domains pre-

National Science Foundation ACI Award 1440715

vent the exercise of centralized control across these multiple domains. In this work, we propose a novel architecture and system named Resource Aware DatacentrIc Infrastructure (*RADII*) that bridges this gap by allowing users to programmatically create durable *collaborative virtual infrastructure* to support data-centric collaborations through out their entire life-cycle. We accomplish this by orchestrating the provisioning of such infrastructure and configuration of the data management plane to satisfy the requirements of scientific collaborations, and by enabling fine-grained control over data transfers driven by user-defined policies. *RADII* offers a unique *cross-layer data annotation mechanism* that allows users to encode security and performance policies for each file into the file's metadata. We rely on SDN to translate these policies into network forwarding rules, which ultimately optimize for user and system requirements. on the importance of files while optimizing for network utilization.

In summary, this paper makes multiple contributions: (1) A novel extensible service architecture that integrates data and infrastructure management. (2) A novel approach based on cross-layer data annotation that allow users to encode performance and security policies in file metadata, which are translated into data and network management operations. (3) An implementation of a system, named *RADII*, that demonstrates the feasibility of the approach. *RADII* is deployed on a production national testbed (ExoGENI) consisting of 20 sites distributed worldwide and builds on existing open source Open Resource Control Architecture (ORCA) [6] and data (iRODS) management technologies. All the source code of our project is available online at [1] and a test prototype implementation is available at [2].

The rest of the paper is organized as follows. In Section II we provide some background on the technologies that we use to build *RADII*. We introduce iRODS, ExoGENI and SDN in Section II. We describe the architecture that enables the provisioning of *collaborative infrastructure* in detail in Section III. Following, we outline our novel cross-layer data annotation mechanism and its interaction with SDN to enable the deep integration of data and network management. In Section V we perform experiments on a real production testbed to demonstrate the effectiveness of our cross-layer data annotation mechanisms and SDN to satisfy user-defined performance on file transfers. In Section VI we discuss some of the most relevant work related to ours. Finally, in Section VII we discuss future directions.

## II. Background

In this section we describe three core technologies of *RADII*. Our goal is to provide enough detail to help the reader understand our approach to the best possible.

### A. ExoGENI

The ExoGENI testbed consists of 20 worldwide distributed Cloud sites on campuses, linked with national research networks through programmable exchange points. Distributed resources are offered in connected topologies, called *slices*, to form mutually isolated, networked virtual infrastructures for solving specific problems. ExoGENI obtains compute and storage from private Cloud sites at the edge, and network resources from edge providers and national fabrics using VLAN-based switching and OpenFlow. With ORCA, ExoGENI offers a powerful, unified hosting platform for deeply networked, multi-domain, multi-site applications. ExoGENI is distinguished from other Cloud platforms, e.g., AmazonEC2 [12] with the following advantages: 1) it is able to allocate bandwidth-provisioned, dedicated layer-2 private networks between resources across sites; 2) it enables modification to slices dynamically, allowing users to modify resource capacities and networking topology of the infrastructure; 3) it provides a programmatic interface that uses Network-Descriptive-Language (NDL) to allow the clean integration with applications. These features are essential for *RADII* to enable the provisioning of durable virtual infrastructure and couple infrastructure and data management. As such, ExoGENI is an exemplary of a worldwide distributed and federated compute environment that with the appropriate middleware software can address the challenges discussed in Section I.

### B. iRODS

iRODS federates distributed, heterogeneous data systems into a single logical file system and grants data access through a uniform client API. The core of iRODS is a rule engine that supports sophisticated data management logic and operations using data rules, including archive processes, publication and dissemination processes, analysis, synthesis and access processing. The programmability of the data rules facilitates the integration of iRODS with advanced management logic. In addition, iRODS owns a metadata system that supports user-defined metadata. The adoption of metadata in iRODS complements rule-oriented data management by enabling more flexible data rule enforcement. A typical iRODS deployment consists of an *iCAT* server, which stores system status and metadata, and a number of peer *resource* servers, which maintains data files. Rule engines run on resource servers to ensure rule enforcement. The unique programmability and metadata support of iRODS make it our best choice for integrating data and infrastructure management in *RADII*.

### C. SDN Network

In wide area network environments data transfers experience poor network performance caused by high latency, packet loss and network congestion. Advanced networking techniques [24] try to address the issues but are restricted by the rigid architecture of traditional networks, where routing algorithms are hardcoded and can only rely on local information available on switches. These static routing schemes are inefficient in reacting to frequent network state changes therefore, they can seldom mitigate the impact that

high latency and packet loss has on applications. Moreover, traditional networking techniques are application-oblivious and therefore can't optimize performance for application requirements.

SDN addresses these challenges by enabling dynamic, programmable network configuration tailored to service needs SDN detaches the hardcoded control plane from the network switches and uses *controllers* deployable on servers to manage the network in a centralized manner. Within the *controllers*, custom network planning algorithms can be developed to support global network optimization. In addition, the *controllers* can obtain application requirements via its custom northbound API to drive network management operations. The *controller* communicates with the network switches using the OpenFlow [23] protocol via its southbound API. An SDN network can be constructed from virtual resources with VMs configured as software switches [27] and *controllers* connected through dedicated virtual network links. In view of all these qualities we believe that SDN technologies present a unique opportunity to resolve the performance issues impeding the adoption of distributed cyberinfrastructure resources for supporting data-driven collaborations.

## III. RADII

*RADII* allows users to programmatically create durable *collaborative infrastructure* to support data-centric collaborations through out their life-cycle. Central to our approach is the orchestration of the provisioning and configuration of the virtual infrastructure and the data management plane in an integrated fashion. In addition, *RADII* offers a unique cross-layer data annotation mechanism that allow users to encode security and performance policies in metadata associated with files. Following in this section we describe the architectural aspects that enable the provisioning and configuration of *collaborative infrastructure*. Later in Section IV we introduce our novel cross-layer data annotation mechanism in detail.

### A. RADII Architecture

RADII's software architecture (Figure 1) follows the Service-Oriented Architecture (SOA) model and consists of a set of three loosely coupled services: *Collaboration Language User InterfacE (CLUE)*, *Orchestrator*, *Compute To Infrastructure (C2I)* and *Data To Infrastructure (D2I)*. Each service provides a subset of the system's functionality and operates independently. We note that the architecture lays over an Infrastructure-as-a-Service (IaaS) Cloud model but the architecture is generic and is amenable to different models.

### B. Collaboration representation and data model

We use a dataflow formalism to represent data-centric collaborations. A dataflow diagram (DFD) is a graphical representation of the flow of data through an information

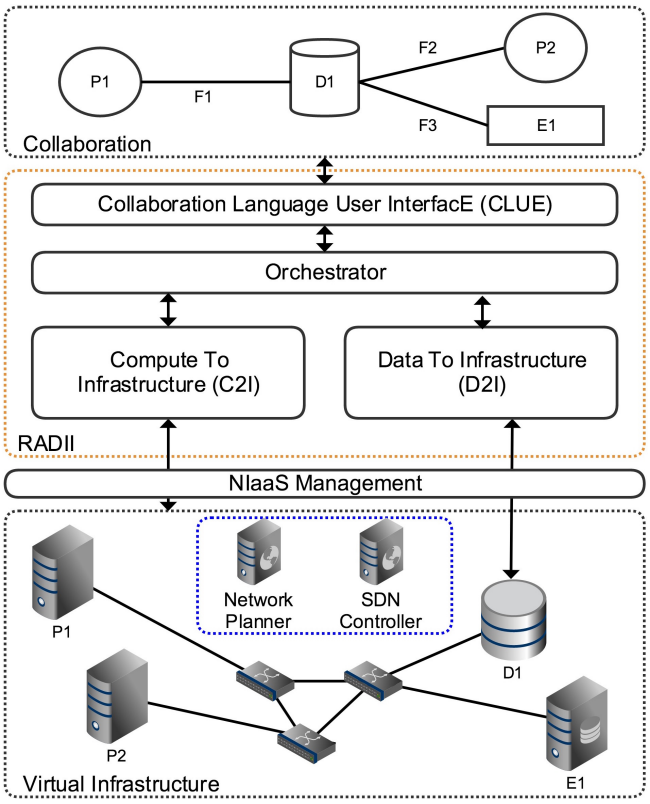

Figure 1: RADII design

system, modeling its process aspects. DFD is a model widely adopted by scientists to describe complex data activities without having to dwell into infrastructure and implementation issues. A DFD consists of four types of entities (refer to Figure 1): *processes* (P1 in figure; activities that transform data from one form to another), *datastores* (D1 in figure; where the data is stored), *externals* (E1 in figure; what sends data into a system or receive data from the system) and *dataflows* (F1-F3 in figure; routes by which data can flow). In *RADII* these entities are high-level artifacts that are mapped to underlying physical resources. In a *collaborative infrastructure* a *process* is a compute artifact that performs computational tasks, e.g., a virtual machine; a *datastore* represents a storage resource to host data at rest, e.g., VMs configured with iRODS; an *external* entity corresponds to a system outside the collaboration acting as the input or output of the collaboration, *e.g.*, storage facility hosting a public data set; and, a *dataflow* corresponds to network connectivity between entities, *e.g.*, virtual network links deployed across multiple network providers. The box labeled `DFD` in Figure 3 illustrates an example of a DFD model and its corresponding underlying physical representation. Note how each entity is mapped to a pre-configured *specialized infrastructure* dedicated to support tasks in the scope of the entity. For instance, a *process* for genomic analysis can be mapped to an inter-connected virtual cluster of VMs pre-configured with genomic analytic software. Specialized infrastructures are connected by *dataflow* entities (networks links) to form

an end-to-end virtual infrastructure capable of supporting all activities related to the collaboration.

**Infrastructure Management Attributes** Each entity is associated with *attributes* that describe its underlying physical representation in detail. For instance, the `location` attribute in a process or a datastore alludes to the specific site where the underlying resources should be deployed. If the process is implemented with multiple resources, e.g., cluster of VMs, the attribute is expanded to support multiple entries. These attributes are then translated to actions by the infrastructure during the resource provisioning cycle. Examples of other attributes associated to processes are `number of VMs`, `number of cores`, `RAM size`, `storage size`, *etc*. `bandwidth` and `anti-locations` are attributes associated with a dataflow and refer to the bandwidth that should be allocated to a network path and network providers that must be bypassed when computing the (shortest) path between two end points, respectively.

**Data Management Attributes** A datastore entity is associated with a rich set of attributes that capture data governance within the collaboration. In *RADII* we choose to use iRODS due to its ability to support complex data policies enforced by a rule engine. Therefore, in a *collaborative infrastructure* a datastore is always mapped into an iRODS grid which attributes can be tailored to the collaboration. Common attributes are `locations of resource nodes`, `number of resource nodes`, `capacity of each resource node`, *etc*.

**Location-aware Data Access Control** We consider a simple yet powerful data access control strategy to develop our first prototype. As illustrated in Figure 3, a data policy consists of a number of *tags*, each of which has a user-defined name (*e.g.*, red) and a list of predefined data access operations (*e.g.*, create, read, update, delete). We leverage the metadata capability of iRODS to allow users to encode tags into the metadata associated with each file. In addition, each *(tag,operation)* tuple is associated with an access control list (ACL) with the identifiers of the collaborators authorized to perform the specific operation on files tagged with the given *tag*. For instance, as shown in Figure 3, files associated with *tag* `red` can be only *read* by `charles` and `fan`. The strategy is novel in that it is aware of the physical location of the nodes hosting the data. As shown in the figure, each *tag* can specify `location` for entries of the `create` operation to restrict tagged data items from being created on *datastores* deployed at unspecified Cloud sites, *e.g.*, data item with *tag* named `red` can only be created on *datastores* provisioned by Cloud site at Oakland, CA. This is a simple yet powerful example of *RADII* bridges infrastructure and data management into a unified control management framework seamlessly. It is important to notice that to achieve this capability with state-of-the-art technologies, e.g., iRODS out of the box, a user must have deep knowledge of the physical infrastructure.

### C. Collaborative Language User interfacE (CLUE)

*CLUE* (Figure 2) is the user interface for composing data-centric collaborations. In this interface a collaboration is represented by a *CLUE object* (Figure 3). A CLUE object is a graph-based representation of a DFD in human-readable format. To facilitate the composition of collaborations users are presented with templates (Figure 3) for each dataflow entity. A template is a partial layout of an entity with attributes set to sane default values and enables the level of customization that tech-savy scientists demand.

The CLUE interface exposes a RESTful API and a graphical user interface (GUI) (see Figure 2) via which CLUE objects –in JavaScript Object Notation (JSON) format– are passed onto the *Orchestrator* service for deployment and provisioning.

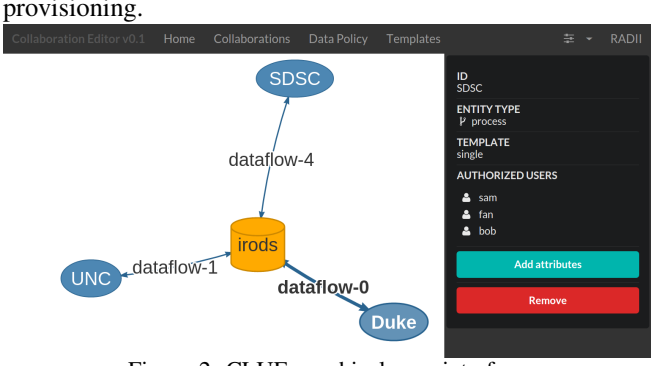


Figure 2: CLUE graphical user interface

### D. Orchestrator

The *Orchestrator* is responsible for orchestrating all the activities pertaining to managing the *collaborative infrastructure* through out its entire lifecycle, i.e., composition, instantiation, modification, query and destruction. The *Orchestrator* takes a *CLUE object* as input, extracts all the information related to the configuration of the infrastructure and data policies and submit them to *C2I* and *D2I*, respectively. To manage the collaboration through outs entire life-cycle it queries *C2I* and *D2I* periodically to obtain information about changes in the infrastructure (e.g., failed resource due to power outage) and data (e.g., storage capacity). State information about the *collaborative infrastructure* is propagated to users via web notifications.

### E. Compute to Infrastructure (C2I)

*C2I* handles all configuration and management aspects of the compute virtual infrastructure. It translates high-level resource requirements extracted from the DFD entities in the *CLUE object* into a concrete request consumable by the IaaS provider underneath. For example, it ensures that VMs have the proper resource capacities, e.g., RAM. As described in Section II we chose ORCA/ExoGENI as the IaaS provider due to its ability to provision durable *networked* infrastructure and its programmability. A built-in ORCA/ExoGENI client in C2I is responsible for creating a NDL request that includes all the requirements specified in the CLUE object

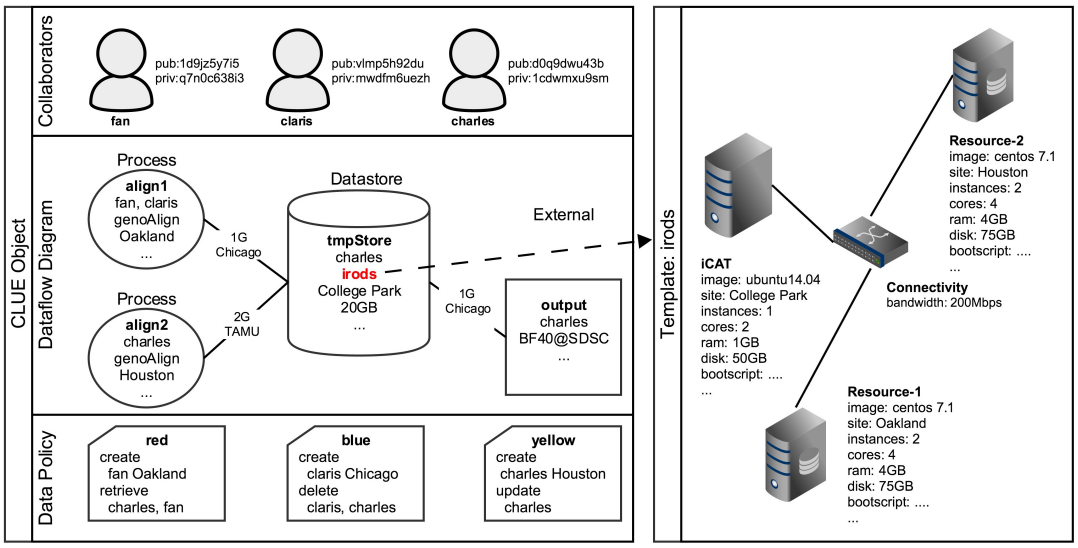

Figure 3: CLUE Object

and submitting it to ExoGENI. Once the request is served, i.e., virtual infrastructure is deployed, active and accessible, C2I continues managing the virtual infrastructure at run time including retrieval of state, modification of individual resource capacity and removal/addition of resources.

### F. Data to Infrastructure (D2I)

*D2I* is the homologous to C2I with respect to data management policies. It translates high-level data policies into rule artifacts that the underlying data management system understands. These rules are ultimately applied and enforced by the data management systems through out the life-cycle of the data.

As we explained in Section II, *RADII* uses iRODS as the underlying data management system. To support the access control mechanisms discussed earlier, D2I maintains an iRODS rule for every <*tag, operation*> tuple. Each rule is a procedure that specify the actions to be executed for the given tuple including the validation logic required. For example, as shown in Figure 3, `fan` is the only user allowed to create files with tag `red` on iRODS *resources* at Oakland site; thus the operation will fail if anyone other than `fan` attempts to perform this action. This mapping is defined at the time of creating the collaboration and maintained as state information in D2I; it can modified throughout the life-cycle of the collaboration. Note that scientists don't have to write the rules, these are pre-coded in advanced and built-into D2I by a system administrator and can be modified at run-time.

## IV. Cross-Layer Data Annotation Mechanism

In the previous sections we described how *RADII* provides an unified interface via which users can describe data and infrastructure management aspects of a collaborative infrastructure, enabling the encoding of low-level infrastructure constraints such as data placement in files metadata. In this section we describe a feature that builds on this capability to allow scientists to encode user-defined performance policies into files' metadata and to enable the translation of these policies into routing and bandwidth allocation plans that meet user and system performance goals.

### A. Dynamic SDN Network

Recall from Section II that data transfers over the wide area network experience poor performance due to high latency exacerbated by network congestion. To mitigate the effect that these conditions have on data transfers *RADII* relies on SDN. More specifically, iRODS resource nodes are connected through a SDN-able network capable of differentiating file transfer intelligently. In our first prototype we support user-defined prioritization of file transfers. Users encode a priority class *low*, *medium*, *high* to a file to indicate the importance that transferring that file has over other transfers in the network. The SDN network maintains a mapping of these priorities to algorithmic artifacts that drive the estimation of bandwidth allocation and routing plans whenever the file is transferred over the SDN network. Later in this section we describe how SDN relies on conventional networking planning algorithms to compute these estimation, translate them into forwarding rules and inject them into the network switches for enforcement.

**SDN Network:** Figure 4 depicts a typical deployment of a SDN network connecting iRODS resource nodes in a collaborative infrastructure created via *RADII*. There is one iRODS resource node in each Cloud site. Each of them is connected to the software switch at the edge of the Cloud site; switches are inter-connected using inter-rack virtual network links. The software switches are controlled by the *SDN controller* to route network traffic. The iRODS *resources* are able to communicate with the *SDN controller* using its RESTful northbound API (Figure 5) to exchange information about data transfers. The *SDN controller* orchestrates data transfer requests and the injection of rules

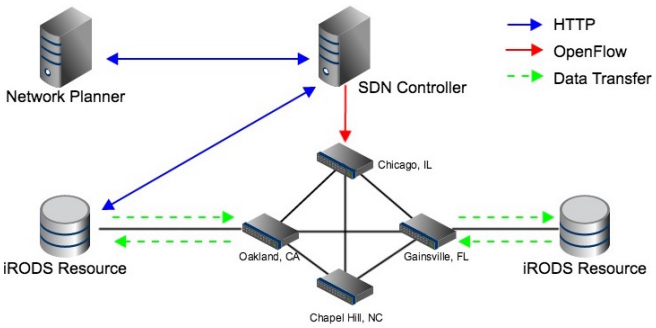

Figure 4: The SDN network

into the software switches, but offloads the network planning including routing and bandwidth allocation to the *Network Planner*. The *Network Planner* is a dedicated, stateless optimization solver deployed as a service on a separate VM host.

**End-to-end workflow:** To initiate a data transfer, the iRODS *resource* passes into the *SDN controller* context information about the transfer and the state of the network; a context includes file metadata, bandwidth demands, IP addresses/ports, *etc*. The *SDN controller* uses the information as the input to the planning algorithm in the *Network Planner*, which generates a solution containing network path assignments and bandwidth allocation for each path. The solution is sent back to the *SDN controller* through its northbound API and translated into OpenFlow rules which effectively executes the routing plan. After the rules have been injected into the software switches, the *SDN controller* notifies the sending *resource* with the bandwidth allocation. The *resource* initiates the transfer and throttles its sending rate accordingly. Other active transfers may also have to readjust their sending rates to accommodate for the new request. Note that file transfers (flows) are uniquely identified and mapped to forwarding rules through a tuple that includes source/destination IP addresses and source/destination port numbers.

| | Entry | Method | Note |
|---|---|---|---|
| iRODS client | /limit | PUT | set a transfer rate limit |
| | /limit/`[id]` | POST | update the rate limit |
| | | DELETE | remove the rate limit |
| SDN controller | /meta/`[tag]` | PUT | map a metadata tag to a priority class |
| | | DELETE | remove the metadata-priority mapping |
| | /transfer | PUT | add the transfer |
| | /transfer/`[id]` | DELETE | delete a transfer |
| | /algorithm/`[id]` | POST | set the routing algorithm for the SDN network |
| Graph Solver | /algorithm/`[id]` | POST | invoke the routing algorithm |

Figure 5: Common APIs in the SDN network

**Network Allocation and Routing Algorithm:** SDN networks offer a high level of flexibility in using custom network algorithms. This is particularly useful in our context to customize network management to the needs of the scientific community it serves. Our *SDN controller* offers a simple API (Figure 5) for network planning configuration including the mapping of metadata tags to priority values. The *SDN controller* keeps this mapping to perform priority assignment upon receiving file transfer requests.

```
{"priority": 1,
 "classes": [1, 2, 4],
 "demands":{
  "s1,s2":[500],
  "s2,s3":[200]
},
 "network":{
  "s1,s2":[300,21.3,0.1],
  "s2,s3":[600,48.6,0.25],
  "s1,s3":[300,12.3,0.15]
}}
```

(a) input

```
{"s1,s2":
 {
  "s1,s2":[300],
  "s1,s3,s2":[200]
 },
 "s2,s3":{
  "s2,s3":[200]
 }
}
```

(b) output

Figure 6: An example of *Graph Solver's* input/output

In our first prototype we use the planning algorithm shown in Figure 7 to compute bandwidth allocations while respecting file transfer prioritization. The algorithm is invoked upon every file transfer request received by the *SDN controller*. It iterates over priority values represented by non-negative integers from 0 to a predefined maximum value. For each priority $p$, the algorithm aggregates bandwidth demands of data transfers with priority $p$ (Line 3). It then invokes the custom routing algorithm on the *Network Planner* (denoted by `solve()`) with current network capacity (Line 4), to calculate path and bandwidth allocation. The allocated paths and bandwidth will be divided among data transfers with priority $p$ in a max-min fairness manner (Line 5). At the end of each iteration, the algorithm updates the network capacity by deducting the bandwidth allocation from each used path (Line 6), so that allocation for data transfers with priority $p + 1$ will be calculated based on the updated network capacity in the next iteration.

In Figure 8, we show the the multicommodity flow (MCF) algorithm proposed in [18] (Figure 8) as an example of a custom routing algorithm, which optimizes for maximum network throughput while preferring shorter paths. Note that the inputs are denoted consistently with those in Figure 7. Although the algorithm optimizes path and bandwidth allocation, it does not assign data transfers to paths, i.e., randomly assignment. To address this, we extend MCF to have transfers of high-priority files favor shorter paths and hence, higher throughput. We use MCF+ to refer to this extension of MCF.

**Flexible and Extensible:** *RADII* administrators can provide new algorithms as long as these follow the input and output specifications shown in Figure 6. The input specifies information about data transfers and current network state. The `demands` field specifies the aggregate bandwidth demand between each pair of edge switches; the `network` field describes the bandwidth availability, latency and loss rate of each link. The output contains path assignments for each pair of edge switches and corresponding bandwidth allocation (in Mbps). In this example, data transfers from switch `s2` to `s3`

**Inputs:** $d_{i,(u,v)}$: demand of transfer $i$ from vertex $u$ to $v$; $\infty$ means no demand specified
$c_l$: capacity of link $l$
$w_j$: weight of path $j$ (*e.g.*, latency)
$I_{j,l}$: 1 if path $j$ uses link $l$ otherwise 0
$Pri$: priority classes
**Outputs:** $b_{i,(u,v)}$: allocation to transfer $i$ from edge $u$ to $v$; $\infty$ means no allocation specified

**Func** `allocate`**:**

$\forall l : c_l^{remain} \leftarrow c_l$
**for** $p \leftarrow 0$ **to** $|Pri|$ **do**
  $\{d_{(u,v)}\} \leftarrow \sum_{i \in Pri[p]} d_{i,(u,v)}$
  $\{b_{(u,v)}\} \leftarrow$ `solve` $(\{c_l^{remain}\}, \{w_j\}, \{d_{(u,v)}\}, \{I_{j,l}\}, Pri, p)$
  $\{b_{i,(u,v)}\} \leftarrow$ `maxMinFair` $(\{b_{(u,v)}\}, \{d_{i,(u,v)} | i \in Pri[p]\})$
  $c_l^{remain} \leftarrow c_l^{remain} - \sum_{(u,v),j} b_{(u,v),j} \cdot I_{j,l}$
**return** $\{b_{i,(u,v)}\}$

**Func** `maxMinFair` $(\{b_j\}, \{d_i\})$**:**

**if** $\infty \in \{b_j\}$ **then**
  $\forall j : b_j \leftarrow \frac{min(\{c_l^{remain} | I_{j,l}=1\})}{|Pri|}$
$fairrate \leftarrow \frac{\sum_j b_j}{|\{d_i\}|}$; $F \leftarrow \emptyset$
**foreach** $d_i$ **do**
  **if** $d_i > fairrate$ **then** $f_i \leftarrow fairrate$
  **else** $f_i \leftarrow d_i$
  $F \leftarrow F + \{i\}$
**return** $\{f_i | i \in F\}$

Figure 7: Allocation algorithm on *SDN controller*

are allocated a total of 200Mbps bandwidth along path `[s2, s3]`. In Section VII we discuss how we will build on this feature to continue our line of work.

## V. Evaluation

In this section we demonstrate that *RADII* is able to improve the performance of data transfers in data-centric collaborations thanks to the cross-layer data annotation mechanism in conjunction with SDN support. We evaluate the impact that network optimization has on the transfer of files in a typical iRODS deployment. Specifically, we implement the MCF algorithm proposed in [18] (Figure 8) for network optimization and enforce data transfer prioritization as we described in Section IV. We compare throughput of data transfers in the SDN network when using MCF as compared to conventional algorithms such as Equal-Cost Multi-path (ECMP) used widely in data-centers and IaaS providers.

### A. Experiment Setup

The experiments are run on a virtual infrastructure (Figure 9) deployed across four Cloud sites on ExoGENI: UH (Houston, TX), UFL (Gainesville, FL), UMass (Amherst, MA) and PSC (Pittsburgh, PA). It consists of four Open vSwitch [27] software switches, each of which is deployed on a VM with 2 cores, 6GB RAM and 50GB disk and connected to an iRODS *resource* on a VM with 4 cores, 12GB RAM and 75GB disk. The switches are inter-connected using inter-site virtual network links with 600Mbps bandwidth. The iRODS *resources* and switches are connected using intra-site links with 1.8Gbps bandwidth. The topology of the virtual infrastructure is representative of a data-centric collaboration: iRODS *resources* are deployed at the network edge of each institution dedicated for inter-institutional data transfers over WAN; clients transfer data to their local edge *resource* before sending to other institution, and retrieve data from the edge *resource* once it arrives. The *SDN controller* and *Network planner* run separately on VMs with 4 cores, 12GB RAM and 75GB at the SL Cloud Site at Chicago, IL. The *SDN controller* is implemented using RYU SDN framework [16], and the *Network planner* is developed with SageMath [31].

**Func** `MCF` $(\{c_l\}, \{w_j\}, \{d_{(u,v)}\}, \{I_{j,l}\}, Pri, p)$**:**
**maximize:** $\sum_{(u,v)} b_{(u,v)} - \sum_{(u,v),j} w_j \cdot b_{(u,v),j}$
**subject to:** $\forall (u,v) : 0 \leq b_{(u,v)} \leq d_{(u,v)}$
$\forall l : \sum_{(u,v),j} b_{(u,v),j} \cdot I_{j,l} \leq c_l$
$\forall (u,v), j : b_{(u,v),j} \geq 0$

Figure 8: MCF algorithm

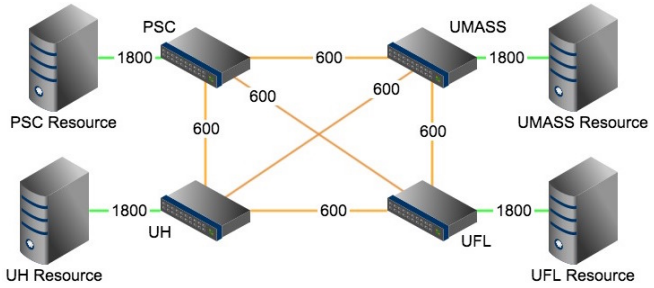


Figure 9: The experimental virtual infrastructure

We select ECMP as our base line routing algorithm for our experiments due to its wide adoption in data centers and IaaS providers. It is able to balance network load across multiple paths by hashing packet headers and selecting paths based on the hash values. The MCF algorithm, as illustrated in Figure 8, takes into account demands and availability of network resources for network optimization, aiming at maximizing the overall throughput while favoring shorter paths. As such, it is an example of how algorithms can leverage the global view of the network available through SDN support to perform network optimization.

To evaluate the effect that prioritizing file transfers have on network performance we associate smaller files and larger files with high priority and low priority, respectively. In other words, we assign high and low priority values in their associated metadata. As described in Section IV-A, the *SDN controller* invokes the routing algorithm iteratively in descending order of the priorities. The intuition behind

this approach is that by prioritizing bandwidth allocation for small files, we prevent large files from hogging the network.

In this evaluation, we use files in size of 256MB, 1GB and 4GB for data transfers to represent small, medium, large files, respectively. This distribution is representative of the communities work with and have used *RADII*. To make our experiments realistic, we produce a background load consisting of 200 simultaneous file transfers and 80% network.

### B. Results

We compare the throughput of data transfers achieved by ECMP an MCF+. We observe that MCF+ outperforms ECMP by 28.2% in overall throughput. Specifically, MCF+ improves the throughput over ECMP by 39.6%, 23% and 20.2% for transfers of 256MB, 1GB and 4GB files, respectively. Through deep observation of our results, we found that ECMP leads to network under-utilization due to the occurrence of collisions of hash values. In the presence of collisions, transfers are concentrated onto a few paths and contend for limited network resources while leaving resources on other paths unused. The contention also leads to network congestion, which causes additional packet loss and decreases the throughput. As a consequence, as shown in Figure 11, the throughput yielded by ECMP is unstable and sometime decreases significantly. The low throughput is also reflected in the network utilization observed in our experiment. As shown in Figure 12, the network utilization with ECMP is lower than 50% in average and sometimes drops to 30% due to network congestion.

In constrast, MCF+ enables data transfers to exploit network capacity. With the primary goal of maximizing the overall throughput, MCF+ spreads data transfers across all paths to utilize the network capacity at its maximum. As shown in Figure 12, MCF+ achieves around 70% network utilization in average, which is significantly higher than that achieved by ECMP. Distributing transfers across paths also avoids network congestion, reduces packet losses and increases throughput. Besides, MCF+ also optimizes the resource allocation by favoring shorter paths. Since network latency is negatively correlated to throughput [22], data transfers can benefit from the lower latency exhibited on the shorter paths to attain higher throughput. As a result, MCF+ achieves higher throughput of data transfers with higher stability than ECMP. Figure 11 depicts a graph showing that throughput when MCF+ is used exhibit low variation consistently.

To investigate the effect that prioritizing data transfers have on the network performance, we compare the throughput of data transfers as a function of file size. As shown in Figure 10, transfers of 256MB files yield the lowest throughput in average using ECMP, while attain the highest throughput using MCF+ among all transfers. The low throughput is partly due to TCP *slow-start* [4]: data transfers

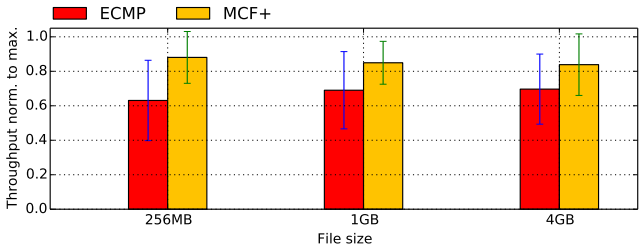


Figure 10: Average throughput comparison between ECMP and MCF+

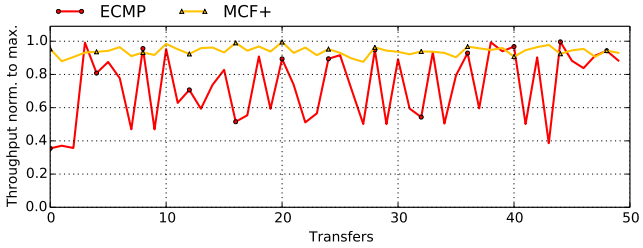


Figure 11: Throughput variation of a data transfer between UFL and UMASS repeated for 50 times

take longer time to probe sending rates on a longer path; on the other hand, packet loss forces transfers to decrease sending rate and prolongs the rate probing process. In other word, longer paths are detrimental to the throughput of small file transfers, since the transfers have small time window to recover from packet loss and can finish before reaching the maximum rates. With MCF+, transfers of smaller files are assigned to shorter paths due to prioritization scheme. As a result, they exhibit larger throughput improvement and higher throughput than transfers of large files.

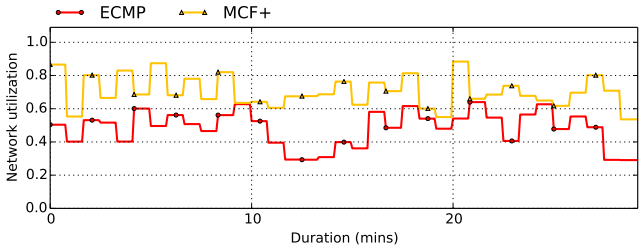


Figure 12: Network utilization variation during data transfers between UFL and PSC

In summary, the evaluation identifies problems with using the generic network provided by IaaS providers and reveals benefits of SDN networks provided in *RADII*. We believe that the integration of custom routing algorithms and cross-layer data annotation in *RADII* can be applied to extensive use cases and open up many opportunities to advance data-driven research cyberinfrastructure.

## VI. Related Work

The emergence of national cyberinfrastructures, e.g., ExoGENI and OSG, provides the scientific community with convenient access to large-scale distributed, heterogeneous systems and software. They provision diverse, on-demand infrastructural resources (*e.g.*, compute, network, storage)

powered by high-performance virtualization and advanced networking. Users specify their resource demands through standard interfaces and subscribe compute resources based on needs. For example, ExoGENI link distributed resources into connected arrangements, *slices* which provide mutually isolated pieces of networked virtual infrastructure built to order for guest applications like scientific workflows. However, deploying custom applications and tuning them for better performance and sharing require a considerable amount of time and expertise. *RADII* automates the deployment and configuration process and provides a simple interface for resource subscription, simplifying the usage of the cyberinfrastructure.

The heterogeneity and distribution of data systems also impose obstacles in data-centric research. Existing approches to address this consist of uniforming uniform data access and operations through an unified virtualization layer [7] [15]. It is also recognized that metadata service is essential in data grids to support advanced data management [8]. Globus Online [14] implements a Software-as-a-Service (SaaS) over distributed data storage to enable reliable and secure data sharing across domains. Nirvana [25] leverages metadata of data to realize fine-grained data sharing and management in collaborative workflows. iRODS enables more sophisticated data management across domains with an integration of a powerful rule engine and a metadata service. Data policies are composed using the rule language and interpreted into computer instructions by the rule engine through out the life-cycle of data managed by the data grid. In order to hide the complexity of the underlying system, these systems treat the infrastructure as an opaque object thus hampering the capabilities of the data management layer. This is in contrast to our approach in *RADII* in that we enable the right level of visibility and controllability between the data and the infrastructure management layer.

Network performance is a crucial aspect of distributed systems since networks enable the exchange of data within and across systems. Distributed file systems like HDFS [33] and GFS [17] account for static network topology for scheduling data replications, to find the path with minimum total distance between the requester and replica. Sinbad [9] monitors the bandwidth utilization at the edge to locate bottleneck links and keeps write operations from those congested links. SDN allows distributed systems to further customize and optimize network management. Hedera [3] balances network flows within datacenters by using SDN to resolve large flow conflicts caused by ECMP forwarding. B4 [21] is a successful SDN deployment over the wide-area-network (WAN), which maximizes network utilization while achieving high fault tolerance. Software-driven WAN (SWAN) [18] integrates MCF function into SDN to globally plan network capacity and schedule network flows, which significantly increases the overall throughput. These solutions are piecemeal in fashion in that they focus on the networking issues and treat data management aspects as a secondary consideration. *RADII* is unique in that it deeply integrates both aspects via advanced orchestration of resource provisioning and the encoding of network performance policies into file metadata which are later translated into actionable forward rules.

## VII. Conclusion

In this paper we have presented *RADII*, an extensible service architecture that enables the rapid provisioning and configuration of durable virtual infrastructure to support data-centric collaborations. The work is unique in two ways: First, *RADII* is representative of a software architecture capable of making national cyberinfrastructure resources available to the scientific community to collaborate around data. Second, the heart of *RADII* is the integration of data management and infrastructure management, a quality achieved through novel mechanisms such a cross-layer data annotation and SDN. To our knowledge, this is the first work that addresses reducing the gap that exists between infrastructure and data. We envision wider applications of *RADII* to serve a diversity of goals in data-centric collaborations such as enabling repeatable data-centric experiments, security policy enforcement at network layer using data properties, among others. In our future work we plan to consider advance security policies to guarantee network isolation of traffic of sensitive data– a common requirement in many domains such as health sciences. We believe that *RADII* can lower the barrier the scientific community face to establish data-centric collaborations on cyberinfrastructure resources and present unique opportunities for performance optimization tailored to the needs of the scientific community.

## References


[1] Radii git repository. https://code.renci.org/gf/project/radii, 2016.

[2] Radii prototype. http://radii.europa.renci.org/collaboration, 2016.

[3] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: dynamic flow scheduling for data center networks. In *NSDI*, volume 10, pages 19–19, 2010.

[4] Mark Allman, Vern Paxson, and Ethan Blanton. TCP congestion control. Technical report, 2009.

[5] Chameleon. https://www.chameleoncloud.org/, 2016.

[6] Jeff Chase, Laura Grit, David Irwin, Varun Marupadi, Piyush Shivam, and Aydan Yumerefendi. Beyond virtual data centers: Toward an open resource control architecture. In *Selected Papers from the International Conference on the Virtual Computing Initiative (ACM Digital Library)*, 2007.

[7] Ann Chervenak, Ewa Deelman, Ian Foster, Leanne Guy, Wolfgang Hoschek, Adriana Iamnitchi, Carl Kesselman, Peter Kunszt, Matei Ripeanu, Bob

Schwartzkopf, et al. Giggle: a framework for constructing scalable replica location services. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–17. IEEE Computer Society Press, 2002.

[8] Ann Chervenak, Ian Foster, Carl Kesselman, Charles Salisbury, and Steven Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of network and computer applications*, 23(3):187–200, 2000.

[9] Mosharaf Chowdhury, Srikanth Kandula, and Ion Stoica. Leveraging endpoint flexibility in data-intensive clusters. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 231–242. ACM, 2013.

[10] COMET. http://www.internet2.edu/, 2016.

[11] NSF Cyberinfrastructure. https://www.nsf.gov/news/special_reports/cyber/, 2016.

[12] Amazon EC2. https://aws.amazon.com/ec2/, 2016.

[13] Chip Elliott. Geni-global environment for network innovations. In *LCN*, page 8, 2008.

[14] Ian Foster. Globus Online: Accelerating and democratizing science through cloud-based services. *IEEE Internet Computing*, 15(3):70, 2011.

[15] Ian Foster, Jens Vockler, Michael Wilde, and Yong Zhao. Chimera: A virtual data system for representing, querying, and automating data derivation. In *Scientific and Statistical Database Management, 2002. Proceedings. 14th International Conference on*, pages 37–46. IEEE, 2002.

[16] RYU SDN Framework. https://osrg.github.io/ryu/, 2016.

[17] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *ACM SIGOPS operating systems review*, volume 37, pages 29–43. ACM, 2003.

[18] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven WAN. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 15–26. ACM, 2013.

[19] Big Data Regional Innovation Hubs. https://www.nsf.gov/pubs/2015/nsf15562/nsf15562.htm, 2016.

[20] Internet2. http://www.internet2.edu/, 2016.

[21] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. B4: Experience with a globally-deployed software defined WAN. *ACM SIGCOMM Computer Communication Review*, 43(4):3–14, 2013.

[22] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review*, 27(3):67–82, 1997.

[23] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[24] M Meyer and JP Vasseur. Mpls traffic engineering soft preemption. 2010.

[25] Nirvana. http://www.ga.com/nirvana, 2016.

[26] National Center of Biotechnology Information. http://www.ncbi.nlm.nih.gov/, 2016.

[27] Ben Pfaff, Justin Pettit, Keith Amidon, Martin Casado, Teemu Koponen, and Scott Shenker. Extending networking into the virtualization layer. In *Hotnets*, 2009.

[28] Ruth Pordes, Don Petravick, Bill Kramer, Doug Olson, Miron Livny, Alain Roy, Paul Avery, Kent Blackburn, Torre Wenaus, W Frank, et al. The open science grid. In *Journal of Physics: Conference Series*, volume 78, page 012057. IOP Publishing, 2007.

[29] Arcot Rajasekar, Reagan Moore, Chien-yi Hou, Christopher A Lee, Richard Marciano, Antoine de Torcy, Michael Wan, Wayne Schroeder, Sheau-Yen Chen, Lucas Gilbert, et al. irods primer: integrated rule-oriented data system. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 2(1):1–143, 2010.

[30] Robert Ricci, Eric Eide, and CloudLab Team. Introducing cloudlab: Scientific infrastructure for advancing cloud architectures and applications. *; login:: the magazine of USENIX & SAGE*, 39(6):36–38, 2014.

[31] SageMath. http://www.sagemath.org/, 2016.

[32] Arie Shoshani and Doron Rotem. *Scientific Data Management: Challenges, Technology and Deployment*. Chapman and Hall CRC Computational Science, 2010.

[33] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, pages 1–10. IEEE, 2010.