Named Data Networking Next Phase (NDN-NP) Project May 2015 - April 2016 Annual Report

Principal Investigators

Van Jacobson, Jeffrey Burke, and Lixia Zhang
University of California, Los Angeles

Tarek Abdelzaher
University of Illinois at Urbana-Champaign

Beichuan Zhang
University of Arizona

kc claffy
University of California, San Diego

Patrick Crowley
Washington University

J. Alex Halderman
University of Michigan

Christos Papadopoulos
Colorado State University

Lan Wang
University of Memphis

Contents

FIA-NP: Collaborative Research: Named Data Networking Next Phase (NDN-NP) 2016 Report

Executive Summary

The heart of the current Internet architecture is a simple, universal network layer (IP) which implements all the functionality necessary for global interconnectivity. This *thin waist* was the key enabler of the Internet's explosive growth, but its design choice of naming communication endpoints is also the cause of many of today's persistently unsolved problems. NDN retains the Internet's hourglass architecture but evolves the thin waist to enable the creation of completely general distribution networks. The core element of this evolution is removing the restriction that packets can only name communication endpoints. As far as the network is concerned, the name in an NDN packet can name anything – an endpoint, a data chunk in a movie or a book, a service, or a command to turn on some lights. This conceptually simple change allows NDN networks to use almost all of the Internet's well-tested engineering properties to solve not only communication problems but also digital distribution and control problems.

Our first several years of NDN design and development efforts tackled the challenge of turning this vision into an architectural framework capable of solving real problems. Our application-driven architecture development efforts force us to fill in details, as well as verify and shape the architecture. We translated our vision into a simple and elegant packet format design, a modular and extensible NDN forwarding daemon, and a set of supporting libraries. This next phase of the project has focused on deploying and evaluating the NDN architecture in four environments: building automation management systems, mobile health, multimedia real-time conferencing tools, and scientific data applications. The implementation and testing of pilot applications in these network environments further demonstrated our research progress in namespace design, trust management, and encryption-based access control. Highlights from this year include:

- Continued evolution the NDN Forwarding Daemon (NFD), to support application-driven experimentation with new NDN protocol features.
- Development of an Android version of NFD to promote NDN experimentation on mobile platforms.
- Implementation of a new transport protocol (InfoMax) that can intelligently filter streams of information in order to reduce transmitted data volume, while minimizing loss of information.
- A growing portfolio of supporting software libraries, including new APIs, transport mechanisms (Sync, information maximization), and security functionality, that leverage inherent capabilities of NDN, e.g., schematized trust, name-based access control.
- Demonstration of extremely scalable forwarding implementation using a billion synthetic names.
- Implementation and evaluation of hyperbolic routing performance to understand its feasibility in supporting NDN's interdomain routing.
- Multi-faceted evaluation of the architecture, from instrumentation of applications on the testbed, to uses of ndnSIM and the Mini-NDN emulator environment.
- Continued uses of NDN in the four courses taught by principal investigators.
- The second annual NDN Community meeting hosted by the NDN Consortium to promote a vibrant open source ecosystem of research and experimentation around NDN.

We have made tremendous progress in the last five years, and a larger community of information-centric networking research has evolved in parallel. We also helped NIST prepare an NDN workshop hosted by NIST at end of May 2016. Our progress revealed the importance of demonstrating NDN capabilities in IoT and big data environments, and highlighted the need for accessible software platform support and emulation capabilities to facilitate R&D on both the NDN architecture and applications that leverage it. We have received a year of supplement funding to complete four tasks: 1) completing and disseminating native NDN applications and associated design patterns, 2) demonstrating NDN scalability; 3) documenting and releasing reference implementations, and 4) documenting NDN design decisions and lessons learned.

Chapter 1

Introduction

This report summarizes our accomplishments during the second year of the “Named Data Networking Next Phase (NDN-NP)” project. This phase of the project focuses on deploying and evaluating the NDN architecture in four environments: building automation management systems, mobile health, multimedia real-time conferencing tools, and scientific data applications.

The first two environments represent critical areas in the design space for health IT and cyberphysical/IoT systems, respectively. For the building automation pilot application, we are in the process of demonstrating two uses on the UCLA campus: real-time monitoring from a web browser via basic Interest-Data exchange, and general SQL query support against historical data for report generation. For the mHealth application, we completed an initial end-to-end design and implementation of the NDNFit application, including a revised namespace design that supports named-based access control. We also developed an identity management Android application based on schematized trust, designed and implemented data transportation protocols, designed and implemented NFN integration mechanism, and implemented name-based access control (NAC). Mobility support in this application has driven several practical elements of the architecture: autoconfiguration, prefix registration, identity management, and encapsulation / LINK support.

Our real-time conferencing application has enjoyed the most real-world testing from the NDN collaborators at various campuses, as we forced ourselves to use it for some of our regular NDN seminars and project meetings to thoroughly stress-test it (and us!). The feedback was invaluable, revealing the need for applications to fully exploit, and verify, network-layer features in supporting the detection of latest data available, adaptive consumer retrieval, and performance optimizations.

The scientific data management environment emerged during the original FIA project as an ideal use case, and came with enthusiastic collaborators who enabled us to show immediate practical use of our research ideas to make their day jobs easier. A complementary project at one of the NDN campuses (UIUC) finalized a new transport protocol that intelligently filters streams of information in order to reduce transmitted data volume, while minimizing loss of information.

This application-driven approach compelled several concrete and fundamental architecture advances, but also required the development and extension of a set of software libraries that not only instantiate the evolving core of the NDN network architecture, but also support the experimentation that informs its evolution. To allow broader experimentation with NDN technology, and in particular experimentation in real mobile environments, we made a prototype port of our reference forwarding daemon implementation on Android platform (NFD-Android). We also implemented two different approaches to routing in NDN networks (link-state and hyperbolic routing with adaptive forwarding) to support comparative evaluation of these approaches in various scenarios.

The software development dimension of this project exceeded even our high expectations – but in the process we created a lasting and powerful artifact of the FIA program: a patent-free and open source software code base to encourage further architecture research experimentation and collaboration with around the world. We made one major and several minor releases of our core NDN forwarder (NFD) and supporting libraries, and support living documentation as well as a mailing list of over 100 people.

Some of the most fundamental architectural advances in the last year have related to security; specifically, codifying the previous year's invention of a security-conscious approach to namespace design that we call *schematized trust*, application of this approach to *name-based access control*, and development of mechanisms to support the authenticity of long-lived data, i.e., packets whose digital signature may have expired at the time of data consumption.

We continued progress on *data synchronization* as a significant NDN communication primitive. In NDN, data synchronization between multiple nodes supports basic services, such as public key distribution, file sharing, and route distribution. In addition to ChronoSync (which uses log-based representations of information) and iSync (which uses a two-level invertible Bloom filter structure to speed up dataset synchronization), this year we developed PartialSync, which enables individual consumers to synchronize with selected part of the dataset namespace, a popular usage pattern in publish-subscribe systems.

In the area of scalable forwarding, we completed our software implementation and evaluation of a scalable FIB longest name prefix lookup design based on the binary search of hash tables. We demonstrated 10 Gbps forwarding throughput with one billion synthetic longest name prefix matching rules, each containing up to seven name components. To the best of our knowledge, this is the largest dataset that has been studied for longest name prefix lookup.

We continued our commitment to multi-faceted evaluation of the architecture, from deployment and instrumentation of applications on the expanding NDN testbed, use of applications by team members, to uses of ndnSIM and the Mini-NDN emulator environment. We also documented four campuses use of NDN in the classroom and curriculum, and the growing NDN consortium to facilitate private and public sector engagement with the NDN project. We ended the second year of NDN-NP project with ten submissions to ICN 2016 conference, which report on various results of the project.

We have made tremendous progress in the last five years, but unexpected collaborations have revealed the value of further demonstrating NDN capabilities in IoT and big data environments, and highlighted needs for accessible software platform support and emulation capabilities to facilitate R&D on both the NDN architecture and applications that leverage it. We are honored to have recently received a year of NSF supplement funding to complete four tasks: 1) completing and disseminating native NDN applications and associated design patterns, 2) demonstrating NDN scalability; 3) documenting and releasing reference implementations, and 4) documenting NDN design decisions and lessons learned. We also hope to use this coming year to further organize and inspire the growing research community to develop new applications around NDN. We plan to offer a whole-day tutorial on NDN application development at the ICN conference, focused on enabling others to build on our success with application-driven architectural research.

Chapter 2

Network Environments / Applications

Contributors	
PIs	Jeffrey Burke, Van Jacobson & Lixia Zhang (UCLA), Tarek Abdelzaher (UIUC), Christos Papadopoulos (Colorado State)
Grad Students . .	Dustin O'Hara, Ilya Moiseenko, Wentao Shang, Yingdi Yu, Haitao Zhang (UCLA); Jongdeog Lee, Shiguang Wang (UIUC)
Undergrads	Akash Kapoor, Yang Sheng (UIUC)
Staff	Adeola Bannis, Jiawen Chen, Peter Gusev, Alex Horn, Jeff Thompson, Zhehao Wang, (UCLA); **Researcher:** Alex Afanasyev (UCLA)

As part of the NDN "Next Phase" research, the NDN project team proposed two network environments, **Enterprise Building Automation & Management** and **Open mHealth**, and one application cluster, **Mobile Multimedia**, to drive our research, verify the architecture design, and ground evaluation of the next phase of our project. The two environments represent critical areas in the design space for next-generation Health IT and Cyberphysical Systems, respectively. They also extend work started in the previous NDN FIA project on participatory sensing and instrumented environments to focus on specific application ecosystems where we believe NDN can address fundamental challenges that are unmet by IP.

2.1 Enterprise Building Automation and Management

For our purposes, Enterprise Building Automation and Management covers the intersection of three critical sub-areas: *industrial control systems* (ICS), including supervisory control and data acquisition (SCADA) and so-called smart grid [5], *enterprise networking*, and the *Internet of Things* (IOT) movement [2]. Our pilot application for this network environment is an initial NDN-based building monitoring system (NDN-BMS) deployment at UCLA. We are building an NDN-based collection, storage, and query system for real UCLA Facilities Management data collected from UCLA's Siemens building monitoring system. (As of April 2016, we have bridged a few hundred live data points from UCLA's campus monitoring to the NDN testbed. We are targeting several thousand points at up to 1Hz sample rate per point by the end of the project.) We are initially focusing on read-only access to sensing data. The 800 or so points of monitoring that we will have access to in 2016 generates 24M rows per year and cover a few buildings and a few data types. They include data from electrical, chilled water, and heating-ventilation-air-conditioning (HVAC) systems, as well as other sensors.

Our high level objectives for the pilot application are to support two dominant uses for this data on the UCLA campus: real-time monitoring from a web browser via basic Interest-Data exchange, and general SQL query support against historical data for report generation. This section summarizes our progress on storage and query support of building management data.

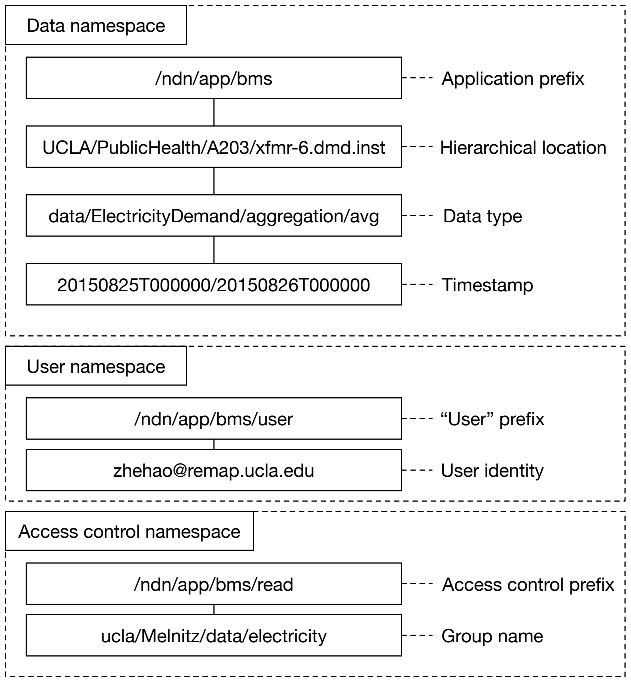


Figure 2.1: Example BMS data name

Mini-BMS [13] extends the NDN building automation management system work. This year's work incorporates data aggregation, schematized trust [14], name-based access control [15], and a visualization unit. Mini-BMS uses Mini-NDN to emulate nodes in a larger BMS network driven by real data from the UCLA campus. Each node keeps outstanding Interests for the data produced by its child nodes to gather the data for aggregation in a fixed time window. The system uses a hierarchical trust schema in which the certificate of a child node is signed by its parent, and a predetermined root of trust installed on each node. Figure 2.1 gives an example of the naming conventions for BMS data, and Figure 2.2 illustrates the structure of the deployed system, which makes historical raw and aggregated BMS data from UCLA campus available on the NDN testbed.

2.1.1 Progress towards milestones

We summarize progress toward our proposed milestones for this specific environment:

- *Review limitations in current IP-based architecture, for Facilities Management needs. (Y1)* In progress, continued in supplement work. Some of the insights from this process are captured in [9].
- *Design NDN namespace, repository, trust and communication model for use cases, such as energy management, new building commissioning, feedback control. (Y1; updated in Y2)* Schematized trust was designed and implemented in Year 2, along with an initial design for name-based access control. Because our EBAMS data starts in hierarchical namespaces on the application side, these approaches work well, though present some limitations, also described most recently in [9].
- *Implement low-level NDN applications, such as energy management data gathering. (Y1)* In progress, to be continued in supplement work. Our expectation in the supplment period is to bridge IoT considerations inspiring for new types of EBAMS with the infrastructure-based systems that we have started with.
- *Preliminary embedded platform support. (Y2)* Completed, described in the previous report. We continue to explore support for embedded platform through hackathons and other efforts outside of the NP funding, which has provided valuable insight into how the NDN approach can be adapted for

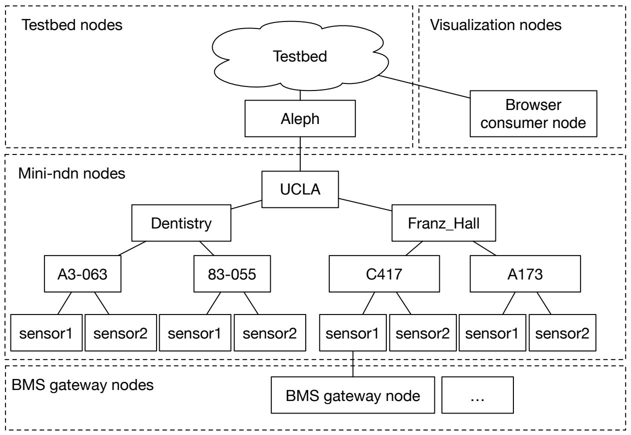


Figure 2.2: BMS deployment structure

constrained and embedded platforms.

- *Integrate UCLA building data (10-20 buildings) into NDN testbed, (Y2)* Done, with a paper forthcoming during the supplement period on the Mini-BMS project described above.
- *Implement high-level NDN application for enterprise building monitoring, applying distributed 3D visualization work done in the first FIA project. (Y2)* UCLA will complete visualization work during the supplement period, with an initial namespace visualizer in September 2016.

2.2 Open mHealth

Mobile health (mHealth) has emerged as an important market and a key area of Health IT, a national priority. The Open mHealth project [3] led by Deborah Estrin (Cornell) and Ida Sim (UC San Francisco) proposes a thin waist of open data interchange standards (Figure 2.3) to enable an ecosystem of sensing, storage, analysis, and user interface components to support medical discovery and evidence-based care. Specifically, the Open mHealth architecture includes standardized personal data vaults and health specific data exchange as the architecture's narrow waist, which provides health-specific syntactic and semantic data standards, patient identity standards, core data processing functions such as feature extraction and analytics and data stores that allow for selective, patient-controlled sharing.

The focus on data exchange as the backbone of the application ecosystem makes Open mHealth an excellent network environment to both drive and evaluate NDN. The NDN architecture embeds data exchange as its narrow waist, which provides a standard way to manage identity and trust relationship, provides provenance via the signature, and secures data at generation, with data owners directly controlling data sharing.

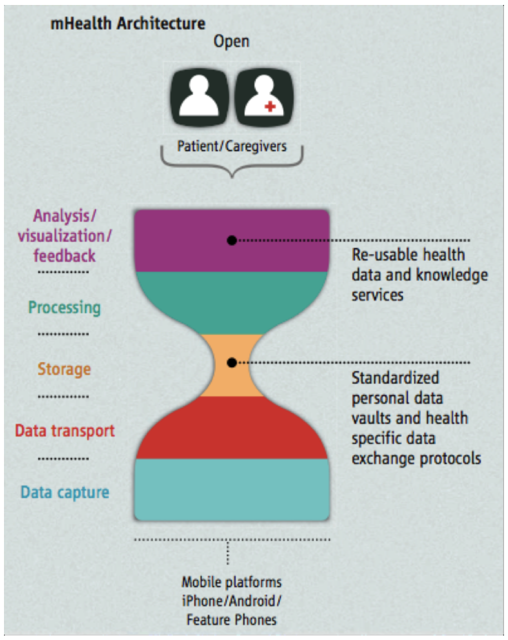

mHealth Architecture
Open
Patient/Caregivers
Analysis/ visualization/ feedback
Processing
Storage
Data transport
Data capture
Re-usable health data and knowledge services
Standardized personal data vaults and health specific data exchange protocols
Mobile platforms iPhone/Android/ Feature Phones

Figure 2.3: The Open mHealth architecture uses a data-centric hourglass model, where the interoperability layer ("thin waist") is based on standardized data exchange. [3]

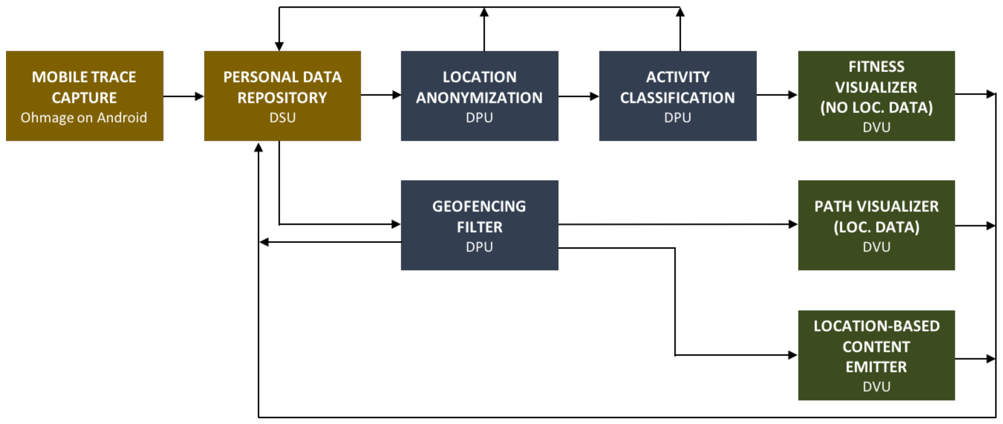


Figure 2.4: data flow for a single user of NDNfit

Our high level goal is to build a pilot NDN-based fitness application and system, which is compatible with Open mHealth paradigm, to capture, process and visualize users' time-location data. Figure 2.4 shows the data flow for a single user who uses NDNFit to retrieve: 1) fitness/activity metrics, 2) walking or running path visualizations, and 3) location-based content during exercise - all through the same ecosystem, but from different content providers.

This year, eight sites are collaborating on design and development of this application:

- UCLA REMAP - application design; library support; web-based visualization; values in design.
- UCLA IRL - architecture implications; repository; library and forwarder support; trust and security
- University of Arizona - forwarder support.
- University of Michigan - trust and security.
- University of Basel - data flow processing using NFN (Named Function Networking).
- Anyang - Ohmage capture application port.
- WUSTL - testbed support.
- UCSD - project management support.

2.2.1 NDNFit

This year, we completed an initial end-to-end design and implementation for the Open mHealth network environment, moving forward with the initial design developed last year. We revised the namespace design based on peer review and to support name-based access control, implemented the mobile capture application, designed and implemented an identity management Android application based on schematized trust, designed and implemented data transportation protocols, designed and implemented NFN integration mechanism, and implemented name-based access control (NAC) [15]. We also started to revise the previous autoconfiguration support mechanism required to deploy this application in the wild. This work is implemented in a demo system running on the testbed supporting a capture application, a data storage unit (DSU) and two data processing units (DPUs): one NFN DPU and one native NDN DPU.

Namespace

Figure 2.5 shows the second version of the proposed data namespace. Over the last year, we updated this namespace based on the requirements of name-based access control. The user publishes encrypted

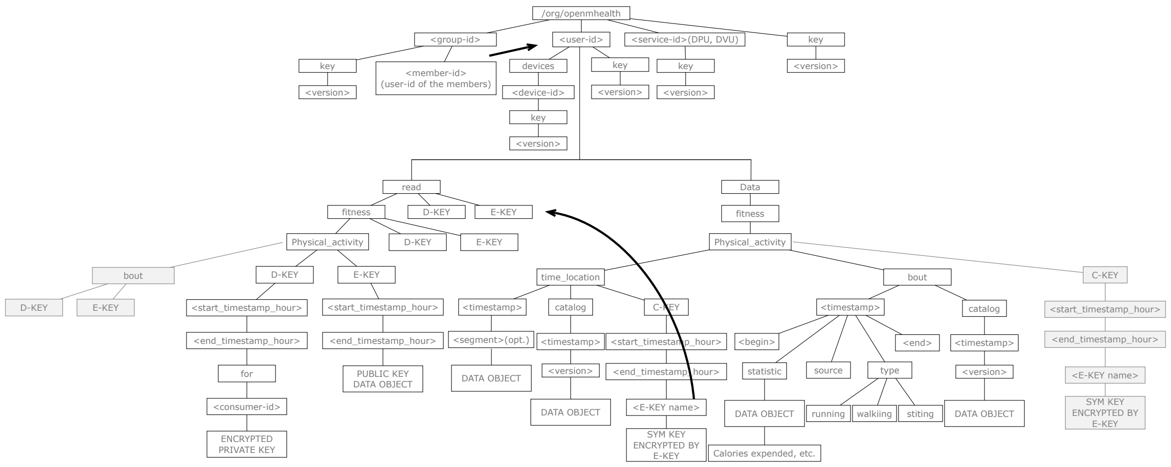


Figure 2.5: NDNFit Namespace, version 2.

health data under *data* branch and publish keys under corresponding entries in the *read* branch. The first implementation of NDNFit captures and processes time-location data. Data is fetched by time range, and data names include the timestamp as the last component.

### NDNFit Android Application

Anyang University and UCLA IRL have built NDNFit, an Android application (interface depicted in Figure 2.6), that captures time-location data, produces Data packets and temporarily stores these Data locally. It uses the NDNFit application protocol (Section 2.2.1) to upload data to the DSU.

### Identity manager

The NDNFit application is one of the first where we explore user-facing application management of the chain of identities (expressed in certificates) used for signing of data packets. The NDNFit application as currently implemented uses three types of identities: user, device, and application. An Android-based identity manager (ID manager) handles user and device identity generation, signature requests, and signing of certificates for these identities as needed. A new user requests their own subnamespace in the Open mHealth top-level namespace, and an authority for that namespace grants the request by signing a certificate corresponding to the user identity. In the current implementation of the system, the Android ID manager generates the user identity and employs web services to request an authority server sign the certificate, using email for user authentication. Lower-value keys are then authorized by the high-value key that is part of the user identity. For example, after the user identity is created, the ID manager generates a per-device identity for the phone it is running on, and the user identity signs the device identity. Then, the NDNFit generates an application identity upon its first launch, which it requests that the ID manager sign with the device identity. (The application then generates an instance identity on startup or every few hours, which is signed by the application identity, though this is not implemented yet.) The lower value keys actually sign data, which can be verified as an authorized part of the Open mHealth namespace by walking the trust chain all the way back to the Open mHealth authority's signature on the user identity. Figure 2.7 illustrates an example trust relationship. The application certificates, device certificates, and user certificates are all published in the DSU to ensure they are available for verification.

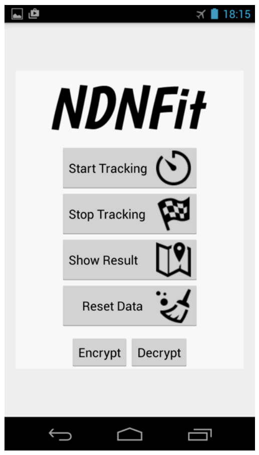


Figure 2.6: NDNFit Application UI

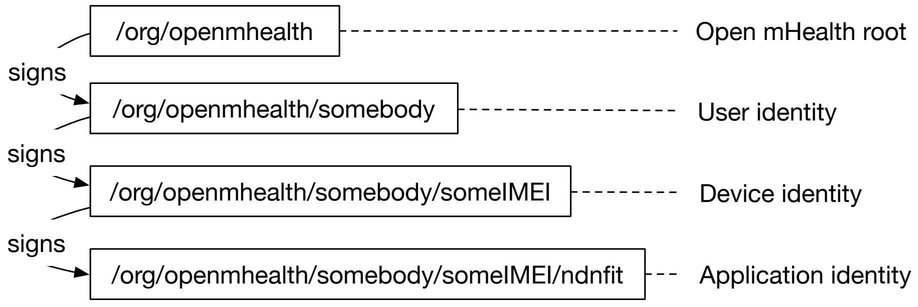

/org/openmhealth
Open mHealth root
signs
/org/openmhealth/somebody
User identity
signs
/org/openmhealth/somebody/someIMEI
Device identity
signs
/org/openmhealth/somebody/someIMEI/ndnfit
Application identity

Figure 2.7: User, device, and app identity for NDNFit application

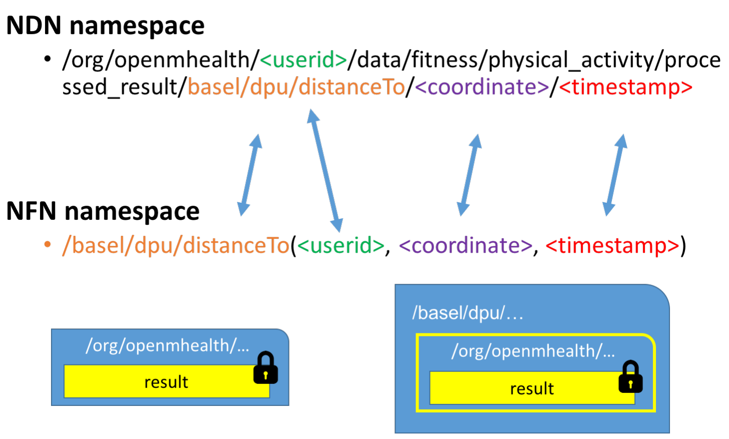

NDN namespace
• /org/openmhealth/<userid>/data/fitness/physical_activity/processed_result/basel/dpu/distanceTo/<coordinate>/<timestamp>
NFN namespace
• /basel/dpu/distanceTo(<userid>, <coordinate>, <timestamp>)
/org/openmhealth/…
result
/basel/dpu/…
/org/openmhealth/…
result

Figure 2.8: Use packet encapsulation to convert NDN packets to NFN packets

Data transport protocol

As part of finalizing the application design, we developed specific data transport protocols for cases where basic Interest-Data exchange was insufficient, including notification of new data and efficient retrieval of data with unpredictable names using catalogs (manifests). The approach taken in implementing the protocol addresses three challenges for communication between mobile devices and data storage units (DSUs): (1) intermittent connectivity; (2) fetching patterns for data named with a timestamp component that can't be predicted by the consumers; (3) power and storage constraints. The DSU implements an application pattern in which permanent storage "watches" namespaces of interest for new data, and then issues Interests to retrieve that data, which it stores and republishes. Briefly, the specific approach includes a *signal interest* that a mobile devices uses to notify a DSU of new data, a catalog the mobile provides to the DSU so the DSU knows what names to fetch, and a confirmation request Interest.

NFN integration mechanism

An interesting research challenge motivated by the Open mHealth environment is how to support distributed data processing. NDNFit uses NFN (Named Function Networking) [12] to implement DPUs. This year, we explored approaches to integrating NFN with NDN. The current implement of NDNFit provides a service that rewrites and translates packets from the NDN to NFN formats while preserving the trust relationships and access control used in NDNFit. Figure 2.8 shows a concrete translation example.

2.2.2 Progress towards milestones

We summarize progress toward our proposed milestones for this specific environment:

- *Review limitations in current IP-based architecture for Open mHealth needs. (Y1)* Completed previously.
- *Design namespace, repository, trust and communication model for use cases, e.g., diabetes or PTSD treatment (Y1; updated in Y2)* Finalized *and implemented* the complete design for the fitness use case.
- *Repository implementation providing backing storage for prototype applications. (Y1)* Complete along with supporting security components.
- *Integrate named data networking into the Ohmage mobile data collection framework. (Y2)* Done on the mobile side. For server-side integration, after further experimentation we deemed it unnecessary to generate a complete application; instead, we designed and integrated a native NDN approach for data processing in collaboration with the University of Basel and Prof. C. Tschudin's group.

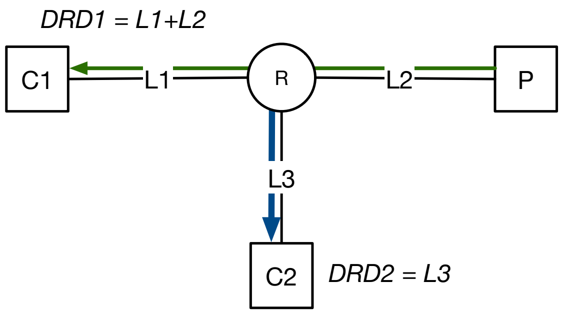


Figure 2.9: Data Retrieval Delays in 1-to-2 RTSD fetching scenario: $C2$ experiences smaller DRD value when it starts fetching after the $C1$ and receives cached data from the R.

- *Pilot user-facing application using NDN, for testing by Open mHealth team. (Y2)* We completed all components and integrated them into a demo system; we plan further work during the supplement to visualize the underlying system operation.

2.3 Mobile multimedia: ndnrtc (NDN real-time conferencing tool)

This year, we continued to develop and evaluate *ndn-rtc*, a complete real-time videoconferencing application based on the WebRTC library and VP8 codec. This application is helping us understand the challenges of low-latency communication over NDN. We are using it to explore the potential to shift real-time communication models from push-based to pull-based. In ndn-rtc, the producer's main task is to acquire video and audio data from media inputs, encode it, pack into network packets and buffer it to respond to incoming Interests. The NDN model shifts capability and control to the consumer, which selects streams based on user requirements and performance, leveraging NDN to enable scalable delivery at the network layer.

In addition to the *ndn-rtc* library, we built *ndncon*, GUI application on top of NDN-RTC that we hope can serve the NDN community's audio-video conferencing needs while leveraging the NDN Testbed. We had the ambitious goal of the NDN team using of *ndncon* for internal weekly meetings and seminars by the end of the project. We have been testing the software since April 2016 in order to improve and fine-tune our algorithms, and to better understand performance on the NDN testbed. We elaborate on the following aspects of testing further in this year's joint publication with Panasonic Research [6]:

- **Detecting the latest data available.** For real-time applications, we have set the design goal that consumers should be able to retrieve the latest data available from a producer without requiring their interests to directly reach that producer. This enables scaling up the number of consumers without impact on the producers. Conceptually, consumers aim to retrieve the most recent data from *the network* rather than from the producer. At a high level, this approach works: our experiments suggest that in multi-consumer scenarios for real-time data, only one consumer typically retrieves data directly from the the original producer, while others receive cached data (Figure 2.9). However, a challenge remains in how to most quickly and accurately establish the latest data available from the network when stream consumption is bootstrapped or when network connectivity changes. We refined techniques this year based on detecting stable values of inter-arrival delays of incoming packets. (Intuitively, cached data arrives as a function of network performance while fresh data arrives according to the original signal sample rate.)
- **Adaptive consumer retrieval.** New Interest pacing techniques enable consumers to tune interest expressions in order to adapt to changing conditions, like buffer starvation or old data arrival.
- **Performance improvements.** We improved overall library performance and robustness by implementing single-threaded consumers, asynchronous non-blocking logging, CPU load and memory footprint optimizations.
- **Real-world testing.** We simulcast several NDN seminars via *ndncon*, which revealed important issues

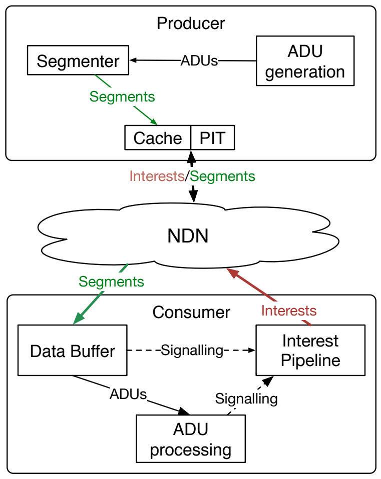


Figure 2.10: NDN consumer and producer conceptual design.

such as CPU and memory load of NFD, interactions of testbed strategy configurations on certain hubs, properly configured and functioning testbed prefix auto-registration and auto-propagation, and user certificate management.

The applications and architecture teams dedicated significant effort to documenting and critiquing generalizations of the *ndn-rtc* design. The applications team has been adapting the design (2.10) to other NDN applications that require real-time data dissemination: live person tracking (OpenPTrack) and distributed control system for live performance (Ananke) [6]. The architecture team provided feedback and offered alternatives based on years of TCP/IP-based conferencing experience. This collaborative approach motivated further implementation work to support new experimentation. To support cross-team experimentation with NDN-RTC, and research on the impact of forwarding strategy and congestion control algorithms on its performance, we developed a new cross-platform headless application, supporting OSX and Ubuntu. It can be built from sources or downloaded as a part of an NDN-RTC Docker image for easy setup.

2.4 Scientific Data Applications

Through a combination of this project funding, other NSF funding, and outside support and collaboration, we continued exploring NDN support for scientific data applications, in particular in the climate science and high energy physical science communities.

2.4.1 Climate modeling applications

Managing climate datasets is challenging due to their size, diversity, vast geographical distribution, number of files and lack of uniform naming structure. For example, CMIP5 (Coupled Model Inter-comparison Project) data is about 3.5 Petabytes in size and distributed among more than twenty institutes worldwide. CMIP5 datasets are distributed via a P2P system called ESGF, composed of about 20 independent nodes located around the world. ESGF nodes are loosely synchronized; they serve CMIP5 datasets, but also serve local datasets, available only from that node. The latter requires that users know which node hosts the local datasets.

We developed an application (*ndn-sci*) that provides efficient dataset publishing, discovery and retrieval using NDN. Last year we reported on software we developed that translates climate file names to uniform NDN names to ease data management. We walked all working ESGF nodes and found approximately 2.7M unique names from those sites. We converted them to NDN names using this translator and included them in our catalog. The purpose of the catalog is to help scientists discover the NDN names of the datasets they want. After discovery, users request each dataset by providing the name to NDN. Discovery can happen in three ways: (a) through a standard name component search and the catalog returns all names containing those components, (b) through typing in the exact name of the dataset, in which case the UI helps through a name component autocomplete function (presenting the user with all possible options as the user types the name), and (c) through a expanding portions of the entire hierarchical namespace and selecting the datasets the user wants. The UI also allows for staged data transfers, where the user initiates a transfer from one remote machine to another. Such transfers are useful for example, when data needs to move from a supercomputer to a repository.

An interesting example is the update control we implemented in the catalog. A common requirement is that only namespace owners can make changes in the catalog about the namespaces assigned to them. In other words, if CSU has or wants to insert names in the catalog, only CSU should be allowed to do so. This simple, yet powerful control was trivial to implement with NDN. We used standard NDN signatures to ensure that only legitimate namespace owners can publish datasets under that namespace by comparing the names of the datasets being inserted or deleted with the signature of the publisher that requests these actions. The catalog is currently serving a subset of the CMIP5 data and is running on our science testbed over ESnet. We demonstrated the software at Supercomputing 2015 [8] and documented it in several publications [11, 4, 10]. All climate science NDN applications and tools are under active development.[1]

2.4.2 NDN in High Energy Particle Physics (HEP)

As we described in last year's report, the HEP community has similar data management requirements as the climate community. For example, the HEP community generates petabytes of data, that is distributed via a tiered system around the world. The HEP community has developed its own distribution system called xrootd, that among other things is responsible for data publishing, discovery and distribution.

At SuperComputing 2015 we also demonstrated that the software we developed for the climate applications can be used in scientific domains such as HEP. We used the same catalog software to serve HEP data. The only change was the use of a different namespace that was appropriate for HEP dataset names. The namespace specification is a parameter provided to our catalog. We are still working to identify access patterns for HEP data, which will help us quantify how much NDN can improve HEP workflows over IP.

2.5 Information Maximization

UIUC (PI Tarek Abdelzaher) finalized a new transport protocol that intelligently filters streams of information in order to reduce transmitted data volume, while minimizing loss of information. Infomax is motivated by the emergence of IoT and sensor-filled environments, and the advent of social networks that democratize information broadcast, propelling us into a world of data overload. The growing disparity between the amounts of data generated and what suffices to meet application information needs suggests that an increasingly important network capability required to support networked applications of the future will be one of data sampling as a means of summarizing large (remote) data sets. Current application models require retrieving the entire data set regardless of the granularity of detailed required, followed by discarding all but the desired samples. Infomax is a generic transport protocol that can sample data in a consumer-controlled manner, while minimizing information loss resulting from non-delivery of the rest of the data set.

Infomax offers a producer/consumer API. To share data, the application on the producer side initializes an Infomax producer construct and specifies a name prefix associated with produced data. The application can then add data to the specified name subtree using a *produce* primitive. On the consumer side, the application specifies data of interest using a named prefix, which identifies a subtree and creates the receiving end for content that lives in this subtree. A *consume* primitive allows the application to retrieve a single sample. The application can call *consume* repeatedly until it retrieves a sufficient number of samples. The protocol ensures that the consumer receives samples in an order that maximizes coverage, i.e., samples of different large data clusters first, followed by samples of smaller sub-clusters that progressively refine the original

[1] `https://github.com/named-data/ndn-atmos`

clusters, until the consumer achieves the desired level of detail. Retrieval continues until the application stops calling the *consume* method.

The protocol assumes that hierarchical data object names reflect object similarity. Hence, more similar objects share a longer name prefix. Accordingly, consumers retrieve samples in a *minimum shared prefix first* order. Note that objects that share the minimum prefix have the least similarity, thus minimizing unnecessary redundancy and overlap among samples. The protocol determines the retrieval order solely according to data names, offering a generic way for summarization/sampling that leverages naming in NDN. This algorithm is similar to a breadth-first traversal of the specified name tree. Since the producer knows the exact tree topology, but the consumer only knows the general prefix that names the tree, the producer composes an ordered list of objects in the tree, in a longest-shared-prefix-first order. When a consumer first contacts the producer, the protocol (of the Infomax socket) requests this list first. Intermediate routers may cache this list per normal NDN operation. Consumers receiving the list then proceed with requesting objects sequentially, thereby receiving a diverse sampling first, then receiving progressive refinements due to the way objects are ordered. Different consumers can stop at different points during this retrieval process, hence receiving summaries of different degrees of detail.

This year, UIUC integrated Infomax with the NDN producer/consumer APIs, which include three data retrieval functions: simple data retrieval (SDR), reliable data retrieval (RDR), and unreliable data retrieval (UDR). SDR retrieves a single NDN data packet up to 8KB. Both RDR and UDR allow larger packets, but segment them into multiple packets. Infomax retrieves (a sampling of) entire content trees, not single data objects, thus it complements the aforementioned data retrieval protocols. The Infomax Data Retrieval (IDR) protocol sits on top of the other three and can use any of them to retrieve the individual objects, but uses RDR by default.

To test and evaluate the Infomax protocol, UIUC developed a Twitter feed summarization application (Iphone app). A producer downloads Twitter feeds on several topics of interest including disasters, civil unrest, and other newsworthy events. Consumers can then download their own customized news summaries of these events, controlling both the topics of their news summaries and the degree of summarization. The key in enabling this application was to create an appropriate name space for tweets that satisfies the Infomax imperative: namely, objects (tweets) that are more similar should share a longer name prefix. The implemented naming solution determined the fraction of shared tokens (words) between pairs of tweets as a measure of logical distance between them, then recursively applied K-means clustering on the resulting graph. Individual tweets were hierarchically named based on their cluster/subcluster. Subsequent testing demonstrated the efficacy of this approach at summarizing events and allowing drill-down into sub events. UIUC is preparing the application for sharing via the Apple AppStore. Infomax implementation and its application examples are currently being prepared for publication in a book chapter [7].

2.6 Libraries

Libraries are a critical part of NDN research as they link work on the protocol and forwarder with application development. We summarize recent activities and their motivations, as well as research in the future of application programming interfaces for NDN.

2.6.1 ndn-cxx: NDN C++ library with eXperimental eXtensions

To promote and support robust, effective, and diverse experimentation with the NDN architecture, and support development of the new forwarding daemon (NFD), in 2014 we forked the NDN Common Client Libraries C++ library development effort (NDN-CPP) and developed ndn-cxx, *C++ with eXperimental eXtensions*, a C++ library that implements all NDN protocol abstractions and provides a foundation for cutting edge experimentation with NDN technology. In particular, ndn-cxx is used to prototype new architectural features, which may then be incorporated into NDN-CPP. The development of ndn-cxx follows an application-driven iterative approach, taking feedback from application developers on how they use and interact with the library, what challenges they experience, and what changes they would like to see. We also strive to maintain stability within ndn-cxx release cycles.

We have continued development of the ndn-cxx library to ensure that it implements all NDN protocol abstractions and provides a foundation for cutting edge experimentation. Since May 2015 we have made five minor releases and one major release introducing variety of functionality:

1. support for Link abstraction, to enable data retrieval when the data's prefix is not globally reachable or when data needs to be retrieved from a moving producer.
2. implementation of the NDNLPv2 NDN Link Layer Adaptation protocol, to enable features such as packet processing instructions (next-hop choice, cache control, etc.), fragmentation and reassembly, and packet loss detection.
3. a Dispatcher helper class to simplify server-side implementation of the NFD management protocol.
4. Support for the upcoming PIB feature[2]
5. the PartialName type, which represents an arbitrary sequence of name components (while Name represents an absolute name).
6. Generalized signing API in KeyChain.

The following applications and projects continue to use ndn-cxx:

- **NFD** - NDN Forwarding Daemon
- **NLSR** - Named-data Link-State Routing protocol
- **repo-ng** - a new implementation of NDN repository
- **ChronoChat** - Multi-user NDN chat application
- **ChronoSync** - Sync library for multiuser real-time applications for NDN
- **ndn-tools** - A collection of NDN command-line tools (ndnping, ndn-traffic-generator, ndndump, etc.)
- **ndnSIM 2.0** - NDN simulation framework for NS-3 simulator engine
- **ndns** - Domain Name Service for Named Data Networking
- **ndn-atmos** - software suite to support ongoing climate model research at Colorado State University, Berkeley and other institutes
- **ndn-group-encrypt** - group-based encryption library for NDN applications

2.6.2 NDN-CCL: Common Client Libraries

The NDN Common Client Libraries (CCL) provide a common application programming interface (API) across multiple languages for building applications that communicate using NDN. They incorporate features often first introduced in ndn-cxx. Currently, the CCL is implemented in C++, Python, JavaScript and Java, with support for pure C and C# .NET. Since May 2015 our library development has included integrating security research results in schematized trust [14] and name-based access control [15], incorporating other new features motivated by the network environments, and improving performance:

1. To support resource-constrained devices in our BMS and mobile health environments, we extended support for signing and verifying HMAC signatures to PyNDN and the in-browser NDN-JS library. We also added support for a generic signature type, which lets the application compute the signature value and supply the encoding.
2. To enable authentication using identity and keys that persist across browser sessions, we added support to NDN-JS for *IndexedDB* storage, a web standard available in major browsers like Firefox and Chrome.
3. We also added support for a new package called *cryptography* [1], which more efficiently handles RSA and ECDSA signatures for devices powerful enough to use them.

[2]The Public Information Base (PIB) stores the public portion of a user's cryptography keys.

4. While signature operations dominate Data packet processing, encoding/decoding operations dominate Interest processing, which can challenge resource-constrained devices. We added optional support for C bindings to PyNDN, to directly use NDN-CPP Lite, which doubles the speed of the encoding/decoding operations.
5. Our mHealth application uses the Link object, so we implemented the capability to encode/decode a Link object and to add it to an Interest packet with a Selected Delegation field.
6. All of our environments are sensitive to power consumption, which means asynchronous I/O capabilities offer a huge advantage. (The CPU can stay relatively idle until it receives a network packet.) We extended the NDN-CCL API to allow an application to use its asynchronous library of choice, in a thread-safe manner to support multi-threading.
7. We added support for notifying applications about negative acknowledgment (NACK) messages, which the new NDN network link protocol uses to signal a reachability failure.
8. For improved security, the real-time videoconferencing application needs to create and serve per-session certificates in addition to regular data packets. So, we updated the MemoryContentCache so the application needs to register only one prefix with the forwarder but can internally dispatch incoming interests based on filters for different types of data being produced.
9. To support requests from the research community using the NDN software platform, we added support for Visual Studio using NDN-CPP, and created an NDN-DOT-NET library to support use of NDN in C# .NET applications. We created the NDN-DOT-NET library by adapting a converter which transforms Java code to C#. Therefore, new features in jNDN are made available for .NET applications without needing to maintain an explicit C# library for NDN.
10. We removed methods from NDN-CCL due to deprecation of the older NDNx forwarder, simplifying the API and code base.

NDN-CCL is used by applications including:

- **ndn-rtc, ndncon** - Real-time audio/video conferencing over NDN (NDN-CPP)
- **ndn-bms** - Building management pilot application (NDN-JS)
- **ndn-opt** - OpenPTrack publisher / consumer (NDN-JS; NDN-CPP)
- **NDNFit** - NDN Fitness application (jNDN, initially)
- **FoS** - Control system for REMAP "future of storytelling" project (NDN-JS; PyNDN)
- **ndn-iot** - IoT toolkit on Raspberry PI (PyNDN)
- **AmbiInfo** - Ambient informatics installation by REMAP (PyNDN on Raspberry PI)
- **ndnfs** - NDN File System, second version (NDN-CPP)
- **Routing status** - NDN routing status page (NDN-JS)

2.7 Open Challenges Raised by Network Environment Development

- Mobility support is a critical and not completely implemented aspect of the NDN platform, though significant progress has been made in the research and design of mobility techniques. The Open mHealth network application has been an excellent driver for several important elements of a practical and holistic NDN approach to mobility - autoconfiguration, prefix registration, identity management, and encapsulation / LINK support.
- Name confidentiality is an open issue, particularly well-motivated by Open mHealth, which has generated significant discussion (though no results yet) over the last year - with potential solutions ranging

from partial name encryption to encapsulation. This discussion also emphasized the importance of explaining general privacy benefits from the NDN architecture in comparison with TCP/IP.

- While name-based access control provides initial insight into how granular data-centric encryption can be achieved in practice, two significant questions have arisen: 1) how to keep APIs and abstractions simple for developers, and 2) how to address ad-hoc group formation. We have also identified other potential solutions to configurable access control using encryption, notably attribute-based encryption, which we hope to compare with this approach in the future.
- User-facing identity management will be an important part of NDN applications. NDN provides the means to map application-level identities into collections of named certificates that are used for signatures, but more work is required to regularize generation, signing, exchange, and revocation of certificates so each application does not need to re-invent an approach. Some progress has been made in the NDNFit and EBAMS applications, which borrow mechanisms originally created for the NDN testbed's routing certificate management.
- Our sample applications, such as *ndn-rtc*, are evolving to the point that certain aspects can be generalized. We have found that it is challenging to balance what has been learned from TCP/IP with a desire to not blindly adopt its abstractions where they do not fit. One specific example is in congestion control. The architecture can provide in-network support for congestion control by leveraging forwarder state, strategy, and packet naming, but requires new designs to support multi-path scenarios in a caching network, rather than direct adoption of techniques from the current internet.

References

[1] Cryptography.io library. `https://cryptography.io/en/latest/`.

[2] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[3] Deborah Estrin and Ida Sim. Open mhealth architecture: an engine for health care innovation. *Science*, 330(6005):759–760, 2010.

[4] C. Fan, S. Shannigrahi, S. DiBenedetto, C. Olschanowsky, C. Papadopoulos, and H. Newman. Managing scientific data with named data networking. In *Proceedings of the Fifth International Workshop on Network-Aware Data Management*. ACM, November 2015.

[5] Xi Fang, Satyajayant Misra, Guoliang Xue, and Dejun Yang. Smart grid—the new and improved power grid: A survey. *IEEE Communications Surveys & Tutorials*, 14(4), 2011.

[6] Peter Gusev, Zhehao Wang, Jeff Burke, Lixia Zhang, Eiichi Muramoto, Ryota Ohnishi, and Takahiro Yoneda. Real-time streaming data delivery over Named Data Networking (invited paper). *IEICE Transactions*, May 2016.

[7] Jongdeog Lee, Akash Kapoor, Md Tanvir Al Amin, Zeyuan Zhang, Radhika Goyal, Zhehao Wang, Ilya Moiseenko, and Tarek Abdelzaher. InfoMax: A Transport Layer Paradigm for the Age of Data Overload. In *Advances in Computer Communications and Networks - from green, mobile, pervasive networking to big data computing (Web of Science Book Citation Index (BkCI))*. River Publisher, July 2016.

[8] C. Olschanowsky, S. Shannigrahi, and C. Papadopoulos. Supporting climate research using named data networking. In *Local Metropolitan Area Networks (LANMAN), 2014 IEEE 20th International Workshop on*, pages 1–6, May 2014.

[9] Wentao Shang, Adeola Bannis, Teng Liang, Zhehao Wang, Yingdi Yu, Alexander Afanasyev, Jeff Thompson, Jeff Burke, Beichuan Zhang, and Lixia Zhang. Named Data Networking of things. In *Proceedings of 1st IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI'2016)*, April 2016. (Invited paper).

[10] S. Shannigrahi, C. Fan, and C. Papadopoulos. Improving large scientific data transfers using ndn strategies. In *ICN 2016*. ACM, May. under submission.

[11] S. Shannigrahi, C. Papadopoulos, E. Yeh, H. Newman, A.J. Barczyk, R. Liu, A. Sim, A. Mughal, I. Monga, J.R. Vlimant, and J. Wu. Named Data Networking in Climate Research and HEP Applications. *Physics: Conference Series*, 664(5), 2015.

[12] Manolis Sifalakis, Basil Kohler, Christopher Scherb, and Christian Tschudin. An information centric network for computing the distribution of computations. ICN, 2014.

[13] Zhehao Wang, Jiayi Meng, and Jeff Burke. Hierarchical storage for NDN building management system. 2nd NDN Community Meeting (NDNcomm), September 2015.

[14] Yingdi Yu, Alexander Afanasyev, David Clark, kc claffy, Van Jacobson, and Lixia Zhang. Schematizing Trust in Named Data Networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, September 2015.

[15] Yingdi Yu, Alexander Afanasyev, and Lixia Zhang. Name-based access control. Technical Report NDN-0034, Revision 2, NDN, January 2016.

Chapter 3

Security

The NDN architecture per se has built-in features (e.g., per packet signature) to enable data-centric authenticity, and rely on content encryption to achieve data-centric confidentiality. Driven by the two network environments (NDNFit and EBAMS), which may deliver sensitive data over the network, we developed a data-centric confidentiality solution, called *Name-based Access Control* [5]. Unlike traditional channel-based confidentiality solutions, which only secure data in motion, our approach also retains confidentiality of data at rest, which better facilitates data distribution systems.

We also investigated approaches to support the authenticity of long-lived data, i.e., packets whose digital signature may have expired at the time of data consumption. Traditional authentication models require signatures to be valid at the time of consumption, which require periodic re-signing to support long-term archival data. We proposed a *look back* authentication model, in which consumers check the validity of a signature at the time of production. We designed *DeLorean* [6] – a system that implements this model using a secure timestamp service that allows consumers to roll back their clocks and authenticate data in the context of its production.

Finally, we designed, implemented and evaluated a content poisoning system for NDN.

3.1 Name-based Access Control

The basic idea of data-centric confidentiality is: a producer encrypts the content in a data packet at the time of production so that only authorized consumers can decrypt the content. Data-centric confidentiality models offer two advantages over channel-based confidentiality: data remains confidential both in motion and in rest; and access control comes automatically via the packet signatures.

A common approach to content confidentiality between two users is to encrypt content using each authorized user's public key directly, but it typically leads to multiple encrypted copies of the same content, and any required packet segmentation (fragmentation) introduces additional storage overhead and reduces the efficiency of content delivery. Moreover, this approach assumes that a producer has complete knowledge about the access control policy of the produced data, which may not be true in some cases, e.g., distributed data production.

We approach the problem by encrypting content using a symmetric *content key* (C-KEY in Figure 3.1) and distributing the content key to authorized consumers, thus resulting in only one encrypted copy of content but multiple encrypted copies of keys. Content keys represent the minimal access control granularity. As a result, we convert the problem of data-centric confidentiality into a problem of content key distribution.

In order to facilitate content key distribution and enforce fine-grained access control, we designed a key distribution protocol called *Name-based Access Control* (NAC) (Figure 3.1). To facilitate distributed data production, NAC also separates the role of namespace owner from data producer. In NAC, the owner of a namespace can publish the namespace's access control policy in terms of named public keys and encrypted private keys, which we call *namespace key* (E-KEY and D-KEY in Figure 3.1). The encryption

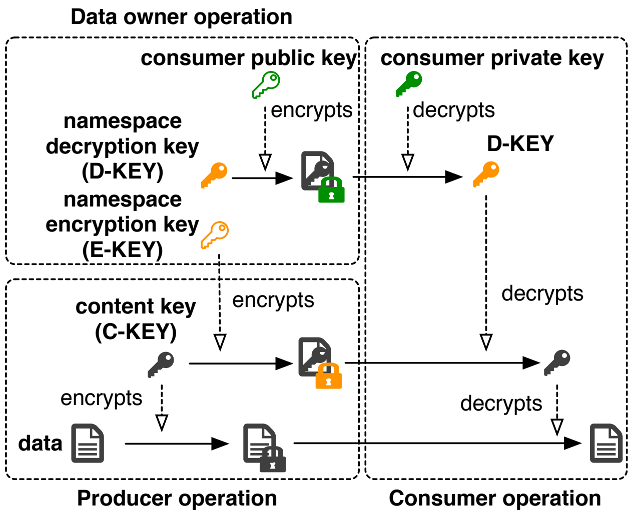
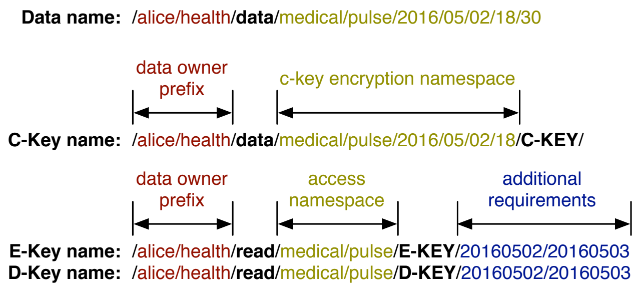


Figure 3.1: Overview of Name-based Access Control. Producers produce data in an encrypted format (encrypted using a content key which is also created by the producer). Producers retrieve namespace encryption keys (published by data owners or namespace owners) to encrypt content keys and publish the encrypted content key as a separate data packet. Data owners (or namespace owners) publish the corresponding namespace decryption key as the third encrypted data packet, which is encrypted using an authorized consumer's public key. An authorized consumer can retrieve the three encrypted data packets and recursively decrypt the keys and the original data.

key name (E-KEY in Figure 3.2) specifies the access namespace under which content keys should be encrypted using the encryption key. The encryption key name also includes additional requirements (e.g., time intervals) to further restrict the access granularity of the encryption key. Using the names listed in Figure 3.2 as an example, a medical monitoring device is producing heart rate data every minute under the namespace "`/alice/health/data/medical/pulse`". The producer (i.e., the monitor device) creates a content key (e.g., "`/alice/health/data/medical/pulse/2016/05/02/18/C-KEY`") to encrypt all the data produced within one hour. Each content key is named after its corresponding hour. A producer also retrieves the namespace key whose access namespace is a prefix of the content key encryption namespace, e.g., "`/alice/health/read/medical/pulse/E-KEY/20160502/20160503`". The producer checks the additional requirements from the name of the namespace key(e.g., time interval) to determine whether a content key falls into the scope of the retrieved namespace key and encrypt the content key only if it is covered by the namespace key.



Data name: /alice/health/data/medical/pulse/2016/05/02/18/30
data owner prefix
c-key encryption namespace
C-Key name: /alice/health/data/medical/pulse/2016/05/02/18/C-KEY/
data owner prefix
access namespace
additional requirements
E-Key name: /alice/health/read/medical/pulse/E-KEY/20160502/20160503
D-Key name: /alice/health/read/medical/pulse/D-KEY/20160502/20160503

Figure 3.2: Encryption key naming convention in NAC. The name of content key (C-KEY) implies that all the data under the content key namespace is encrypted by the content key. The name of namespace key (E-KEY and C-KEY) determines which content key should be encrypted using the namespace key, i.e., the access namespace in E-KEY name must be a prefix of the content key encryption namespace.

The namespace owner encrypts the corresponding decryption key (D-KEY) using the public keys of authorized consumers and publishes the encrypted decryption keys also as data packets (the packet with green lock in Figure 3.1). In NAC, we defined naming convention for encrypted data and name content, content key and namespace decryption key using the same naming convention (Figure 3.3). More specifically, any encrypted data name is a concatenation of the plain text data name, a special name component "`FOR`", and the encrypting key name. Following the encrypted data naming convention, authorized consumers can retrieve data and keys and construct a key chain to decrypt the original data. For example, Alice may grant Bob the access to heart rate data by encrypting the corresponding D-KEY using bob's public key and name the encrypted D-KEY as the last one in Figure 3.3

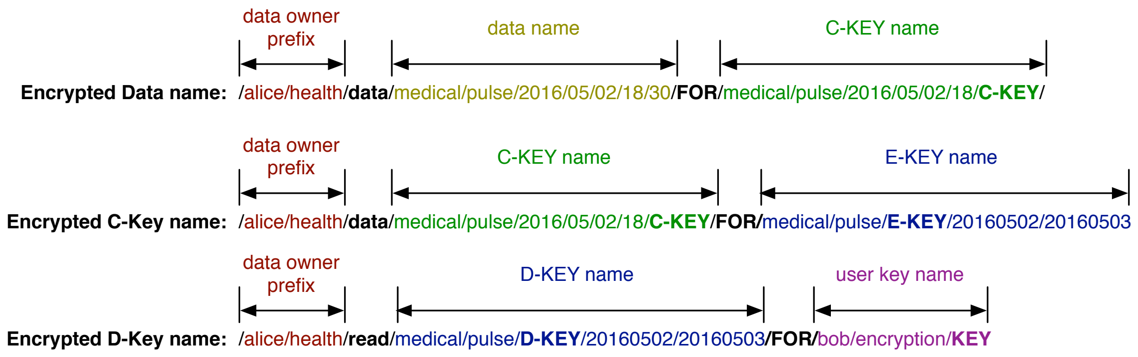

data owner prefix
data name
C-KEY name
Encrypted Data name: /alice/health/data/medical/pulse/2016/05/02/18/30/FOR/medical/pulse/2016/05/02/18/C-KEY/
data owner prefix
C-KEY name
E-KEY name
Encrypted C-Key name: /alice/health/data/medical/pulse/2016/05/02/18/C-KEY/FOR/medical/pulse/E-KEY/20160502/20160503
data owner prefix
D-KEY name
user key name
Encrypted D-Key name: /alice/health/read/medical/pulse/D-KEY/20160502/20160503/FOR/bob/encryption/KEY

Figure 3.3: Encryption key naming convention in NAC. The first name refers to data encrypted using a content key (the packet with black lock in Figure 3.1). The second name refers to a content key encrypted using a namespace key (the packet with orange lock in Figure 3.1). The last one refers to a namespace decryption key encrypted using an authorized consumer public key (the packet with green lock in Figure 3.1).

3.2 NDN DeLorean

NDN enables data-centric authenticity by mandating per-packet digital signature. Unlike physical signatures, digital signatures however may not be considered trustworthy over prolonged time periods: given enough computation power and time, it is possible to reconstruct the corresponding private key and issue impersonated signatures. In addition, since each created signature is indeed an encrypted data digest using a private key, keep using the same private key may render the key vulnerable to known plaintext attack. Moreover there is also a chance that the keys get accidentally or maliciously leaked to adversaries. As a result, current practices recommend the use of relatively short-lived signatures/certificates (from several months to a couple of years) [2]. This limited lifetime span works well for channel-based security model since communication channels have a limited duration, but not so well for NDN's data-centric security model. The lifetime of an NDN data packet can outlive the lifetime of its signature, especially in cases of historical data archives.

We proposed an authentication system for NDN data archives, NDN DeLorean, which uses a *look back* data authentication model: the data authentication process first rolls the clock back to the time of the data's creation. In order to allow consumers to securely rollback the reference time for data authentication, we designed a publicly auditable timestamp service that issues proofs of data creation times by logging the fingerprints of archived data in the form of Merkle tree. Given a data packet, the certificates that authenticate its signature (certification chain), and the proof of the creation time (i.e., the fingerprints created by the timestamp service) of data and certificates, one can always authenticate the data, regardless of the signature expiration and even the fact that the private key may have become public.

DeLorean is an always-on service that publishes a data chronicle (Figure 3.4). The chronicle consists of a sequence of volumes, each containing fingerprints of the witnessed data packets, such as specific USA Today articles, within a fixed time slot, e.g., 10 minutes. The existence of a data packet (its fingerprint) in

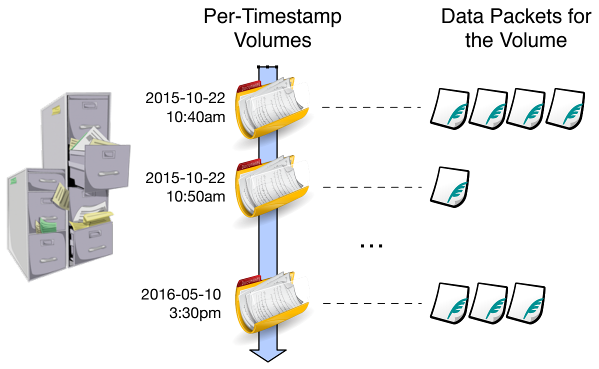
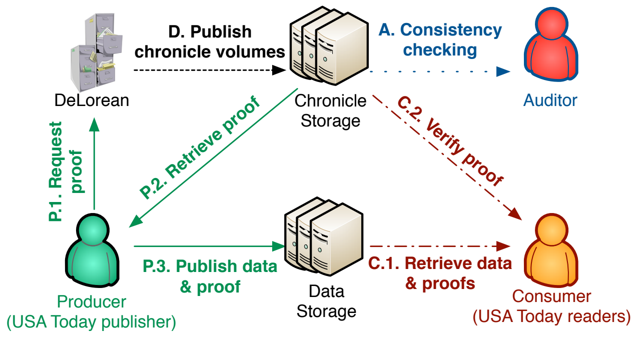


Figure 3.4: DeLorean Chronicle

a particular volume is a timestamp proof that the data packet existed before the end of the corresponding time slot. The DeLorean system finalizes each volume at the end of each time slot and publishes it as a set of data packets. The finalized volume cannot be changed without invalidating consistency with any future volumes.



DeLorean
D. Publish chronicle volumes
Chronicle Storage
A. Consistency checking
Auditor
P.1. Request proof
P.2. Retrieve proof
C.2. Verify proof
Producer (USA Today publisher)
P.3. Publish data & proof
Data Storage
C.1. Retrieve data & proofs
Consumer (USA Today readers)

Figure 3.5: DeLorean Workflow.

At any time, a data producer (article's author) or an archive service on the producers behalf (USA Today publisher) can request a timestamp proof for data (articles) from DeLorean (Flow P.1 in Figure 3.5), supplying a fingerprint of the archived data in form of a hash digest of an individual data packet or a digest of the manifest that represents a data collection. The response to this request is a name of the chronicle volume that DeLorean will publish at the end of the current cycle (Flow D) and the index of the fingerprint in the volume. After waiting until the volume is ready (on average a 5 minute wait in our example), the producer can retrieve the volume to verify whether DeLorean has included the data fingerprint in the volume (Flow P.2). In the end, the producer can publish the timestamp proof, which includes the full name of the volume and the index of data fingerprint, alongside the data (Flows P.3).

To verify data independently of its signature validity, consumers need to look back to the time point when data was produced (or time stamped). A consumer first obtains the corresponding timestamp proof, which can be stored alongside the data (Flow C.1), and verifies the data's existence by retrieving several additional DeLorean volumes. Similarly, the consumer verifies the existence of the data's signing key certificates.4 With all certificates proving their existences, the consumer can verify the data signature as if it were the time of the data's production.

In order to ensure the correct and truthful operations of DeLorean, a set of third-party auditors continuously check the consistency of the chronicle (Flow A), i.e., checking that DeLorean has not modified the previously published volumes. If auditors detect that DeLorean has modified the chronicle, the users of the service (auditors, data producers, and consumers) will take immediate actions to either fix the issue or abandon the specific instance of DeLorean service. In order to guarantee consistency, at least one auditor must retrieve each DeLorean volume around the time it is published. The more auditors in the process, the less frequently each individual auditor needs to perform consistency checking. Note that although consumer and producer roles are separate from the auditor role in Figure 3.5, they can be (and, from security perspective, should be) combined.

3.3 Content Poisoning Mitigation

Content poisoning attacks are a significant problem in Information Centric Netw orks ICN), such as Named Data Networking. In a content poisoning attack, an attacker injects bogus content into the network with a legitimate name. While users will reject the content because of signature mismatch, the network is largely unaware of the problem due to the computational burden of on-the-fly packet verifi cation. Thus, subsequent requests may trigger the return of bogus content, constituting a denial of service attack. While NDN could resist poisoned content by restricting prefix advertisement, the latter interferes with the "content from anywhere" principle, which we consider to be a great advantage of NDN. We investigated NDN content poisoning and surveyed the state-of-the-art in mitigation mechanisms. We developed a novel system for detecting, reporting, and avoiding poisoned content that leverages the verification work that users must do anyway. We evaluated two evasion strategies, Immediate Failover and Probe First, that capture the spectrum of possible solutions to avoiding bad content. Our conceptualization of content poisoning in NDN as essentially a forwarding problem enables NDN-based mitigation mechanisms via the use of adaptive forwarding strategies [3].

3.4 Plan For Next Year

We hope to complete three tasks in the next year: 1) automation of certificate management, 2) automation of data signing and verification based on schematized trust, and 3) applying name-based access control design to NP environments, specifically NDNfit and EBAMS (Enterprise Building Automation & Management).

For task (1), our plan is to adapt ACME, the Automatic Certificate Management Engine[2] protocol designed for Lets Encrypt, for NDN certificate issuance. ACME will replace the old manually managed NDN certificate system. Compared to the existing system, NDN-ACME will be able to support multiple certificate hierarchies, instead of supporting only the NDN testbed certificate hierarchy. The new system will enable multiple validation approaches, such as email, shared secrets, and DNS, instead of email being the only one validation mechanism. Moreover, the new system will provide automated certificate renewal and key rollover, which are sorely missing now, as well as the use of short-lived keys, which will reduce the need for key revocation.

Task (2) has three sub-tasks. First, we plan to overcome extend the current trust schema syntax [4] in order to allow users to describe more application semantics in trust schema, such as *userId* or *timestamp*, as well as supporting user-defined syntax in the new trust schema format. Second, we plan to remove the current limitation of having users manually write trust rules for a trust model, which is prone to errors. We will build a GUI tool that enables users to easily express the trust model in a graphical form, and that automatically converts the graph into trust schema. The tool will also provide a verification interface, through which users can supply test cases in terms of data name and key name pair. The tool can output the result of compliance checking for user to verify the correctness of the trust rule expression. The tool may also accept a single data name as verification input and output all the possible signing key namespace as an alternative correctness verification. Finally, we plan to integrate signing part of trust schema with the new NDN certificate system mentioned above. In the current implementation, if there are no available signing keys, the library will refuse to sign data. By integrating trust schema with the certificate system, the library can automatically generate missing keys and request certificates using the new ndncert [1].

Task (3) is to turn the Name-based Access Control (NAC) into functioning code, integrate it into the NDNfit and EBAMS applications, discover potential issues in order to and further improve its design and implementation.

Task (4) is to complete communication confidentiality support in NDN based on NAC, which provides content confidentiality. Data names carrying application semantics may also reveal sensitive information and may cause privacy issues in some cases. We discussed the privacy issues of NDN during the March 2016 NDN retreat, and agreed to explore two communication scenarios: point-to-point ephemeral communication and data distribution. Our initial plan is to provide a DTLS equivalent solution for the point to point ephemeral communication and provide an anonymizing proxy based solution to maximize the efficiency of data distribution with data name obfuscation.

References

[1] Alexander Afanasyev. NDNCERT: User public key certification system for NDN Testbed. 1st NDN Community Meeting (NDNcomm), September 2014. `http://www.caida.org/workshops/ndn/1409/`.

[2] Richard Barnes, Peter Eckersley, Seth Schoen, J. Alex Halderman, and James Kasten. Automatic Certificate Management Environment (ACME). `https://github.com/letsencrypt/acme-spec`, September 2014.

[3] S. DiBenedetto and C. Papadopoulos. Mitigating poisoned content with forwarding strategy. In *The third Workshop on Name-Oriented Mobility: Architecture, Alg orithms and Applications (NOM)*, April 2016.

[4] Yingdi Yu, Alexander Afanasyev, David Clark, kc claffy, Van Jacobson, and Lixia Zhang. Schematizing Trust in Named Data Networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, September 2015.

[5] Yingdi Yu, Alexander Afanasyev, and Lixia Zhang. Name-based access control. Technical Report NDN-0034, Revision 2, NDN, January 2016.

[6] Yingdi Yu, Alexander Afanasyev, and Lixia Zhang. NDN DeLorean: An Authentication System for Data Archives in Named Data Networking. Technical Report NDN-0040, NDN, May 2016.

Chapter 4

Distributed Dataset Synchronization

Early in the NDN project, we identified a common need by all distributed applications: synchronization of shared dataset, or Sync (See [4], Section 2). A list of names (or name space) represents a dataset, that may be shared among distributed parties. While various applications differ in their goals and communication patterns, the need to synchronize the data represented by a collection of names or namespaces is identical. We have come to consider Sync a fundamental NDN communication primitive. Since distributed applications are a generalization of 2-party communications, we can view sync as a generalization of end-to-end reliable data delivery. However Sync differs from the existing end-to-end reliable transport protocols, such as TCP, in two fundamental ways. First, Sync reliably synchronizes application named datasets among multiple parties, while a TCP connection delivers byte streams identified by its two endpoints. Second, a TCP connection delivers byte streams between two specific endpoints synchronously (i.e. both must be online at the same time), while NDN's Sync can fetch data packets from anywhere it can find the matching data items, either in real time or asynchronously. The ability to reconcile set differences asynchronously is especially useful in constrained environments with ad hoc and intermittent connectivity.

Over the last few years we have explored various different approaches to Sync support. We initially developed *ChronoSync* to support a Multi-User Chat application [9]; *ChronoSync* has since evolved to a general library utility used by other NDN applications including NLSR [5], ChronoShare [1], and scientific data management applications [6, 2]. Another approach, *iSync*, differs from ChronoSync in that it utilizes an Invertible Bloom filter (IBF) to reconcile data collections between nodes [7]. Our progress during this reporting period includes a) development of *Round-Sync*, an improvement over ChronoSync, and b) *PartialSync* [8] which allows individual consumers to synchronize a *subset* of the dataset with the producer.

In the rest of this chapter we first give a brief summary of ChronoSync and iSync to illustrate their different design approaches, followed by a description of Round-Sync (which is based on ChronoSync) and PartialSync (which adopted certain design choices from both ChronoSync and iSync). We then summarize our current understanding of the Sync design space, and discuss our ongoing effort and plan for the coming year in the Sync area.

4.1 Chronosync and iSync

The **Chronosync** design exploits naming rules to ease dataset synchronization. It assumes that each participant in a distributed application names its produced data sequentially. Therefore Chronosync only needs to keep track of the latest sequence number of each participant, and the latest state of the dataset can then be represented by the cryptographic digest over everyone's identifier plus sequence number.

Chronosync participants interact with each other using two types of Interest-Data exchanges: synchronization (sync) and application data. A sync interest carries the name of the application instance plus the cryptographic digest that represents the sender's knowledge of the current dataset which ChronoSync will deliver to all other participants. For example, a sync interest for Chronochat application instance looks like `/ndn/chronchat/lunch-talk/a1324asd9`, where `/chronchat/lunch-talk/` names the application instance, and `a1324asd9` is the digest. ChronoSync will deliver this interest to all participants who announce `/ndn/chronchat/lunch-talk/` prefix.[1] This sync interest serves two purposes: it checks to see whether anyone else holds a different digest, and if so, it tries to resolve the difference; it also checks to see whether anyone has produced new data since the state represented by the digest. To be able to tell whether a received sync interest represents the latest sync state, each participant maintains the history of its observed dataset changes in form of digest logs.

Any recipient of a sync Interest S_i who has newer information can respond with a data packet that includes one or more pieces new data (in the form of [producer name, sequence number] pairs) missed by the sender of S_i. Once a participant learns of changes to the dataset, it can decide whether and when to retrieve the missing data from its producers. However when two or more nodes produce data at the same time, they will all respond to S_i, but the sender of S_i can receive only one node's data, and has to invoke a recovery mechanism to retrieve the rest. ChronoSync also has sync recovery mechanisms that enables all participants to reach a common state after the network heals from a partition.

The iSync protocol utilizes the invertible Bloom filter (IBF) to reconcile sync collections between participants [3]. IBF is a hash-indexed, redundant table and supports item insertion, deletion, and limited inversion (i.e., extraction of stored items). Moreover, one can obtain a difference set between two IBFs by subtracting their IBF entries. iSync uses this unique subtraction operation to discover multiple differences in a single subtraction, which leads to an efficient implementation with a low computation and communication overhead. iSync consists of a repository and a sync agent. When the repository adds a new item to the collection, it notifies the sync agent which in turn indexes the inserted content name and updates a digest that reflects all names in the collection. The sync agent notifies other participants of its local digests by periodically broadcasting the digest (equivalent to sync interest in Chronosync), while receiving remote digests. When a remote collection digest does not match the local one, a reconciliation process starts, which involves repeatedly requesting, receiving, and comparing remote IBF against the local IBF.

iSync uses a hierarchical two-level IBF: *Repository IBF* and *collection IBFs.* The former records the status of the entire repository, while the latter logs insertions and deletions of each sync collection separately. An update to a collection changes the collection's IBF in only one second level digest, by adding the hash of the new data name into the corresponding *collection IBF*, which then invokes an update to the repository IBF and digest. We (Washington U.) designed and implemented the iSync protocol [7], and evaluated iSyncs performance by comparing it to the CCNx synchronization protocol. Experiments show that iSync is about eight times faster across a range of network topologies and sizes, and that it reduces synchronization traffic by about 90%.

Comparing Chronosync and iSync reveals two basic differences. First, with its use of an IBF, iSync can discover multiple changes to the dataset through one sync exchange, while Chronosync may need to iterate through all changes, especially with multiple simultaneous producers. Second, and related, the storage and computation cost of IBF is nontrivial. iSync makes no assumption about data names, and its volume of name collections grows over time, and so does the IBF data structure overhead. For example, each iSync repo node may need to create multiple IBFs in a sync period if the number of updates since the previous sync interest has exceeded the capacity of the IBF; other repos need to retrieve and process all those IBFs in order to discover all the updates in the last sync period.

Currently, both Chronosync and iSync must deliver sync interests to all participants in a distributed application, highlighting the need for more scalable solutions, still an open research topic.

4.2 Round-Sync

ChronoSync uses a Sync interest for two purposes: a) to check to see whether anyone else holds a different digest; and b) to check to see whether anyone has produced new data since the state represented by the digest. This double-meaning of Sync interests leads to certain performance limitations of ChronoSync. First, each node must resolve any differences before producing new data, otherwise additional new data

[1]An alternative to letting participants make routing announcements is to use broadcast, i.e. sending sync interest using the name `/ndn/broadcast/chronchat/lunch-talk/a1324asd9`, which is the current default implementation. A scalable and efficient solution for delivering sync interests remains an active research effort.

would make reconciliation more difficult. Second, if multiple nodes produce new data simultaneously, they cannot recognize the digest in the Sync interests generated by others, thus they must reconcile first before producing any new data. Besides delaying the new data propagation, reconciliation itself also carries a cost.

Round-Sync aims to reduce the invocation of state reconciliation by decoupling the function of detecting out-of-synchronization from fetching new data. Round-Sync divides data generation into rounds, that are identified by monotonically increasing numbers. When a node A produces a piece of data at round n, it immediately increases the round number to $n + 1$ and sends a Data interest to solicit data for round $n + 1$. This reduces, although does not eliminate, the chance of multiple nodes producing new data in the same round.

Each round n has a digest G_n which is a cryptographic hash of all data names produced at round n. The dataset also has a cumulative digest G_N which is the hash of all the round digests up to round n. Round-Sync uses two different types of interests, one for data fetching (Data interests), and one for detecting out-of-synchronization state in a round (Sync Interests). These mechanisms enable Round-Sync to synchronize data production for each round separately. When a node A produces a piece of data at round n, it increases the round number to $n + 1$ and sends a Data interest to solicit data for round $n + 1$. If another node B also produces data at round n before it receives A's data, B will have a different round digest than A's. While A and B resolve their differences, node C may produce data at round $n + 1$ which does not interfere with A and B's reconciliation.

In summary, by cutting data production into different rounds and decoupling the detection of out-of-synchronization from fetching new data, Round-Sync reduces the chance of data production collision (one sync interest solicited multiple pieces of data produced by multiple nodes) and allows data production in parallel with dataset synchronization. At the time of this writing we are evaluating the performance of Round-Sync as compared to ChronoSync through simulation, hoping to achieve quantitative results soon.

4.3 PartialSync

PartialSync protocol [8] is designed to efficiently synchronize datasets between consumers and producers, where individual consumers may be interested in only a subset of the whole data collection, and may synchronize with any producer sharing the same dataset, assuming that a) all producers keep synchronized with each others, say through either ChronoSync or iSync, and b) they can be reached by using the same routable name prefixes.

PartialSync uses regular Bloom Filters (BF) to let consumers express their subscriptions (*Subscription List*), and uses Invertible Bloom Filter (IBF) to represent a producer's latest dataset (*Producer State*). A consumer keeps the producer's IBF and queries for an update periodically by attaching both its BF and the producer's previous IBF to each query, expressed as a PartialSync interest. By comparing the differences between the previous IBF carried in the query and its current IBF, the producer can generate a list of names corresponding to the new data that has been produced in the period between the previous and new IBFs. Using the subscription list carried in the query, the producer can notify the consumer of all changes in the subset to which the consumer has subscribed.

To address the scalability issue with the number of names encoded in an IBF, PartialSync adopts ChronoSync's approach of letting each producer name data sequentially. Therefore the latest dataset can be represented by an IBF which contains only one data item from each producer, which is the producer's name plus its latest sequence number. When a producer P generates a new data item and increases its sequence number from N to $N + 1$, PartialSync will remove the item $[P, N]$ from the IBF and add $[P, N + 1]$ to the IBF. In doing so, the IBF contains only M items, where M is the number of producers.

Upon its start, a consumer first sends a PartialSync Interest to the Sync prefix, the interest includes a Subscription List BF and an empty IBF. This interest may reach any of the producers. The producer sends back a PartialSync Reply which includes its latest IBF, as well as the latest names, in the form of [producer, seq#], of all the data streams as part of the content of the message. The consumer can then retrieve the latest data in its subscription list accordingly. In its subsequent queries, the consumer includes in its PartialSync interest the subscription BF and the last IBF it obtained from the producer. Whenever the producer receives a PartialSync Interest from the consumer, it first parses the BF and the IBF in the consumer's PartialSync Interest. If the IBF from the consumer and its own IBF are the same, the producer waits until new data arrives that matches the consumer's subscription list. Otherwise, if they are different, the producer checks whether there are any updates to data to which the consumer is subscribed. In both cases, the producer sends a PartialSync Reply which contains a list of updated data names.

When a consumer receives a PartialSync Reply, the consumer first checks whether each received data name is in the subscription list; due to the false positive property of BFs, the producer may occasionally send data names not on the consumer's subscription list. For each subscribed data name, it then checks whether the sequence number is newer than its own version, and if so, the consumer sends an Interest to fetch the newer data. The consumer will send another PartialSync Interest to the producer, either upon receiving a PartialSync Reply or upon the timeout of the previous PartialSync interest, which includes the received IBF.

In summary, PartialSync supports scalable and efficient subscription service through the following means. 1) Like ChronoSync, it uses naming conventions to keep data items in the IBF constant. 2) Like iSync, it makes use of an IBF to easily detect multiple updates in one sync step. 3) It uses BF to express subscriptions, so that producers only inform a consumer about updates to data of interest to the consumer. 4) It carries the consumer state in a PartialSync interest, which frees the producers from keeping consumer state and allows consumers to sync with any producer (assuming producers keep sync'ed with each other). In contrast to ChronoSync and iSync, whose sync interests are multicast to all participants, PartialSync interests are anycast to any one of the producers.

4.4 Summary and Plan for Next Year

Data synchronization is an important communication mechanism for distributed applications over NDN. Existing Sync protocols make different design tradeoffs in how they achieve dataset reconciliation: ChronoSync and RoundSync leverage the naming convention of using sequence numbers in data names to allow efficient encoding of the synchronized namespace as a list of [producer name, latest sequence number] pairs; iSync supports arbitrary namespace structure at the cost of higher storage and processing overhead associated with IBF. PartialSync can be applied in distributed applications to support synchronization between consumers and multiple producers, but requires producers to synchronize using ChronoSync or iSync.

Our plans for the next year include: 1) continuing to explore efficient dataset reconciliation mechanisms that can improve the existing Sync protocols, 2) evaluating the communication cost of multicasting sync interests to all participants and designing more scalable and efficient solutions for delivering sync interests, 3) improving the application interface and library support for the Sync protocols by providing more powerful and developer-friendly abstractions over the core Sync functionality, and 4) developing new distributed applications over the Sync protocols and evaluate their correctness, efficiency, and usability.

References

[1] Alexander Afanasyev, Zhenkai Zhu, Yingdi Yu, Lijing Wang, and Lixia Zhang. The Story of ChronoShare, or How NDN Brought Distributed Secure File Sharing Back. In *Proceedings of IEEE MASS 2015 Workshop on Content-Centric Networks*, October 2015.

[2] Chengyu Fan, Susmit Shannigrahi, Steve DiBenedetto, Catherine Olschanowsky, Christos Papadopoulos, and Harvey Newman. Managing scientific data with Named Data Networking. In *Proceedings of the Fifth International Workshop on Network-Aware Data Management*, November 2015.

[3] Michael T. Goodrich and Michael Mitzenmacher. Invertible bloom lookup tables. In *In 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2011.

[4] Van Jacobson, Jeffrey Burke, Lixia Zhang, Beichuan Zhang, kc claffy, Dima Krioukov, Christos Papadopoulos, Tarek Abdelzaher, Lan Wang, Edmund Yeh, and Patrick Crowley. Named

Data Networking (NDN) Project 2010 - 2011 Report, 2011. `http://named-data.net/project/annual-progress-summaries/project/ndn-ar2011-html/`.

[5] Vince Lehman, A K M Mahmudul Hoque, Yingdi Yu, Lan Wang, Beichuan Zhang, and Lixia Zhang. A secure link state routing protocol for NDN. Technical Report NDN-0037, NDN Project, January 2016.

[6] C. Olschanowsky, S. Shannigrahi, and C. Papadopoulos. Supporting climate research using named data networking. In *Local Metropolitan Area Networks (LANMAN), 2014 IEEE 20th International Workshop on*, pages 1–6, May 2014.

[7] Fu Wenliang, Hila Ben Abraham, and Patrick Crowley. Synchronizing namespaces with invertible bloom filters. In *To appear in ANCS 2015*, 2015.

[8] Minsheng Zhang, Vince Lehman, and Lan Wang. PartialSync: Efficient Synchronization of a Partial Namespace in NDN. Technical Report NDN-0039, NDN, June 2016.

[9] Zhenkai Zhu and Alexander Afanasyev. Let's ChronoSync: Decentralized dataset state synchronization in Named Data Networking. In *IEEE ICNP*, 2013.

Chapter 5

Networking

Contributors	
PIs	Beichuan Zhang (Arizona), Van Jacobson & Lixia Zhang (UCLA), Lan Wang (Memphis), Christos Papadopoulos (Colorado State University), Patrick Crowley (Washington University)
Grad Students . .	Junxiao Shi, Teng Liang, Weiwei Liu, Klaus Schneider (Arizona); Spyridon Mastorakis, Ilya Moiseenko, Wentao Shang, Yingdi Yu (UCLA), Muktadir R. Chowdhury, Minsheng Zhang (U. Memphis), Steve DiBenedetto, Chengyu Fan (Colorado State), Haowei Yuan, Hila Ben Abraham (Washington University)
Undergrads	Ashlesh Gawande, Benjamin Murphy (U. Memphis), Eric Newberry (Arizona)
Staff	Vince S. Lehman (Memphis), John DeHart, Jyoti Parwatikar (Washington University) **Research Scientist:** Alex Afanasyev (UCLA)

In the past year, we continued our research to develop the underlying network layer of the NDN architecture, with an emphasis on how to best meet the needs of the target network environments. More specifically, we made significant progress in the areas of network protocols and software, routing protocols, forwarding strategy, and scalable forwarding.

5.1 NDN Forwarding Daemon (NFD)

Since May 2015, we made one major and four minor releases of NDN Forwarding Daemon (NFD), evolving it from v0.3.1 to v0.4.1 as of this report. Over the year, we have designed and implemented a network adaptation layer NDNLPv2, simplified configuration tasks in environments with one or more remote NDN forwarders acting as NDN gateways, improved management protocols, as well as ported NFD to the Android platform. NFD now runs on Linux, FreeBSD, Mac OSX, Android, DD-WRT/OpenWRT (home routers), Raspberry Pi, and a couple of other embedded platforms, as well as in virtualized environments. The development of NFD continues to use an open source and distributed model, involving the broader community. We use Redmine for issue tracking, Gerrit for code review, and Jenkins for automatic build and continuous integration. NFD has over 30 contributors from 10 different institutions, as well as several contributions outside the NSF-funded NDN team. To coordinate NFD development, we use the NFD developer mailing list with more than 100 members currently and conference calls twice weekly. We also continually update the NFD Developer's Guide and other related documents to provide implementation details and suggested development practices for new developers and researchers.

5.1.1 NDN Link Protocol v2

To accommodate NDN communication to different underlying links, it may be necessary to include additional information in the NDN interest and data packets. We designed and implemented a network adaptation protocol, NDN Link Protocol v2 (NDNLPv2). With NDNLPv2, routers can: (1) fragment and reassemble NDN packets that exceed a link's maximum transmission unit (MTU); and (2) send feedback upstream in form of network NACKs to signal failures of interest forwarding. Special NDN applications can use NDNLPv2 to: (1) explicitly indicate which face to forward an interest; (2) obtain information about the face that originally received a packet; and (3) request special cache behavior for the data packets. The latter may be useful when an NDN application wants to cache data and not use a router's cache. In the future, we plan to further extend the network adaptation features.

In implementing NDNLPv2, we decided to refactor NFD's Face system, specifically, we split the NDN Face abstraction into Transport and LinkService components. *Transport* provides delivery of the data blocks over specific underlying channels (raw ethernet packets, unicast/multicast UDP datagrams, TCP and WebSocket streams). *LinkService* provides a "network adaptation" layer to translate between NDN packets and data blocks communicated through the channels. The new Face abstraction serves as a container of Transport and LinkService instances and provides a high-level interface to send/receive NDN packets. This new structure makes it easy to combine different transports with different implementations of network adaptation.

5.1.2 NFD Management Protocol

Our original design of NFD included several management protocols, implemented using Interest/Data exchanges. To obtain information about the state of an NFD module (status datasets) or change its state (control commands), one sends an interest or signed interest for the data or for the confirmation of the command specified in the interest name. These include obtaining basic statistics of NFD, a list of Faces, RIB, and FIB entries, creation, removal, and state modification of Faces, manipulating forwarding information base (FIB) and routing information base (RIB) entries, as well as selection of per-namespace interest forwarding strategies. In addition, NFD management protocols include *notification streams*, a pub/sub-like protocol based on interest/data exchange to get notifications when specific NFD events happen. For example, when a face is created or destroyed, an NFD module (RIB manager) and other special applications (e.g., nfd-autoreg) will receive a corresponding notification. NFD Management uses the name-based schematized trust model for command interest authentication and authorization. In the future, we plan to include encryption of the status datasets using the name-based access control mechanism.

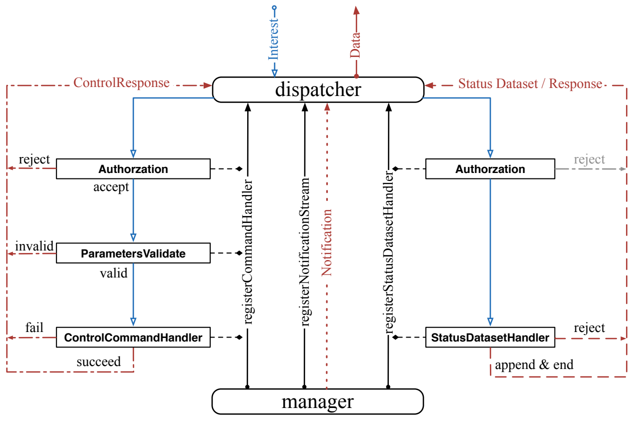


Figure 5.1: Overview of the manager Interest / Data exchange via the dispatcher

The development model for the NFD management protocol is generic and applies to other applications. For example, NLSR [6] and repo-ng[1] implement management the same way. To simplify implementation of new management protocols, we designed and implemented the management dispatcher abstraction. With the dispatcher, the applications can easily implement a new management protocol by registering necessary control commands, status datasets, and notification streams with the dispatcher. The dispatcher will classify and authenticate incoming interests in a centralized and consistent way, allowing applications to focus on command logic rather than implementation details.

[1] `https://github.com/named-data/repo-ng`

5.1.3 Automatic Prefix Propagation

In NDN, when a producer application wants to make its data available for retrieval, it registers the data's prefix with the NDN forwarder on the same host machine (i.e., *local registration*). Additional signaling is needed to make data available for fetching by remote NDN forwarders. This can be achieved in one of the following ways: *manual configuration*, *dynamic name announcement protocols* (e.g., NLSR), or *opportunistic data discovery strategies* (e.g., the access strategy, and auto-registration of the specified prefix(es) upon creation of on-demand faces). Different methods have different tradeoffs in ease of use, implementation complexity, and communication overhead. We have designed and developed an *Automatic Prefix Propagation* (APP) protocol to address the above issues.[2]

As shown in Figure 5.2, APP enables the local NDN forwarder in Host d to automatically propagate local prefix registrations to the remote forwarder on Gatway Router B. Local prefix registration triggers such propagation when (1) an active remote NDN gateway exists, as indicated by the "`/localhop/nfd`" registered prefix in the local RIB, and (2) the local forwarder possesses a private key and the corresponding NDN certificate that matches or covers the locally registered namespace. If there are two or more candidates for the keys/namespace, the key that corresponds to the shortest namespace is selected. This allows the forwarder to aggregate multiple local registrations into one remote action, reducing communication and bookkeeping overheads.

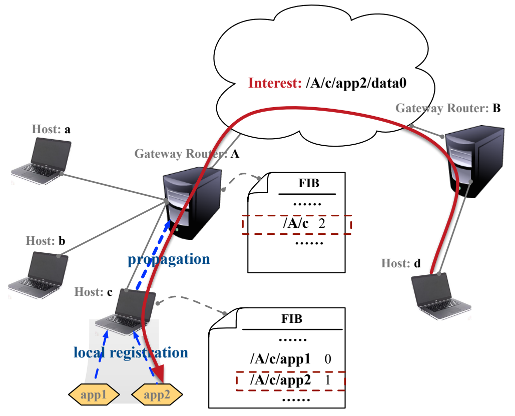


Figure 5.2: Make a producer application's data available for fetching from remote NDN forwarders.

After the initial prefix propagation, NFD periodically refreshes the registration, unless the prefix is no longer registered locally or there are no NDN gateways configured. The protocol maintains a finite state automaton for each propagated entry, that not only schedules propagation refreshment and handles failures with configured strategies, but also dynamically deals with connectivity changes and other state transitions.

5.1.4 Supporting Mobility and FIB Scaling with LINK

In previous work we discussed a possibility of using a LINK object to control the number of entries in the forwarding information base [1]. We can use the same mechanism to support mobile publishing, when it is necessary to track down the moving producer and not just its data [14]. In this approach, data names map to a set of globally routable names, which interests can include as "LINKS" to inform (hint) the forwarding system of the whereabouts of the requested data.

Over the last year, we added support for LINK objects in all our libraries and in NFD. This support included several components: (1) definition and implementation of the LINK abstraction as a data packet containing a list of namespace-to-namespace mapping (delegations), (2) extension of the NDN interest packet format to allow inclusion of a LINK object, and (3) extension of the interest processing logic in NFD forwarding pipelines. As part of the changes in the forwarding pipelines, we have introduced a *network region table* that contains a list of prefixes considered local to the producer. Whenever an interest with a LINK object reaches the router, if one of the delegations specified in the LINK is a prefix for some entry in the network regions table, the router processes the data name in the interest, otherwise, NFD uses LINK delegations to decide how to forward the interest.

Our current implementation of the LINK support has several limitations that we plan to address in the future. First, the network region table requires manual configuration of NFD instances running on

[2]The currently implementation of the protocol assumes a single NDN gateway; future versions will support multi-gateway environments.

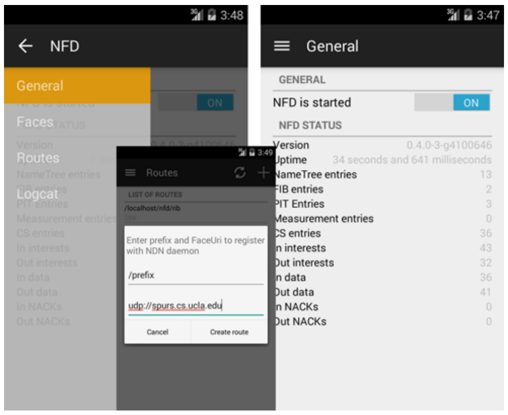
routers and end hosts, which we plan to address by extending the auto-configuration protocols. Second, our implementation does not currently partition the content store for data with the same prefixes, retrieved using different links. As we showed in a brief analysis [1], such partitioning would be necessary in real systems to prevent denial-of-service attacks through targeted cache poisoning. In addition to that, we have not yet fully utilized the implemented support in the developed applications, which could expose limitations of the designed logic.

5.1.5 Permanent Faces

Initial deployment of our applications in the BMS environment highlighted a limitation of the implemented Face system in NFD: the tunneled faces only exist while there is underlying network connectivity, and must be re-created after connectivity changes. To handle this situation, we added support for permanent faces (currently, only UDP-based faces) that persist throughout the lifetime of the NFD instance, along with any associated RIB and FIB entries. We also plan to support Face and RIB/FIB entry permanency across NFD restarts. All native faces (e.g., Ethernet faces associated with physical network interfaces) are permanent by design. As long as an interface exists, there is a corresponding face. The limitation applies specifically to *tunneled* faces, i.e., where it is infeasible or prohibitively expensive to enable native NDN support in the transit networks, such as a campus WiFi network.

5.1.6 NFD on Android

To allow broader experimentation with NDN technology, and in particular experimentation in real mobile environments, we made a prototype port of our reference forwarding daemon implementation on Android platform (NFD-Android). The port is fully based on and evolves with the original NFD source code and in addition includes several user interfaces to manually start/stop NFD service and get basic operational statistics, create faces, and register prefixes (Figure 5.3). The initial port can run on any Android device, including any unrooted devices, and can be either manually compiled from the publicly available source code or be directly installed from the Google Play store.



/localhost/nfd/rib

Enter prefix and FaceUri to register with NDN daemon

/prefix

udp://spurs.cs.ucla.edu

NFD is started ON

```
Version                          0.4.0-3-g4100646
ptime       34 seconds and 641 milliseconds
ameTree entries                                13
IB entries                                      2
IT Entries                                      3
easurement entries                              0
S entries                                      36
n interests                                    43
ut interests                                   32
n data                                         36
ut data                                        41
n NACKs                                         0
ut NACKs                                        0
```

Figure 5.3: User interfaces to NFD-Android

The initial NFD-Android port has a number of usability limitations: it lacks auto-configuration capability, including after power cycling a device; security limitations prevent non-rooted Android platforms from communicating over raw Ethernet sockets; and users must manually establish a direct WiFi session, discover peer IP addresses, and then configure proper faces and routes toward the peer-to-peer faces.

Nevertheless, the NDN team with the help of students from UCLA's CS217B class in Spring quarter of 2015 developed several pilot applications that demonstrate NDN advantages. These applications include:

- NDN Whiteboard (`https://github.com/named-data-mobile/apps-NDN-Whiteboard`): a real-time shared whiteboard that works over Named Data Networking (NDN).
- PhotoSharing app (`https://github.com/ohnonoho/photoSharing`): a simple photo sharing application that leverages sync and local sharing of data (the app manages Direct WiFi sessions)
- ndn-hangman (`https://github.com/dchandc/ndn-hangman`): a simple Hangman game that uses Chronosync to synchronize state of the game in delay-tolerant fashion.

These pilot application design and development efforts allowed the students to deepen their understanding of NDN through first hand experience, to appreciate NDN's advantages in easing distributed application

Pre-signed Data				Dynamically Generated Data			
Time Between Interests (ms)	Avg NFD CPU %	Avg File timeout %	Throughput (Mb/s)	Time Between Interests (ms)	Avg NFD CPU %	Avg File timeout %	Throughput (Mb/s)
11	3.98%	0.29%	5.5	11	3.56%	0.33%	5.48
8	3.34%	0.31%	7.5	8	3.65%	25.9%	5.39
5	5.81%	0.34%	11.66	5	4.19%	94.7%	0.59

Table 5.1: 2.5 GHz Phone. Left: Serving pre-signed Data. Right: Serving dynamically generated Data

Pre-signed Data				Dynamically Generated Data			
Time Between Interests (ms)	Avg NFD CPU %	Avg File timeout %	Throughput (Mb/s)	Time Between Interests (ms)	Avg NFD CPU %	Avg File timeout %	Throughput (Mb/s)
11	2.31%	0.34%	5.53	37	2.24%	0.32%	1.67
8	2.85%	6.4%	6.94	34	2.23%	0.34%	1.81
5	3.11%	48.97%	5.74	31	1.97%	90.79%	0.18

Table 5.2: 1.2 GHz phone. Left: Serving pre-signed Data. Right: Serving dynamically generated Data

development, as well as allowed us to identify the following common issues that require emphasis with NDN newcomers:

- How best to choose application data names, establish naming conventions, and make best use of metadata, to simplify the overall design;
- How to use versioning to handle data changes (this reflects an inadequate understanding of immutable data concept), and
- How best to work with in-network caches.

These efforts also identified a number of issues with using the NDN prototype code base beyond the Android package; the most important ones are the lack of documentation and lack of sample code to ease the steep learning curve. We have proposed to address these issues in the supplement period.

We tested the performance of NDN (including NFD, applications and libraries) on two Android devices with different computational abilities. Our testing was centered around using an Android device as a producer. For each test, the Android device would serve Data in one of two ways when it received an Interest: either by loading pre-signed Data saved in the device's storage, or by generating and signing Data on-demand for each Interest. While the test was running, we ran a statistics collection application that gathered information about the producer application's CPU usage and NFD's CPU usage. Each test consisted of a consumer laptop expressing Interest for 100 MB worth of Data which was served from the Android device. These tests were conducted with two phones: a Samsung phone with a 2.5 GHz quad-core CPU and a Motorola phone with a 1.2 GHz quad-core CPU.

The results for the 2.5 GHz phone are shown in Table 5.1, and the results for the 1.2 GHz phone are shown in Table 5.2. Our test results showed that both lesser and more powerful devices are able to serve pre-signed Data with similar throughput when the request rate is lower. When Interests are sent every 11ms, both the 1.2 GHz and 2.5 GHz phones serve pre-signed Data with around 5.5 Mb/s throughput. More powerful devices also dynamically generate and sign Data at a rate not too far behind serving pre-signed Data; the 2.5 GHz phone serves both types of Data with around 5.5 Mb/s throughput when Interests are sent at an 11ms interval. The 1.2 GHz phone requires much more computational time to dynamically generate and serve Data than the 2.5 GHz phone, thus the Interest rate needed to be lowered in order to get results for the 1.2 GHz phone. Lowering the Interest rate on the 1.2 GHz phone with dynamically served Data caused lower throughput than serving pre-signed Data (1.67 Mb/s vs. 5.53 Mb/s). We also saw that requesting Data at too frequent an interval resulted in more timeouts at each increasing Interest rate; dynamically served Data showed substantial timeouts on both the 1.2 GHz phone and the 2.5 GHz phone when Interests were sent at a 5ms interval.

On the 2.5 GHz phone, NFD's CPU usage increased slightly as the Interest rate increased in both Data serving schemes. On the 1.2 GHz phone, serving pre-signed Data also caused only a slight increase to NFD's CPU usage. But, when the 1.2 GHz phone served dynamically generated Data, NFD's CPU usage decreased due to the applications inability to serve Data quickly. In that case, NFD actually had to forward less Data back to the consumer application which lowered NFD's CPU usage.

5.2 Routing Protocols

Our work on NDN routing protocols continued in two parallel directions: conventional link-state routing (Named-data Link State Routing, NLSR [6]) and update-less greedy routing (Hyperbolic Routing, HR [5]).

5.2.1 NLSR

NLSR is a name-based routing protocol that supports multi-path forwarding and uses a hierarchical trust model to secure routing information. The global NDN testbed has operated NLSR since August 2014. In the past year, we worked on refactoring the NLSR code to improve readability and modularity, as well as adding unit tests to provide more thorough test coverage for our code. In addition, we worked with users of NLSR to track down and correct bugs. Through growing deployment of NLSR on the NDN testbed, currently 32 nodes and 87 links, we have learned of issues in NLSR's operation that we would not have otherwise caught, and resolved them successfully. We also continued to adapt NLSR to operate with the newest versions of both ndn-cxx and NFD, and released version 0.2.1 on June 30, 2015. Finally, we updated the NLSR paper to reflect our design and implementation changes [6]. We have also used NLSR to conduct our comparison of link-state and hyperbolic routing algorithms.

5.2.2 Hyperbolic Routing with Adaptive Forwarding Strategy

Hyperbolic routing (HR) presents a potential solution to address the routing scalability problem in NDN. HR does not use traditional routing tables or exchange routing updates upon changes in the network topology, but uses pre-assigned hyperbolic coordinates to direct interests toward data. However, it introduces potential drawbacks of sub-optimal routes for some destinations. To overcome these drawbacks, we designed and implemented a new forwarding strategy called Adaptive Smoothed RTT-based Forwarding (ASF) and have been evaluating the performance of hyperbolic routing with ASF [5]. Our experiments compare the packet loss ratio, RTT, message overhead, and failure response time of data retrieval under link-state routing and hyperbolic routing with various forwarding strategies, failure conditions, and topologies.

Design of Adaptive Smoothed RTT-based Forwarding (ASF)

To allow for variations in each next hop's RTT measurement, ASF maintains a Smoothed-RTT (SRTT) for each next hop. ASF groups and sorts the next hops based on their SRTT performance as well as their ability to respond with Data. ASF prioritizes next hops that have previously returned Data and that have lower SRTT measurements. When ASF receives an Interest to forward, it first tries to select the next hop that is returning Data and has the lowest SRTT measurement. In this way, ASF is able to choose a path that results in the lowest delay and that results in Data being returned. If no next hops fit this criteria, ASF will attempt to forward the Interest to any lowest routing cost next hop that has not yet been used. Finally, if ASF still has not picked a next hop, it will select the lowest routing cost next hop that was previously not returning Data. This logic allows ASF to prioritize both delay and successful Data retrieval when making forwarding decisions.

Since hyperbolic routing does not respond to short term changes in the network topology, new or recovered links will not be detected if ASF continues to use an existing working path that does not include the link. By periodically probing less used or unused next hops, ASF can learn about the performance of these new

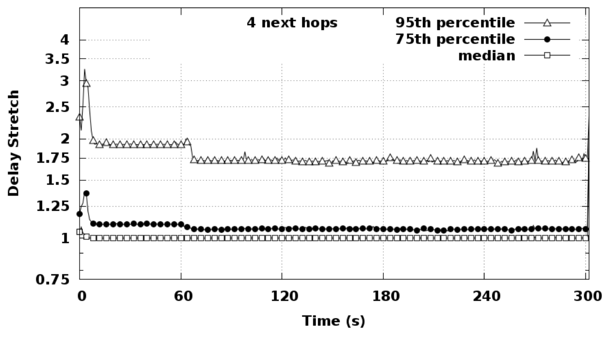


Figure 5.4: Delay stretch of HR with ASF over link-state routing (60-second Probing Interval)

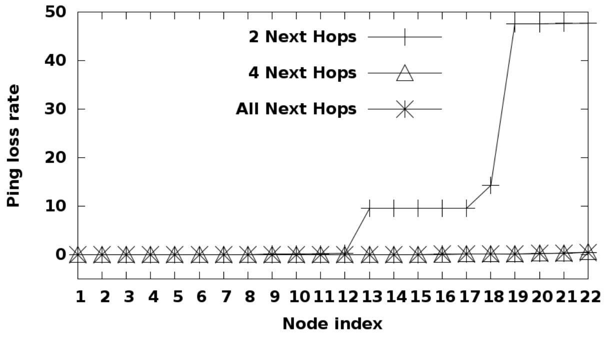


Figure 5.5: Loss rate of HR with ASF (60-second Probing Interval)

or recovering links. When ASF is due for a periodic probe and receives an Interest to forward, it will also attempt to select a different next hop than the primary forwarding next hop to probe. ASF will first try to probe the lowest routing cost next hop that has not yet been used, which allows for the strategy to quickly learn the performance of new next hops. If all next hops have measurements from their use, ASF will sort the next hops by their SRTT value if they are returning data and by their routing cost if they are not; next hops that return Data take precedence over next hops that do not. ASF then assigns a probability to each next hop where next hops that are higher priority receive a higher probability and lower priority next hops receive a lower probability. ASF will then probabilistically select one of the next hops to forward the probe Interest. Assigning probabilities in this manner allows ASF to more often probe next hops that have previously performed well.

Preliminary Results

Below we present some preliminary results on the forwarding performance and cost of hyperbolic routing with the support of ASF (HR/ASF) in a small topology. We measured the delay stretch and loss rate of HR/ASF compared to a shortest path routing protocol, in this case link-state. We also compared the overhead of HR/ASF (i.e., probes in ASF) with that of link-state routing (routing update messages). We used a snapshot of the real NDN testbed with 22 nodes as the evaluation topology. The routing cost of each link in the topology is set to the delay between the two neighboring nodes, and the hyperbolic coordinates of the nodes are set equal to the coordinates of the AS to which these nodes belong [7]. Testbed nodes that belong to the same AS have small disturbances added to make the coordinates unique. Each node advertises one name prefix and produces ping data under that name prefix, and every node sends ping Interests to every other name prefix once a second.

We ran evaluations with a probing interval of 30 and 60 seconds for ASF as well as different multi-path factors (an upper limit on the number of next hops per name prefix) on the NDN testbed topology. Figure 5.4 shows the delay stretch of hyperbolic routing with a 60 second probing interval and a multi-path factor of 4. The median stretch is very close to 1 and the 75th percentile stretch is slightly higher than 1. This shows that the packet delays under HR are quite close to those under link-state routing except in a small fraction of the cases. Figure 5.5 shows the loss rate for each node under hyperbolic routing (there are no losses in link-state routing). We can observe that the more next hops available for the strategy, the lower the loss rates, and a multi-path factor of 4 has nearly no loss for all the nodes. In terms of overhead, HR with a 60-second probing interval and multi-path factor of 4 has a lower overhead (0.98 pps per node) than that of link state (1.05 pps per node). Our experimental results on topologies of up to 99 nodes show that HR's delay stretch has a median close to 1 and a 95th-percentile around or below 2. Our results also show that the ASF probing overhead in dynamic topologies is much lower than the control overhead in link-state routing, because HR does not incur any traditional routing protocol overhead other than probing.

While in our preliminary evaluations HR/ASF performed surprisingly well, there are many factors and scenarios, e.g., larger topology size, mobility, and routing policies, that we have not yet considered, so it is unclear whether the performance and overhead will scale in realistic Internet-scale settings. We also did not optimize ASF's various parameters, such as probing period and probability, dynamically based on observed performance, or compare it with other previously proposed strategies [3, 8, 9, 11] to identify areas of improvement. These research issues will be addressed in future work.

5.3 Scalable Forwarding

In the second year of the project, the Washington University team led efforts and made substantive contributions in four primary areas: scalable forwarding, forwarding strategy, synchronization (discussed in Section 4.1), and testbed development (discussed in Section 6.1).

In the area of scalable forwarding [10], we proposed, implemented, and evaluated a scalable FIB longest name prefix lookup design based on the binary search of hash tables [13]. As reported in last year's report, we implemented the proposed design in software, using Intel DPDK [4] packet processing framework, and demonstrated 10 Gbps forwarding throughput with one billion synthetic longest name prefix matching rules, each containing up to seven name components. To the best of our knowledge, this is the largest dataset that has been studied for longest name prefix lookup. As for in-network caching elements, we explored Content Store design issues and evaluated the performance of a modified NDN repository based on the Redis key-value store [12]. Our experimental work has shown that existing storage systems and databases, such as Redis, can be employed in NDN to implement terabyte-scale repositories.

NDN introduces a new forwarding model, in which the forwarding plane can choose between multiple interfaces when forwarding a packet. This forwarding strategy mechanism brings new opportunities but it also introduces challenges when the application's performance or correctness is affected by a conflict between the application design and the assigned forwarding strategy; in practice, the forwarding strategy is determined by network operators but at the same time application developers make implicit assumptions about how packets are forwarded. Our early work [2] demonstrated the impact of the forwarding strategy decision on the performance and correctness of existing NDN applications, and strongly suggests that work remains in defining forwarding strategy as a reliable architectural component of NDN.

5.4 NDN/OSCARS Integration

OSCARS is a bandwidth reservation service available on certain ESnet nodes that enables layer-2, user driven, end-to-end reservations to isolate large data transfers from other traffic. We have integrated NDN with OSCARS through a protocol that allows users to specify how fast data is required. If reservation is required, the NDN strategy layer communicates through OSCARS to create a reservation on certain reservation-capable NDN faces. On successful reservation, NDN strategy uses the newly created high-speed path for data transfer.

5.5 Plans for the Next Year

In the next year, we will develop features to fill some gaps in our architecture, motivated by the requirements of the network environments, as well as pushing forward the experimentation and documentation.

- *Autoconfiguration.* Painless autoconfiguration is essential for usability, especially for mobile and intermittently connected environments. Autoconfiguration involves: (1) establishing NDN connectivity; (2) obtaining appropriate keys; and (3) producers registering name prefixes to attract interests for their data. In local environments where NDN can run directly over layer-2, nodes can locally broadcast Interests to discover neighbor gateways and available data, and figure out where to forward future Interests by observing from which direction data packets return. When the NDN stack runs as an overlay on IP, it can perform local NDN gateway discovery in IP multicast-enabled domains or using DNS and DHCP-derived local domains, or use home agent gateways based on pre-configured settings. Next year's effort will build on our preliminary work and develop a complete autoconfiguration functionality.

- *Congestion Control.* To support high-throughput applications (e.g. scientific big data transfer) and low-latency applications (NDN-RTC), we need congestion control in the reference implementation and deployed on the testbed. We plan to (1) design mechanisms for congestion detection, back-pressure signaling, multi-path strategies, and consumer rate adjustment; (2) simulate and evaluate these mechanisms; (3) implement them in NFD, and deploy them on the NDN testbed, and (4) develop library APIs to provide rate control functionality to applications, and integrate them into both applications.
- *Hyperbolic Routing (HR).* Now that the hyperbolic routing code is stable, we plan to do real testbed experiments to discover potential issues that may not surface in the emulated environment, and demonstrate its feasibility together with applications.
- *Documentation* Our most successful documentation effort has been the NFD Developer's Guide, which explains how to interpret, use, and modify the NFD code base. In the next year we will follow this model to expand documentation on other important components, particularly NLSR, automatic prefix registration, and NFD management protocol.

References

[1] Alexander Afanasyev, Cheng Yi, Lan Wang, Beichuan Zhang, and Lixia Zhang. SNAMP: Secure namespace mapping to scale NDN forwarding. In *18th IEEE Global Internet Symposium (GI 2015)*, April 2015.

[2] Hila Ben Abraham and Patrick Crowley. Forwarding strategies for applications in named data networking. In *Proceedings of the ACM IEEE Symposium on Architectures for Networking and Communications Systems*, 2016.

[3] Raffaele Chiocchetti, Diego Perino, Giovanna Carofiglio, Dario Rossi, and Giuseppe Rossini. Inform: A dynamic interest forwarding mechanism for information centric networking. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ICN '13, pages 9–14, 2013.

[4] Intel Corp. DPDK: Data Plane Development Kit, 2016. `http://www.dpdk.org`.

[5] Vince Lehman, Ashlesh Gawande, Rodrigo Aldecoa, Dmitri Krioukov, Lan Wang, Beichuan Zhang, and Lixia Zhang. An Experimental Investigation of Hyperbolic Routing with a Smart Forwarding Plane in NDN. In *Proceedings of the IEEE IWQoS Symposium*, June 2016.

[6] Vince Lehman, A K M Mahmudul Hoque, Yingdi Yu, Lan Wang, Beichuan Zhang, and Lixia Zhang. A secure link state routing protocol for NDN. Technical Report NDN-0037, NDN Project, January 2016.

[7] F. Papadopoulos, D. Krioukov, M. Boguñá, and A. Vahdat. Greedy Forwarding in Dynamic Scale-Free Networks Embedded in Hyperbolic Metric Spaces. In *IEEE Conference on Computer Communications (INFOCOM)*, San Diego, CA, Mar 2010. IEEE.

[8] Haiyang Qian, R. Ravindran, Guo-Qiang Wang, and D. Medhi. Probability-based adaptive forwarding strategy in named data networking. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 1094–1101, May 2013.

[9] Klaus M Schneider and Udo R Krieger. Beyond network selection: Exploiting access network heterogeneity with named data networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 137–146. ACM, 2015.

[10] Tian Song, Haowei Yuan, Patrick Crowley, and Beichuan Zhang. Scalable name-based packet forwarding: From millions to billions. In *Proceedings of the 2nd ACM Conference on Information-Centric Networking*, September 2015.

[11] Cheng Yi, Jerald Abraham, Alexander Afanasyev, Lan Wang, Beichuan Zhang, and Lixia Zhang. On the role of routing in named data networking. In *ACM SIGCOMM ICN Conference*, 2014.

[12] Haowei Yuan. Scalable NDN Forwarding, 2015. Doctoral Thesis. Washington University in St. Louis, Department of Computer Science & Engineering.

[13] Haowei Yuan and Patrick Crowley. Reliably scalable name prefix lookup. In *Proceedings of the ACM IEEE Symposium on Architectures for Networking and Communications Systems*, May 2015.

[14] Yu Zhang, Alexander Afanasyev, Jeff Burke, and Lixia Zhang. A Survey of Mobility Support in Named Data Networking. *Proceedings of the third Workshop on Name-Oriented Mobility: Architecture, Algorithms and Applications (NOM'2016)*, April 2016.

Chapter 6

Evaluation

Contributors	
PIs	Beichuan Zhang (Arizona), Van Jacobson & Lixia Zhang (UCLA), Lan Wang (Memphis), Christos Papadopoulos (Colorado State University), Patrick Crowley (Washington University)
Grad Students . .	Junxiao Shi, Jerald Abraham, Yi Huang (Arizona); Yingdi Yu, Wentao Shang, Spyridon Mastorakis (UCLA), Steve DiBenedetto, Chengyu Fan (Colorado State), Haowei Yuan, Hila Ben Abraham (Washington University)
Undergrads	Ashlesh Gawande, Vince S. Lehman (5/2014 – 11/2014) (Memphis)
Staff	John DeHart, Jyoti Parwatikar (Washington University), Vince Lehman (Memphis)
	Researcher: Alex Afanasyev (UCLA)

6.1 NDN Testbed: Deployment, Management, Expansion

The NDN team at Washington University operates and manages the global NDN testbed, which as of March 2016, has 31 nodes in 14 countries. The Washington University team also manages integration testing and deployment activities. They have enhanced the monitoring tools for the NDN Testbed [1], and updated the NDN real time testbed usage mapping tool to use a pure NDN paradigm. A central server on the Washington University campus sends NDN Interests to client daemons that run on each of the testbed nodes. These client daemons collect usage data and return it as content in data packets to the server. A web page at http://ndnmap.arl.wustl.edu displays the usage data.

6.1.1 Lessons Learned from Testbed Use

During NLSR's development we found use of the NDN testbed valuable for exposing and debugging issues in implementation. In a controlled test environment, certain problems and bugs occur less frequently than in a real testbed deployment. For example, we encountered an issue in NLSR when a router, configured as a neighbor for other routers, was not enabled for a long period of time. This bug prevented other routers' name prefixes from having their expiration time extended. We also observed a situation where a router tried to fetch its own outdated Link State Advertisements (LSA) due to updates it received from elsewhere in the network. This put the router in an infinite loop where it continuously tried and failed to retrieve its own outdated LSA. Further, we discovered a sequence number problem that we detected in the testbed network due to an incorrect bit mask used to copy the name LSA sequence number. For each of these problems, we used the logs collected from the testbed to track down the cause of the issue. We had not considered test cases for these scenarios and due to real world use in the testbed, we were able to discover and diagnose these serious bugs in NLSR.

6.2 Mini-NDN

In the past year, we transitioned our development of the Mini-NDN emulator from a prototype to publicly released software. We added a user-friendly experimentation framework for users to define and run NDN experiments using simple syntax; the framework includes basic experiment examples for reference. We want Mini-NDN to be as simple to use for as many users as possible, so we wrote and released documentation for the major components of Mini-NDN and created a mailing list for user support. We released version 0.1.0 of Mini-NDN on July 15, 2015 and the current version 0.1.1 on November 4, 2015 (`https://github.com/named-data/mini-ndn/`).

At the 2nd NDN Hackathon, we worked on a project to simplify statistic collection and data analysis for Mini-NDN's users. The Hackathon project moved the status information for each node in the network from plain console output to a web interface. A user's browser connects to the Mini-NDN network to gather information. The interface includes reporting the status of NFD and NLSR, real-time metrics for each link's bandwidth consumption, and a dynamic graph that plots measurements recorded by NFD (e.g., number of Interests, number of PIT entries, etc.). The project won the Best Internal Impact prize for its potential in improving the analysis of various NDN applications for the NDN team.

References

[1] Zeév Lailari, Hila Ben Abraham, Ben Aronberg, Jackie Hudepohl, Haowei Yuan, John D. DeHart, Jyoti Parwatikar, and Patrick Crowley. Experiments with the Emulated NDN Testbed in ONL. In *Proceedings of the 2nd ACM Conference on Information-Centric Networking*. ACM, 2015.

Chapter 7

Impact: Education

Contributors
PIs . . . . Christos Papadopoulos (CSU), Lan Wang (Memphis), Beichuan Zhang (Arizona), Van Jacobson, Jeff Burke, Lixia Zhang (UCLA)

7.1 Education Efforts

The NDN group continues to produce a significant amount of educational material, which can be found at the following URL: `http://www.named-data.net/education.html`.

7.1.1 UCLA

Lixia Zhang taught a graduate course "CS217A: Internet Architecture & Protocols" during Winter 2016 quarter, where she devoted three lectures on an introduction to NDN and made compared NDN with today's TCP/IP protocol architecture. About half of the students in the class signed up for optional term projects on NDN research topics, which they carried over to CS217B in Spring 2016 quarter to finish up.

During the Spring 2015 and Spring 20116 quarters, Lixia Zhang taught CS217B on "Advanced Topics in Internet Research", a graduate seminar course focused on the NDN architecture design and application development. In addition to covering NDN design literature, the course traces historical architectural ideas from a series of research papers spanning the last few decades (e.g. [1, 4, 2]). Students also learned about other architectural designs under NSF's FIA program, in particular eXpressive Internet Architecture [5] and Mobility First [6]. In parallel to in-class discussions, the students carried out research projects on NDN design and development. The students finished the following projects during Spring 2015 quarter. The first three projects utilized named data communication to enable *serverless* distributed applications.

- Shared WhiteBoard Over Named Data Network (NDN) on Android,
- Photo Sharing NDN Applications on Android, and
- Porting a turn-based Android game to NDN (Android).
- Wireshark Dissector for NDN Packet Format, now used by the NDN network management toolkit.
- Android Key Manager, now part of the NDNfit implementation (Section 2.2).
- nTorrent: BitTorrent over NDN – now evolved into a complete design (please see nTorrent project under Spring 2016).
- NDN Security Certificate Logger Design to support long-lived data validation, now incorporated into the NDN signature logger implementation, reported in a paper submitted to ICN 2016.
- NDN Security Schema Design And Implementation, now incorporated into the NDN security library.

Three of the above projects became student master projects that fulfilled the M.S. degree requirements.

Research projects carried out during Spring 2016 quarter include the following:

- Mitigating content poisoning attacks in NDN, extending results in [3].
- Evaluation of the design and performance of ChronoSync 2.0 using ndnSIM (see also Section 4).
- Implementing NDN Sync Protocol on RIOT OS for Internet of Things.
- Local NDN hub discovery for automatic NDN network attachment.
- Secure poker games on an Named Data Network, which used NDN's security primitives to address a difficult application scenario.
- Implement nTorrent: A Peer-to-Peer file download system over NDN. We expect to experiment with this new application over the NDN testbed in June 2016.
- NDN Open mHealth pilot applications, as part of the NDNfit (Section 2.2) development efforts.

7.1.2 University of Memphis

Lan Wang supervised Alejandro Gil Torres, an exchange student from Spain who did his undergraduate thesis on NLSR. The student implemented additional statistics collection to the NLSR code and used these statistics to evaluate the performance impact of several NLSR default timer value settings.

Lan also gave a seminar talk at University of Mississippi on NDN, titled "Architectural Development and Routing Design in Named Data Networking" (http://www.cs.olemiss.edu/node/435).

7.1.3 Colorado State University

CSU continued to teach NDN and assign NDN-related programming assignments in the graduate networking course. We did not include NDN in the undergraduate networking course in the Fall 2015 because PI Papadopoulos was on sabbatical. PI Papadopoulos included a strong NDN component in his graduate class in Spring 2016, which used NFD and the ndn-cxx C++ library. The class included an implementation of a content distribution network using IP and then using NDN. There were strong comments from the students about how much easier it was to implement the project with NDN. Graduate students working on NDN taught part of the class. The class used project templates developed at CSU and published on the NDN web site.

Papadopoulos gave a webinar on NDN at ENCITE (Enhancing Cyberinfrastructure by Training and Engagement) titled "An Overview of NDN" on 11 March 2016. The recording can be found at `https://events-na12.adobeconnect.com/content/connect/c1/1282664749/en/events/event/shared/default_template/event_registration.html?sco-id=1484223888&_charset_=utf-8`.

7.1.4 Washington University in St. Louis

John Dehart taught Introduction to Networking in Fall 2015 and had Hila Ben Abraham, a graduate student, give a guest lecture on NDN during the course.

7.2 NDN Tutorial at ICN 2015

During ACM's Information-Centric Networking 2015 conference, we held a full day tutorial on synchronization and security in Named Data Networking (NDN). This tutorial shared important architectural concepts that we are exploring in these areas, the software we have built to perform these tasks, and remaining open issues. In particular, it emphasized how the existing open source toolset provides a platform for exploring the open research questions. The tutorial introduced conference participants to emerging features of the available toolsets. In addition to referencing a variety of existing examples, the tutorial used the creation of a modern browser-based application to illustrate writing NDN applications, leveraging 1) multi-party synchronization, 2) schematized trust, 3) encryption-based access control.

The main objective of the tutorial was to introduce the practical role of sync as a communication protocol, available tools, and envisioned use cases. Specifically, we wanted students to understand how to move from a general sync concept to specific sync designs, and the role that sync plays in the sample application. We gave an introduction and brief comparison of the application design patterns based on the current prototypes of applications, including ChronoShare, NLSR, NDNFit.

7.3 NDN Weekly Seminars

During this reporting period we continued our biweekly NDN seminar series among participating universities. NDN seminars cover a wide range of topics that reflect ongoing NDN design and development efforts and promote inter-campus exchanges and collaborations. Most speakers are graduate students from different universities, presenting their work to a broad audience to exchange ideas, solicit feedback, and explore cross-campus collaboration opportunities. In 2014 Shiguang Wang, a UIUC graduate student, and A K M Mahmudul Hoque, a Memphis graduate student, collected topics and created the schedule. In 2015 Jongdeog Lee, a UIUC graduate student started coordinating the seminars, followed by Spyros Mastorakis at UCLA.

This year's seminars took place from 5/15 - 4/16 and covered the following topics:

- May 27: Prof. Pedro de las Heras Quires (UCLA Visiting Scholar), ”Application of Macaroons for distributing encryption keys and providing access control in NDN applications”
- July 15th Ashlesh Gawande (Univ. of Memphis), ”Mini-NDN: A light weight emulation tool for Named Data Networking”
- August 12th Susmit Shannigrahi and Chengyu Fan (Colorado State Univ.), ”NDN for Scientific Data Applications”
- August 26th Eva M. Castro and Pedro de las Heras Quiros (Universidad Rey Juan Carlos, Spain), ”Redesign of ChronoSync”
- October 7th Ryan Bennett (Colorado State University), ”NDN-IO: Rapid Prototyping for Node.js and Browse”
- October 28th Jongdeog Lee (UIUC), ”InfoMax-An Information Maximizing Transport Layer Protocol for Named Data Networks”
- December 2nd Rodrigo Aldecoa (Northeastern University), ”Hyperbolic Routing (Theory)”
- December 16th Vince Lehman (University of Memphis), ”Hyperbolic Routing (Implementation)”
- 10 February 2016 Yingdi Yu (UCLA), ”Schematizing Trust in Named Data Networking”
- 24 February 2016 Peter Gusev (UCLA REMAP), ”Real-Time Videoconferencing over NDN”
- 2 March 2016 Susmit Shannigrahi (Colorado State University), ”Scientific Data Applications in NDN”
- 16 March 2016 Ben Murphy (University of Memphis), ”Performance measurements of Android devices in NDN”

References

[1] David D Clark and David L Tennenhouse. Architectural considerations for a new generation of protocols. *ACM SIGCOMM Computer Communication Review*, 20(4):200–208, 1990.

[2] David D. Clark, John Wroclawski, Karen R. Sollins, and Robert Braden. Tussle in cyberspace: defining tomorrow's internet. In *ACM SIGCOMM*, 2002.

[3] Stephanie DiBenedetto and Christos Papadopoulos. Mitigating Poisoned Content with Forwarding Strategy. *Proceedings of the third Workshop on Name-Oriented Mobility: Architecture, Algorithms and Applications (NOM'2016)*, April 2016.

[4] Sally Floyd, Van Jacobson, Steven McCanne, Lixia Zhang, and Ching-Gung Liu. A reliable multicast framework for light-weight sessions and application level framing. In *SIGCOMM*, 1995.

[5] David Naylor, Matthew K. Mukerjee, Patrick Agyapong, Robert Grandl, Ruogu Kang, and Michel Machado. XIA: Architecting a More Trustworthy and Evolvable Internet. *ACM SIGCOMM Computer Communication Review (CCR)*, 44(3):50–57, Jul 2014.

[6] Arun Venkataramani, James Kurose, Dipankar Raychaudhuri, Kiran Nagaraja, Suman Banerjee, and Morley Mao. MobilityFirst: A Mobility-Centric and Trustworthy Internet Architecture. *ACM SIGCOMM Computer Communication Review (CCR)*, 44(3):74–80, Jul 2014.

Chapter 8

Impact: Expansion of NDN Community

The NDN project is attracting ever-increasing attention from the global networking community. The PIs participated in conferences and speaking engagements, as listed below, and engaged with a variety of interns and visiting researchers from universities and industry. These formal and informal efforts have helped to disseminate research results and project ideas, as well as practical information about the NDN code base.

8.1 The Second NDN Community meeting

The second NDN Community Meeting was held at UCLA in Los Angeles, California on September 28–29, 2015 [2]. The meeting provided a platform for the attendees from 63 institutions across 13 countries to exchange their recent NDN research and development results, to debate existing and proposed functionality in NDN forwarding, routing, and security, and to provide feedback to the NDN architecture design evolution.

8.2 Expansion of the NDN Consortium and Testbed

The NDN consortium established in 2014 continued its expansion, adding two industrial and five academic members:

1. ViaSat
2. Juniper Networks
3. Federal University of Par, Brazil
4. National Institute of Technology Karnataka, Surathkal
5. Northeastern University
6. TNO, the Netherlands Organization for Applied Scientific Research
7. University of Maryland, College Park

Over the last year we added new nodes to the NDN testbed [1] growing it to 31 Nodes with 84 links:

1. the Norwegian University of Science and Technology (NTNU)
2. the Korea Institute of Science and Technology Information (June 15)

3. COPELABS (Cognition and People Centric Computing) at University of Lusofona in Portugal
4. the University of Indonesia, Depok Indonesia.
5. University of Goettingen,
6. University of Indonesia,
7. Osaka University,
8. University of Minho in Portugal.

8.3 The 2nd ACM Information Centric Networking Conference

PI Van Jacobson presented the keynote at the 2nd ACM Conference on Information-Centric Networking. The team presented four papers.

8.4 NDN-Related Workshops

Listed below are the NDN-related workshops held at different venues:

1. The Third Workshop on Name-Oriented Mobility: Architecture, Algorithms and Applications (NOM'2016), in conjunction with IEEE INFOCOM, San Francisco, CA, April 11, 2016.

   The NDN team presented two papers at this workshop.
2. Workshop on Multimedia Streaming in Information-/Content-Centric Networks (MuSIC 2016), in conjunction with IEEE INFOCOM, San Francisco, CA, April 11, 2016.
3. Workshop on Content-Centric Networking (CCN 2015), in conjunction with IEEE International Conference on Mobile Ad hoc and Sensor Systems, Dallas, TX, USA, October 19, 2015

   The NDN team presented one paper at the workshop.
4. Workshop on Information Centric Networking Solutions for Real World Applications (ICNSRA), in conjunction with IEEE GlobeCom 2015, San Diego, CA, USA, December 6, 2015.

Chapter 9

NDN Presentations

Listed below are presentations given by NDN-NP team members during the second year of the project.

1. Alex Afanasyev, “Named Data Networking: Data-Centric Future of the Internet”. Aerospace, May, 2016
2. Alex Afanasyev, “Supporting Mobility in Named Data Networking”, ICNRG Meeting, January 2016.
3. Alex Afanasyev, “Shaping a New Architecture by Architectural Principles”, ICNRG Meeting, November 2015.
4. Alex Afanasyev, “NDN protocol development: status of reference implementations, supporting software releases, open architecture research issues”, ICNRG Meeting, October 2015.
5. Alex Afanasyev, “Schematized Trust: Design and Application”, NDN Community Meeting, September 2015
6. Alex Afanasyev, “Trust Schema: Name-Based Trust Management”, NDN Tutorial at ICN’2015, September 2015
7. Lan Wang, “Architecture Development and Routing Design in Named Data Networking,” Department of Computer and Information Science, University of Mississippi, Oct. 2015
8. Jeff Burke, “NDN “NP” Application Update: Building Automation & Management / Io,” Osaka University Murata Lab, Osaka, Japan, November 12, 2015.
9. Jeff Burke,“Named Data Networking,” Panasonic Advanced R&D, Osaka Japan, November 11, 2015.
10. Jeff Burke,“NDN Network Environments Update”, NSF FIA PI Meeting, Arlington, VA, 2015
11. Jeff Burke, “ICN Roadmaps for the next 2 years,” ACM ICN 2015, Panel participant with Cisco, Orange, ESnet, PARC, Huawei, September 2015, San Francisco, CA.
12. Jeff Burke, “The Future of the Internet is the Future of Storytelling,” NDN Community Meeting, Los Angeles, CA, September 29, 2015.
13. Jeff Burke, “From the Internet of Things to the Internet of Experiences,” ICCCN Cyberphysical Systems Panel, Las Vegas, NV, August 3, 2015.
14. Jeff Burke, “ICN as an Enabler for New Forms of Multimedia Experience,” IEEE ICME MuSIC Workshop, Invited Keynote and panel moderator, Torino, Italy, July 3, 2015.
15. Jeff Burke, Small Data CRI 2015, Invited workshop participant for Cornell Tech Small Data workshop, New York, June 15-16, 2015.

16. Jeff Burke, Invited panel participant for Network Architecture panel with Mark Stapp of Cisco, GQ Wang of Huawei, and Ignacio Solis of PARC, CCNxCon, San Jose, May 21, 2015.

17. kc claffy, "A Brief History of a Future Internet", Usenix LISA 2015, Washington DC, November 13, 2015.

18. kc claffy and Lixia Zhang, "A Brief History of a Future Internet", Usenix LISA Conversations, live talk show, YouTube LISA Conversations Channel, April 26, 2016.

19. Patrick Crowley, "Named Data Networking, New Frontiers in Network, MIT. Cambridge, MA, April 30, 2015.

20. Patrick Crowley, "Global NDN Testbed," NDN Community Meeting, UCLA. Los Angeles, CA, September 29, 2015.

21. Patrick Crowley, "NDN Startups," NDN Community Meeting, UCLA. Los Angeles, CA, September 29, 2015.

22. Patrick Crowley, "Named Data Networking," Internet2 TechEx Conference. Cleveland, OH, October 5, 2015.

23. Patrick Crowley, "Named Data Networking," Washington University OIN Conference. St. Louis, MO, October 20, 2015.

24. Christos Papadopoulos, "Named Data Networking: An Internet Architecture for the Future.", Invited talk, ESGF meeting, Monterey CA, Dec 2015.

25. Christos Papadopoulos, "Named Data Networking: An Internet Architecture for the Future.", Invited talk, LHCOPN-LHCONE meeting, Amsterdam NL, Oct 2015.

26. Christos Papadopoulos, "Named Data Networking: An Internet Architecture for the Future.", Invited talk, University of Memphis, Oct 2015.

27. Christos Papadopoulos, "Named Data Networking: An Internet Architecture for the Future.", Keynote presentation, NSF SwitchOn Workshop, Sao Paolo, Brazil, Oct 2015.

28. Christos Papadopoulos, "Managing Scientific Data with Named Data Networking", NSF PI meeting, Austin TX, Sept 2015

29. Christos Papadopoulos, "A Catalog for Scientific Data.", CERN, Switzerland, July 2015.

30. Beichuan Zhang, "Information-Centric Internet of Things: Driving the Future Network Architecture," Beijing Jiao Tong University, June 2015

31. Beichuan Zhang, "Named Data Networking (NDN)," Beijing University of Post and Telecommunications, June 2015

32. Beichuan Zhang, "NDN Live Video Broadcasting over Wireless LAN," IEEE ICCCN, Aug 2015

33. Beichuan Zhang, "Named Data Networking (NDN)," Future Network Development and Innovation Forum, Dec 2015

34. Beichuan Zhang, "Routing in Named Data Networking," NIST, Feb 2016

35. Lixia Zhang"Evolving Internet into the future via Named Data Networking," Jilin University, China, September 2015.

36. Lixia Zhang"Moving Internet into Future via Named Data Networking," Fujitsu Corporation, Japan, October 2015.

37. Lixia Zhang"Named Data Networking," Keio University, Japan, October 2015.

38. Lixia Zhang“Named Data Networking: the Design of a New Internet Architecture,” Waseda University, Japan, October 2015.

39. Lixia Zhang“NDN Design and development: recent progress,” Workshop on Research Activities and Future of EU/US/JP ICN Projects, Tokyo, Japan, October 2015.

40. Lixia Zhang“Networking via Named Data,” Mitre Corporation, December 2015.

41. Lixia Zhang“Securing the Internet by Securing Data Directly,” National Chiao Tung University, Taiwan, December 2015.

42. Lixia Zhang“New Applications via Opportunistic Peer-to-Peer Wireless Communications,” NSF Wireless Cities Workshop, February 2016.

43. Lixia Zhang“Named Data Networking of Things,” US-Europe Workshop on the Next Generation Internet of Things, March 2016.

44. Lixia Zhang“Looking back, looking forward: why Internet needs a new protocol architecture,” UCSD, April 2016.

45. Lixia Zhang“Challenges in the Internet of Things Realization,” PKU-UCLA Joint Research Institute In Science And Engineering 7th Annual Symposium, May 2016.

References

[1] The NDN project testbed. `http://www.named-data.net/testbed.html`, 2012.

[2] Alexander Afanasyev, Yingdi Yu, Lixia Zhang, Jeff Burke, kc claffy, and Joshua Polterock. The second named data networking community meeting (NDNcomm 2015). *ACM SIGCOMM Computer Communication Review*, January 2016.

Chapter 10

Publications

Listed below are publications by NDN-NP team members during the second year of the NP project (1 May 2015 – 30 April 2016).

[1] Vince Lehman, Ashlesh Gawande, Rodrigo Aldecoa, Dmitri Krioukov, Lan Wang, Beichuan Zhang, and Lixia Zhang. An Experimental Investigation of Hyperbolic Routing with a Smart Forwarding Plane in NDN. In *Proceedings of the IEEE IWQoS Symposium*, June 2016.

[2] Peter Gusev, Zhehao Wang, Jeff Burke, Lixia Zhang, Eiichi Muramoto, Ryota Ohnishi, and Takahiro Yoneda. Real-time streaming data delivery over Named Data Networking (invited paper). *IEICE Transactions*, May 2016.

[3] Yu Zhang, Alexander Afanasyev, Jeff Burke, and Lixia Zhang. A Survey of Mobility Support in Named Data Networking. *Proceedings of the third Workshop on Name-Oriented Mobility: Architecture, Algorithms and Applications (NOM'2016)*, April 2016.

[4] Stephanie DiBenedetto and Christos Papadopoulos. Mitigating Poisoned Content with Forwarding Strategy. *Proceedings of the third Workshop on Name-Oriented Mobility: Architecture, Algorithms and Applications (NOM'2016)*, April 2016.

[5] Wentao Shang, Adeola Bannis, Teng Liang, Zhehao Wang, Yingdi Yu, Alexander Afanasyev, Jeff Thompson, Jeff Burke, Beichuan Zhang, and Lixia Zhang. Named Data Networking of things. In *Proceedings of 1st IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI'2016)*, April 2016. (Invited paper).

[6] Alexander Afanasyev, Yingdi Yu, Lixia Zhang, Jeff Burke, kc claffy, and Joshua Polterock. The second named data networking community meeting (NDNcomm 2015). *ACM SIGCOMM Computer Communication Review*, January 2016.

[7] Chengyu Fan, Susmit Shannigrahi, Steve DiBenedetto, Catherine Olschanowsky, Christos Papadopoulos, and Harvey Newman. Managing scientific data with Named Data Networking. In *Proceedings of the Fifth International Workshop on Network-Aware Data Management*, November 2015.

[8] Alexander Afanasyev, Zhenkai Zhu, Yingdi Yu, Lijing Wang, and Lixia Zhang. The Story of ChronoShare, or How NDN Brought Distributed Secure File Sharing Back. In *Proceedings of IEEE MASS 2015 Workshop on Content-Centric Networks*, October 2015.

[9] Yingdi Yu, Alexander Afanasyev, David Clark, kc claffy, Van Jacobson, and Lixia Zhang. Schematizing Trust in Named Data Networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, September 2015.

[10] Ilya Moiseenko, Lijing Wang, and Lixia Zhang. Consumer/producer communication with application level framing in Named Data Networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, September 2015.

[11] Peter Gusev and Jeff Burke. NDN-RTC: Real-time videoconferencing over Named Data Networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, September 2015.

[12] Tian Song, Haowei Yuan, Patrick Crowley, and Beichuan Zhang. Scalable name-based packet forwarding: From millions to billions. In *Proceedings of the 2nd ACM Conference on Information-Centric Networking*, September 2015.

[13] Jongdeog Lee, Akash Kapoor, Md Tanvir Al Amin, Zhehao Wang, Zeyuan Zhang, Radhika Goyal, and Tarek Abdelzaher. InfoMax: An information maximizing transport layer protocol for Named Data Networks. In *24th International Conference on Computer Communication and Networks (ICCCN)*, August 2015.

[14] Giulio Grassi, Davide Pesavento, Giovanni Pau, Lixia Zhang, and Serge Fdida. Navigo: Interest forwarding by geolocations in vehicular Named Data Networking. In *IEEE 16th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2015.

[15] Haowei Yuan and Patrick Crowley. Reliably scalable name prefix lookup. In *Proceedings of the ACM IEEE Symposium on Architectures for Networking and Communications Systems*, May 2015.

[16] Fu Wenliang, Hila Ben Abraham, and Patrick Crowley. Synchronizing namespaces with invertible bloom filters. In *To appear in ANCS 2015*, 2015.

NDN Technical Reports

All the reports are available online at `http://named-data.net/publications/techreports/`

[1] Minsheng Zhang, Vince Lehman, and Lan Wang. PartialSync: Efficient Synchronization of a Partial Namespace in NDN. Technical Report NDN-0039, NDN, June 2016.

[2] Yingdi Yu, Alexander Afanasyev, and Lixia Zhang. NDN DeLorean: An Authentication System for Data Archives in Named Data Networking. Technical Report NDN-0040, NDN, May 2016.

[3] Wentao Shang, Yingdi Yu, Ralph Droms, and Lixia Zhang. Challenges in iot networking via TCP/IP architecture. Technical Report NDN-0038, NDN, February 2016.

[4] Vince Lehman, A K M Mahmudul Hoque, Yingdi Yu, Lan Wang, Beichuan Zhang, and Lixia Zhang. A secure link state routing protocol for NDN. Technical Report NDN-0037, NDN Project, January 2016.

[5] Wentao Shang, Yingdi Yu, Teng Liang, Beichuan Zhang, and Lixia Zhang. NDN-ACE: Access control for constrained environments over Named Data Networking. Technical Report NDN-0036, NDN, December 2015.

[6] A. Bannis and J. Burke. Creating a secure, integrated home network of things with Named Data Networking. Technical Report NDN-0035, NDN, 2015.

[7] Yingdi Yu, Alexander Afanasyev, and Lixia Zhang. Name-based access control. Technical Report NDN-0034, Revision 2, NDN, January 2016.

[8] P. Gusev and J. Burke. NDN-RTC: Real-time videoconferencing over Named Data Networking. Technical Report NDN-0033, NDN, 2015.

[9] NFD Team. NFD developer's guide. Technical Report NDN-0021, Rev. 6, NDN, March 2016.