Managing Scientific Data with Named Data Networking

Chengyu Fan
Colorado State University
chengyu@cs.colostate.edu

Susmit Shannigrahi
Colorado State University
susmit@cs.colostate.edu

Steve DiBenedetto
Colorado State University
dibenede@cs.colostate.edu

Catherine Olschanowsky
Colorado State University
cathie@cs.colostate.edu

Christos Papadopoulos
Colorado State University
christos@cs.colostate.edu

Harvey Newman
Caltech
newman@hep.caltech.edu

ABSTRACT

Many scientific domains, such as climate science and High Energy Physics (HEP), have data management requirements that are not well supported by the IP network architecture. Named Data Networking (NDN) is a new network architecture whose service model is better aligned with the needs of data-oriented applications. NDN provides features such as best-location retrieval, caching, load sharing, and transparent failover that would otherwise be painstakingly (re-)implemented by each application using point-to-point semantics in an IP network.

We present the first scientific data management application designed and implemented on top of NDN. We use this application to manage climate and HEP data over a dedicated, high-performance, testbed. Our application has two main components: a UI for dataset discovery queries and a federation of synchronized name catalogs. We show how NDN primitives can be used to implement common data management operations such as publishing, search, efficient retrieval, and publication access control.

1. INTRODUCTION

Improvements in simulations and data collection over the last decade have transformed scientific research. The impact of the resulting data explosion can be seen in domains such as High Energy Particle Physics (HEP) and climate modeling. However, while scientific communities benefit from more data, the increased volume also presents new management challenges. Consequently, scientific communities must explore new data management techniques in addition to new software and hardware designs. Both HEP and Climate research communities use multi-petabyte distributed datasets. Effective use of this large amount of data requires seemless publication, simplified but efficient discovery mechanisms, and fast, reliable transfers.

We argue that many problems faced by today's data management applications are caused by the architecture of the network itself. IP networking uses *hosts* as the core primitive; one must first identify the host responsible for the desired content or service. As a result, data management is complicated by needing to keep location state up to date in the face of changing network dynamics as well as content entering and leaving the system. This complicates discovery and makes data transfers brittle; a failed node requires the consumer application to identify an alternative content source and try to resume. These issues, and others, lead us to suggest that a new Internet architecture is needed.

Named Data Networking (NDN) [12] is a proposed future Internet architecture where *namedcontent* is the first-class network entity, not the hosts. This model enables applications to directly fetch data by asking the network for it by name, instead of first determining the IP address of the hosting server. Furthermore, NDN data is digitally signed by its publisher, thus providing provenance for every data packet. Additionally, NDN routers are stateful [11] and can make intelligent forwarding decisions based on past measurements. Consequently, *every* application benefits from NDN features such as transparent failover and content retrieval from optimal sources without any additional efforts.

In this work, we present a distributed data management system based on NDN. Our solution provides seamless access to scientific data using well defined NDN naming schemes. We use a data discovery mechanism based on names. Our system also supports secure dataset publication and retrieval. Data publishers derive NDN names for their datasets from real world file names and write them into a synchronized federation of *catalogs*. Users may query any catalog instance and discover datasets by their NDN names. Then, users can simply ask the network for datasets by name without needing to first locate the hosting server.

2. CURRENT SCIENTIFIC DATA MANAGEMENT SYSTEMS

Sheer size and complexity of scientific data as well as its distributed nature make data management complex. Climate datasets are typically generated at geographically distributed sites, curated, and made available to remote users. Powerful resources are required to rapidly accommodate all requests for retrieval, post-processing, visualization, and analysis of the data. While large super-computing centers can accommodate such requirements, resources are scarce, and large amounts of data must be moved to various repositories to eliminate bottlenecks. High-Energy Physics datasets are typically generated at a single location and then distributed around the world via a tiered system. The latter also need to support distribution of derivative data such as simulation

NDM'15 November 15-20 2015, Austin, TX, USA

ISBN ACM 978-1-4503-4037-3/15/11…$15.00

DOI: http://dx.doi.org/10.1145/2832099.2832100

output.

Scientific communities' only problem isn't data volume or distribution mechanisms. Various data naming schemes, data formats, and lack of structured metadata makes data discovery very difficult. First, a scientist needs to know where data is located and how they are named. This is a hard problem since most scientific data isn't indexed either by search engines or existing scientific data management systems. Even when data location and names are known, moving data between repositories often requires advanced planning and operator intervention. Finally, existing IP network uses a point-to-point model which requires discovering and connecting to a host hosting the data. If the host fails during retrieval, the system needs to find another host, connect to it and restart the retrieval.

Since the IP networking protocols don't provide uniform frameworks to address these common problems, various communities have designed and developed customized data management software at significant effort and cost to satisfy their needs. The climate community uses ESGF [6] for searching and accessing CMIP5 [10] data. Similarly, HEP community has developed xrootd [7] for their data. These IP based systems do not provide an appropriate network service model to smoothly facilitate data discovery and retrieval operations. Moreover, much of the functionality of these software is similar and could potentially be served by a common infrastructure. We next describe these two data management systems and their limitations followed by how NDN can serve as a substrate for a common data management system, reducing both complexity and cost.

2.1 Xrootd

Xrootd [7] is a sophisticated data management system created by the HEP community that provides scalable discovery and retrieval capabilities. In xrootd the data resides on multiple servers organized in a system that dynamically matches clients with servers that have the desired data. Xrootd's architecture is shown in Figure 1. The system consists of a manager, several data servers and the clients. All data servers register themselves with the manager. When a client wishes to open a file the client sends a request to the manager with the desired filename. Upon receiving the request the manager multicasts the request to all the data servers. Only those servers that have the desired file respond. The manager decides which data server should serve the request and informs the client accordingly. The client then contacts the server directly.

In the Xrootd system a server can manage resources by asking clients to delay contacting the server. Furthermore, clients can be redirected to another server at any time, an approach that improves fault tolerance. When a server becomes unavailable the client launches another search. Xrootd employs several mechanisms to optimize performance and minimize resource usage such as multiple independent streams on a single socket.

Xrootd has several limitations. Its security framework allows any authentication protocol and channel-based security to be used but does not currently provide provenance. Clients must contact a manager who may be unavailable, overloaded, or distant, creating a choke point or a central point of failure. Xrootd's redirection and fallback mechanisms introduce delays, and the algorithms and parameters for failover must be implemented and tuned case by

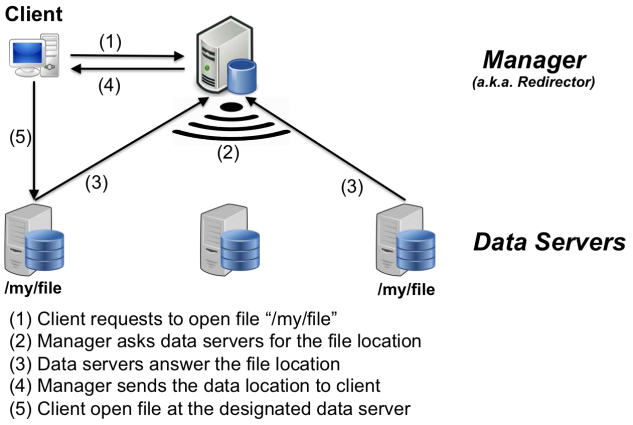

Figure 1: Xrootd Architecture[1]

case, which becomes difficult in the HEP case where the data is distributed at sites in all world regions, some with poor or highly variable network connections. Failover is not transparent to clients since they are active participants in remedying failure. Functions such as caching of extracted data object collections, and dynamic relocation of datasets are possible, but not implemented automatically due to the complexity.

2.2 ESGF

The Earth System Grid Federation (ESGF) [6] is a data management system used by the climate community to distribute CMIP5 data[10]. ESGF adopts a federated software architecture consisting of multiple geographically distributed nodes that coordinate through a peer-to-peer (P2P) protocol. Institutions publish data at their local node. Nodes can join or leave the federation dynamically and the system synchronizes itself automatically. Each node hosts a set of services and applications that collectively enable data and metadata access. There are four types of nodes: secure data publication and access (Data Node), indexing and metadata search (Index Node), user authentication and secure delivery of user attributes (Identity Provider) and data analysis and visualization (Compute Node). Each node can be configured to host one or more of these services. Users access the system through the ESGF web portals or desktop client that connects to a local node which distributes the user query to other nodes in the federation. The local node assembles the results and return them to clients as scripts containing hyperlinks, which clients can then invoke to download data through various tools such as wget, GridFTP[2] or OpenDAP[4].

ESGF has several limitations as well. Querying different ESGF nodes with the same set of parameters often generates different results. Data distributed through ESGF lacks built-in provenance. Climate community acknowledges the need to support subsetting operations[5] which isn't currently supported. Many retrieval tools associated with ESGF cannot provide advanced capabilities such as parallel retrieval or transparent failover. ESGF uses various technologies such as OpenID/PKI-based authentication methods and custom middleware for user authorization and myproxy for PKI infrastructure. However, configuring and maintaining such software is burdensome for both users and administra-

tors.

3. NAMED DATA NETWORKING

NDN [12] is a Future Internet Architecture where data in the network is accessed directly by its name rather than through the host where it resides. Naming the data allows the network to participate in operations that were not feasible before. Specifically, the network can participate in discovering and local caching of the data, merging similar requests, intelligent retrieval and more. NDN uses two types of packets for data transport - Interest and Data. Interest packets carry data requests; they are sent by consumers and carry the content name. Any entity that has the data may reply with one or more Data packets. For scalability, NDN uses hierarchical name prefixes for routing. Interest packets are routed towards publishers that advertise prefixes matching the requested data. Interests leave behind breadcrumbs for data packets to follow on the return, resulting in symmetric routing. In NDN, Data packets have publicly verifiable built-in signatures. This decouples content from its original publisher; therefore, content can be received from anyone, such as a repository, a router cache, or a neighbor, as well as the original publisher. Due to the pervasive caching in the network, NDN is able to achieve very efficient distribution, virtually eliminating hot spots. While NDN helps addressing some previous challenges faced by scientific communities, , many complexities still remain. One example is name discovery; it's fundamental to any NDN based applications but isn't easily implemented using vanilla NDN. Happily, as we will show in this paper, such functionality is easily added on top of NDN, taking advantage of many of NDN's properties.

3.1 Forwarding Strategies

NDN can intelligently use multiple available paths for data retrieval. Forwarding strategies decide which path(s) to use, based on per-prefix, user-defined policies. This is a unique feature of NDN that can greatly benefit scientific applications.

NDN maintains information about paths in the network in a new layer called the *strategy layer*. The strategy layer monitors two-way traffic and changes local forwarding decisions based on those observations. For instance, if a NDN router has two paths to the same prefix, a strategy can choose the least congested one; or a strategy may deploy BitTorrent-style forwarding for parallel retrieval. Network operators can configure static or dynamic per name-prefix strategies. For parallel retrieval a strategy can load balance between multiple servers using round-robin or weighted round-robin Interest forwarding. Custom forwarding strategies also help with failover. In IP, link or server failure means the host-to-host channel is lost and applications must set up another channel, which is a costly operation. NDN's strategy layer is capable of maintaining multiple viable channels and rank them according to performance. When a channel fails, the forwarding strategy simply chooses the next best path. The switch-over is much faster since it happens in the network (which has direct knowledge about network conditions) as opposed to the application (which must estimate network conditions).

3.2 Advantages of NDN for Climate and HEP Data

Naming the data enables the network to take a much more active role in data management by making it aware of all data locations, enabling informed decisions about directing Interests, thus effectively implementing anycast. Named data also allows the network to merge and satisfy similar Interests, effectively implementing multicast. It allows caches popular data in the network to satisfy future Interests far more efficiently than end-to-end retrieval and eliminate potential hotspots.

NDN pushes functionality currently implemented at higher layers to the network layer. At first this might seem as increasing network complexity, but these functionality is more effectively implemented at the network layer. Higher layer implementations result in a more complex systems since they require state only known to the network. At higher layers applications either have to consult the network to determine the required state, or must deduce it via external measurements. Take for example the selection of the best server out of a pool of available servers. Applications either have to create complex monitoring protocols for a highly dynamic environment, or run experiments in order to dynamically determine the best server (a good such example is the server selection in Netflix). Neither is necessary in NDN, which is aware of all feasible paths and maintains state information about each of them at the strategy layer.

NDN natively supports other common application-level requirements such as data prefix announcement. NDN announces hierarchical names directly into the network, so once a namespace has been defined by a community and a prefix has been assigned to an organization (operations that are similar to the DNS allocation we do today), all datasets named under that prefix are automatically published. This has two positive implications: (a) announcing a dataset name into the network is now a low-effort task - one simply needs to name the dataset appropriately, and (b) there are now strong incentives to follow the naming scheme from the start.

NDN provides provenance and integrity via mandatory digital signing. A NDN user will never accept corrupt, fake, or forged data. This decoupling of the content from the host enables caching anywhere, which in turn drastically reduces the distribution and retransmission costs of popular data. NDN is also in a better position to mitigate DoS attacks at all layers. Symmetric routing means that NDN routers can correlate Interests with replies and determine when Interests receive no response, which may indicate a deliberate attack or overload, and respond by choosing an alternate route or deploying pushback.

We believe that the capabilities provided by NDN are of use to the majority of scientific data management systems. Liberating such systems from (re-)implementing common functionality leads to simplified systems that focus on implementing domain specific differences rather than the similarities. As the number of datasets increases, however, discovery in NDN becomes inefficient. The solution is to build a searchable, replicated data name catalog that substantially speeds up discovery. As it turns out, the advantages of NDN are essential in simplifying the implementation of such a catalog. In the next section we describe a distributed highly scalable,federated catalog system that supports publishing, access control and name discovery.

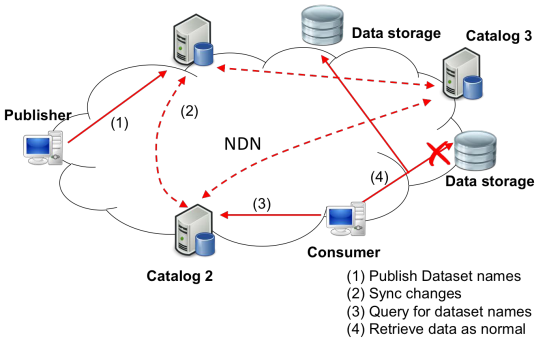

Figure 2: The Catalog Architecture

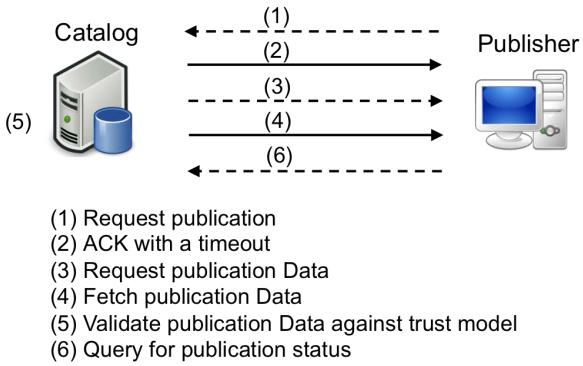

Figure 3: Data Publication

4. A SCALABLE NDN NAME DISCOVERY SYSTEM

We now present a distributed catalog system built on top of NDN that facilitates scientific data discovery. Our design is independent of data types or specific namespaces, we only require the published names to be hierarchical. We assume these NDN names, while not required, are under the jurisdiction of some specific entity, such as CMIP5. Inspired by the similarly named effort, we will use the prefix /CMIP5 for illustration and brevity. We consider CMIP5 to be the effort providing the catalog and use NDN dataset names inspired by their Naming specification [9]. We make no claim that the NDN names we use in this paper should be the final names but simply use them as illustrative examples. We build our system as a distribute federation of multiple independent catalogs. We use appropriate forwarding strategies in NDN to ensure that publishers and clients reach the closest or the least congested instance when making requests. They don't have to stick with that instance, the catalog allows them to move between instances even in the middle of a query. Also, the system allows publishers to publish data into any catalog instance using their NDN publishing key - no other form of authentication is required.

The catalog architecture is shown in Figure 2. The various steps are numbered and we will describe them shortly. First, we define some terminology; A *publisher* is the entity that owns the data and wants to publish it. A *consumer* is the user who wants to retrieve the data. The *catalog* is the system providing the service. A *data-storage* node offers persistent NDN-based storage.

4.1 Publishing

We assume a distributed publishing model where publishers are various institutions that own datasets and wish to publish them under the same /CMIP5 prefix. The owner of /CMIP5 prefix delegates publisher keys to each institution. This publisher key enables each institution to publish names under a prefix such as </CMIP5/.../institution>/. In our example, all catalog instances serving CMIP5 names will operate under the same namespace, e.g., /CMIP5/catalog . We want to emphasize that the catalog holds only names of datasets, not the actual data. The actual data is stored in the repositories operated by the publisher. Figure 3 describes the publication protocol. When a catalog is launched, it registers a prefix with the local NFD. We defined the registration prefix to be */CMIP5/catalog/publish*. For publication. a publisher encapsulates and signs a list of NDN names along with associated actions such as "add" or "remove" in one or more NDN data packets. The publisher then sends an Interest to the nearest catalog communicating its intention to update the catalog database. This is the typical way in NDN uses to upload information to a server - the publisher asks the server to pull newly available data. Upon receiving this Interest the catalog replies with an acknowledgment. There are several reasons for sending an acknowledgment rather than immediately asking for the data; the update may be big, and if the catalog is currently busy it may not want to pull the data immediately. Without an acknowledgement, this would result in the publisher timing out and trying again. An acknowledgment can carry back an estimate for when the catalog will retrieve the data, allowing the publisher to retry on an informed timeout. An acknowledgment also erases state in the network, thus saving router resources. We mentioned that the catalog pulls a set of instructions by the publisher. These instructions may include anything that the protocol supports, which, in our system, a list of new dataset names to be added or removed from the catalog. We implement these instructions in JSON format described below.

```
{
 "add" : [
   "/publisherA/file/1",
   "/publisherA/file/2",
   "/publisherA/file/3",
 ],
 "remove" : [
   "/publisherA/file/4",
   "/publisherA/file/5",
 ]
}
```

4.1.1 Access Control

As we mentioned earlier, catalogs make use of the digital signatures provided by NDN to enforce access control for publishers. NDN distributes keys for these digital signatures are distributed by default. NDN doesn't dictate how the signatures are distributed; they may be distributed through a PKI, web-of-trust mechanisms, or anything else agreed upon by the applications. When an Interest initiates a pull for instructions from a publisher, the catalog will fetch the instructions but not accept them if they are not correctly signed. This forms a first line of defense against illegitimate

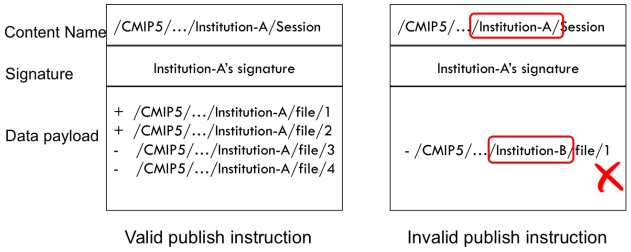

Figure 4: Valid vs. Invalid Catalog Instructions

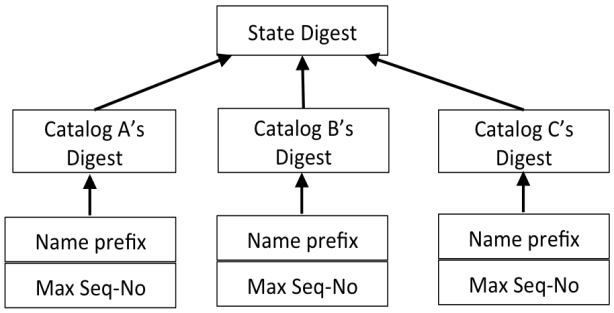

Figure 5: An example of digest tree used in catalog federation

[13]

publishers from making changes to the catalog. Although we have not implemented this, the catalog may also maintain a list of approved publishers distributed out of band. The next line of defense is designed to prevent publishers from making unauthorized changes to catalog entries. We assume that the policy in place is to allow a publisher to make changes only to entries published under the publisher's prefix; a publisher in CSU is only allowed to make changes to datasets under the prefix /CMIP5/CSU/. This process is depicted in Figure 4.

4.1.2 *Catalog Federation*

A single centralized catalog instance can easily become a bottleneck when subjected to large number of queries or publication requests. Moreover, it becomes a single point of failure and introduces increased latency for distant clients. To make the catalog scalable, we have adopted a federated architecture of several catalogs running a synchronization protocol. The catalog instances in the federation announce a single prefix; this allows the network to route queries to the most appropriate instance as determined by the NDN strategy. Depending on the strategy queries may go to the nearest catalog instance or the least congested one. We use a synchronization protocol called ChronoSync [13] to ensure consistency. Chronosync is a digest tree-based state exchange protocol. As shown in Figure 5, the state is calculated based on the published content and its sequence number. When a new content is published, a new state is calculated and sync messages are propagated to other nodes. A multicast namespace /ndn/broadcast/sync is reserved to exchange sync messages. In our application, the sync messages are simply the instruction messages described in the previous section. Whenever a catalog instance receives such a message it validates the payload and applies the updates to its local database. It then propagates the sync message to other catalog instances. A periodic database snapshot is generated to serve as recovery state in case of failure or a database restart.

4.2 Name Discovery

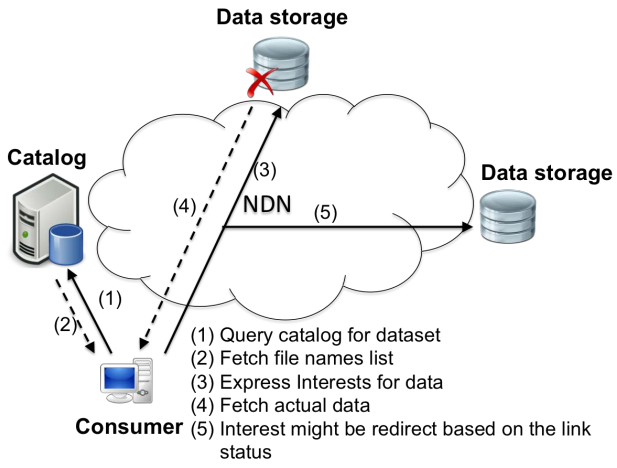

Figure 6: Data Retrieval over Catalog

In this section, we describe consumers' interaction with the catalog federation for discovering NDN names. The message exchanges between consumers and the catalog are shown in (Figure 6). Our catalog design doesn't restrict consumers to a specific catalog instance. Since the query name captures all the necessary parameters, all catalogs will generate consistent results for a given query.

When the query Interest arrives the catalog responds with an ACK. The ACK name contains the catalog ID, the query parameters, and the local database version (/CMIP5/catalog /query/<catalog-id>/<query-params>/<version>). The freshness time for the ACK is set to 0, which means it will not be cached in NDN routers. This ensures consumers receive responses not from a cached response but a live catalog instance. After the ACK is sent, the catalog converts the query parameters into an appropriate SQL string and issues a SQL request to retrieve the results. The query results contain a list of names. They are packetized and published into the memory storage under /CMIP5/catalog/query-results namespace. As soon as the ACK arrives, the consumer constructs the corresponding query-result content names by replacing "query" with "query-results" (/CMIP5/catalog/query-results/<catalog-id>/<query-params>/<version>). Currently, NDN strategy directs queries consistently to the same catalog instance. Note that query-results packets are cacheable and therefore same query Interests don't trigger multiple database queries which also protects the catalog from DoS attacks.

4.3 User Interface (UI)

We have implemented a Web UI loosely based on the ESFG UI to help consumers discover and download datasets. The UI is shown in Figure 7. Users discover datasets in two ways: (a) by specifying a set of filters, or (b) typing the name prefix directly. The filters are selected from namespace components and return all dataset names that matches the selected components. To help users type a name prefix

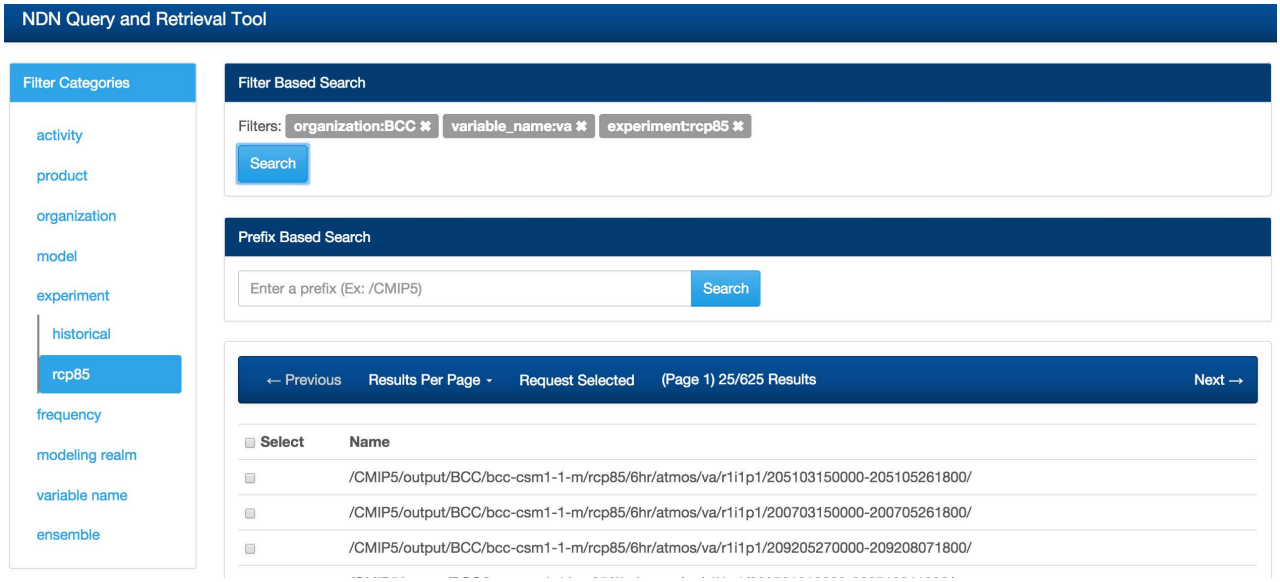

Figure 7: Data Discovery UI

directly, the UI provides name auto-completion as users type in the search box.

4.4 Catalog Federation deployment over NDN-SCI testbed

Figure 8: NDN Science Testbed

We maintain a seven node NDN testbed with high-end servers connected with 10Gbps links. This testbed allows us to deploy and test NDN applications. To demonstrate NDN's features that we described in earlier sections, we have developed a prototype application [3] that implements those functionality. The application has two main components - an UI and a backend catalog. This testbed currently hosts four synchronized catalog instances and one instance of the UI. Since the front-end is not tied to a specific catalog instance, the number of catalogs and front-ends does not need to match. The deployed catalog federation currently supports data publication, publisher verification, name synchronization, user queries and data retrieval.

4.5 Evaluation

In our deployment, we have used real CMIP5 datasets from ESGF. We took about 38K netcdf files and translated the file names into NDN names. For the translation, we had to parse actual filenames, metadata and user configurations. We translated and inserted these 38K NDN names into one of the catalog instances. The process of translation and insertion took a few seconds. Once the names were inserted, the catalog instances exchanged and synchronized the names. Since we are not synchronizing the actual data but only the names, the operation is fairly lightweight, also needing only a few seconds. Once the names are synchronized, users are able to query any catalog instance. While cost of a certain query depends on the network communication delay as well as database look-up time, simple queries from our UI were satisfied in tens of milliseconds.

5. FUTURE WORK

We want to expand this work on several ways. Our first goal is to demonstrate NDN's plugin support for different forwarding strategies that optimize data retrieval throughput. One example strategy can be parallel data transfer from multiple sources. We also want to understand which strategies will be most beneficial for various communities.

We plan to implement transparent query routing to any available catalog instance in case of a mid-query catalog failure. When a catalog instance fails, the query results Interests are redirected to other available instances. Since all catalog instances are synchronized and the query name in our protocol captures all the necessary query parameters, catalogs should be able to generate and return the same results. This approach avoids maintaining state between a consumer and a specific catalog instance. In this protocol, all catalog instances listen for Interests under the same namespace, /CMIP5/catalog. Upon receiving a query Interest, a catalog instance generates the results, packetizes them and sequentially publishes them into a memory storage under /CMIP5/catalog/<query-params>/<version>. In case the catalog fails after generating the results, subsequent Interests will be rerouted to another catalog. The new catalog may take some time to regenerate query results, thus causing the consumer to retry. This does not result in additional

traffic since the catalog is aware of queries in progress and will satisfy any retransmissions as soon as possible. Furthermore, generated results are reusable when a consumer issues the same query.

We also plan to show how modern network management tools such as SDN or ESNet's OSCARS can be combined with NDN. Scientific data transfers consume large amounts of bandwidth and can disrupt other network flows. In use cases such as HEP with deadlines for the delivery of high priority large datasets or LSST with real-time image delivery, there are specific QoS requirements such as dedicated bandwidth for a certain amount of time across multiple network domains. To address these issues, scientific traffic is often transferred over dedicated networks such as ESNet and Internet2. However, dedicated network paths do not reduce data management complexities. NDN, on the other hand, can provide intelligence at the network layer but it cannot control underlying network parameters such as bandwidth or network topology. We propose to interface ESNet's high speed layer 2 circuit reservation system, OSCARS[8], with NDN. We can also achieve the same functionality by combining NDN with Software Defined Networking (SDN). This hybrid approach will allow us to have dedicated high speed paths identified by special NDN namespaces. NDN will recognize these pre-established high speed paths or create them dynamically when large scientific data need to be transferred. High speed paths combined with NDN's advanced features such as parallel retrieval and transparent failover will make scientific data transfers faster and less complex to manage.

6. CONCLUSIONS

In this paper we presented the design of an NDN federated catalog system to enable scientific datasets discovery. While we used real world climate data as the driving example, we also argued that NDN names provide a common substrate. Specifically, the catalog application need only concern itself with the addition/remove of names rather than heavyweight dataset synchronization or even services. Consequently, our catalog design can support the needs of any community that is able to define and use consistent naming schemes. This design also avoids managing dataset replication and service hosting issues. The catalog need not keep track of what data or service is hosted where. Instead, the NDN network itself ensures that users requests will be routed to the optimal source without complicating any future applications that may be developed.

As we continue to experiment with different data communities, such as HEP, there is no need to change the catalog system. Each catalogs process is simply configured with the naming schema for the community it wishes to support. Multiple catalogs can even coexist on the same hardware. Similarly, our query UI is also dynamically populated during during its start up, thus updating its search fields appropriately.

To summarize, we have demonstrated how an intelligent network layer can make data management easier for scientific communities as long as they converge on common naming schemes. We presented a design for reusable distributed federated catalog system that facilitates NDN name discovery. Finally, we discuss a real-life deployment of the catalog along with the complementing UI and also provide some coarse-grained evaluation of the deployed system.

7. ADDITIONAL AUTHORS

Additional Authors: Edmund Yeh (Northeastern University, Email:`eyeh@ece.neu.edu`) and Jean-Roch Vlimant (CERN, Email:`vlimant@cern.ch`) and Azher Amin (Caltech, Email:`azher@hep.caltech.edu`) and Dorian Kcira (CERN, Email:`dkcira@cern.ch`) and Iosif Legrand (Caltech, Email: `Iosif.Legrand@cern.ch`) and Ramiro Voicu (Caltech, Email:`Ramiro.Voicu@cern.ch`) and David Randall (CSU, Email:`randall@atmos.colostate.edu`) and Kelley Wittmeyer (CSU, Email:`kelley@atmos.colostate.edu`) and Mark Branson (CSU, Email:`mark@atmos.colostate.edu`) and Don Dazlich (CSU, Email:`dazlich@atmos.colostate.edu`).

8. REFERENCES

[1] A. Hanushevsky, Potential Data Access Architectures using xrootd. http://xrootd.org/presentations/OSGAHM_1103. Plenary.pptx.

[2] Globus, www.globus.org.

[3] ndn-atmos, https://github.com/named-data/ndn-atmos.

[4] openDAP, www.opendap.org.

[5] 4th annual Earth System Grid Federation and Ultrascale Visualization Climate Data Analysis Tools face-to-face conference report. Technical Report LLNL-TR-666753, Lawrence Livermore National Laboratory, Livermore, CA, 2014.

[6] L. Cinquini, D. Crichton, C. Mattmann, J. Harney, G. Shipman, F. Wang, R. Ananthakrishnan, N. Miller, S. Denvil, M. Morgan, et al. The earth system grid federation: An open infrastructure for access to distributed geospatial data. *Future Generation Computer Systems*, 36:400–417, 2014.

[7] A. Dorigo, P. Elmer, F. Furano, and A. Hanushevsky. Xrootd-a highly scalable architecture for data access. *WSEAS Transactions on Computers*, 1(4.3), 2005.

[8] C. Guok, D. Robertson, M. Thompson, J. Lee, B. Tierney, and W. Johnston. Intra and interdomain circuit provisioning using the oscars reservation system. In *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on*, pages 1–8. IEEE, 2006.

[9] K. E. Taylor, V. Balaji, S. Hankin, M. Juckes, B. Lawrence, and S. Pascoe. Cmip5 data reference syntax (drs) and controlled vocabularies, 2010.

[10] K. E. Taylor, R. J. Stouffer, and G. A. Meehl. An overview of cmip5 and the experiment design. *Bulletin of the American Meteorological Society*, 93(4):485–498, 2012.

[11] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang. A case for stateful forwarding plane. *Computer Communications*, 36(7):779–791, 2013.

[12] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang, et al. Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73, 2014.

[13] Z. Zhu and A. Afanasyev. Let's chronosync: Decentralized dataset state synchronization in named data networking. In *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, pages 1–10. IEEE, 2013.