Supporting Climate Research using Named Data Networking

Catherine Olschanowsky, Susmit Shannigrahi, Christos Papadopoulos
Computer Science Department, Colorado State University, Fort Collins, Colorado - 80521
{cathie, susmit, christos}@cs.colostate.edu

***Abstract*—Climate and other big data applications face substantial problems in terms of data storage, retrieval, sharing and management. While several community repositories and tools are available to help with climate data, these problems still persist and the community is actively looking for better solutions.**

In this project we apply NDN to support climate modeling applications. The information-centric nature of NDN, where content becomes a first class entity, simplifies many of the problems in this domain. NDN offers lightweight data publication, discovery and retrieval compared to IP-based solutions. However, introducing a new network architecture to a mature domain that routinely produces petabytes of datasets and a plethora of assorted tools to manipulate them, is a risky proposition. The advantages of NDN alone may not be sufficient to overcome the natural inertia. Our approach is to introduce NDN while carefully avoiding undue disruption to existing workflows. To that extent we employ a user interface that employs familiar filesystem operations to publish, discover and retrieve data, integrated with domain-specific translators that automatically convert and publish datasets as NDN objects. We outline the advantages of NDN in this application domain and the challenges we faced during the adaptation. We believe this is the first exercise in applying NDN in an existing, large, mature application domain.

I. Introduction

Named Data Networking (NDN) [9] is a new Internet architecture where content is addressed directly rather than through an end host. NDN is an instance of Information Centric Networking (ICN), an area that is receiving significant interest in the networking community lately.

The ability to request content directly has some significant advantages. For example, content can be retrieved from any available location: the publisher, a repository, a router cache, or even a friendly neighbor. This flexibility, however, comes with costs, one being the need for provenance. NDN solves the problem by requiring that all content is signed, which not only assures provenance but also ensures that bad content can be detected and discarded. Names in NDN are also human readable providing some additional level of assurance. NDN names, along with the use of *selectors* can be used to perform a search because sending an Interest returns any content that matches the Interest. As we will show later, this is a powerful feature in NDN and greatly facilitates both discovery and efficient content retrieval.

NDN places few restrictions on naming, simply requiring that, (a) the name structure is hierarchical, (b) naming rules are globally agreed among content users, and (c) name prefixes are allocated to publishers (similar to how the current DNS allocates domain names). NDN names can conceptually be divided into two variable parts, the routing prefix, stored in router FIBs and used to route Interests, and the application-specific portion of the name. Routers match the full name when de-duplicating Interests or matching content in their cache, but only use the routing prefix to forward Interests.

978-1-4799-4894-9/14/$31.00

To date, very little research exists on naming schemes. Questions such as efficiency, aggregation, ease of lookups, proper hierarchy structure, etc. are very important, yet not well understood, especially for global names. To make matters worse, applications today use a myriad of application-specific naming schemes created without global uniqueness or structure in mind. For example, naming in HTTP is often tied to the filesystem structure, which makes it non-portable. Moreover, there are millions of scripts and other tools in existence that make assumptions on the structure and location of the data. Thus, an important question is, do content based architectures such as NDN offer sufficient momentum to transition from a world of numerous local, ad-hoc naming schemes into global namespaces?

The NDN team is beginning to address such questions by creating or adapting existing applications to NDN. The team have created several new applications such as chat, lighting control and building automation [5]. However, we have not yet demonstrated success in adapting existing applications.

In climate simulation and other sciences there are strong indications that global namespaces are desirable. For example the CMIP5 project [4] that collects datasets for global climate research, prescribes precise specifications and tools to convert local names into CMIP5-compliant names. The Data Reference Syntax (DRS) document [3] describes in detail these specifications. Among scientists in big data domains there is a general consensus that global namespaces are needed, especially given the current explosion of data.

In this paper we present our efforts to adapt NDN into a large and demanding application domain, climate modeling. The climate community has a long history of collecting massive amounts of diverse data over tens of years and from numerous teams. It is not uncommon for a research team to run massive simulations on supercomputers and generate data volumes that reach into the petabytes. In addition, the climate community has a long history of sharing data among scientists around the globe, which means they are constantly challenged by discovery and distribution problems. Indeed, scientists often still share data by shipping physical disks around despite the availability of 10G and 100G links. While there are several reasons why we cannot fully utilize such high-speed links, part

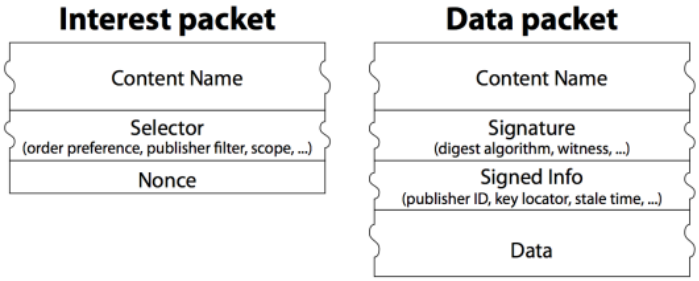

Fig. 1: NDN Packets - [9]

of the problem is fine granularity access, which often requires a new (and slow) round of requests from a repository.

We believe that NDN can address many of the problems faced by the climate community today. With NDN for example, requests to repositories can be substantially simplified and automated, with requests issued transparently to repositories that have the data without the need for application infrastructure to locate them (and users to establish login credentials, login to various portals, search, and issue targeted requests); data can be fetched from the nearest location, often a boon to collaborative efforts; and data can be published by the simple act of naming it. To keep the problem tractable, we limit our scope as follows: (a) we investigate naming schemes for a significant *category* of climate data, namely datasets in the NCDF format [12]; (b) we build an interface where climate scientists can publish, retrieve and discover climate data by name; (c) we write translators to convert dataset names from existing, ad-hoc namespaces into NDN compliant names, and (d) we deploy a 10G testbed between three institutions that are involved in climate research to test our approach.

While the 10G testbed is still being deployed, we have already demonstrated some of the advantages of NDN over the current workflow by building a proof-of-concept environment. An important consideration was to ensure the least disruption to current workflows, thus we created an environment where a scientist simply drags and drops an object into an NDN-aware filesystem. This action has the effect of invoking a translator that analyzes the object, constructs an appropriate NDN name for it, and publishes the name. Objects become available through repositories that synchronize through the NDN *sync* protocol and can be retrieved through a simple mouse click.

II. Background

In this section we first give a brief overview of NDN [9] and the climate modeling application to which we apply NDN.

A. Named-Data Networking

NDN is an example of Information Centric Networking (ICN). In ICN data is accessed by name rather than through the host where it resides. In NDN the content may be retrieved from the original publisher, a repository, a router cache or a neighbor; all content is authenticated and its integrity verified by the network, so the user is assured that an untainted copy was received. By using extensive caching in the network, NDN allows more efficient distribution (the more popular the content the more available it becomes). This can greatly benefit parallel transfers such as bulk data distribution and archiving, and collaborators working simultaneously on the same datasets.

The receiving end, i.e., the data consumer, drives communication in NDN. NDN uses two types of packets for transport (Figure 1) - Interest packets and Data packets. Interest packets are used to express a query for particular content; they are send by a consumer or client when specific content need to be retrieved. When an Interest packet reaches a node having the data, the node packs the data in one or more Data packets and sends it back. Packets move through the network through a series of routers. A router remembers the interface from which the request comes in and forwards the Interest by looking up the name in its Forwarding Information Base (FIB), which is populated by a name-based routing protocol. Once the Interest reaches a node that has the requested content, a Data packet is sent back, which carries both the name and the content, together with a signature by the producer. This Data packet follows in reverse the path taken by the Interest to get back to the consumer. Note that neither Interest nor Data packets carry any host or interface addresses (such as IP addresses); Interest packets are routed towards content producers based on the names carried in the Interest packets, and Data packets are returned based on the state information set up by the Interest packets at each router hop.

While NDN is intended to be a general-purpose network architecture, it has great potential to simplify content discovery and transport efficiency for large datasets. NDN names data with user-selected, flexible and expressive names, offloading discovery from the user to the network. Data is named in a hierarchical fashion, but typically only a portion (a global prefix) of the name is needed for routing.

B. Climate Modeling Data

The infrastructure under development is designed to support the data management needs of the Center for Multiscale Modeling of Atmospheric Processes (CMMAP) [1] at Colorado State University. CMMAP has over 80 users distributed around the country. Data organization and movement are critical functions for CMMAP researchers, yet complicated due to the volume and complexity of the simulation data being studied.

The science of global atmospheric modeling is in a period of profound change, as dramatic as any in its 50-year history, due to the amazing increases in computational power that have carried us to the brink of an Earth-science revolution. It is now possible, though still very expensive, to simulate the global atmosphere using a grid fine enough to crudely resolve the larger individual clouds, i.e., to run a global cloud resolving model (GCRM) [11].

GCRMs are currently under development at several modeling centers in the U.S. and abroad. CMMAP is actively engaged in this work. They have successfully tested model components running across 80,000 cores, and expect to scale much further within the next year or so. Within ten years machines will speed up, and the models will mature to the point that they will be used routinely in weather prediction, and even in short climate simulations. Eventually, within the careers of today's graduate students, GCRMs will be routinely used to simulate climate change.

The outcome will be very rewarding, but there will be a learning curve due not only to the conceptual complexity of

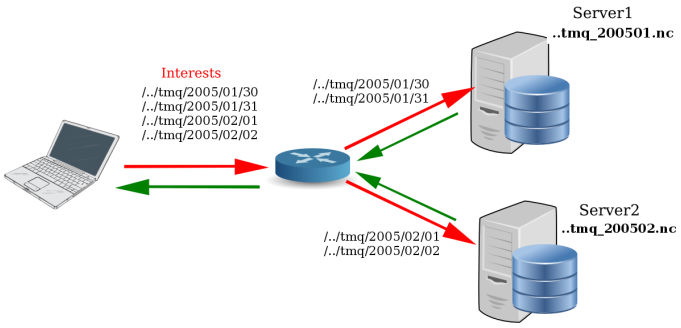

Fig. 2: NDN retrieval example

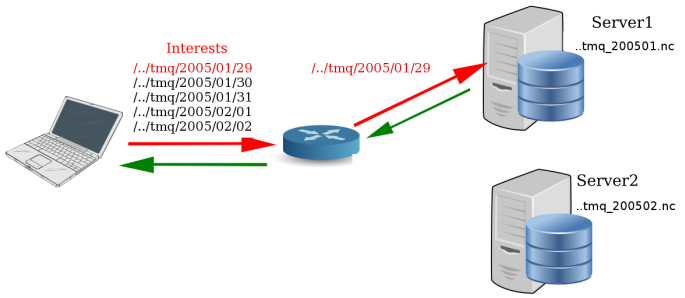

Fig. 3: NDN partial retrieval from cache

GCRMs, but also due to new problems posed of technical, practical, and fiscal natures. This is where the need for new infrastructure arises. The NDN-CMMAP infrastructure project relates to data management, analysis, and visualization. GCRMs produce terabytes to petabytes of model output; Every conceivably useful field must be written because the computational expense of the model is such that one cannot afford to rerun to create additional output. This includes restart records, i.e., snapshots of the models state that can be used to restart the model for a later simulation or in case of a system crash. Secondly, diagnostic data is written to understand what the model is simulating.

In short, the very large models used in cloud-climate studies must be supported by a suitably designed infrastructure, both hardware and software, for data management, transport, analysis, and visualization. These needs are community wide and should be addressed in a coordinated fashion that serves the community as a whole, from an end-to-end, scientific workflow perspective. We believe that NDN offers a better network architecture to address the needs of climate (and other big data) applications, as we describe next.

III. Integration with Current Scientific Workflow

Scientific workflows in the atmospheric sciences are complicated not only by the sheer size and complexity of the data created, but also by the distributed fashion of the workflows. The data must be moved shortly after creation, and subsequent moves may be necessary for post-processing, visualization, analysis, or collaboration.

The CMMAP simulation data is typically created at supercomputer centers. It must be transported, archived, curated and made available to users at remote sites. Users must make difficult choices about which output to transfer; for example, choosing which fields and what subsetted spatial and temporal resolutions. The sheer volume of data produced, the wide range of scales represented and the diverse phenomena included complicate the extraction of useful information from the GCRM output.

In this section we distill the data movement needs of the CMMAP community into two basic steps: publishing and retrieval. Through an illustrative example we explain how the use of NDN simplifies the implementation and execution of these steps.

A. Data Publishing

We are developing a model in which data is published by researchers by providing an appropriate NDN compatible name with the help of automated tools. Interested parties (users, repositories, archives, etc.) pull the data from the nearest available location by simply asking the network for it by name. Locating, verifying the integrity, retrieving and delivering the data is all done by the underlying network infrastructure.

Currently there is no uniform publishing process in place for climate modeling data. Several methods are used ranging from formal publication to informal private data exchange. Formal publishing can be achieved through public repositories such as UCAR [8] or NOAA [6]. The publishing step is challenging because the data has to be made available at a place where users can discover the data and have access to retrieve it. This requires organizing the data and advertising its location.

The underlying functionality of the NDN infrastructure relieves the user of the responsibility of deliberately organizing the datasets and advertising the location of each. Once appropriate name prefixes have been determined (for example named after the institution and project) and published into NDN, all datasets that fall under that prefix become content objects and are automatically advertised. Through the simple act of assigning an NDN name, these objects become discoverable and the naming makes the content organization implicit.

B. Data Retrieval

A common data movement operation is to retrieve a subset of an object in order to stage it near the compute resources being utilized for analysis. The ability to retrieve only a specific subset of an object is critical due to the volume of the data stored after the initial model simulation. The following is an illustrative example using a $water\_vapor$ objects from UCAR.

The dataset we use in this example is organized by date (an object is one month's worth of data) and contains several water vapor related metrics. Assume that a user would like to retrieve all of the metrics from objects spanning 4 days of January and February of a given year. Further, assume that each monthly object resides on a different server.

Using the current Internet architecture to retrieve the desired subset a user would (a) have to determine where the two files reside, (b) manually fetch files from each server, and (c) merge and trim the files into the final dataset. Note that in addition to requiring a priori knowledge of the location of the

datasets (often very hard in itself) this process involves several manual steps.

Contrast the above steps with those required with NDN. A researcher interested in 4 days of data sends out one Interest for each day. Since data is distributed among multiple servers, the intermediate routers forwards Interests to the appropriate servers automatically; the user need not take any manual steps. Only the required subset of the data arrives, which is in turn presented to the user. Figure 2 shows the process - a researcher asks for data from January 30th to February 2nd. Note that the data is distributed between two servers as indicated by the name prefixes they advertise. The intermediate router can now forward requests for January 30th and 31st to server1 and February 1st and 2nd to server2. Upon receiving the Interests, each server sends back the matching data. The retrieved data objects are cached at all intermediate nodes. In situations when the same data or any subset of a previously requested data is requested again, it is served from the intermediate caches without the need to go to the original data sources. As shown in Figure 3 - beyond the obvious speed-up advantage in data retrieval, the data may still be available in case server1 or server2 goes down.

IV. Proposed Naming Schema

A naming convention is required to move from the filesystem to an NDN realm. However, we do not propose renaming of all existing data but a file (or dataset) name to NDN name mapping so the files are available over NDN with minimal user intervention. The requirements for NDN names compliment natural naming schemas for most data. As an example, a large portion of the climate community utilizes the Coupled Model Intercomparison Project (CMIP) [10] naming schema. This schema is fully NDN compatible without any changes.

Here we present an overview of the guidelines that should be applied when creating a new naming schema, and then present the specifics of a naming schema for a subset of the data produced and used by CMMAP that does not currently utilize the CMIP schema.

A. General NDN Guidelines

NDN names for climate data should be both expressive and human readable. The advantages of long expressive names versus short easy to remember names have to be balanced. However, NDN allows multiple names to refer to the same object, perhaps easing the decision process.

NDN names are not restricted to static data items. The names may refer to derived data objects, for example subsets of existing files, or a dataset, which may be composed from sets of many files. A hierarchical naming pattern makes this varying level of granularity possible.

In addition to providing for varying levels of granularity, the hierarchical names are used in the routing of the data. A data provider publishes a data prefix which acts as a globally routable prefix covering all of the data made available by the provider.

We propose three primary guidelines for designing an NDN compatible namespace for climate data.

1) Names should be built as a set of well defined *components*
2) The components should be organized in a hierarchical pattern
3) More general (more common) name components should appear earlier in the hierarchy

B. Proposed CMMAP Naming Schema

The naming schema we developed for CMMAP data is designed to be compatible with existing climate modeling schemas (CMIP) while allowing for some flexibility for project-specific requirements. We describe the schema by first defining the name components to be used and their expected order.

NDN names for CMMAP applications are divided into the routable prefix of the name that has a relatively fixed structure, and the model-specific portion of the name. The goal of the routable portion of the name is to get an Interest routed to the set of machines that host the data. We anticipate the following name components to be common across many different projects.

Activity
Sub-activity or Product
Organization
Model
Ensemble
Experiment or Field Campaign

Thus, a routable name prefix may be defined as follows:

/Activity/[Sub-Activity or Product]/Organization/Model/

For example, one such name might be $/coupled/control/CMMAP/gcrm$. In NDN, organizations are free to chose any unique name they want, not just those limited to DNS domains.)

For more specific names such as those from the output of a specific climate model, we add components unique to the model. These may include the following:

Sample Granularity or Frequency
Start Time
Variable Name
Ensemble Member
Grid Resolution
Specific Properties: For instance optical properties.

In designing the model-specific portion of the NDN names appropriate for CMMAP applications we determined that names must include enough information to uniquely identify simulation runs of a specific model. Examples of such models include CESM [2] and SAM [7], which are well known climate models. The model-specific names then will be structured as follows:

CESM /Ensemble/Experiment/Sample Granularity/Start Time
CMIP5 /Experiment/Frequency/modeling realm/ variable name/ensemble member/
SAM /field campaign/optical properties for radiation/grid resolution/output type/timestamp/

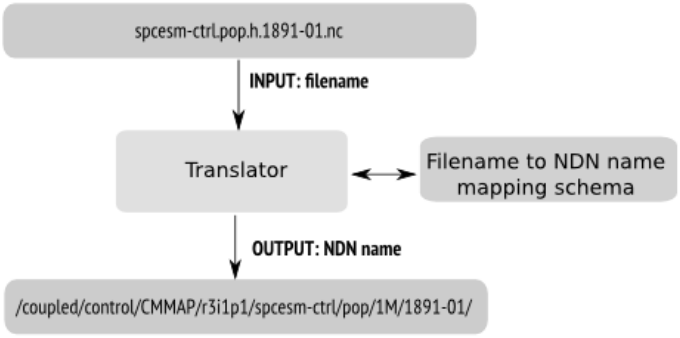

Fig. 4: Name translator

It is fairly easy to see how to extend the names above to support operations such as subsetting. For example, we can add a portion (e.g., a suffix) to the NDN name to indicate which variables the subset should contain, for example using key value pairs such as $subset\_variable = temperature$.

We consulted with CMMAP climate scientists and confirmed that these naming rules are acceptable for their datasets and are appropriate for global distribution. However, not all name components can be gleaned from the directory structure and filename of the dataset. We discovered that to construct many of these names our translators have to parse metadata in the files, and in some cases even the data itself to mine for missing name components. This was an important lesson in building translators and an indication that the task will not be straightforward.

V. Software Architecture

We are actively developing infrastructure to support publishing and retrieving data for CMMAP users through NDN. Our software acts as an integration layer between the CMMAP users and the NDN infrastructure. In order to integrate seamlessly with existing workflows and make the barrier of acceptance low, we provide the following:

- Translators converting existing filenames to NDN object names
- A Filesystem interface for publishing/retrieving data
- Specialized applications (if needed) to carry out operations on data, such as extracting subsets

A. Translators

The problem of organizing climate modeling data is hard due to the size, diversity and number of datasets generated. Climate modeling software often contain simple rules about how to organize their output, typically in directories and files with meaningful names. This helps manage datasets in local file systems, but is not an adequate naming solution for sharing data. This deficiency is well recognized by the climate community. Current efforts with a global scope address this problem by imposing strict naming and organizational structures on model output. One prominent example is CMIP5 [4]. The CMIP5 effort goes to great lengths to enforce a common namespace, providing tools to translate from common formats to the standardized names.

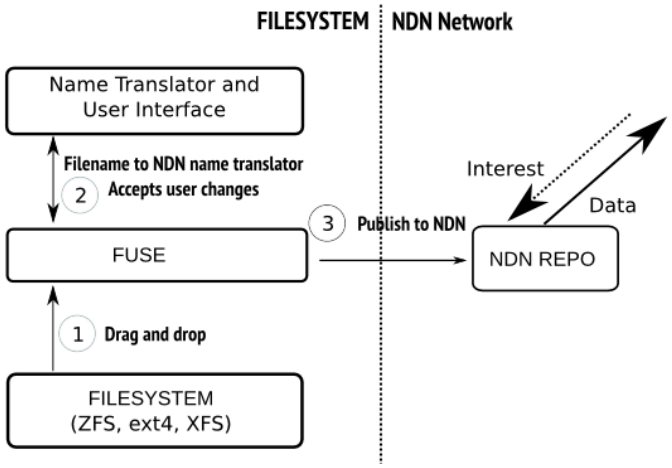

Fig. 5: Integrating NDN with filesystem using FUSE

We follow a similar approach in our work and use name translators. Recognizing that it is simply impossible to rename the vast collection of datasets and update all the assorted tools that manipulate them, we are developing translators that ease the burden of migrating existing datasets to NDN.

We envision that there will be multiple domain specific translators depending on the climate models and even the version of the model. We have determined that our translators need information gleaned from the filesystem path, the filename, limited user-provided configuration information, and metadata mined from within the data itself in order to construct appropriate NDN names. Thus, we expect the need for some intelligence in our translators. We expect that our experience in writing a few such translators will inform later efforts in domains other that climate data.

The behavior of a translator is determined by a metadata to NDN name mapping (shown in Figure 4). This mapping is expressed as a configuration file comprising a list of ordered NDN name components along with instructions for retrieving the data from the directory structure, filename, file content, or user defined configuration. Note that the use of a schema allows name specifications to be shared, and if agreed, the schema can be easily updated to add or remove fields in the derived NDN name. We expect the routing prefix to be part of the schema, so once the NDN name is translated, the content is automatically published under that particular name prefix.

Since NDN naming is flexible and virtually any appropriate translation schema can be plugged into, a translator works across many current naming schemes as long as the existing name can be broken down in hierarchical components.

B. Filesystem Integration

While the translators are critical for the automation of data publishing, they are only part of the user interface. Recall that our goal is to present scientists with a simple, intuitive interface to export data to NDN with minimal effort. We have developed such a user interface that allows for the publishing of hundreds of files while maintaining the familiar filesystem interface. Figure 5 provides an overview of the architecture.

Our interface uses the Filesystem in USErspace (FUSE) software. As the name implies, FUSE is a user-space filesystem

that allows us to make the necessary changes without the need to tamper with the OS filesystem. While FUSE can act as a independent filesystem, we only use it for interfacing between the OS filesystem (e.g., HFS, ext4, XFS) and the NDN network. In our system, the FUSE filesystem is simply mounted as a folder in the user's machine. FUSE provides a flexible interface for associating functions with normal file operations such as read and write. Once the publisher copies (or links) the data files into FUSE the following happens:

1) FUSE calls the appropriate translator to translate the filename into an NDN name
2) Using a GUI, the original name and the NDN name are shown to the user. This presents an opportunity for the user to inspect and correct the name
3) Once the user confirms the name, the file is published under the translated NDN name.

C. Support for Specialized Applications

The use of NDN names makes it trivial to run specialized applications directly on top of the network. One example is an archiver, an application that archives a copy of every object under a particular prefix. For example, an archiver can express a standing Interest for all objects that start with $/coupled/control/CMMAP$. A copy of all objects under that name will be delivered to the archiver in addition to the original requester. Another example is a remote subsetting service provided by the publisher that allows researchers to retrieve only the portion of an object (e.g., temperature for January 31st) rather than the entire object. Similar to an archiver, a subsetting service is also trivial to register by publishing longer prefixes that indicate the capability to provide the service. Contrast this to IP-based solutions: for an archiving each retrieval would require a call to the archiver to deliver a copy of the object; for subsetting, a special API is needed to communicate the parameters for subsets, and changes to the subsetting operations would require a change to the API.

NDN provides the ability to register a strategy with every name prefix in the FIB, thus more complex strategies can be implemented by the NDN strategy layer. For example, if multiple publishers are available, one possible strategy is to implement load sharing among them by distributing Interests appropriately (e.g., in a round-robin fashion).

In summary, NDN provides a flexible interface to implement services that circumvents the need for complex middleware to enable specialized applications.

VI. Conclusion

While it is conceivably easier to apply NDN to new applications, an important question is how to apply NDN to existing, well-established domains. In this paper we begin to answer this question by presenting a case study of applying NDN to large climate applications. We selected climate applications because these, similar to other big data applications, have needs that are well-suited for NDN.

We ask several broad questions, including: (a) What are appropriate naming schemes for climate applications? (b) Does the name-based approach in NDN help solve important domain problems, such as object discovery and retrieval in addition to specialized operations such as archiving and subsetting? (c) Is there a plausible path for these applications from the existing IP-based world to an NDN world?

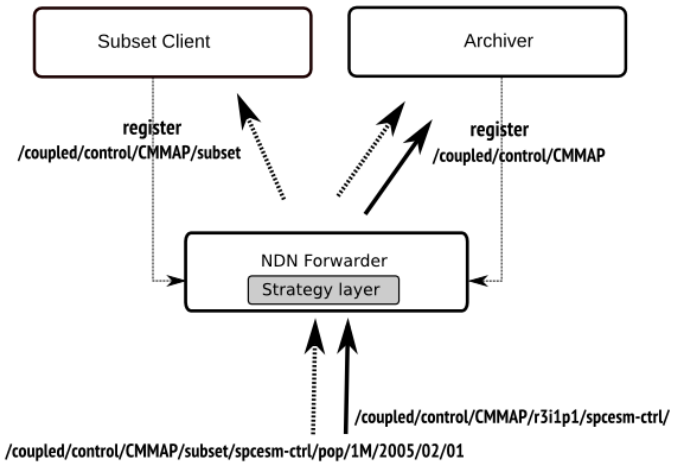

Fig. 6: Applications running on Top of NDN

While this is preliminary work, we believe we have strong evidence that the approach might work. The problems we try to address with NDN are well-recognized by the climate community; for example, there are already efforts toward standardizing dataset names. Object publishing and retrieval, operations for which NDN is well suited, are routinely done in the climate community, but through numerous repositories that are still hard to use.

We are in the process of deploying a dedicated climate testbed, which should be operational in the summer of 2014. The testbed will link scientists in three major locations, CMMAP, LBNL and UCSD. Our future plan is to deploy NDN and our software on the testbed, import existing datasets and run experiments with climate scientists to assess its potential.

References

[1] Center for Multiscale Modeling of Atmospheric Processes. www.cmmap.org/.

[2] CESM. http://www2.cesm.ucar.edu/.

[3] CMIP5 Data Reference Syntax. http://cmip-pcmdi.llnl.gov/cmip5/docs/cmip5_data_reference_syntax.pdf.

[4] CMPI5. cmip-pcmdi.llnl.gov/cmip5/.

[5] NDN Apps. http://ndn.ucla.edu/projects-and-code/apps/.

[6] NOAA Repository. http://nomads.ncdc.noaa.gov/data.php?name=access.

[7] SAM. https://sam.nrel.gov/.

[8] UCAR Repository. http://motherlode.ucar.edu/.

[9] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. Networking Named Content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM, 2009.

[10] Gerald A Meehl, George J Boer, Curt Covey, Mojib Latif, and Ronald J Stouffer. The coupled model intercomparison project (CMIP). *Bulletin of the American Meteorological Society*, 81(2):313–318, 2000.

[11] David Randall, Marat Khairoutdinov, Akio Arakawa, and Wojciech Grabowski. Breaking the cloud parameterization deadlock. *Bulletin of the American Meteorological Society*, 84(11), 2003.

[12] Russ Rew and Glenn Davis. NetCDF: an interface for scientific data access. *Computer Graphics and Applications, IEEE*, 10(4):76–82, 1990.