Named Data Networking Next Phase (NDN-NP) Project
May 2014 - April 2015 Annual Report

Principal Investigators

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

Beichuan Zhang
University of Arizona

Kim Claffy
University of California, San Diego

Christos Papadopoulos
Colorado State University

Tarek Abdelzaher
University of Illinois at Urbana-Champaign

Lan Wang
University of Memphis

J. Alex Halderman
University of Michigan

Patrick Crowley
Washington University

Contents

FIA: Collaborative Research: Named Data Networking Next Phase (NDN-NP) 2015 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 command to turn on some lights, *etc.* 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 four years of NDN design and development efforts (which has a 4-month overlap with NDN-NP) 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 architectural details, and most importantly, verify and shape the architectural direction. We translated our vision to a simple and elegant packet format design, a modular and extensible NDN forwarding daemon, and a set of libraries, including security support, to support application development. These achievements establish a platform that enabled us to tackle new application environments as we stated in the NDN-NP proposal: open mobile health applications, building automation and management systems, and multimedia applications. We achieved all our major milestones for the first year of the NDN-NP project. Highlights include:

- Design, implementation, public release, and testbed deployment of a new NDN Forwarding Daemon (NFD). The basic NFD design goals are modularity and extensibility to enable experimentation with new protocol features, forwarding data structures and algorithms, and applications.
- Design and preliminary implementation of pilot applications for our network environments, addressing namespace design, trust management, and end-user requirements. We also designed and implemented a real-time videoconferencing application to explore how to leverage NDN features in low-latency communication.
- Continued work on a portfolio of libraries used to develop NFD, network environment pilots, and other applications, including a new high-level API and research in transport mechanisms uniquely supported on NDN, such as SYNC and information maximization.
- Further development of NDN security support, implemented in the libraries and motivated by the requirements of the network environments: trust management, encryption-based access control, signing and verification schema, and support tools.
- Development of NDNS, a DNS-like resolution system that can securely store any named data, including certificates, and NDN links used for publisher mobility support in Open mHealth.
- Evaluation of hyperbolic routing performance to understand its feasibility in supporting NDN's inter-domain routing.
- The establishment of the NDN Consortium to promote a vibrant open source ecosystem of research and experimentation around NDN, and the organization of the first NDN Community meeting.

In the process of these achievements we have created and enjoyed many opportunities for training and education of the NDN team, our students, and collaborators, expanding the global community of researchers with skills and experience in the thriving field of Internet architecture research and development.

Next year, we will dive more deeply into research issues raised by the environments, including privacy support through group encryption, support for mobile publishers, and scaling and feedback control in BMS. These activities will bring new understanding to issues in the NDN architecture: namespace design, distributed data synchronization, usable trust management, and extremely scalable routing and forwarding. We have also participated in strategic planning conversations with the Department of Energy, who is considering restructuring their own network research program to support their persistent networking problems, most of which have their roots in limitations of TCP/IP that have motivated the design of NDN.

Chapter 1

Introduction

This report catalogs a wide range of our accomplishments during the first year of the "NDN Next Phase (NDN-NP)" project. This phase of the project is *environment-driven*, in that we are focusing on deploying and evaluating the NDN architecture in two specific environments: building automation management systems and mobile health, together with a cluster of multimedia collaboration tools.

NDN architecture development has always been application-driven: even before this phase of the project began, we already developed several pilot applications to verify and validate the architecture design. The NDN-NP application environments are fundamentally different from these earlier pilots in several ways:

- They have stringent requirements on identity, security and privacy. NDNFit is a personal fitness-tracking app we are developing to experiment with the mobile health environment; it collects and processes personal exercise data that may be sensitive and private (Section 2.2).
- They are much bigger in scale. We will build a large scale repository to support UCLA's building management system, which raises challenging issues in handling granular and complex data queries (Section 2.1).
- They stress-test NDN's architectural promise in offering data access by topologically independent names in scale. Both NDNFit and NDNrtc (our video conferencing tool, Section 2.3) require support for mobile producers.
- Rather than being short-lived demoware, they are expected to evolve into daily usage.

We have made substantial strides into the design and development of these new application environments (Chapter 2). In terms of protocol development, we defined a new NDN packet format, finished developing and implementing a complete prototype NDN codebase including a modular and extensible NDN Forwarding Daemon (NFD) and a set of libraries with security support, and deployed the code on the global NDN testbed which now connects 26 sites across three continents. These achievements established a platform that enables us to tackle the new application environments proposed for this project.

What we have learned during the first year of NP

First, the application environments continued to drive and inform architectural development.

- To support the security needs of the selected application environments, we developed a trust schema to automate signing and verification rules (Section 3.2) and support group encryption (Section 3.3). We also designed extensive functionality to facilitate NDN security operations (Sections 3.4 – 3.7).
- To facilitate experimentation with the architecture, we developed a modular and extensible NDN Forwarding Daemon (NFD), which implements rigorous security checking throughout its execution (Section 4.2).
- To support mobility, we prototyped a name-based lookup service, NDNS (NDN DNS, Section 4.3).
- To enable globally scalable NDN networks, we continued investigating advanced routing and forwarding

techniques and methods (Sections 4.4-4.5).

- To explore boarder NDN application scenarios, we demonstrated the applicability of NDN to solve networking problems facing other computational and big data scientific domains, and for which today's TCP/IP architecture does not have good solutions (Section2.4). In particular we collaborated with climate scientists funded by the Department of Energy to show how NDN natively supports data retrievals from multiple data sources and via multiple paths in parallel, providing for simplified data dissemination and robust failover in case of network partition, node or end-site failure.

Second, security has been a central motivation for the NDN architecture, but it is also the most challenging. Security researchers from the University of Michigan joined our project team, and we received additional security-related guidance from NDN Project Advisory Committee member David Clark. We still have a long way to go toward offering *usable security* to a broad set of applications and to the developer community who will be writing them, and we continue to prioritize our retreats and PI conversations around this challenge.

Third, we succeeded at our primary community-building objective: maintaining and extending free and open source development of the entire NDN codebase, ranging from NFD to libraries to applications (see `https://github.com/named-data`). The development team has grown far beyond the NSF-funded NDN-NP campuses, and now includes people from Europe and the Asia Pacific region. We have received feedback that the ecosystem of software and supporting research tools is key to the vibrancy of the extended NDN community, as evidenced by the NDN network simulator (ndnSIM) tool. ndnSIM's user mailing list has 300+ members, many of them not only use the tool but actively contribute to its development.

We conclude the year with a surprising amount of momentum. Commercial interest in NDN and its industrial applications continues to grow dramatically. The NDN industrial consortium now features ten industrial members, and many of these, along with others not part of this formal group, are pushing forward aggressively to standardize and commercialize NDN in niches such as video streaming. As our software infrastructure grows in popularity, we find that these application use cases both inform and stress the importance of pursuing the important basic research challenges that form the core of the NDN-NP research agenda, including routing, support for distributed applications, trust management, and usable security. In the final year of the NDN-NP project, we hope to further organize and inspire the growing research community developing around NDN. We also plan to articulate a long-term vision for NDN to enable the global research community to embrace and collaborate with us on the critical research challenges and opportunities that we see.

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, Lijing Wang, Haitao Zhang (UCLA); Jongdeog Lee, Shiguang Wang (UIUC)
Undergrads	Akash Kapoor, Yang Sheng (UIUC)
Staff	Adeola Bannis, Peter Gusev, Alex Horn, Jeff Thompson, Zhehao Wang, (UCLA); Hongyan Wang (UIUC)
	Postdoc: 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 [6], *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 (Figure 2.1). 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. (At the time of this report, 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 in 2016.) 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 2015 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. Figure 2.2 lists examples of currently monitored data points, selected to provide a representative example of the types of data monitored in a typical building at UCLA, to inform namespace, data payload, and other design considerations.

Our high level objectives for the pilot application are to support two dominant uses for this data on

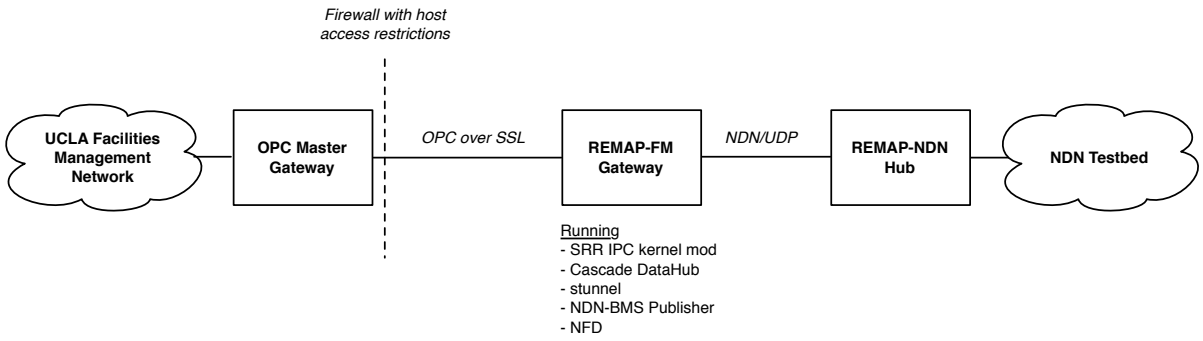


Figure 2.1: Bridge to UCLA campus monitoring system.

KERCKHOFF_A264A.CHWS.RT_CV	CHW Return Temp
LIFESCI_1313.XFMR-B.DMD.INST_CV	Electricity 2 - Instant Demand
KNUDSEN_2-218.AH10.CT_CV	Cold Deck Supply Temperature
OOARH_CV	Outside Air Humidity
KNUDSEN_4-151.SF6.VFD:FREQ OUTPUT_CV	Supply Fan VFD Speed

Figure 2.2: Example point names from the UCLA monitoring system.

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. Section 3.1.2 summarizes our progress with an encryption-based access control approach (that builds on previous work) and initial trust model.

The exploration of this network environment pilot application involves a cross-campus collaboration:

- UCLA REMAP - application design; interface to building management systems.
- UCLA IRL - repository design; implementation; trust and privacy design and implementation.
- University of Michigan - overall security approach.
- University of Arizona - forwarder support.
- University of Memphis - routing support.
- WUSTL - testbed support.
- UCSD - project management support.

Additionally, UCLA Facilities Management has collaborated with the NDN team to enumerate their requirements, challenges, and limitations with the existing system, which motivated the pilot application. As part of the NP work completed this year, UCLA installed additional server hardware and configured a software tunnel to provide the NDN team access to live data.

Existing building management systems (BMS) often rely on relational databases to support data analysis, such as tracking average power consumption or detecting abnormal system behavior. A typical approach is to store collected data into a (usually centralized) relational database and run data mining jobs using Structured Query Language (SQL), which provides powerful language constructs that express common data processing functionalities. To facilitate transition to an NDN-based BMS, we are developing a similar SQL interface, while also providing access to data via names derived from the existing application namespace.

Our initial approach to providing SQL support is to store the data in SQL databases and build NDN repo semantics on top of that. NDN-over-SQL is easier to implement than SQL-over-NDN, because SQL (which

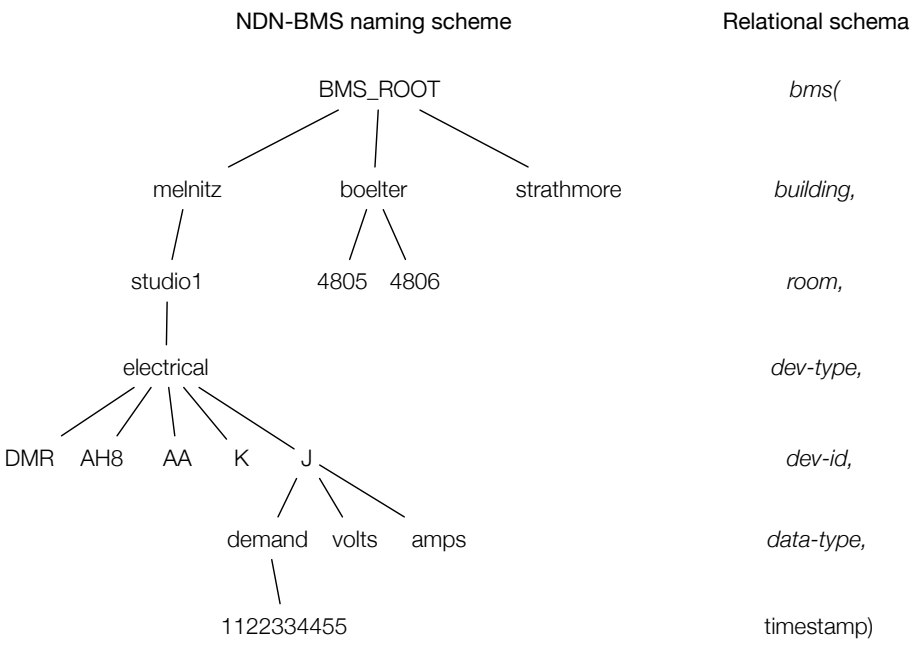


Figure 2.3: An example of mapping between NDN naming hierarchy and SQL schema

implements relational algebra) can express most NDN Interest semantics once given a mapping between the NDN naming hierarchy and the SQL table schema. For example, Figure 2.3 depicts a simplified BMS data schema, in which the following Interest:

```
/<prefix>/melnitz/studio1/electrical/AA/voltage, Exclude = (ANY, T1), ChildSelector = 0
```

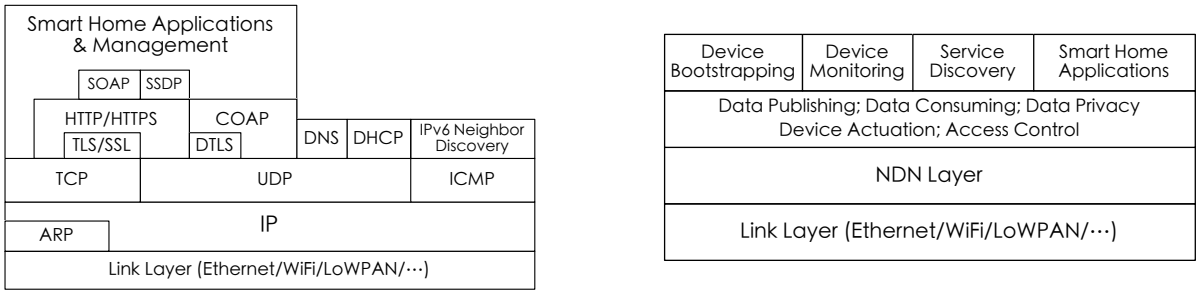
could be translated into the following SQL query:

```
SELECT * FROM bms
WHERE building = melnitz AND room = studio1 AND devtype = electrical
AND devid = AA AND devtype = voltage
AND NOT (timestamp <= T1)
ORDER BY timestamp ASC LIMIT 1;
```

This mechanical translation generalizes to arbitrary Interests. We implemented a prototype of this "NDN-over-SQL" approach, which provides two interfaces: one for normal NDN Interest/Data exchange, and one for traditional MySQL queries that pass directly to the underlying SQL database.

We also considered creating a shim layer that translates SQL queries into NDN Interests, similar to how other projects have developed modules to decompose SQL queries into Map-Reduce jobs that are natively supported by Hadoop File System (HDFS) [11, 13, 1]. However it is non-trivial to do efficient SQL decomposition; re-using the above modules could add layers of indirection between the SQL and the NDN layer and prohibit direct experimentation with NDN features such as naming and security. Thus, we are using the more tractable "NDN-over-SQL" approach in the short term, and plan to explore new design directions that can fully utilize NDN features to support secure and scalable data queries.

Security is another dimension on which NDN offers advantages over the TCP/IP-based BMS systems, which mostly rely on physical and isolation to implement security, hindering innovative integration efforts and often proving easy to circumvent. NDN's intrinsic security features – data verification via NDN signatures and encryption-based access control – enable integrated security functionality not feasible with today's systems. For example, the hierarchical namespace of the NDN-BMS (Figure 2.3) provides a starting point



(a) Typical IoT stack over IP.

(b) Analogous IoT stack over NDN

Figure 2.4: Comparison of IoT stacks explored in the NDN-IoT project.

for a trust management approach that leverages this real-world hierarchy reflecting the physical organization of buildings and systems (Chapter 3).

2.1.1 NDN-IoT: NDN on the Raspberry PI and Arduino

To complement the BAMS work at the enterprise level, we also explored the potential for NDN to enhance emerging ubiquitous computing environments, or the so-called "Internet of things." While the NDN-BMS application focuses on administrative and plant-level concerns without addressing computational and performance constraints, the IoT space provide a new class of NDN-capable devices that possess limited resources. We developed multiple platforms to support IoT experimentation.

- **Raspberry PI support** With additional support from Qualcomm Research, we completed an open source (GPL) "NDN of Things" toolkit [1] to support exploring NDN for the Internet of Things using the popular Raspberry PI platform. This platform includes functions for bootstrapping and device discovery, as well as sensor data publication and actuation. The Raspberry PI is an accessible embedded device capable of acting as a sensor aggregation point or processing node in the middle of an IoT network. NDN offers promising simplifications and efficiencies over TCP/IP-based solutions (such as COAP/DTLS, see Figure 2.4) and easier integration with a future global internet. Initial performance comparisons between these two stacks using our implemented version of the NDN-IoT stack are promising; we will publish a detailed comparison this coming year.
- **Arduino support** In addition to the Linux-based Raspberry PI platform, we explored NDN's viability for the type of low-cost, low-power, microcontroller-based devices that will pervade IoT environments, and eventually feed into BMS systems as well. As a proof-of-concept for NDN functionality at the IoT edge, we developed and tested library support for the Arduino platform, an open-source microcontroller [2]. Our demonstration included the ability to query a local sensor, encode the result in the NDN-TLV packet format, add an HMAC-based authenticator, and respond to incoming Interests with a corresponding Data packet[3]. To show that a common stack for all platforms is feasible for NDN, we built the library for this application using same C-language core as the NDN-CPP library, with simplified C++ bindings.
- **NDN-OPT Personnel Tracking**
  To explore NDN support for real-time infrastructure sensing services such as automatic personnel tracking, we have developed an NDN publisher/consumer for data produced by UCLA REMAP's NSF-supported open source personnel tracking system, OpenPTrack.[4] We targeted this system for real-world use, where we would have otherwise built a custom messaging solution or used common middleware. We integrated our time-series data producer (C++) with the Robot Operating System

[1] `https://github.com/remap/ndn-pi/releases`
[2] https://en.wikipedia.org/wiki/Arduino
[3] Since the NFD function is not included, the last step requires prefix registration with a remote NDN forwarder.
[4] `http://openptrack.org/`

(ROS); we provide our (Javascript) consumer as a browser-side library. NDN-OPT realizes low-latency (soft real-time) data transmission at higher data rates than standard BMS sensors (currently at 30Hz), and provides an easy way of sharing tracking data via browser-based applications.

These examples span a range of uses of NDN in the building monitoring and industrial automation space, from plant-wide monitoring (NDN-BMS) to embedded aggregators and interfaces (Raspberry PI) including at the microcontroller level (Arduino), to real-time sensing in infrastructure (NDN-OPT). They illustrate how NDN can provide a common layer of authenticated communication across a variety of scales, reducing the requirement for middleware and simplifying the approach to security (Chapter 3).

2.1.2 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)* Worked with UCLA facilities management to discuss and review limitations of the current systems, and reviewed key security and management challenges in such networks. This process will continue in Year 2.
- *Design NDN namespace, repository, trust and communication model for use cases, such as energy management, new building commissioning, feedback control. (Y1; updated in Y2)* Developed namespace, repository, trust, and initial communication models for monitoring and configuration, and will test them in Summer 2015. The NDN-OPT project has enabled us to work with low-latency, higher data rate sources for control of multimedia applications. We will consider feedback control in Year 2.
- *Implement low-level NDN applications, such as energy management data gathering. (Y1)* In progress.
- *Preliminary embedded platform support. (Y2)* Done.
- *Integrate UCLA building data (10-20 buildings) into NDN testbed, (Y2)* Done.
- *Implement high-level NDN application for enterprise building monitoring, applying distributed 3D visualization work done in the first FIA project. (Y2)* Starting summer 2015.

2.2 Open mHealth

Open mobile health (mHealth) has emerged as an important market and a key area of Health IT, a national priority. This network environment continues our work in the first NSF-supported NDN project on participatory sensing. We collaborated with the Open mHealth project [3] led by Deborah Estrin (Cornell) and Ida Sim (UC San Francisco) to understand their requirements. The Open mHealth team advocates an interoperable, Internet-inspired approach. They propose a thin waist of open data interchange standards (Figure 2.5) to enable an ecosystem of sensing, storage, analysis, and user interface components to support medical discovery and evidence-based care. 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.

Nine 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.
- Tsinghua University - repository.
- 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 Pilot Application

NDNFit is a mobile physical activity monitoring application (fitness tracker) that supports location-based notifications. Commercially available TCP/IP-based parallels include Nike+, Fitbit, Endomondo, etc. Consistent with the Open mHealth philosophy, NDNFit is designed to participate in an **ecosystem of composable services** rather than be a single siloed application. NDN can support these high-level goals of NDNFit more securely and efficiently because it eliminates the need to map massive amounts of private health data to many unrelated endpoints in the ecosystem. NDN accomplishes the sharing of protected data more simply and securely using asynchronous named data publishing and consumption following an established trust schema. Similarly, OAuth-style authentication, which works for user-facing components but is quite burdensome for composable services, can be replaced in NDN with encryption-based access control. This year we made progress demonstrating these simplifications in our design and implementation of NDNFit.

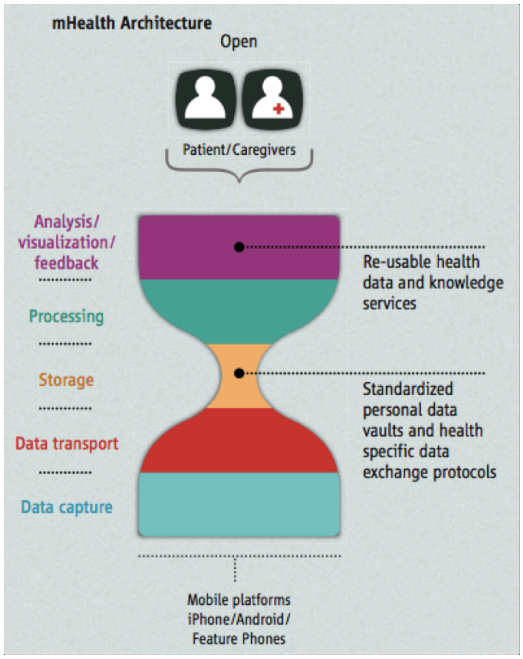


Figure 2.5: The Open mHealth architecture proposes an hourglass model, where the interoperability layer ("thin waist") uses standardized data exchange [5]. In NDN that layer is a topology-independent namespace with payload conventions.

NDNFit targets these features for the user:

- Capture and report walking, jogging, and running activity data via a mobile device and upload to a mobile-friendly data repository.
- Calculate and report activity metrics based on GPS and accelerometer data for both automatically and manually identified rounds of exercise.
- Support structured comparison of data within ad-hoc and formal groups or teams.

Our design raised other challenges relevant to the end-user experience, including data producer mobility (so users can change data hosting providers without disruption), identity management (to manage multiple identities per user across platforms); and user-friendly encryption-based access control.

This year we developed an **overall application architecture**, including a **trust model** and **encryption-based access control design**, an approach to **publisher mobility support**, and an initial direction for **distributed computation** based on the University of Basel's *Named Function Networking* research [10]. We also began to define **autoconfiguration support**, the NDN equivalent of DHCP needed for data publishers. Specific results include:

- a conceptual system architecture and namespace design (Figure 2.6);
- support for mobile publishers;
- support for public key certificates;
- a scheme for group-based encryption to support fine-grained access control (initially based on an adaptation of the approach for the building monitoring network environment);
- a personal repository to enable of sharing of sensed data; and
- an evolving, draft-status technical report.[5]

[5] `https://github.com/remap/ndn-netenv-techreports`

This work informed our design of a more comprehensive application architecture for NDNFit (Figure 2.7) from which we will start to build out the pilot application.

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)* Done. Included conversations with Open mHealth architects and developers, code review and porting of existing applications, and literature review. Will continue in Year 2.
- *Design namespace, repository, trust and communication model for use cases, e.g., diabetes or PTSD treatment (Y1; updated in Y2)* Developed initial design.
- *Repository implementation providing backing storage for prototype applications. (Y1)* Designed repository, began coding.
- *Integrate named data networking into the Ohmage mobile data collection framework. (Y2)* Starts summer 2015.
- *Pilot user-facing application using NDN, for testing by Open mHealth team. (Y2)* In progress.

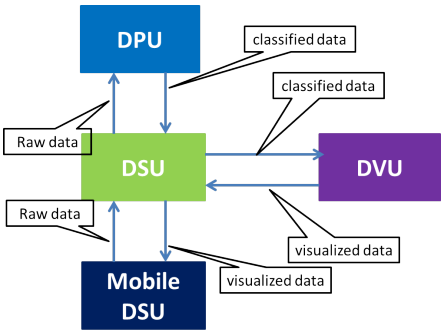
2.3 Mobile multimedia collaboration: ndnrtc (NDN real-time conferencing tool)

ndnrtc is a fully implemented real-time videoconferencing application based on the WebRTC library and VP8 codec, which we developed to understand the challenges of low-latency communication over NDN. We used it to explore the potential to shift real-time communication models from push-based (when producer writes data to the socket which then transmits the data to consumer) to pull-based (producer publishes data at its own pace, while consumers request what they need and manage incoming segments). In this case, 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. We assume the producer publishes at multiple bitrates from which the consumer can select. The NDN model shifts capability and control to the consumer, enabling scalable delivery at the network layer.

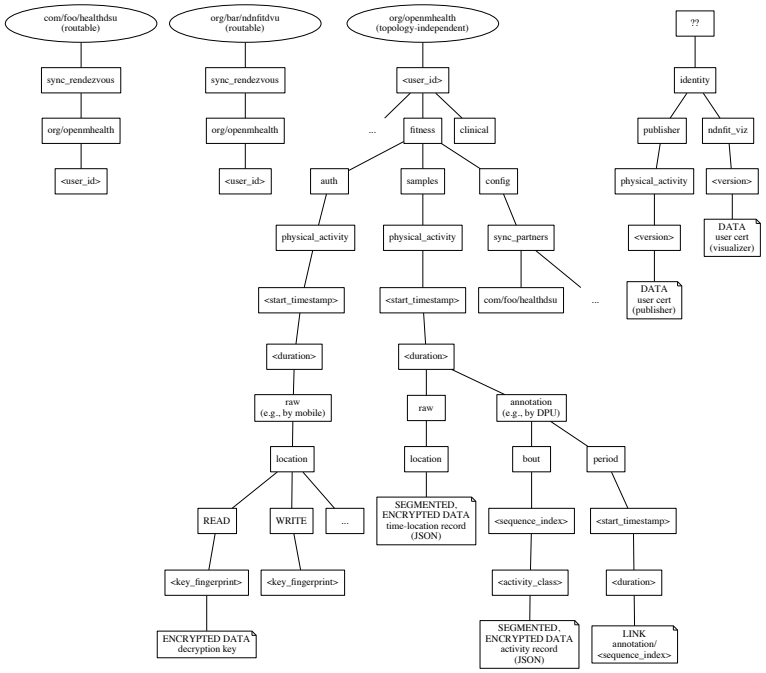
The key challenge in a consumer-driven model for videoconferencing (and other latency-sensitive applications) is to *ensure the consumer gets the latest data in a network with internal storage*, without resorting to direct producer-consumer communication (which limits scaling). Since network-cached video data can be retrieved faster than it is produced, the consumer can first try to retrieve a series of packets quickly and monitor inter-arrival gaps of the packets. It can leverage its knowledge of segment publishing rate(s) to guess whether it should continue to retrieve cached data, or retrieve data directly from the producer.

In more detail, the approach is the following. During bootstrapping, the consumer aims to quickly exhaust all segments in network caches first. By increasing the number of outstanding interests beyond the round-trip time times video production rate, the consumer pulls cached data out of the network until the freshest data starts to arrive. The consumer then keeps the number of outstanding interests below a threshold. It can adapt this threshold as a function of observed inter-arrival delays to achieve the best synchronization state with the producer.

So far, this consumer-driven model has generated usable results with sub-second latencies required for interactive videoconferencing, while enabling scaling via NDN network caching combined with multicast delivery properties. We would like to further explore the value of the producer providing meta data to enable more efficient data retrieval. Adding such information to returned data segments, e.g, interests nonce values, interests arrival timestamps and data generation delays, can help consumers match interest generation rate to data production rate, estimate network delay, and detect congestion. Keeping historical data on the consumer side, e.g, an average number of segments per video frame type, may also enable more efficient interest pipelining by helping the consumer select a proper number of required initial interests to fetch upcoming frames.



(a) Conceptual application architecture with data flow, including primary component types in the ecosystem: Data Storage Unit (DSU); Data processing unit (DPU); Data visualization unit (DVU).



(b) Initial NDNFit namespace design.

Figure 2.6: NDNFit conceptual architecture and namespace design

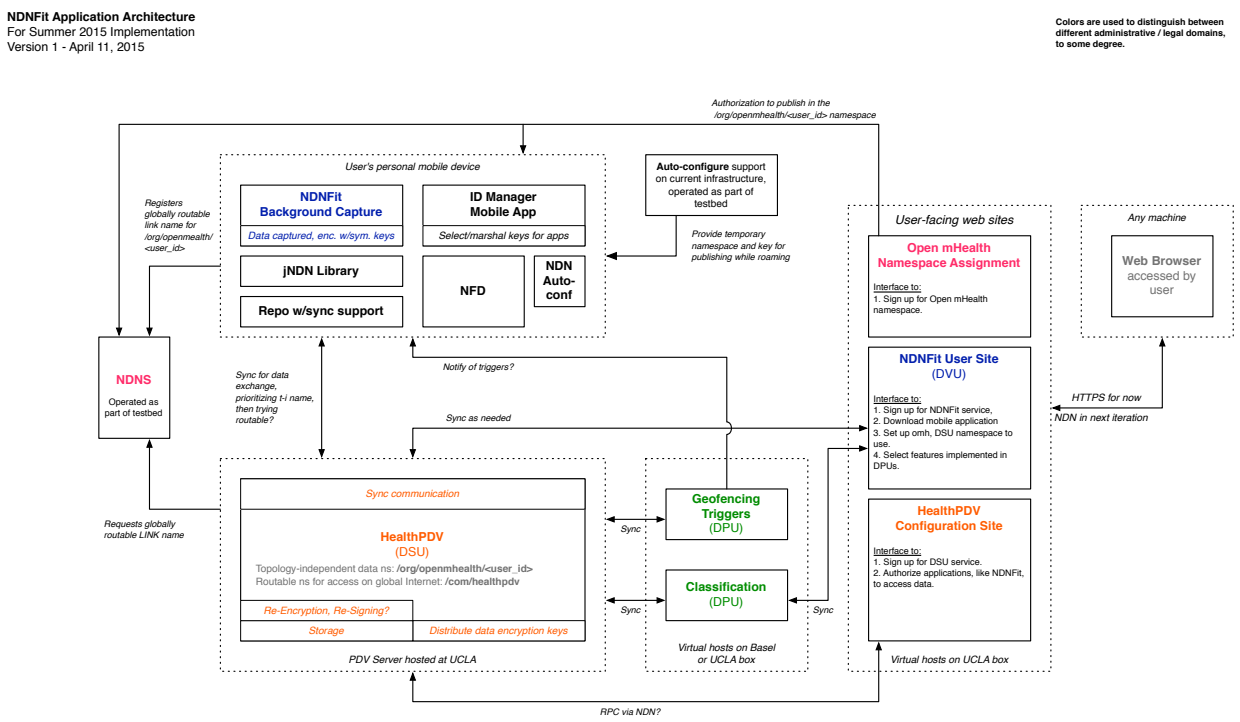


Figure 2.7: Comprehensive NDNFit architecture. (Colors indicate administrative domains.)

2.4 Scientific Data Applications

Through a combination of this project funding, other NSF funding, and outside support and collaboration, we began to explore 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 standards. For example, CMIP5 (Coupled Model Intercomparison Project) data is about 3.5 Petabytes in size and distributed among more than twenty institutes worldwide. We are developing an application that provides efficient dataset discovery and retrieval using NDN. We developed software that translates climate model file names to uniform NDN names to ease data management. Individual filenames contain several fields describing parameters such as variable name, granularity and recorded time period. Below is an example of an individual file name along with component names.

ua_6hrPle_IPSL-CM5B-LR_rcp85_r1i1p1_2026010103-2035123121.nc
(variable name, mip table, model id, experiment id, ensemble, start time, file type)

Since atmospheric filenames are hierarchical, they easily translate to NDN names which facilitates data discovery and management. With the help of the atmospheric scientists, we added new components to the NDN names that were not in the original atmospheric filenames; these compoents embody meta-data and manual specifications of the data.

/CMIP5/output/IPSL/IPSL-CM5B-LR/rcp85/6hr/atmos/ua/r1i1p1/2026010103-2035123121/
(**activity, product, institute**, model id, experiment id, **freq., modeling realm**, variable name, ensemble, start time)

Once the community agrees on a standard NDN naming convention, they can use it to name files for discovery and retrieval. To speed up discovery we also created a database catalog containing the NDN names of all datasets published in the system. Applications can query the catalog using a limited number of components to learn full NDN names of requested data. Our implementation supports data publication by translating a filename into an NDN name and indexing it in the catalog, as well as data retrieval by enabling querying the catalog from anywhere in the network to retrieve matching NDN names, which can then be used to retrieve the files via NDN. Details are provided in [9]. All climate science NDN applications and tools are under active development.[6]

2.4.2 NDN in High Energy Particle Physics (HEP)

Our success with the climate application inspired us to expand our scope to the High Energy Particle Physics community. Computing models of The Large Hadron Collider (LHC) experiments generate petabytes of data every day. Like the climate community, the data is dispersed across tens of institutes across the world. However, unlike the climate domain, LHC data storage and usage have followed a strict hierarchy. CERN generates data using LHC and represents Tier 0. Several (Tier1) institutes hold this generated data; smaller organizations form Tier 2 and 3, which use the data hosted in Tier 1 sites. Expanding needs for CPU, storage and network capacity are leading to more agile and pervasive models that optimize highly distributed heterogeneous resources to exchange data among many Tier 2 and 3 sites. The HEP community uses distributed services (e.g., CMS's AAA, ATLAS' FAX projects) to access and analyze the data.

Several NDN features offer advantages to the HEP computing use case. Data sources publish new content according to an agreed naming convention. Network nodes dynamically cache data based on content popularity and satisfiy subsequent interests directly from a cache, thus lowering the load on servers. NDN natively supports data retrieval from multiple data sources and via multiple paths in parallel, providing robust failover in case of network partition, node or end-site failure. We presented this work at the Conference on Computing in High Energy Physics (CHEP2015) in Okinawa, Japan, where two other presentations also referred to the use of NDN in high energy physics, including a keynote speech.

2.5 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.

2.5.1 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 certain level of stability within ndn-cxx release cycles.

Since the initial release on May 7, 2014, we continued to advance ndn-cxx to support updated definitions of NDN protocol, support new features of the NDN forwarder daemon (NFD), and simplify the tasks of writing NDN applications. More recently, ndn-cxx has moved to be based on C++11, to leverage advantages of modern programming techniques and simplify library and application maintenance. Basic features added

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

to ndn-cxx since its initial release include 1) dispatching of interests using filters based on name prefixes and regular expressions; 2) support for new naming conventions [7]; tagging of interest and data packets; 3) application-defined meta-information blocks; and 4) producer-generated NACKs that authoritatively indicate non-existence of requested content. Security-related updates to ndn-cxx include 1) updating the signature data structure to the latest NDN specification [8] (which has SHA256 digest signature support[7]), and 2) periodic reloading of trust anchors to support dynamic trust models. To facilitate implementation of common tasks in NDN applications, the ndn-cxx library now supports 1) per-application in-memory storage with LRU, LFU, and FIFO replacement policies; 2) detection of network state changes, and 3) fetching of multi-segmented data. To support evolution of the NFD management protocols, the ndn-cxx library has two new APIs. One API enables requests of specific action from NFD's content store when making packets available (e.g., not to cache certain data packets). The second API implements NFD face query operations, to retrieve a subset of faces that match a filter.

The following applications and projects currently 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 realtime 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

2.5.2 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. Over the last year, it has developed significantly to track features required by the new forwarder (NFD), applications, and experimental projects by the team.

NDN-CCL features to support NFD

- The crucial update required was to support NFD prefix registration, with automatic fallback to register with the older NDNx forwarder so that applications could experiment with either. The libraries support NFD's optional LocalControlHeader to provide low-level operational details, e.g. on which face NFD sends an interest.
- NFD uses signed command interests, so the API supports signing and verifying an interest. In many types of command interests, a name component is a TLV-encoded message with extra parameters. We developed the ProtobufTlv class, where a message object is constructed in a common manner across libraries using the Google Protobuf API and is encoded or decoded in TLV by ProtobufTlv. Thus the application does not need to work with the low-level details of TLV encoding. NFD requests may return a large answer split into multiple segments; the SegmentFetcher class abstracts away the details of fetching these segments.
- We used these API utility classes to provide example programs for NFD configuration commands such as "list RIB", "list faces", "list channels" and "register route".

[7] Signing interests with SHA256 digest provides a simple way to disable authentication without breaking the protocol, helpful in already secure and/or low-power environments.

- NFD deprecated support for "poking" data packets directly into the forwarder's content store. Therefore, the MemoryContentCache stores data packets in application memory and automatically uses them to reply to incoming interests. NDNrtc development provided feedback to refine the behavior and features of the MemoryContentCache.

NDN-CCL features motivated by network environments

- The BMS network environment includes low-power devices which may not be capable of producing computationally-intensive RSA signatures quickly enough. The libraries support elliptic-curve signatures (ECDSA) which can be produced more quickly, e.g. on a Raspberry PI running NDN-CPP. (We are adding ECDSA support for Python and JavaScript.)
- A low-power device in the BMS environment may not be able to run a local NFD configured with routes from other forwarders. We updated the client libraries to support "remote registration" where the application receives interests directly from a remote NFD.
- For devices in the BMS environment that only have a microcontroller, we developed the NDN-CPP Lite API, which is a light-weight C++ layer on top of underlying C code that does not rely on the C++ standard library or heavyweight classes. The Lite API combined with HMAC signature support allows an NDN application to fit in a few tens of kilobytes.
- For security in the BMS and mHealth applications, we adapted the ConfigPolicyManager from ndn-cxx for verifying a certificate chain of trust using NFD's policy configuration file format. This allows application writers to experiment with trust models at a higher level without needing to code the mechanics of fetching certificates, etc.

Usable security improvements in NDN-CCL

- Recently, browser vendors have implemented the *crypto.subtle* API which allows JavaScript code in the browser to use faster native cryptography. We updated the NDN-JS library to automatically use this when available so that signing can be 40 times faster than with pure JavaScript code.
- Libraries can now be installed with standard packaging systems: *easy_install* for PyNDN, *npm* for NDN-JS and *Maven* for jNDN.
- All libraries now have unit tests and integration tests. A Jenkins server does continuous integration by automatically running these tests whenever code is committed to the Git repository.

NDN-CLL is used by the following applications:

- **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)

2.5.3 A New Consumer-Producer API

Applications work with Application Data Units (ADU) – units of information represented in a most suitable form for each given use case [4]. For example, a video playback application typically handles video frames; multi-user game ADUs represent user status; and intelligent home application ADUs may be sensor readings. NDN enables applications to communicate using ADUs by exchanging interest/data packet with data

names. However the simple APIs provided by ndn-cxx and NDN-CCL for application development leave the application itself with responsibility to bridge the gap between handling ADUs and producing/fetching NDN packets: packet segmentation, queuing, reassembly, retransmission, error recovery, plus data signing and verification. This imbalance significantly increases application development time. To offer a programming abstraction that gives application developers flexibility and simplicity when working with ADUs, we have developed a higher level API, dubbed the *Consumer-Producer API*.

To understand this API, we first consider the *design patterns* that NDN introduces for applications. To make its data available, a producer process must make design choices ranging from name structure and security model to more basic things such as data segmentation. For a consumer to fetch data, it also must make decisions ranging from interest pipelining to data validation, in addition to conventional issues of packet losses and error corrections. In today's TCP/IP networks, the *socket* abstraction and associated protocols provide the API so that applications do not deal directly with raw IP packets. We believe we need an equivalent API and a set of supporting protocols for NDN networks.

Consumer / Producer communication abstractions

The TCP/IP socket API presents a container for data transfer parameters holding the current state of transmission in a virtual channel between processes running on two end hosts. Because a socket creates a duplex pipe for data to flow in both directions, server and client applications use sockets in more or less the same way with a few minor differences (e.g. *listen()* and *accept()* calls). A socket is only useful when attached to a communication channel (e.g. through *bind()* or *connect()*). To support "time asynchrony" or delay tolerance, application developers resort to higher level abstractions (e.g. ZeroMQ) suitable for queuing and passing messages.

In NDN, consumer and producer applications have different data transfer parameters, and warrant different programming abstractions. Producers care about Interest demultiplexing, ADU segmentation, securing Data packets and controlling data freshness. Consumers care about receiving all data segments, delivery reliability, data integrity and authenticity, and flow and congestion control.

Sending data over the Internet requires segmenting large ADUs to fit the network MTU. There are two major differences between how TCP/IP and NDN handle segmentation. First, TCP segment numbering conceals boundaries between ADUs, which can only be discovered after segment reassembly. In NDN, data names expose the ADU boundaries, therefore segmentation obeys ADU boundaries (Figure 2.8). A second, and related, difference is the degree of insight and control applications have during data transfer. In TCP/IP, if several ADUs are transmitted back-to-back and one segment is lost in transit, all subsequent ADUs are blocked (known as head-of-line (HOL) blocking). NDN applications (whether consumer or producer) produce and fetch each ADU independently, according to their own priorities. The Consumer-Producer API abstraction accommodates this need for flexibility.

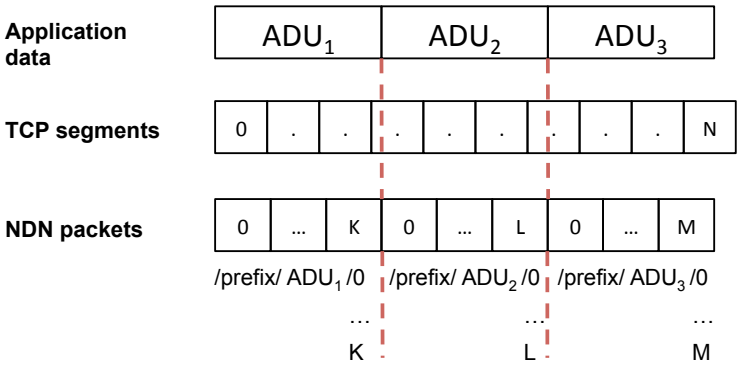


Figure 2.8: TCP/IP segmentation does not preserve boundaries of application data units (ADUs). NDN segmentation exposes these boundaries through data naming.

Table 2.1: Comparing four stages of data transfer in TCP/IP and NDN architecture

Function	**TCP/IP socket API**	**NDN Consumer/Producer API**
Creation and deletion	socket(),close()	consumer(), producer(), delete()
Configuration	setsockopt(),getsockopt()	setcontextopt(), getcontextopt()
Attaching to network	bind(), connect()	attach()
Data transfer	send(), receive()	consume(), produce(), nack()

Similar to TCP/IP's use of client- and server-side sockets, our API has *consumer and producer contexts* which are used for transferring ADUs. These contexts have lifecycles similar to sockets (see the four stages in Table 2.1). Unlike Internet sockets, which cannot operate if not attached to the network via *bind()* and *connect()*, NDN producer context can publish data to in-memory buffer or persistent storage (e.g. a Repo) even when the produce has no network connectivity.

2.5.4 Information maximization

UIUC developed a NDN-based prototype implementation of InfoMax and its *information maximization* data retrieval protocol, which allows consumers to request a representative sampling of data that matches a given query. InfoMax is designed for situations where the amount of data generated and stored exceeds application (or human) consumption requirements. This adaptive, receiver-driven data sub-sampling capability represents a new data transport paradigm. The current Internet architecture places this sort of sub-sampling burden on the information client. Information-centric networking changes the equation by facilitating the development of generic lossy transport that has the effect of sub-sampling data in a manner that achieves minimum information loss, while reducing the volume of information retrieval in a consumer-controlled fashion. This protocol may be of increasing importance to a growing range of data-centric applications.

InfoMax offers a primitive for requesting a data set specified by a name prefix that identifies the root of a content tree. The requesting node (the consumer) does not need to know what content is stored in the tree, nor its name space. It just wants the data under the tree at a configurable level of summarization (i.e., sub-sampling), which it controls by specifying how much information to pull from the named tree.

The novelty of InfoMax lies in the order that objects belonging to the requested tree are transferred. InfoMax assumes that objects that share a longer prefix within the tree have more semantic overlap. Hence, the marginal utility of sending an object that shares a longer prefix with a previously transmitted object is less than the marginal utility of sending an object that shares a shorter prefix. This differential leads to a shortest-shared-prefix-first transmission order for objects in the requested tree. We show that this order maximizes the marginal utility of transmitted objects and hence the total accumulated utility for a given transmitted data volume. InfoMax complements a previous transport layer abstraction, called the Information Funnel [12], which was geared for data collection by a single consumer from multiple producers. InfoMax is geared for the complementary case of data dissemination, where multiple consumers pull data from one producer, possibly at different levels of summarization. Together, the Information Funnel and InfoMax dissemination protocol may constitute two building blocks of many data-intensive services. The funnel will collect data into repositories from multiple sources, and InfoMax will serve it to multiple consumers at different degrees of detail. A paper based on this work has been accepted by ICCCN 2015.

2.6 New Architectural Findings from Network Environment Development

Work towards pilot applications for the NP network environments, along with other application development (including that of the new NFD forwarder itself), informed our architectural research, as we hoped. The team focused on several significant challenges that are purposefully inescapable in these network environments, such as publisher mobility support in Open mHealth and encryption-based access control in both environments. We derived practical designs for each area, and identified broader research considerations

for longer-term work. Each environment now has its own weekly conference call with a cross-cutting group of participants focused on the design and implementation of pilot applications. We classify the impacts of network environments and applications on NDN architectural design in the following list:

Hierarchical trust - Considering the requirements for both data verification and encryption-based access control in both network environments led to deeper consideration of how to express trust relationships in data namespaces themselves (for hierarchical trust relationships that follow the namespace) and a corresponding key namespace that parallels the data namespace (for granular expression of group-based access to parts of the data tree). While the basic mechanism for access control is hierarchical, an unresolved challenge is how to best support applications in creating and publishing appropriately granular keys to the right parties.

Cross-namespace trust mapping - For cases where access rights are delegated to entities in a namespace that is different from the data namespace, e.g., users in the BMS system who are organized in a different hierarchy than the building system itself, we developed an approach to mapping across namespaces by simply publishing appropriately signed certificates that delegate authority. This decision motivated continued discussion of LINK objects for the purpose of linking hierarchical trust relationships across domains. Both security and publisher mobility needs have motivated the need for LINK objects, and both sets of constraints helped clarify our approach to designing LINK objects as part of the protocol (see Section 4.1.3).

Alternate signature support - For more efficient signing required by the device-side of building management systems, we explored alternate signature strategies including HMAC-based authenticators and ECDSA, which we are incorporating into the NDN libraries and protocol specification.

Role of topology-independent namespaces - The Open mHealth environment finally drove us to solve the mobile publishing problem, i.e., allowing producers to move without changing namespace. For NDNFit, we developed an approach that uses topology-independent namespaces to name data uniquely and portably, but these names are not necessarily globally routed. By providing one level of indirection (e.g., via an NDNS lookup), requests in these namespaces can be forwarded to different providers or mobile publishers.

Support for publisher mobility - In addition to topology-independent namespaces, publisher mobility requires a data retrieval protocol that is robust to disruption and changing routable prefixes. We are exploring a synchronization-based protocol based on earlier work [14] that will address this need comprehensively.

Importance of application-accessible, temporary storage for the "publish and forget" app design pattern - To support low-capability publishers and simplify application development, many applications being developed have independently arrived at a design pattern in which new data is published to a repository or in-memory cache, which then serves requests asynchronously and without burden on the application. We likely need deeper library support for this capability, which sometimes has required an in-application pending interest table.

Protocol details and conventions - Application requirements have helped to clarify and sometimes resolve questions about conventions for segmenting, versioning, zero-length components and other open issues. Finding the appropriate balance of specificity ("one good convention") and expressiveness is a challenge and opportunity afforded by NDN's flexibility in user-specific naming.

Impact of strategy choice on applications - Different applications, especially those with low-latency requirements, may require different forwarding semantics, although there may be an effective default strategy for most situations. For example, to ensure low-latency data delivery and as a protective measure for data loss, a latency-sensitive consumer may re-express interests before they time out, at periods of O(100ms). The NFD default forwarding strategy initially suppressed these retransmitted interests, however the need from NDN-RTC development leads to a new BestRoute2 strategy in NFD which allows a consumer to re-express

pending interests up to every 150ms. A related issue is interest prioritization by applications; the NDN-RTC video conferencing application suggests the need for some type of one-hop priority (e.g., to prioritize audio fetching over video) for consumer-driven media playback.

References

[1] Michael Armbrust, Reynold S Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K Bradley, Xiangrui Meng, Tomer Kaftan, Michael J Franklin, Ali Ghodsi, et al. Spark sql: Relational data processing in spark.

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

[3] Connie Chen, David Haddad, Joshua Selsky, Julia E Hoffman, Richard L Kravitz, Deborah E Estrin, and Ida Sim. Making sense of mobile health data: An open architecture to improve individual-and population-level health. *Journal of medical Internet research*, 14(4), 2012.

[4] 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.

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

[6] 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.

[7] NDN Project Team. Ndn technical memo: Naming conventions. Technical Report NDN-0022, NDN, 2014.

[8] Ndn packet format specification. `http://named-data.net/doc/ndn-tlv/`.

[9] 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.

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

[11] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu, and Raghotham Murthy. Hive: a petabyte scale data warehouse using hadoop. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 996–1005. IEEE, 2010.

[12] Shiguang Wang, Tarek Abdelzaher, Santhosh Gajendran, Ajith Herga, Sachin Kulkarni, Shen Li, Hengchang Liu, Chethan Suresh, Abhishek Sreenath, Hongwei Wang, William Dron, Alice Leung, Ramesh Govindan, and John Hancock. The information funnel: Exploiting named data for information-maximizing data collection. In *10th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, May 2014.

[13] Reynold S. Xin, Josh Rosen, Matei Zaharia, Michael J. Franklin, Scott Shenker, and Ion Stoica. Shark: Sql and rich analytics at scale. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, 2013.

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

Chapter 3

Security

Contributors	
PIs	J. Alex Halderman (UMich), Jeff Burke, Van Jacobson, Lixia Zhang (UCLA), Christos Papadopoulos (CSU)
Grad Students	Steven DiBenedetto (CSU), James Kasten (UMich), Yingdi Yu (UCLA)
Undergrads	Michael Sweatt, Prashanth Swaminathan (UCLA)
Staff	**Postdocs:** Alexandar Afanasyev (UCLA)

Security is a major theme of NDN-NP. In particular, we are motivated to meet the specific but representative challenges posed by the network environments. The team now includes security-focused participants from the University of Michigan, who have come up to speed with NDN and are providing security expertise across the project. This chapter provides an overview of the security requirements derived for the network environments (Section 3.1), and work in a number of areas to support them, as well as other applications including the forwarding and routing components of NDN.

3.1 NDN Security and the Network Environments

To identify pressing issues in the NDN architecture and explore issues raised by both the environments and active work on NFD (the forwarding daemon) and NLSR (link state routing daemon), we held security-focused project meetings at UCLA in January and July 2014. These meetings helped familiarize participants with the security mindset, and allowed us to articulate challenges that drove our research and results this year. In February 2015, we held a project retreat structured around the security needs of the network environments. Since each environment presents a different threat model, they represent points in the broad set of possible future applications. By working within these threat models and pursuing solutions to address core security challenges such as confidentiality, integrity, and authentication, we are beginning to better understand NDN's security strengths while developing mechanisms that address unmet security needs. In developing the first pilot applications for these environments, we tried to focus on important, representative security requirements while keeping them implementable so that we can learn from practical coding, testbed deployment, and end-user experience of the pilot applications.

Both environments (mobile health and building management) require privilege authentication and data confidentiality / access control. The requirement for automated privilege authentication led us to invent a new concept of *trust schema*, which expresses a trust model in terms of a set of *trust rules* that map data names to the names of trusted signing keys (Section 3.2.) The confidentiality requirement led us to design a group-based encryption protocol, which allows data producers, rather than data hosts, to ultimately control access to data (Section 3.3.) Key distribution is critical to the network environments we have chosen, so we are designing a certificate system to issue and distribute certificates (Section 3.4), which requires a

rigorous definition of the certificate format (Section 3.5). Key verification performance concerns, such as those in low-latency and resource constrained applications found in the multimedia applications and EBAMS environment, led us to explore a key-bundling approach (Section 3.6) that wraps all information necessary to verify a key in one or more data packets and associates them with the key. Finally, since both environments involve archival data, the need to verify data with signatures using otherwise expired keys led us to design and implement a signature logger (Section 3.7). Before describing these results in detail, we elaborate on the security requirements of the network environments and their pilot applications.

3.1.1 Trust and Security in Open mHealth

We selected the Open mHealth network environment for NDN-NP in part because it has important confidentiality requirements easily understood by users but not easily implemented in practice. Through our explorations this year, we have come to articulate the challenge as follows: In this network environment, the person whose health is being monitored and assessed is the ultimate owner and controller of his/her data, and wishes to *selectively share* the minimum amount of data necessary to support an application they wish to run. The threat model includes both standard attack scenarios in consumer applications with sensitive data and the desire to limit leakage (and thus risk) of data to unauthorized systems. At the same time, the proposed environment is an ecosystem of composable services that operate on this data to provide value to users. The challenge is how to make it possible for independent application developers to easily enable the end user to authorize granular, selective sharing of their own data, and for that data to pass through a processing chain that may generate and store derivative data, which is also under the user's control.

Since confidentiality is the major security concern of Open mHealth, the security requirements of the NDNFit pilot application focus on data access control. We have three requirements: the owner must be able to control access to his/her own data; the app must be able to apply this access control at arbitrary level of data granularity; and the potential data requesters should not be restricted within a single application.

In Open mHealth today as implemented over IP, like many current applications, access to sensitive data is managed by the host of data, which encrypts the data using a session key negotiated between the data host and the data requester. The owner of data has to trust its data host to faithfully apply the access control policy. In NDNFit, the owner of sensitive data produces, encrypts, and signs it before transmitting it, as we expect will be the case for many NDN applications. These operations require marshaling of keys, the best practices for which are an open challenge. But as a result of this process, a data owner controls access to her own data through distribution of the decryption key. Unlike IP-based health applications where the data is likely visible to its host, in NDN applications like NDNFit, even data hosts may not be able to access the data without the owner's permission.

To enforce access control at different granularities, one must know the semantics and structure of data. In an IP network, data semantics and structure are defined by data hosts, which manage access to sensitive data. In NDN, applications can name data in a well-defined naming hierarchy, and data owners can use different encryption keys for data under different namespaces. One objective with the network environments is to explore how well the data namespaces can express the granularity of selective sharing desired in the applications. For example, in the Open mHealth namespace design (Figure 2.6(b)), the position of the timestamp in the hierarchy is driven not only by its role as the primary selection criteria for data retrieval, but also because it is a primary dimension for access control–e.g., a given application might be allowed to look at only the past day's worth of data and thus retrieve keys associated with that portion of the namespace.

In an IP network, the fact that data hosts perform access control implies that these hosts authenticate all potential data requesters, which leads to a problem if the host does not recognize an entity known to the data producer. NDN simplifies data access by supporting encryption of the data decrypt key using the requester's public key, therefore the requester authentication depends on only the data *owner*, rather than the data *hosts*.

Remaining challenges include creating a reasonable user experience for managing the keys, identities, and data access, and the challenge of "read audits" on NDN, while still leveraging in-network caching.

3.1.2 Trust and Security in Building Automation and Management Systems

We selected the Enterprise Building Automation and Management Systems environment for NDN-NP in part because the challenge of securing critical infrastructure, such as industrial control systems, has become a national priority over the last decade that we believe is not well-met by TCP/IP-based solutions and channel-based security more generally. In particular, the increasing desire for ubiquitous monitoring and control of critical systems that is integrated with web access and traditional applications has proved almost impossible to meet securely by methods typically relied on by the industrial control community, such as physical or logical isolation (air gaps, VLANs, etc.). The challenge that we have focused on is how to provide easily configurable, campus-scale access control to data that is named through its relationship to the physical world, by individuals who are organized according to a different, administrative hierarchy. The threat model is organized around standard attack objectives and approaches for industrial controls, but in the context of their integration with more traditional IT and internet-based systems, which we expect NDN to support better than IP. We believe it is possible to do this without interactive security services, but rather a set of key publishing and naming conventions using NDN as a certificate store. Further, we began to support NDN security functionality, in particular content signing and authenticated control, on resource-constrained platforms such as the Arduino.

EBAMS shares with Open mHealth some security requirements, such as the data access control. BAMS also requires users in the same access control group share the same access privilege. EBAMS must support authentication for control and actuation commands with low-latency, challenges we explored in the first NDN project [7]. In exploring the NDN-BMS pilot application, we have articulated a two-part definition of groups: 1) *primary groups* emerge directly from the data namespace, for example a certain building in the BMS namespace, and can be expressed directly in terms of that data namespace; 2) *secondary groups*, which are organized in another namespace (e.g., a hierarchy of personnel), whose access maps to one or more primary groups through records stored in the namespace.

This NDN group-based encryption scheme distributes the group's decryption key to group members. The group's decryption key is named after the group name and is encrypted using each member's public key. Granting access to data is simplified as encrypting the data using the group's encryption key. We are in the process of finalizing the design, library support, and implementation, which we will describe below. We believe a significant simplification over IP is emerging if we can solve the key distribution problem. In an IP network, since access control is performed by end hosts, group-based access control requires the end host to maintain a roster for each group and to associate the group with certain access privilege. In a distributed system, this implies *all* the end hosts must manage the group configuration correctly, otherwise, an attacker can by-pass the access control through a misconfigured end host.

The above approach covers data access in the NDN-BMS pilot. We are exploring how to use the same namespace and same type of trust model to authenticate control and actuation commands. Control command authentication requires checking a command (e.g., sent as a remote procedure call expressed in an Interest) against its issuer's privilege. Therefore it requires a mechanism to authenticate a user's privilege, and map a control command to its corresponding privilege. In NDN, both control commands and privileges are explicitly expressed as names. This year, we defined a certificate format that can support associating a command issuer's key with its control privilege. The new certificate format is just a data packet with a key as its content and the privilege as its name. We also defined a trust schema to explicitly specify the mapping between control commands and privileges.

Some significant challenges remain, including secure key distribution (partially addressed below) and for our pilot application, the integration of web browsers (for data access, device configuration, and other purposes) into the industrial control scenario of EBAMS. While we have many browser-based NDN experiments in this and other application, we have not comprehensively considered the overall security environment, which involves a host operating system, browser process, sandboxed browser tab environment, and one or more remote data consumers and publishers. We plan to tackle this complex environment during next year, motivated by how EBAMS devices and applications often use embedded web servers and web applications to support both configuration and everyday applications.

3.1.3 Trust and Security Mobile Multimedia

The security of media applications is motivated from many directions: personal privacy, digital rights management, and even regulatory requirements. Our initial pilot application, real-time videoconferencing, has more straightforward security requirements than the above two network environments. But as we extend low-latency content distribution into other domains, such as vehicular networking or contested environments, we expect more significant challenges to emerge.

The NDN-RTC application generates and consumes a few different types of data: media streams and related metadata per participant, chat text, conference participant lists and conference discovery/rendezvous information. The application requires that individual data chunks of each type can be verified as from their claimed senders even if they are not encrypted, although we expect that most media traffic will be encrypted. As a peer-to-peer system, NDN-RTC uses a "web-of-trust" model for managing user relationships based on past work by the NDN team in endorsement-based key management for chat applications [8]. This model contrasts with the hierarchical trust models of the two other network environments.

While the security work for NDN-RTC is still in early stages, over the course of this year, we have clarified the following challenges based on the project's implementation and testing. First, low-latency (and low CPU overhead) for each stream's signing and encryption is critical, which led us toward symmetric key operations and perhaps even HMAC authenticators, as currently being explored for resource constrained IoT applications in the BMS network environment. Second, we will support certificates that are either testbed-signed (from a hierarchical identity system created for the testbed) or self-signed, and we will leverage lessons learned from our *Let's Encrypt* project (Section 3.4) to make it easy to create and exchange certificates. Third, we need to handle revocation of a given person's access to an encrypted media stream while others can continue to view it without interruption (i.e., removing someone from a videoconference so they can't silently decrypt a conversation they've officially "left"). We may be able to use pre-existing broadcast encryption techniques and/or other techniques like pre-warnings to other participants' applications that new encryption keys are being used, allowing them time to fetch the new keys.

3.2 Expressing and Evaluating Trust in Applications

One threat model of the network environments is to compromise data or control command interests, for which data and signed interest authentication is critical. Data (or command interest) authentication requires a well-defined trust model that is adopted by both data producer (or signed interest issuer) and data consumer (or signed interest receiver).

The most fundamental security advantage of NDN is that it incorporates content signing into the core network architecture. However, thus far, the question of how NDN applications will actually sign and verify content, and how they will evaluate application-specific rules for managing trust, has been left to individual application developers to resolve. That is, developers must explicitly handle initialization of keys and certificates for their applications, and explicitly specify the signing key for each data packet published by the application. Further, developers must handle packet verification including checking the privilege of a signing key, fetching the signing key, and verifying the validity of the signing key. Such a burden on the developers is similar to re-inventing higher-level protocols like TLS or SSH for every application. Experience with network security outside the NDN project also suggests that it is error-prone to require application developers to implement trust model evaluation, key exchange, and other security tasks[1].

Before the NDN-NP project, the NDN libraries provided support for data signing and verification operations, but lacks support for trust model or key exchange. However, the network environments make such operations unavoidable. For example, Open mHealth has user-centric confidentiality (data access control managed by end-users) at its core; it is not an optional aspect that one can graft on later. To meet needs of the NDN-NP environments, over the past year we have made significant progresses toward appropriate

[1]For example, the existing HTTPS certificate system prescribes a single trust model for web applications. This model is a poor fit for many applications, such as those mobile apps that communicate with only a single server, thus developers often work around the trust model. Unfortunately, implementations that do this often fail to perform certificate chain validation properly, which leaves users vulnerable to man-in-the-middle attacks.

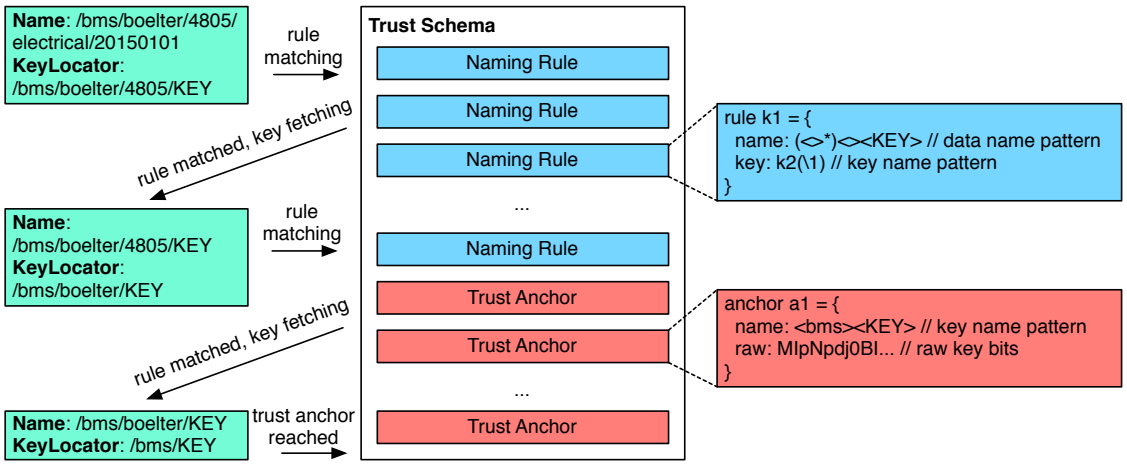


Figure 3.1: An example of trust schema, based on the EBAMS network environment.

higher-level security support in the NDN libraries.

We identified and specified a set of trust models that suit the needs of the chosen environments (EBAMS, MHealth, NDNrtc) and supporting network modules (NFD, NLSR), that we can implement in a set of configurable security libraries. We believe it is possible in many cases, and in particular for the network environments we have chosen, to enable application developers to configure one of these trust models for their applications' namespace and security relationships, and then let a lower level security library perform signing and verification according to that trust model. For example, if data producers and consumers use the same trust model, then data consumers can verify the producer's data. Therefore, for application developers the challenge of data authentication integrity is simplified to the task of selecting an appropriate trust model for their application, and configuring the mapping of application namespace(s) to this model.

The core of a trust model is the privilege of keys, i.e., mapping keys to their corresponding data packets. Since everything in NDN is named, our approach is to regulate these privileges by defining the rules between data names and signing key names. For example, given a data name, we can define the rules to derive the name of the trusted signing keys; or given a key name, we can define the rules to derive one or more trusted name spaces for published data. With a set of rules defined in a trust model, a data consumer can then construct a chain of certificates that can derive the trust of the data producer from a common trust anchor, which is also specified as a part of the trust model.

But these rules must be expressed in a form that is easy for security experts to use as templates, for application authors to configure, and for lower level libraries to interpret. We call a set of rules that express a trust model a *trust schema* (see Figure 3.1 for an example), and we are working the development of an expressive syntax for defining trust schemas.

3.3 Group-based Encryption to Support Flexible Access Control

In NDN, sensitive data is protected with a producer-generated encryption key, thus we need to translate access control policies into appropriate key distribution behavior to enable data decryption consistent with these policies. Taking the EBAMS as an example, one or more data producers (such as sensors or monitors in a building) may publish data under their corresponding namespaces, in a time sequence and encrypted. Access to encrypted data can be granted to qualified consumer groups (such as a group of facility staff, or a group of building occupants), where members share the same key and access privileges. To support this scenario, we designed a prototype group-based encryption protocol that involves interactions of three entities: the data producer, privilege manager, and consumer group. The data producer generates symmetric keys to

encrypt its own data and delegates the access control of its data to its privilege manager, by encrypting its symmetric key using the privilege manager's public key. Such delegation accommodates deployment scenarios where data producers may be small devices, such as embedded sensors, that may not be powerful enough to support complex access control. Such delegation also makes it feasible to manage the access control at finer granularities. The privilege manager can grant access by distributing its private key specifically for the given application to authorized consumer groups. This level of indirection separates consumer group management from privilege delegation; changes to consumer group membership do not affect privilege delegation between the data producer and privilege manager.

We are still evaluating the requirements of the Open mHealth network environment, but we believe a variation of this approach may be applicable for the NDNFit pilot application. Next year, we will implement and evaluate this protocol within the context of the two network environments.

3.4 Certificate Issuance and Distribution

NDN benefits from the property that all data is cryptographically signed. However, in most application scenarios, the classic PKI problems of key management and certificate distribution must be overcome. For prototype purposes, NDN has implemented a web interface, NDNCERT [2], which allows users to submit certificate signing requests and receive certificates for use within the NDN testbed. This solves the problem of certificate distribution for users following current industry practices (i.e., in a similar manner to that used by most browser-trusted HTTPS certificate authorities), but it remains a manual process. Given the potential adoption and scale of key management for NDN, usability and cost effectiveness are of primary importance. The NDN participants from University of Michigan have been working on new PKI concepts that can dramatically simplify certificate acquisition, issuance, and deployment, with applications both to NDN and to the existing global HTTPS PKI.

Existing web PKI certificate authorities tend to use a set of ad hoc protocols for certificate issuance and identity verification. A typical user experience involves making a payment to a chosen certificate agent (CA), generating a PKCS#10 certificate signing request (CSR), cutting and pasting the CSR into a CA web page, proving ownership of the domain through an email- or web-based validation method, downloading the issued certificate, and configuring web server software to use it. Most of these steps are accomplished by getting the human user to follow interactive natural-language instructions from the CA rather than by machine-implemented published protocols. These steps are both error-prone [6] and time consuming[2]. The manual nature of this process presents an obstacle to the wide deployment of HTTPS and to applications of NDN, since it inhibits mechanization of certificate management.

To overcome these problems, we are developing an extensible framework for automating certificate issuance and validation, the Automated Certificate Management Environment (ACME) protocol [5]. ACME aims to abstract away and automate the process of obtaining, authorizing, and maintaining certificates from system administrators and application developers. A lightweight ACME client can prove authority over an identifier with an ACME server and automatically install the associated certificate with little to no interaction. ACME will allow servers, routers, and infrastructural software to obtain certificates without user interaction (Figure 3.2). Use of this protocol can radically simplify the deployment of HTTPS and the practicality of PKI-based NDN applications.

(In a parallel project, UMich NDN participants are working with collaborators at EFF and Mozilla to create an HTTPS certificate authority that speaks ACME. This service, called Let's Encrypt, will begin providing zero-cost certificates to the public in the summer of 2015. Experience with Let's Encrypt will help ACME develop into a robust, field-tested protocol, and we plan to propose it for IETF standardization within the year.)

In general the ACME client desiring signed authority over an identifier or attribute will send a challenge request to the ACME server. The ACME server issues challenges whose successful completion would prove ownership over the desired identity. The client sends responses to the challenges and the server verifies them. If there is no error, the client sends a certificate signing request for the identifier(s) to the server and the

[2]Usability tests indicate that webmasters often need 1–3 hours to obtain and install a certificate.

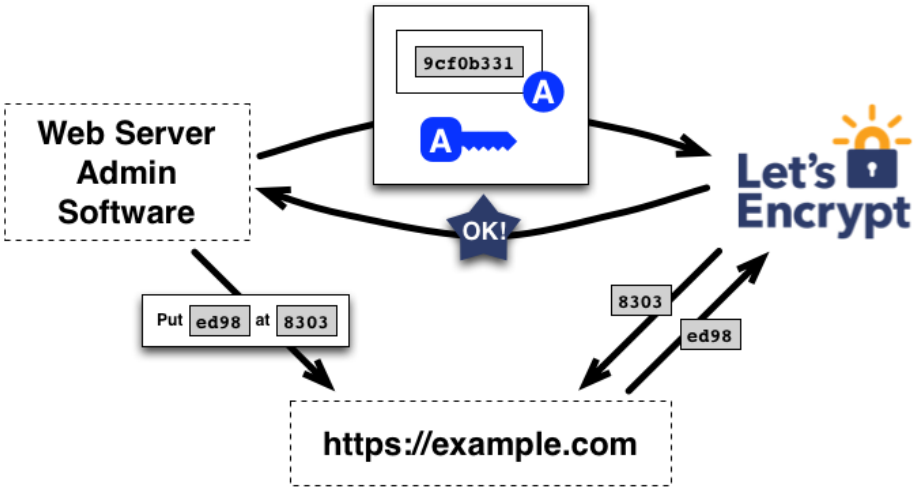


Figure 3.2: ACME validation example for a traditional website.

server responds with a certificate. In ACME, the private keys never leave the client and all of the steps happen seamlessly.

Over the next year, we will work to implement ACME for NDN by developing challenges types that convey the appropriate authority over identifiers within NDN. We are investigating several potential challenge types. One way to implement ACME challenges in NDN is with scoped interests, which confirm that the requesting device (client) is within a particular network location. This approach would be applicable to the BMS environment where devices can authenticate themselves to commanding or authoritative servers. Another general class of challenges is bearer tokens, which allow for authorization over general resources, without requiring the authoritative entity to generate the private keys for its clients. Open mHealth's mobile components might use SMS messaging as an out-of-band approach for exchanging these.

Challenges may also be devised for more than names within NDN. For instance, the group-based encryption protocol attributes can also be authenticated within ACME and automatically applied to clients. The framework that ACME provides is general enough to handle all types of identifiers and privileged resources. The application of ACME within NDN will greatly simplify key management and distribution, fulfilling the trusted computing promise of NDN.

3.5 New Certificate Format

Another key area of work this year related to the mechanics of certificates within the NDN system. Both Open mHealth and EBAMS, as well as other NDN applications, require a mechanism to authenticate a user's identity or privilege. Identities and privileges are expressed as names in NDN, so binding the key to an identity or privilege naturally results in an NDN data packet having the key as its content.

In the first phase of the NDN project, we defined the original certificate format and accumulated experience with certificate management. Over the last year we developed a new version of the NDN certificate format, informed by our experience deploying NLSR and NFD, and by the needs of the new application environments. The new NDN certificate format simplifies the assumptions of certificate management to facilitate broader use of certificates. It also introduces extensions to support more certificate management operations, such as certificate revocation, which were infeasible in the old version. These changes provide improved support for deployment flexibility, help maintain trust in long-lived content (e.g., archival content in both network environments), and lay foundations for trust agility.

The first version of an NDN certificate was designed as a type of NDNS resource record[3]; certificate are

[3]NDNS [3] is a DNS-like name lookup system for NDN (see Section 4.3).

published to, and retrieved from, NDNS. As the diversity of NDN applications grows, however, it is no longer feasible to assume that every node can always access certificates through NDNS. Applications will need to access certificates from anywhere they are stored, including NDNS.

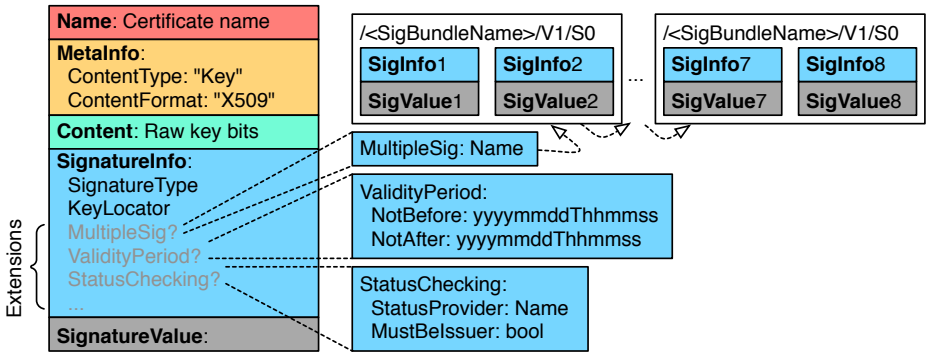


Figure 3.3: New certificate format

The new version of the NDN certificate makes certificates independent from where they are stored. This change is reflected in the certificate naming convention, which now only describes the key subject, the key itself, and the certificate version. With this convention, certificates can be served as normal data from anywhere and by any application. For local communications, an end node can serve its own certificates directly from a local certificate publishing service (e.g., the Public key Info Base [1]) to whoever is requesting them. When one's certificates are stored elsewhere, a forwarding hint can help direct certificate requests in the right direction [4] (see Section 4.4.4). How to obtain a forwarding hint is up to applications; it may or may not require NDNS.

The second major change to the certificate format is support for new extensions in the SignatureInfo field. The old NDN SignatureInfo contained only the signature type and a pointer to the signing key (KeyLocator). This information would be sufficient for a normal data packet, but not when the data packet is used as a public key certificate. For example, there is no validity period information about the signature, so a data consumer cannot tell whether the certificate has expired or been revoked.

To allow SignatureInfo to carry more information, we introduced extension fields in SignatureInfo. Extension fields can also handle other additional information. We defined the StatusChecking extension to contain information about how frequently the signature status will be updated, and who can be trusted to assert the signature status. An NDN security library that supports this extension can convert this information to an Interest for the signature status, and the response to which will carry the signature status and the revocation reason if the signature is revoked. The validity period of a signature is also defined as an extension, ValidityPeriod, which contains the timestamp when the signature takes effect and the timestamp when the signature expires.

We also defined a third extension, MultipleSignature, to facilitate collecting multiple signatures for the same public key. This extension can support the Web-of-Trust model, in which a user's public key may be asserted by other users, thus facilitating publications of content with trust chains tracing to multiple roots of trust. (We expect to use this mechanisms to support WoT trust model in group-based applications such as ChronoChat and NDN-RTC.) With the first NDN certificate format, each packet could carry only one signature, thus collecting multiple signatures required a data consumer to actively discover all existing signatures and to send a separate interest to fetch each signature. The whole process could take a long time and make the Web-of-Trust unusable. With the MultipleSignature extension, a public key owner can prepare a signature bundle of its public key (described below), and put the name of the signature bundle into the MultipleSignature extension, so that data consumers can efficiently retrieve all in a short time.

3.6 Key Bundling

To facilitate the authentication of every NDN data packet, the NDN data packet format design specifies both a signature that binds the data name to the content, and a *KeyLocator* field to carry the signing key name. With that name, a data consumer can fetch the signing key to verify the signature. Since the signing key is returned as the content in a data packet which is signed and contains its own KeyLocator, completing the validation requires the consumer to recursively fetch and verify keys until it reaches a trust anchor.

Such an iterative certificate fetching process can introduce high latency, since a data consumer must receive a certificate before fetching the signing key of the certificate. The time to verify the first data packet is proportional to the number of certificates in the chain of trust, which hinders performance and efficiency of applications. Furthermore, the actual signature verification starts only after the trust anchor is reached. If a data consumer fetches an invalid intermediate certificate, then all subsequent certificates fetched based on the invalid intermediate certificate become useless, which may expose the consumer to a denial-of-service attack. These disadvantages motivated our search for a more efficient data authentication mechanism.

We have developed a new certificate-fetching model in which the data producer or provider collects all intermediate certificates necessary to authenticate the data packet. This model is based on two rationales. First, it is to the interest of a data producer or provider to help data consumers validate the data. Second, since a producer's data may be consumed by many data consumers, it is more efficient for the data producer to collect all related certificates at once and share with all data consumers, rather than leaving to individual consumers to perform the collection.

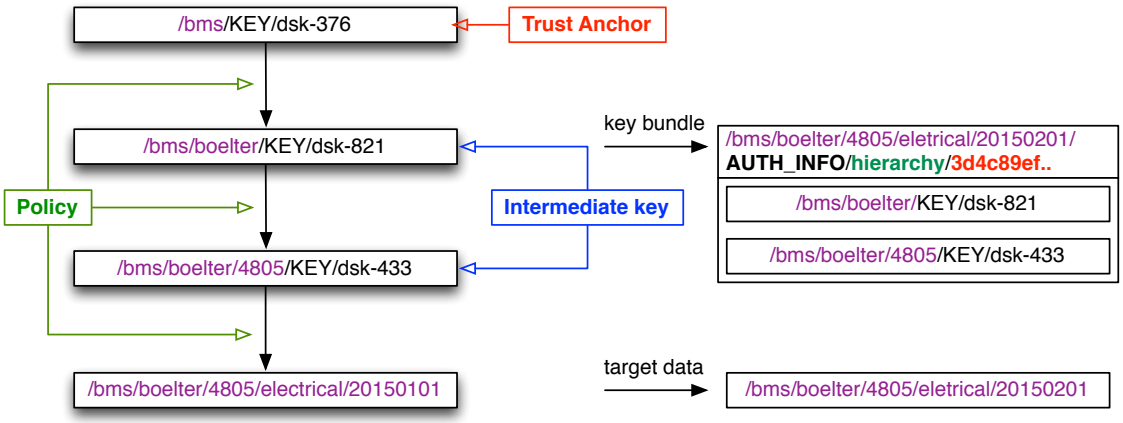


Figure 3.4: An example of key bundle in the NDN-BMS namespace.

We designed a new type of NDN data packet called a *key bundle* to enable the new certificate fetching model. Given a data packet, its key bundle contains all certificates that can construct a trust chain from a particular trust anchor to the data's signing key. The name of a key bundle consists of two parts: the name of its associated data packet, and the digest of the corresponding trust anchor. Data producers or providers can publish each data packet along with its key bundle. By requesting a key bundle for the signing key of a data packet, data consumers can retrieve all certificates needed at once to authenticate the data packet.

3.7 Signature Logger

The security model in NDN is fundamentally different from TCP/IP's traditional channel-based models such as TLS. TLS security equates to securing a communication channel between two endpoints. Since all data received via this secure channel is considered valid, the data validity only depends on whether the certificate of an endpoint is valid at the time when the channel is established.

In NDN, data production and consumption may not, and often do not, happen at the same time, which leads to the possibility that the lifetime of a data packet may be longer than the lifetime of the signatures in its trust chain. Such scenarios are particularly likely in the network environments we have chosen: a person's life-long health records in Open mHealth easily outlive their signatures, and the same can be try for archived building management data in EBAMS. We worked over the last year to understand the implications of this new property brought up by NDN's security model (securing the data directly instead of its containers), and to develop effective solutions.

A naive solution to maintaining long-lived data could be re-signing the data when its signature expires. However, this solution requires continuous active involvement of the publisher, which can be expensive and may decrease availability of the data. An alternative is to securely bind a timestamp to the signature when it is generated, so that even after the signature has expired, verifiers can still check whether the data's signing key was valid when the data was produced.

We have sketched out the design of a secure timestamping system called a *Signature Logger*. For long lived data, the producer can insert the data into the Signature Logger. The response of the Signature Logger contains a timestamp and a secure hash, through which the Signature Logger asserts that the data signature is generated before the timestamp. The data packet and its timestamp can then be served by any data provider, which may or may not be the original data producer. When validating the data packet, a data consumer can request a proof of existence for the signature timestamp from the Signature Logger and check if the signature is valid at the time the data was produced.

We have prototyped a simple implementation of the Signature Logger, however it has several limitations. First, the design assumes that the logger is trustworthy, i.e., it always presents the true information to everyone. Moreover, the current implementation only supports a single trust model. In other word, only data that is signed within a particular trust model can be recorded in the logger. Over the next year we plan to address the first issue by designing a gossip protocol that enables users to audit the behavior of the logger, so that we can deter the logger from presenting different answers to different users, and to address the second issue by extending the current implementation to support multiple trust models.

References

[1] Public-key information service. `http://redmine.named-data.net/projects/ndn-cxx/wiki/PublicKey_Info_Base`.

[2] Alexander Afanasyev. NDNCERT: User public key certification system for NDN Testbed. 1st NDN Community Meeting (NDNcomm), September.

[3] Alexander Afanasyev. *Addressing Operational Challenges in Named Data Networking Through NDNS Distributed Database*. PhD thesis, UCLA, September 2013.

[4] 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.

[5] 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.

[6] Zakir Durumeric, James Kasten, Michael Bailey, and J. Alex Halderman. Analysis of the HTTPS certificate ecosystem. In *Proc. 13th ACM Internet Measurement Conference (IMC)*, October 2013.

[7] Wentao Shang, Quihan Ding, Alessandro Marianantoni, Jeff Burke, and Lixia Zhang. Securing building management systems using named data networking. *IEEE Network*, 28(3):50–56, 2014.

[8] Yingdi Yu, Alexander Afanasyev, Zhenkai Zhu, , and Lixia Zhang. An endorsement-based key management system for decentralized ndn chat application. Technical Report NDN-0023, Revision 1, NDN, July 2014. `http://named-data.net/publications/techreports/ndn-tr-23-chronochat-security/`.

Chapter 4

Networking: Routing and Forwarding

Contributors	
PIs	Beichuan Zhang (Arizona), Van Jacobson & Lixia Zhang (UCLA), Lan Wang (Memphis), Christos Papadopoulos (Colorad State University), Patrick Crowley (Washington University)
Grad Students	Junxiao Shi, Yifeng Li, Yi Huang, Teng Liang (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, Vince S. Lehman (5/2014 – 11/2014) (U. Memphis)
Staff	Vince S. Lehman (11/2014 – present) (Memphis), John DeHart, Jyoti Parwatikar (Washington University)
	Postdoc: Alex Afanasyev (UCLA); Syed Obaid Amin (U. Memphis)

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 chosen environments. This chapter reports on the following contributions in this area:

- Documented a set of guidelines for the NDN packet format design.
- Extended packet format with new content types (Link, NACK, and Encapsulation).
- Extensive development of a free and open-source NDN forwarder (NFD), which is in use by the entire project and deployed on the testbed.
- Design and implementation of a general namespace mapping system called NDNS.
- Design of a namespace mapping solution to scale global routing and support mobility.
- Performance evaluation and improvements to name-based link-state routing and hyperbolic routing approaches.
- Design, implementation and performance evaluation of a binary-search based name prefix lookup algorithm, which can achieve 10Gbps throughput with one billion synthetic name prefixes.
- Development of the iSync protocol to efficiently synchronize namespaces between neighbor nodes.

4.1 NDN Packet Format

The NDN project started in 2010 with a packet format originally defined as part of the CCNx project (`http://www.ccnx.org/releases/latest/doc/technical/index.html`). As our understanding of the NDN architecture's capabilities and needs deepened through experimentation over the last few years, and based on inputs from the broader NDN research community, we transitioned from CCNx's binary XML encoding to

NDN Type-Length-Value (TLV) encoding. We also made a number of important changes to various fields in the packet format. The current interest and Data packet fields are illustrated in Figure 4.1. In July 2014 we released an initial draft of naming conventions as an NDN technical memo [32] to start the discussion of an initial standardization for common use cases, e.g., segmentation, versioning, time stamping, and sequencing. In early 2015, we extended the packet format with new data types to support failure handling, mobility, and routing scalability.

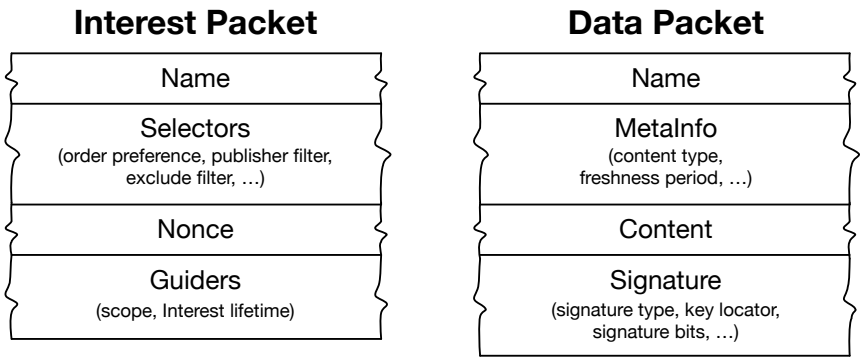


Figure 4.1: NDN Interest and Data packet structure

4.1.1 Design Requirements

Several iterations on the packet format design have taught us the challenge of getting it right to meet the needs of both the current and emerging technologies and applications. Inspired by the original Internet design requirements [14], we articulated the design requirements for the NDN packet format [7] to help guide design decisions and to avoid premature constraints. When requirements conflict with each other, prioritizing them determines the outcome of the packet format design. We prioritized our design requirements in the following order:

1. **Universality (Elasticity)**

   As the new narrow waist of the Internet architecture, NDN must support a diversity of usage scenarios and underlying network technologies, from constrained IoT environments [5] to ultra high-speed network channels [24]. Such diversity has convinced us that the fixed-length header of the IP architecture, despite its obvious advantages of implementation simplicity, is inappropriate for NDN. Instead the packet format should allow individual applications to choose a packet size appropriate for the intended deployment environment, which may range from a few tens of bytes in constrained IoT environments to jumbo frames to take advantage of future ultra high speed links. One way to provide this flexibility is to make the packet format, including individual header fields, use a variable-length encoding scheme.

2. **Extensibility**

   Given the experimental nature of the NDN architectural research, its packet format should stay as flexible and extensible as possible, avoiding premature standardization due to performance optimization. We want to make it easy to add or remove elements as our understanding of the architecture grows, while minimizing the number of required fields. Drawn from the rich lessons cumulated over 30+ years of Internet protocol designs, we have chosen a type-length-value (TLV) encoding format, which supports extensions through creation of new TLV blocks, and allows old TLV blocks to be deprecated over time without a flag day[1]. A TLV format also allows for encoding of actions to be taken by those

[1]BGP [20] is an example of an Internet protocol that, after rapid version changes in its first few years, adopted (in BGP4) a TLV encoding format which stopped the version change

nodes who do not recognize a newly defined TVL block[2], a feature that facilitates changes.

3. **Processing Efficiency**

   Optimizing packet processing efficiency is always desirable, but often conflicts with the elasticity and extensibility requirements, e.g., variable length packet headers may require more processing than fixed length headers. In considering the tradeoff, we must not constrain our view based on current processing capabilities and implementation choices, both of which can evolve quickly in response to market demands, investment, innovations, and new discoveries[3]. Thus we put efficiency after elasticity and extensibility[4].

Our 5-year experience with the NDN packet format design also taught us a number of specific approaches to packet processing optimization without sacrificing other design goals, which we summarized in [7].

4.1.2 Separating Information-Centric and Multi-Hop Forwarding Elements

NDN network elements communicate using application data units, thus the primary function of the NDN packet format is to include all information necessary to describe the requested data and to secure the data, such as data name, data name constraints, payload, security context, security context constraint, etc.. A separate function of the NDN packet format is to aid data retrieval over multi-hop environment. For example, the interest lifetime identifies obsolete data requests; the nonce field carried in an interest helps detect duplicate or lopping interests; QoS labels can support prioritized data retrieval; NACKs signal problems in interest forwarding so that routers can try alternative paths; and information about packet fragmentation can help reassembly of packets at the next hop. The above elements may change hop-by-hop.

Therefore there exist two distinct classes of protocol elements: one type is information-centric, used in retrieving the desired data; the other is network-centric, used to aid hop-by-hop (HBH) packet forwarding but is not directly related to the data itself. The information-centric protocol elements must be implemented in all consumers and producers; the need for the network-centric elements depends on specific forwarding scenarios and environments.

A fundamental engineering design question is what elements should be considered as part of the the narrow waist in NDN network architecture. Different ways of encoding information-centric and HBH elements in the packet format define tradeoffs in implementation, usability, and protocol extensibility. Combining the information-centric and multi-hop forwarding elements into a compact format can lead to simpler implementations and faster processing for the time being, but may lead to inclusion of unnecessary elements in the protocol. For example, when data packets are cached in the network and retrieved later by different consumers, most if not all elements of HBH information are no longer meaningful. We believe that encoding the two classes of protocol elements separately can facilitate the evolution of the base format, and the HBH elements can be carried via a shim layer, which we call the NDN Link Layer. We are still analyzing tradeoffs and considering design alternatives; implementation and test deployment will inform our eventual choice.

4.1.3 New Content Types

NDN has two types of packets: Interest and Data. Applications provide their own header to identify the type of content and other application-specific information. If a particular feature or function repeatedly shows up in multiple applications, we may consider codifying this feature into network packet header and supporting libraries. For example, since many applications will use (and retrieve over the network) public keys and certificates, we have defined a subtype of Data payload called a *Key* object, for public keys and

[2]Optional TLVs in the IPv6 extension header [15] reserve the first two bits of the type to specify the behavior of routers that do not recognize the TLV; SCTP [45] also reserves two bits to indicate the desired processing for packets with unrecognized chunks.

[3]For example, Internet's switch to classless inter-domain routing (CIDR) in mid-90's challenged the then-best line speed router implementations, but new lookup algorithms and implementations were quickly invented to meet the needs.

[4]We recognize the desire from industrial players to commercialize and monetize NDN technologies on a 2-3 year time horizon, they may set different priorities, especially if they monetize in hardware.

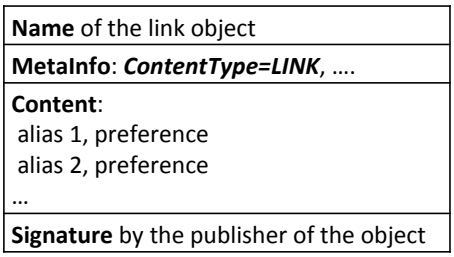


Figure 4.2: LINK Object

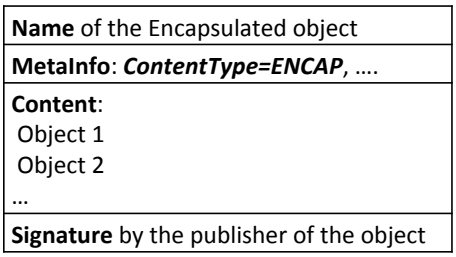


Figure 4.3: Encap Object

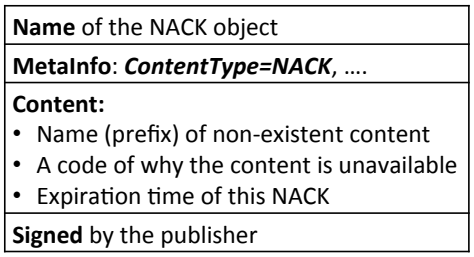


Figure 4.4: NACK Object

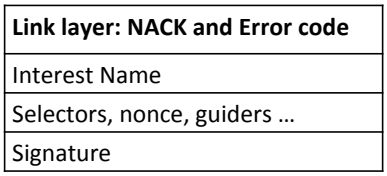


Figure 4.5: NACK packet

certificates. This approach allows third parties to handle packets carrying key objects separately (if needed) from general data packets.

Our experience with protocol and application development has revealed the need for additional types of network objects. This year we have defined three new object types:

- **LINK object**: A content object is usually published under the problisher's name (e.g., /net/ndnsim), with a signature by the publisher's key to bind the name and the content. But consumers may need to retrieve this content through another namespace, such as the namespace of an ISP hosting the content, e.g., /att/user/alex/net/ndnsim. The knowledge that these two namespaces point to the same content can be published as a LINK object as shown in Figure 4.2. A LINK object is a Data packet whose ContentType is LINK. Its name contains a special marker LINK, e.g., /net/ndnsim/LINK, and its payload contains a list of names or name prefixes that point to the same content as /net/ndnsim. Each name prefix has a preference value, which can be used by the the consumer to choose among the multiple choices. LINK object has semantics similar to that of UNIX file system's symbolic link. It allows flexible, alternative naming structures without republishing (i.e., re-signing) the data under each name. We use LINKs in our name mapping service NDNS (Section 4.3) as part of our current solution to scale global routing (Section 4.4).
- **Encap object**: this new type allows encapsulation of one or multiple content objects under a different name (Figure 4.3), a useful mechanism as we learned from working with the existing Internet protocols. Each enclosed object is a complete packet on its own, including the signature of the enclosed object. To minimize signature computation, the signature of the Encap object covers the name of the Encap object and the signatures of all the enclosed objects; it does not cover the content of each enclosed object. Producers can use Encap objects to return content whose original name does not match the requested content, or to return multiple content objects at once.
- **Negative Acknowledgment (NACK) object**: In the original design, when the producer receives an interest but does not (or not yet) have the content, the producer simply drops the interest. The consumer may re-try multiple times and give up. The NACK object enables a producer to send an explicit response indicating that the requested data is unavailable, so that the consumer learns this information quickly and accurately. This NACK object (Figure 4.4) is a Data packet. Like any other Data packet, it is signed, fills a PIT entry in routers, and can be cached in content stores. Its payload contains the name whose content is not available, a code that explains the reason, and the expiration time of this NACK to notify the consumer when it may retry. Multiple application development efforts

have made use of the NACK objects; more experimentations will inform our design choices of what information a NACK packet should include.

4.1.4 Interest NACK

Similar to an NACK object that informs consumers of data unavailability at the *application* level, we introduced an *Interest NACK* to inform a router of its upstream neighbor's inability to forward an Interest packet at the *network* layer. In the original NDN design, routers discover failures by interest timeout only. More specifically, when a router N forwards an Interest I, it starts a timer based on the estimated RTT to the data producer. If the corresponding Data packet comes back before the timer expires, N updates the RTT value and forwards the data back to consumer; otherwise N may try alternative paths to forward I, or give up. However, this timer-based failure detection is slow. Furthermore, when N simply gives up by discarding I, this unsatisfied Interest I, which we call *dangling state*, remains in the PIT of routers between the consumer and N until I's lifetime expires. Such state can block other consumers from getting the same data, since routers holding the dangling state believe that they have already forwarded I, and thus simply wait for the Data to return.

Interest NACK is introduced to address the above problem. When an NDN node can neither satisfy nor further forward an Interest, it sends an Interest NACK back to the downstream node. If this downstream node has exhausted all its forwarding options, it will send an interest NACK further downstream. An interest NACK carries the original Interest as the payload, and an error code explaining why the interest cannot be satisfied. Current error codes include *duplicate*, *congestion*, and *no route*. The packet format is sketched in Figure 4.5.

Interest NACKs bring two benefits to the system: removing dangling interest state faster than waiting for a timeout, and allowing downstream nodes to learn the specific cause of a NACK and take informed recovery actions. An Interest NACK differs from an ICMP message; the former goes to the previous hop while the latter is sent to the source host. As such interest NACKs cannot be used for DDoS attacks as ICMP messages have been since they can only be sent strictly between direct neighbors.

4.2 NDN Forwarding Daemon (NFD)

We continued development of the NDN Forwarding Daemon (`http://named-data.net/doc/NFD`). As our understanding of the NDN architecture matured over the last few years, we needed an platform that implements the latest architectural components to support easy experimentation with new protocol features, algorithms, and applications. To this end, we developed a new NDN forwarder (NFD) that meets NDN research needs most effectively (see Chapter 4 of last year's annual report for more details on this decision and the major components and functionality of NFD [25].)

NFD implements and evolves together with the NDN protocol, and is deployed on the NDN testbed. More specifically, NFD supports the new TLV packet format, and prioritizes code modularity and extensibility to facilitate research. NFD represents the centerpiece of our commitment to making NDN's core technology free and open to all Internet users and developers in order to facilitate research, development, and use. (GPL licensing details are at: `https://github.com/named-data/NFD/blob/master/COPYING.md`). We worked closely as a team over multiple conference calls per week for ten months, involving six PIs and dozens of students, postdocs, and staff members. We made the first public release of NFD v0.2 in August 2014, followed by v0.3.0 in February 2015, and v0.3.1 in March 2015. NFD is now part of the NDN platform release. The NDN testbed is running NFD natively as well as NDN-based routing protocols. While the packet format, NFD codebase, and testbed continue to evolve, they now provide a solid foundation for future NDN development.

To simplify and promote the adoption of NFD and NDN technology in general, in addition to providing full access to the source code, we also provide official binary packages for a set of supported platforms, including Ubuntu Linux 12.04, Ubuntu Linux 14.04, and Mac OSX with MacPorts. NFD runs on other operating systems including Fedora, Gentoo, FreeBSD, embedded systems like Raspberry Pi, OpenWRT/DD-WRT, Galileo, and virtualized environments like Vagrant and Docker. The public release of the forwarder and

other related tools was accompanied by the release of a set of extensive documentation on NFD's homepage (`http://named-data.net/doc/NFD/current/`) and a developer's guide [36]. We developed NFD tutorials for the NDN Community meeting in August 2014 and the ACM ICN conference in September 2014.

The current release supports the following forwarding strategies: broadcast, best-route, client control, and ncc (the strategy used in CCNx). In testing the video streaming application on the NDN testbed, we learned of the need for, and then developed, an "access hub strategy" between user computers and campus NDN hubs to improve the accuracy and efficiency of interest forwarding. We also added support for interest retransmission. We expect to add additional new strategies in the coming year to accommodate different network environments.

All management actions that change NFD state use signed Interest *control commands* [31] protocols. These allow NFD to determine whether the issuer is authorized to perform the specified action based on fine-grain access controls. Management modules respond with *control responses* to inform the user of the command's success or failure. Control responses have status codes with similar semantics to HTTP responses. NFD also supports unauthenticated query and status/dataset requests. These datasets provide information about available FIB entries, faces, and more.

Most managers use dispatch tables to route Interests to the correct handler (see Figure 4.6). All management protocol Interests, whether commands or dataset requests [33], follow a namespace pattern of `/localhost/nfd/<manager-name>/<verb>`. Here, *verb* describes the action that the *manager-name* manager should perform. For example, `/localhost/nfd/fib/add-nexthop` directs the FIB Manager to add a next hop (command arguments follow the verb).

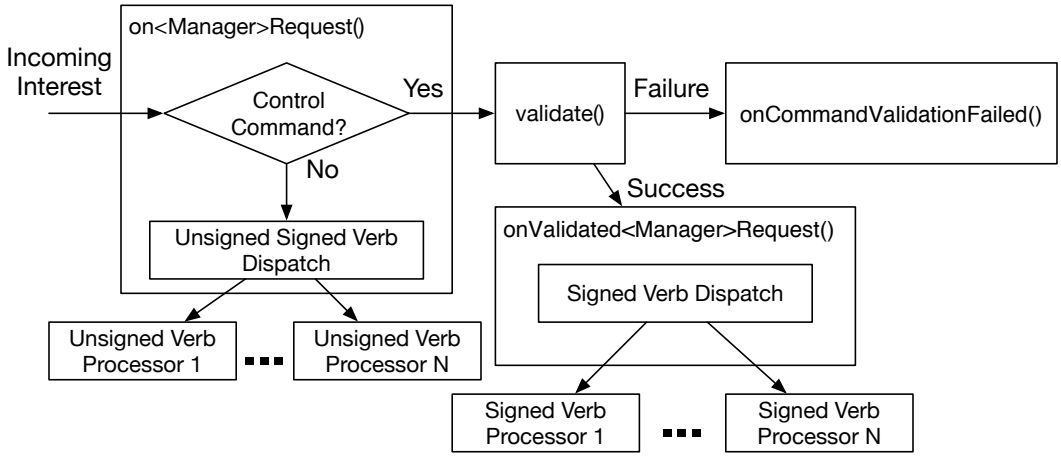


Figure 4.6: Overview of the manager verb dispatch workflow

4.3 Universal Infrastructural Lookup Service: NDNS

Several challenges have arisen that demonstrate the need for an infrastructural component to discover and retrieve information based on NDN names: ensuring the uniqueness of globally scoped names, retrieving public keys even when the owners are offline, supporting mobile publishers, and scaling the routing system. DNS for NDN (NDNS) is our attempt to address the above challenges as well as future deployment and operational needs in NDN. NDNS has the following design goals: 1) *scalability*, to support information discovery and retrieval at Internet-scale; 2) *autonomy*, to enable authorized administrators to control trade-offs among the cost of acquiring data, speed of updates, and accuracy of caches; 3) *universality*, i.e., supporting storage of named content in any format in the system; and 4) *security*, ensuring authenticity and integrity of the data retrieved from NDNS.

NDNS borrows many elements from DNS/DNSSEC, including concepts of domain and zone, iterative and recursive queries for information discovery, and hierarchical trust model. Zones include referrals to subzones in the name tree, effectively providing a structured hierarchy. Each record is cryptographically secured with the zone key, which is stored in NDNS itself and secured with the key of the parent zone, with cryptographic protection down to the common trust anchor. This security model resembles that of DNSSEC, but also tries to leverage the information-centric and data-centric security features of NDN. Specifically, NDNS uses multicast to support request aggregation and in-network caching, and our synchronization [49] tools to update dynamic information in a distributed data-centric manner.

4.3.1 Design of NDNS

The NDNS architecture has four components: namespace, information discovery, information security, and information maintenance. NDN uses the namespace of Internet applications, which for today's global-scale applications is the hierarchical DNS namespace. NDNS reflects the NDN namespace (Figure 4.7). NDNS constructs a tree of zones, starting with a root or set of top-level domains. Each NDNS zone contains information about the name, e.g., public key of the authorized owner or LINK to aid information retrieval across network domains [10]. To distribute control over namespaces and allow information discovery by traveling from the top to leaves, each zone contains referrals (delegations) to its children zones. Zones can be replicated and distributed across multiple authoritative NDNS name servers.

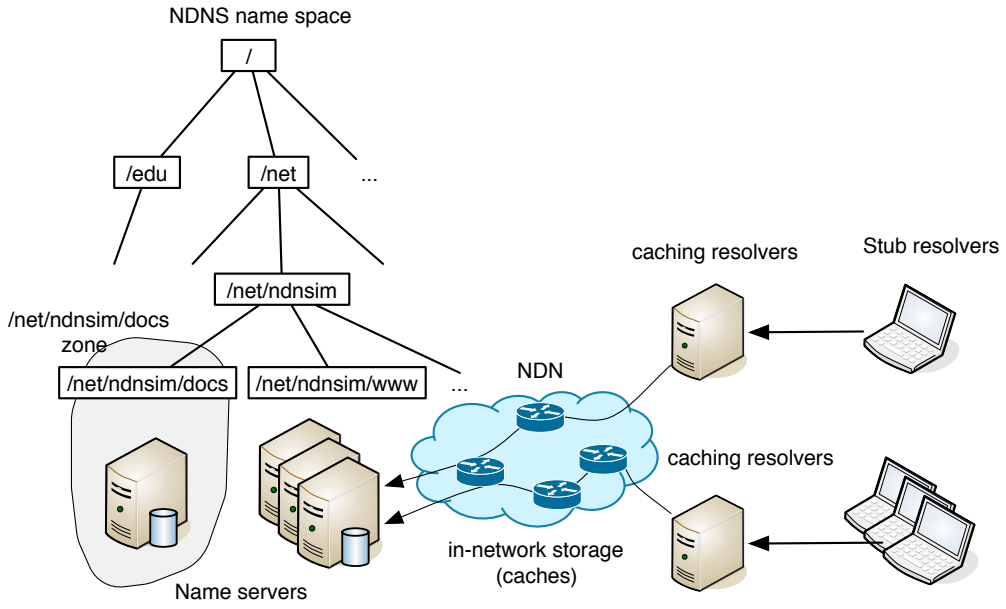


Figure 4.7: NDNS elements: forest of zone trees, name servers, caching resolvers, stub resolvers, and authorized namespace users. Hosts are connected via NDN network

NDNS defines two modes of information discovery: recursive and iterative queries. Stub resolver clients (often with limited processing power) sends recursive queries to caching resolvers, which can aggregate requests and cache responses. A series of iterative queries discovers information associated with a name, and consists of two stages. The first stage discovers which target zone is responsible for the requested information, starting from either the root or a top-level domain zone, then continuing to one of its children, and then children of the children. At each iterative step, the querier requests delegation information about the next level zone. For example, for "`/net/ndnsim/www`", the querier

- first requests "`/NDNS/net/NS`" to determine whether the zone "`/net`" is delegated,

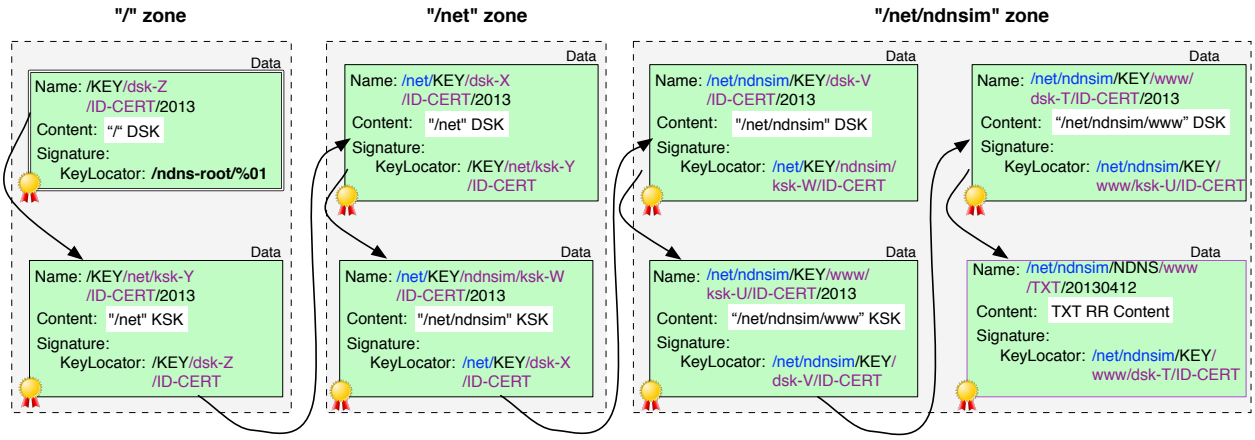


Figure 4.8: Hierarchical trust model example: data signing key (DSK) is certified by a key signing key (KSK) inside a zone; KSK, except for a root zone's KSK, is certified by DSK of the parent zone.

- then "`/net/NDNS/ndnsim/NS`" to determine whether "`/net/ndnsim`" is delegated, and
- then "`/net/ndnsim/NDNS/www/NS`" to check whether "`/net/ndnsim/www`" is further delegated.[5]

When the query for the next sub-namespace delegation fails (e.g., "`/net/ndnsim/NDNS/www/NS`" returns a negative answer), or no more component in the target name can be added to detect a sub-namespace delegation, the process switches to the second stage. Now one knows that the last detected zone is the only zone where the requested information may reside, so end clients can send the final question to get the answer. In this way, information from higher-level zones by the virtue of NDN in-network caching propagates to all corners of the network, reducing query load on name servers.

To support mobility, NDNS adopts a multi-master model to handle zone maintenance: any authorized party (i.e., with a key for the zone) can generate an update resource record and deliver it toward any of the zone's name servers. NDNS synchronizes the zone contents across all name servers using Chronosync [49].

To ensure validity of the data, NDNS adopts a hierarchical trust model so that clients can validate provenance of the data discovered using NDN mechanisms. The trust model requires that each NDNS record is cryptographically signed with a key that belongs to the authorized user of the namespace. The authorization fact is defined by signing this key with another key that belongs to the parent zone (i.e., the parent zone authorizes use of a child zone) or by a key that belongs to the zone itself (see Figure 4.8); all these keys are stored inside the NDNS system.

4.3.2 Current Status and Future Plan

NDNS is deployed on the NDN Testbed (5.1) with a few dedicated zones and several zones created for participating institutions:

- "`/ndn/NDNS`": NDN Testbed root zone
- "`/ndn/edu/ucla/NDNS`": NDNS zone for University of California, Los Angeles (UCLA) site of NDN Testbed
- "`/ndn/edu/arizona/NDNS`": NDNS zone for University of Arizona site of NDN Testbed
- ...

[5]In addition to the delegation information, the query also discovers LINK objects to be used to retrieve information from name servers of the delegated domains.

Currently, the ChronoChat [49] and NDN Video Player [29] applications use NDNS to discover public key certificates during their data validation stages. As part of our routing scalability solution, NDNS provides the LINK objects to map all names that are absent from the global routing table to globally routable names.

4.4 Routing

During the 4-year NDN project, we developed and deployed a secure routing protocol, *Named Data Link State Routing Protocol (NLSR)* [22], which supports name-based multipath forwarding in a single domain. To support testbed growth and deployment of new NDN applications with a geographically distributed user base, we need an **inter-domain** routing protocol with four properties and capabilities:

- Scalability: the protocol can scale to a large topology and a large number of name prefixes.
- Security: every node can verify the authenticity of received routing information;
- Multipath: the protocol can provide multiple paths to each name prefix if alternative paths exist; and
- Policy routing: network operators can express and apply routing policies to influence traffic paths.

There are three major candidates for NDN's inter-domain routing: path-vector, link-state, and hyperbolic routing [13, 27, 39, 40, 41]. Link-state routing has several advantages compared to the path-vector approach: (1) it separates topological connectivity from name prefixes which reduces routing churn caused by a link fails; and (2) it converges faster. Unfortunately, link-state protocols require ISPs to disclose all business relations with their neighbors. Hyperbolic routing is a type of geometric routing – each name prefix is mapped to a set of hyperbolic coordinates and routers use greedy forwarding based on hyperbolic distances. This approach can greatly increase the scalability of the routing system as there will be no FIB or routing updates. Moreover, there are no routing advertisements to authenticate, so the routing protocol is secure as long as the mapping from name prefixes to coordinates is secure. One issue with this approach is how to support routing policies, since policies cannot be expressed in coordinates and routers do not have full topology information to apply policies.

During the first year of the NDN-NP project, our investigation has mainly focused on the following issues:

- **Routing's role in NDN (Section 4.4.1)**: today's IP networks place intelligence into the control plane (routing layer), while the data plane merely forwards packets according to the FIB. However, NDN has an intelligent and adaptive forwarding plane, which raises the question of whether routing protocols are still needed. If so, for what purpose and to what extent? We found that a routing protocol is highly beneficial in bootstrapping the forwarding plane for effective data retrieval, and in efficient probing of new links or recovered links. However, *NDN routing does not need to converge fast following network changes*, which adaptive forwarding can handle more promptly.
- **Hyperbolic routing (Section 4.4.2)**: we evaluated the performance of hyperbolic routing, which has potential to become a feasible candidate for NDN's inter-domain routing if its performance closely approximates that of link-state routing in a large topology. (We would still have to investigate routing policy support, trust models, and other deployment related issues.) Our experiments thus far have compared 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.
- **Routing protocol enhancements (Section 4.4.3)**: we used NLSR for this evaluation, as it supports both link-state and hyperbolic routing algorithms (by disseminating hyperbolic coordinates in link-state announcements). During the evaluation, we discovered and fixed many bugs in the NLSR code. In addition, we added two important features to NLSR: (1) runtime advertisement and withdrawal of name prefixes which allows network operators to manage their name prefixes without causing unnecessary disruption to the network; and (2) on-demand publishing and retrieval of NLSR's Link-State Database (LSDB) which enables operators to monitor and diagnose the routing protocol.
- **Scaling routing using secure namespace mapping (Section 4.4.4)**: we applied a well-known concept of *Map-and-Encap* [16] to provide a simple and secure namespace mapping solution to the scalability problem. Whenever necessary, application data names can map to globally routable names

used to retrieve the data. By including such sets in data requests, we are informing (more precisely, hinting) the forwarding system of the whereabouts of the requested data, and routers use such hints when they do not know from where to retrieve data using application data names alone. This solution enables NDN forwarding to scale with the Internet's well-understood routing protocols and operational practice, while keeping all the benefits of the new NDN architecture.

The rest of this section provides more details about our activities and findings.

4.4.1 Lessons Learned on the Role of Routing in NDN

A unique feature of Named Data Networking (NDN) is that its forwarding plane can detect and recover from network faults on its own, enabling each NDN router to handle network failures locally without relying on global routing convergence. This feature prompts us to re-examine the role of routing in an NDN network: does it still need a routing protocol? If so, what impact may an intelligent forwarding plane have on the design and operation of NDN routing protocols? Through analysis and extensive simulations, we found that routing protocols remain highly beneficial in an NDN network. Routing protocols disseminate initial topology and policy information as well as long-term changes in them, and computes routing tables which guide the forwarding process. However, because the NDN forwarding plane can quickly detect and recover from failures, there is no need for a routing information layer to accommodate churn in connectivity. But NDN networks can use routing protocols to significantly improve network scalability and stability, by using larger keep-alive timer values that ignore short-term failures. Furthermore, routing algorithms that would not work well in today's IP networks may work fine in an NDN network due to the reduced role of routing in bootstrapping adaptive forwarding.

We ran experiments in the QualNet simulator [42] which provides complete implementations of OSPF and RIP routing protocols. We implemented basic NDN operations and the forwarding strategy presented in [47] in the simulator. We used the Abilene topology [1] and selected Rocketfuel topologies [44] in the experiments. We injected random link failures into the topologies, and used a shifted Pareto distribution to generate time-to-failure and time-to-recover values for each link independently. When a link fails, both directions of the link stop working. With this model, multiple network events (failures and recovery) can happen concurrently. We learned the following lessons from the routing simulation experimentations:

1. Although NDN has a powerful forwarding plane that is able to handle link failures on its own with only local information, the interface ranking provided by a routing protocol can make the local search more effective.

2. Routing protocols provide information that allows routers to, upon a link failure. optimize (minimzing) probing in search of working paths.

3. Routing can have slower convergence and lower overhead.

4.4.2 Hyperbolic Routing

Hyperbolic routing is a geometric routing scheme that relies on hyperbolic coordinates to send packets through a network. Assuming a mechanism for retrieving the coordinates of a name prefix exists (e.g., through NDNS), each Interest can carry the coordinates of the name prefix, and routers can use greedy forwarding to forward the Interest, i.e., choose the next hop(s) for the Interest based on the distances between a router's neighbors and the Name prefix. This scheme is highly scalable as there is no need to maintain a routing table or FIB, and there are no dynamic routing updates.

To investigate whether hyperbolic routing is a viable solution for NDN, we compared it with our link-state routing protocol under different forwarding strategies, failure scenarios, number of multi-path faces, and topologies. The performance metrics we used are RTT, packet loss ratio, number of messages generated, and failure response time. To show that hyperbolic routing is feasible for NDN, we must show that its performance metrics closely approximate (or are better than) those of link-state under various conditions.

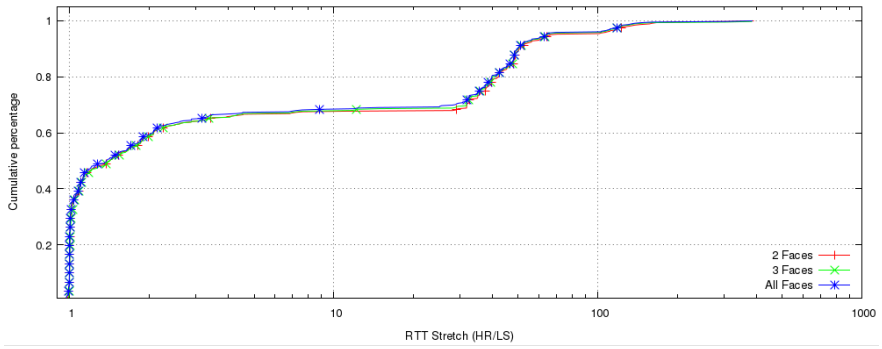


Figure 4.9: CDF of RTT Stretch (hyperbolic/link-state) with Best Route Strategy

We conducted emulation experiments on a 22-node topology based on the current NDN testbed topology. Our results are available at [23]. Our experiments uncovered an undesirable interaction between the ncc forwarding strategy (described in Section 4.2) and the duplicate suppression implemented in NFD, which leads to strange unpredictable paths [34]. Due to this problem, we decided to stop using the *ncc* forwarding strategy and instead use the *best route* strategy as a baseline measure. Since the best route strategy uses the highest routing ranked face (except for consumer retransmissions), the forwarding plane performance basically reflects how good the top routing-ranked face is. Our results show a high percentage of packet losses for some of the node pairs in hyperbolic routing with the *best route* strategy; around 31% of the node pairs had 100% timeouts. In comparison, there were no timeouts when using link state since the topology is connected. We also computed the RTT stretch of hyperbolic routing compared to link-state routing (Figure 4.9) and found that the stretch is higher than 1.5 times of the shortest path in 48% of the cases. These results suggest that *best route* is not a feasible forwarding strategy when hyperbolic routing is in use. Therefore, we are designing a forwarding strategy to avoid problems in both *ncc* and *best route*. We will use this this strategy in our experiments once it is available.

We will continue the comparison between hyperbolic routing and link-state routing with larger topologies, more failure scenarios, and smart flooding (the latter mechanism floods the first Interest and uses the lowest delay path for subsequent Interests). For hyperbolic routing to be a viable solution for NDN, its forwarding plane overhead wil have to be much lower than that of smart flooding.

4.4.3 Named Data Link State Routing Protocol (NLSR)

We continued research and development on NLSR [22]. Although the implementation met our testbed and hyperbolic routing evaluation needs, we made enhancements to functionality and performance this year.

First, while NLSR was able to disseminate routing information, detect routing changes, and advertise name prefixes, it did not allow dynamic changes to the set of name prefixes advertised by a router once the process was launched. To change the advertised name prefixes, an operator needed to stop NLSR, add or remove name prefixes from the configuration file, and launch NLSR again. This requirement not only complicates network operations, but also causes unnecessary interruptions to the routing system. We have implemented a command-line tool, NLSRC, for users to communicate with a running NLSR process to advertise and withdraw name prefixes using signed Interest Commands. NLSR has the capability to receive Interest Commands from NLSRC, validate signatures on the commands, and then advertise or withdraw name prefixes accordingly.

Second, we found that routing convergence took much more time than expected. We implemented failure and recovery response optimizations to speed up convergence, and added multiple parameters so users can tune the frequency of routing computations to balance convergence time and routing protocol overhead (i.e., routing messages and computation time).

Third, acquiring information on a running NLSR process required tedious analysis of log files. To make it easier for operators to monitor NLSR and diagnose problems, we added the ability for NLSR to publish its

link-state database (LSDB), which contains all the name prefixes, router adjacencies, and router coordinates (used for hyperbolic routing) that a router has advertised and received. Our command-line tool NLSRC can display the contents of the LSDB on the console. We plan to redesign our web-based monitoring tool to use the LSDB published by NLSR instead of obtaining such information from log files.

Finally, in the previous project year, we deployed NLSR on the NDN testbed without turning on key validation in the configuration, as we first wanted to ensure that the routing messages were propagated as expected. Since then, we have enabled security for NLSR with the necessary certificates and keys configured, and continue to evaluate and support NLSR on the testbed.

The testbed deployment and our own experiments exposed many bugs in the code; we have incorporated the bug fixes, as well as improved the structure and readability of the code. We released NLSR 0.2.0 on April 30, 2015 with all of the previously mentioned improvements [37].

4.4.4 SNAMP: Secure Namespace Mapping to Scale NDN Forwarding

One frequently raised concern about NDN is the scalability of its name-based forwarding ([12, 30]). Given that the number of data names is unbounded, can one keep the size of name-based NDN forwarding information base (FIB) under control? In the traditional IP architecture, the basic approach to scalability of global routing is IP address aggregation, a practice embedded in global policies for address allocation: end users and small networks get IP addresses from their access providers, who can aggregate many of these smaller chunks of address space and inject only the aggregated prefixes into the global routing system. Such route aggregation has been largely successful, but other factors have driven growth of the global routing table, including a growing demand for provider-independent addresses [28], network-layer traffic engineering and load balancing [9], and pratices to mitigate DDoS attacks and prefix hijacks.

Another approach to limiting growth of the global routing table is to introduce a layer of indirection: one can reach addresses that are not on the globally routed by mapping them to addresses that are. This is known as "map and encapsulate" [16], or tunneling, and is used in several proposals for scaling the global IP routing system [38, 17, 11, 26]. Building upon this Map-and-Encap idea, we proposed a solution, dubbed SNAMP (Secure NAmespace MaPping), to address NDN's named-based forwarding table scalability [10]. SNAMP enables the network to forward all interest packets toward the closest data even if not all data name prefixes are present in the global routing table. Data whose name prefixes do not appear in the table can be retrieved using a securely mapped set of globally routed name prefixes (Figure 4.10). We prototyped a distributed mapping system, NDNS (Section 4.3), to maintain a database of mappings from names to globally routed prefixes. Data retrieval require SNAMP mechanisms only if the data is not from a local producer, not in any nearby cache, and not in the global routing table.

When a data name does not have its prefix in the DFZ, routers can still direct interest packets carrying that name toward the data, provided that the interest packet includes one or more globally routed prefixes of the network(s) that can provide the requested data. For example, Fig. 4.10 shows that, although "`/net/ndnsim/www/index.html`" cannot find a matching prefix in the DFZ FIB, interest packets carrying this name can be forwarded in the DFZ using FIB information for "`/ucla/cs`" or "`/telia/terabits`" prefixes. Once inside the local network environments (the UCLA CS department or the Terabits network), the Interest packet can retrieve the data by its name directly.

Figure 4.11 presents an overview of the proposed NDN map-and-encap process. We briefly describe two important components of our design – the *link* object and link discovery – and then describe the SNAMP-enabled NDN interest forwarding process.

The Link Object. In SNAMP, if a data producer wants to make data available globally but its prefix is not in the DFZ, the producer must establish an association between the name prefix (e.g., "`/net/ndnsim/www`") and the globally routed prefixes of its Internet service providers (e.g., "`/telia/terabits`" and "`/ucla/cs`" in our example). We call this association a *link*, as described in Section 4.1.3. We define the name of the link object to be N_o with a special mark (to avoid confusion with other types of data under the same name), and the data portion to be a list of delegated namespaces in some priority order. By creating and signing the link object, the owner of the original namespace N_o delegates its namespace to a set of namespaces

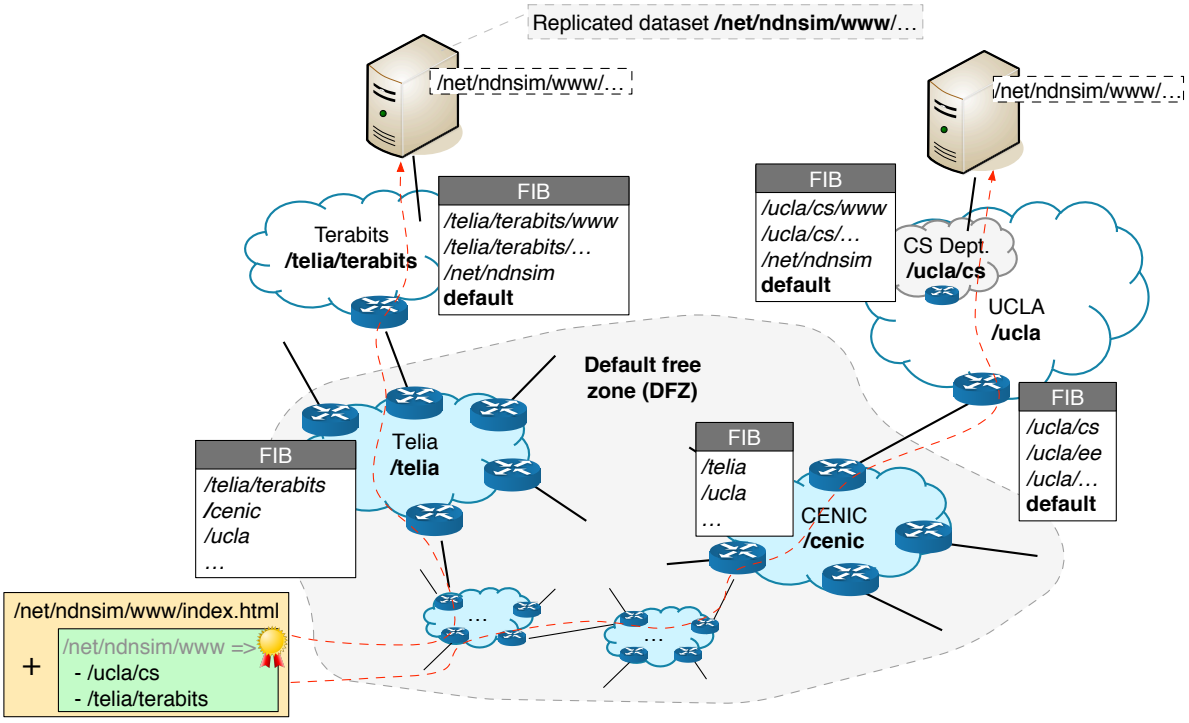


Figure 4.10: Example of FIB state in an NDN network with Map-and-Encap: the "`/net/ndnsim/www`" dataset is replicated at Terabits and UCLA Computer Science Department, "`/net/ndnsim/www/index.html`" data can be reached directly inside Terabits and UCLA CS network, and can be reached globally if "mapped" to "`/telia/terbits`" or "`/ucla/cs`" prefix

$N_1, \ldots N_n$, essentially endorsing that the data under namespace N_o can be retrieved if interest is forwarded toward $N_1, \ldots,$ or N_n.

Discovery of the Prefix Delegation Set. DNS-Based Authentication of Named Entities (DANE) [21] provides an attractive means to use DNSSEC infrastructure to store and sign keys and certificates used by today's applications. Because the entities that vouch for the binding of public key data to DNS names are the same entities responsible for managing the DNS names in question, DANE restricts the scope of assertions that can be made by any entity, thus embodies the security "principle of least privilege." Inspired by DANE, we developed NDNS (DNS for NDN) [8] (Section 4.3), a scalable federated database system that retains these DNS/DANE properties and serves a similar purpose in the NDN world. One use is to serve prefix delegation maintenance and lookups. Whenever needed, the owner of a namespace N_o can store its link objects in NDNS. Others can look up N_o's name to find its link using an iterative or recursive resolution process [8] similar to the DNS query process. During iterative resolution, either a dedicated caching resolver on behalf of the consumer or the consumer itself retrieves a set of link objects, gradually discovering the delegation namespace, one level at a time. In our example (Figure 4.12), the process starts with retrieval of delegation information for "`/net`" namespace, followed by retrieval of information about "`/net/ndnsim`", and concluded by retrieval of the "`/net/ndnsim/www`" namespace delegation (see [8] for more details).

Note that NDNS requires only the "`/NDNS`" prefix to be present in the DFZ FIB, pointing toward multiple replicas of the root NDNS server. This constraint guarantees that all requests to the NDNS root zone can be answered. However, if all of the top level domain names (e.g., "`/com/NDNS`" and "`/net/NDNS`", etc.) are present in the DFZ, then the resolution process can start from the top level domains.

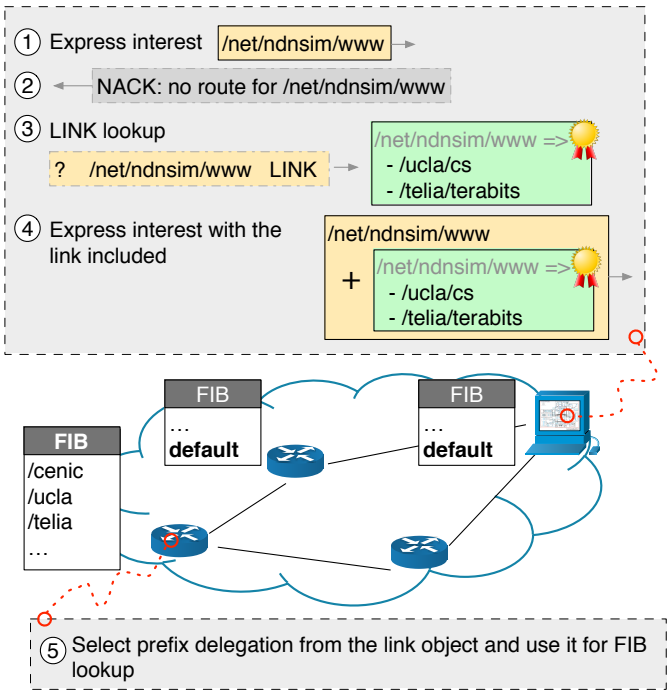


Figure 4.11: Overview of map-and-encap NDN process

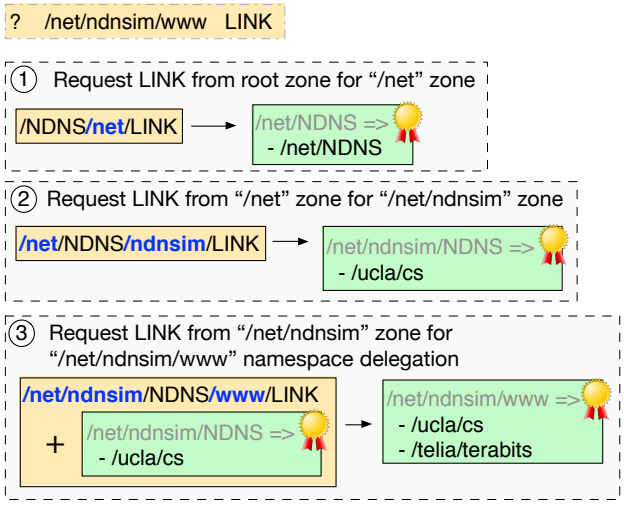


Figure 4.12: Example of iterative NDNS query (performed by the recursive NDNS resolver on behalf of the consumer or by the consumer himself)

Retrieving Data with SNAMP. SNAMP is transparent from the application's perspective: consumer applications send out ordinary NDN interests for the data, and producer applications publish data in the namespace they own. When an expressed interest happens to carry a globally routed prefix (e.g., "`/ucla/cs/www/index.html`"), the router receiving the interest can fetch the matching data either from a local cache or forwarded the interest toward the data producer. If the interest carries a name that is not present in the global routing table (e.g., "`/net/ndnsim/www/index.html`"), it still can bring back data if the data happens to be in a local cache along the way, otherwise the interest reaches the first router that does not have "default" route and cannot be further forwarded. This is the place where SNAMP kicks in (steps 2–5 in Fig. 4.11).

A default-free router at the network edge will respond to this unroutable interest with a network NACK [47], indicating that it does not have a route for the interest's name and needs more information to forward the interest. This NACK eventually propagates back to the consumer node, and the local NDN forwarder for the consumer application will retrieve the link object from NDNS and verify the validity of the delegation by checking the signature of the link object.

After the link object is retrieved and verified, the node will embed it into the original interest and send it out again. When this modified interest reaches the first router that cannot find a matching FIB entry, the router will extract the prefix delegations from the attached link object, select the best candidate (e.g., using routing cost or previous traffic measurements), and forward the interest based on this selection.

Note that in the above procedure, routers that do not have a FIB entry for the interest name may need to perform multiple additional FIB lookups to determine the best namespace to use for further forwarding (e.g., based on the routing cost).

Optionally, the first default-free router may record its decision in the forwarded interests, e.g., by putting the index for the selected delegate in the optional field inside the interest. Downstream routers can then rely on this pre-recorded selection, or perform the selection again. Also note that every router can elect to verify the validity of the attached link object, and forward only interests with a valid link object (i.e., the name of the link object must match the prefix of the interest name and have a valid signature) toward legitimately delegated prefixes. Routers can store verified link objects locally so that they do not have to verify the same link objects carried in subsequent interests.

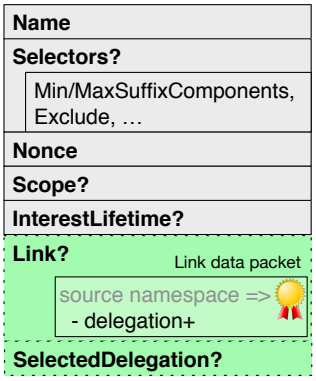


Figure 4.13: New "Link" field in interest packet

Eventually, the interest either reaches a cache that can satisfy it or propagates down to one of the data producers. The cached or produced data is then returned to the original requester using standard NDN data forwarding mechanisms, following the state created by the forwarded interests.

Attaching Link Objects to Interests. In order to attach a link object to an interest packet, we need to extend the NDN-TLV interest packet specification [35] by including two additional optional fields (Fig. 4.13): "`Link`" to carry the link object and "`SelectedDelegation`" to carry the delegation namespace (its index inside the link object) chosen by the previous hop.

4.5 Scalable Forwarding

With respect to forwarding research, our goal is to develop fast, scalable NDN node prototypes. By fast, we mean that we intend to support 1 Gbps links at line-rates in software implementations; we expect our hardware-accelerated implementations to exceed this rate by at least an order of magnitude. By scalable, we mean that we intend to develop an NDN forwarding plane that can support millions and even billions of names, each of arbitrary length. In addition, we provide feedback to the areas of architecture and routing whenever their design choices might have substantial performance consequences in the data plane. We also need to keep NDN forwarding comparable to other name-based forwarding schemes with respect to features and performance.

This year our scalable forwarding research focused on whether it is feasible to perform line-rate *longest* name prefix matching (LNPM) with a large FIB table *regardless of the specific characteristics of the forwarding rules*. We proposed and implemented a scalable name prefix lookup design based on binary search of hash tables [48]. With this design, the worst-case lookup requires at most $log(k)$ hash lookups for prefixes with up to k name components, regardless of the characteristics of the FIB. We implemented the proposed design on a general purpose multi-core platform using the Data Plane Development Kit (DPDK) [4] as the underlying packet I/O and multicore framework. The implementation was capable of achieving 10 Gbps throughput with *one billion* synthetic longest name prefix matching rules, each containing up to seven name components. Additionally, in recognition of the central and growing role that namespace synchronization plays in emerging NDN applications and network environments, we have developed and evaluated a scalable approach to synchronization, called iSync. We describe both research achievements in this section.

4.5.1 Binary Search of Hash Tables for Name Prefix Lookup

The binary search of hash tables organized by prefix lengths was originally proposed for accelerating IP prefix lookup [46]. With this idea, the number of hash lookups is reduced to $log(32) = 5$ and $log(128) = 7$ for IPv4 and IPv6, respectively. NDN names are much longer than IP addresses in terms of bits, but these names contain explicit delimiters that separate the name components. As a result, the hash tables can be organized by the numbers of components in the prefixes. That is, prefixes with the same number of name components are stored in the same table. Using a binary search of hash tables requires only $log(k)$ hash table lookups for rules that have up to k components in each prefix, regardless of their other characteristics.

For names with up to k components, k hash tables are created and form a balanced binary search tree. Locating the longest matching prefix for a name requires performing a binary search on hash table nodes. At each node, if there is a matching prefix in the corresponding hash table, then the algorithm proceeds to the right subtree to search for a potential longer matching prefix; if there is no match, then the longest matching prefix must be in the left subtree. The lookup procedure terminates when the bottom binary search level or a leaf FIB entry is reached. The total number of hash lookups is bound by $log(k)$. As shown in [46], additional *marker* entries are required to ensure that prefixes can find shorter matching prefixes, and the number of additional marker entries is bounded by $log(k)$ for each prefix.

Storing name prefixes requires much more memory than IP prefixes. Fingerprint-based hash tables can reduce the required number of memory accesses, where fingerprints are hash values of the keys in the table. Typically, fingerprint-based hash tables have cache-line sized buckets, where each bucket stores a constant number of fingerprint entries. Each fingerprint entry contains a fixed-length fingerprint and also stores the string address. In our implementation, each fingerprint entry contains a 20-bit fingerprint and a 44-bit memory pointer, which is the address where the forwarding information and the actual prefix string are stored. To keep the average number of memory accesses low, we chose to trade memory space for speed. Similar to [43], the hash table has a relatively low load factor so that most hash lookups require only one hash bucket access, and chaining is used to resolve bucket overflows. String matching is performed when there is a fingerprint match. Our design employs two string matching strategies. The first strategy, denoted as CityA, performs string matching whenever there is a fingerprint match, and it uses CityHash [3], a fast non-cryptographic hash function. The second strategy, denoted as SipE, performs string matching only at the end, and uses the SipHash [6] cryptographic hash function. Our performance evaluation (with synthetically generated names to approximate a worst-case scenario) demonstrated that this design supports

large FIB tables efficiently in software with modest performance optimization (increased page sizes and software prefetching); detailed analysis and the multicore performance are available in [48].

Table 4.1: Throughput with 256-Byte Packets

	CityA		SipE	
	MPPS	Gbps	MPPS	Gbps
None	4.1	9.1	3.8	8.4
First	4.2	9.4	4.0	8.9
All	4.4	9.7	4.1	9.1

To further evaluate the name prefix lookup performance with real network traffic, we developed an NDN forwarding engine on top of the DPDK packet I/O and multicore framework [4]. The forwarding engine performed only longest name prefix lookup; we will add Pending Interest Table and Content Store in future work. For fast prototyping, we employed a simplified NDN packet format [48], and transmitted NDN packets on top of UDP. The DPDK-based Pktgen program reported forwarding throughput on the receiver side (observed values listed in Table 4.1). When all hash buckets were prefetched, the CityA and SipE strategies yielded 9.7 Gbps and 9.1 Gbps throughputs, respectively. When eight worker threads were employed using both processors, 10 Gbps throughput was achieved for all of the cases listed in Table 4.1.

4.5.2 iSync - Synchronizing Namespaces with Invertible Bloom Filters

In named data networking (NDN), data synchronization across nodes is used to support basic services, such as public key distribution, file sharing, and route distribution. The essential requirement of a data synchronization service – keeping a dataset (or a *collection*) up-to-date among distributed participants – requires replicating the dataset among participating hosts. In NDN, a content item is represented by a namespace, and a synchronized NDN dataset consists of the content names: the list of namespaces. While the implementation of NDN applications that request to synchronize their information varies according to their goal and needs, the primitive of synchronizing a collection of namespaces is identical. We designed and implemented iSync, a high performance and scalable synchronization protocol, to provide data synchronization as a service to multiple applications.

The process of data synchronization includes three tasks: 1) understanding whether a set is up-to-date or out-of-date, 2) finding set differences, and 3) retrieving missing items. The second task has the largest impact on the efficiency and scalability of a synchronization service. Existing NDN synchronization implementations rely on log-based representation of information, which limit their performance and scalability. We proposed a new synchronization layer for the NDN stack: the iSync protocol, which utilizes the invertible Bloom filter (IBF) to reconcile sync collections between nodes [19]. IBF is a hash-indexed, redundant table that hashes each item into at least two cells. It supports item insertion, deletion, and limited inversion (i.e., retrieval of stored keys). The ability to retrieve the stored items differentiates IBF from other compact set representations. Moreover, one can obtain a difference set between two IBFs by subtracting their IBF tables. 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. The repository offers an interface for applications to insert files and declare synchronized collections. Like CCNx Sync [2], iSync defines a collection as the set of content items sharing a common prefix. The iSync repository notifies the sync agent when a new content name matches one of the locally declared collections. Then, the sync agent indexes the inserted content name and updates a digest that reflects all names of the collection. The sync agent notifies remote nodes of its local digests by sending periodic broadcasts, while receiving remote ones. When a remote collection digest does not match the local collection digest, a reconciling process starts, and the set difference is found by repeatedly requesting, receiving, and comparing remote IBF tables against local and global IBF tables.

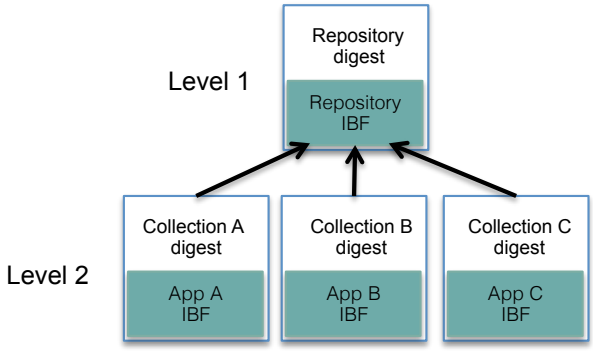


Figure 4.14: Hierarchical Synchronization Data Structure.

iSync uses a hierarchical two-level IBF: *Repository IBF* and *collection IBFs* (Figure 4.14). The higher level IBF records the status of the entire repository, while the lower level logs file insertions or deletions of each sync collection separately. An update to a collection changes the collection's IBF in only one second level digest, by hashing the new namespace into the corresponding *collection sync IBF*. The change in a collection digest invokes an update to the repository IBF and digest in the higher level.

This design achieves the three goals of a synchronization protocol: 1) The first level digest holds the status of the entire repository. 2) Currency is maintained by subtracting a remote first level IBF from the local first level IBF. Traditional NDN synchronization protocols iteratively send the digests of each collection, but iSync requires only one data exchange to discover all out-of-date collections. 3) In a similar way, the set difference of the changed collection is obtained by subtracting a corresponding remote IBF from the local second level IBF. iSync sequentially requests all remote out-of-date second level IBFs, subtracts them from the local ones, obtains the updated namespaces, and fetches the content.

We compared the performance of iSync to the CCNx synchronization protocol [2] across a range of network topologies and sizes [18]. While CCNx Sync requires multiple data retrievals and comparisons for a single update, iSync's approach reduces this workload, thus reducing latency and traffic overheads. Our experiments showed that iSync was about 8 times faster in most of our test cases, and that it reduced synchronization traffic (i.e., packets sent) by about 90%.

References

[1] Abilene TM. `http://www.cs.utexas.edu/~yzhang/research/AbileneTM/`.

[2] CCNx synchronization protocol. Online: `https://www.ccnx.org/releases/latest/doc/technical/SynchronizationProtocol.html`.

[3] Cityhash. Online: `https://code.google.com/p/cityhash`.

[4] DPDK: Data plane development kit'. Online: `http://www.dpdk.org`.

[5] IEEE std. 802.15.4, part. 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks. IEEE std.

[6] Siphash. Online: `https://131002.net/siphash`.

[7] A. Afanasyev, R. Ravindran, L. Wang, B. Zhang, and L. Zhang. ICN packet format design requirements. Internet-Draft draft-icn-packet-format-requirements-00, 2015.

[8] Alexander Afanasyev. *Addressing Operational Challenges in Named Data Networking Through NDNS Distributed Database*. PhD thesis, UCLA, September 2013.

[9] Alexander Afanasyev, Neil Tilley, Brent Longstaff, and Lixia Zhang. BGP routing table: Trends and challenges. In *Proc. of High Tech. and Intell. Systems conf.*, 2010.

[10] 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.

[11] R. Atkinson, S. Bhatti, and S. Hailes. ILNP: mobility, multi-homing, localised addressing and security through naming. *Telecommunication Systems*, 42(3), 2009.

[12] A. Baid, T. Vu, and D. Raychaudhuri. Comparing alternative approaches for networking of named objects in the future Internet. In *Proc. of NOMEN*, 2012.

[13] Marián Boguñá, Fragkiskos Papadopoulos, and Dmitri Krioukov. Sustaining the Internet with Hyperbolic Mapping. *Nature Communications*, 1:62, 2010.

[14] David Clark. The design philosophy of the DARPA Internet protocols. *ACM SIGCOMM Computer Communication Review*, 18(4):106–114, 1988.

[15] S Deering and R Hinden. internet protocol, version 6 (IPv6). RFC 2460, 1998.

[16] Steve Deering. The map & encap scheme for scalable IPv4 routing with portable site prefixes. *Presentation Xerox PARC*, 1996.

[17] D. Farinacci. Locator/ID separation protocol (LISP). Internet draft (draft-farinacci-lisp-00), 2007.

[18] Wenliang Fu, Hila Ben Abraham, and Patrick Crowley. isync: A high performance and scalable data synchronization protocol for named data networking. In *Proceedings of the 1st International Conference on Information-centric Networking*, ICN '14, pages 181–182, New York, NY, USA, 2014. ACM.

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

[20] Susan Hares, Yakov Rekhter, and Tony Li. A border gateway protocol 4 (BGP-4). RFC 4271, 2006.

[21] P. Hoffman and J. Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698, 2012.

[22] AKM M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang. NLSR: Named-data link state routing protocol. In *ACM SIGCOMM ICN Workshop*, 2013.

[23] Hyperbolic routing evaluation results. `http://netwisdom.cs.memphis.edu/mini-ndn/`.

[24] ITU-T. Interface for the optical transport network (OTN). G.709 Recommendation, February 2012.

[25] 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 2013 - 2014 Report.

[26] D. Jen, M. Meisel, D. Massey, L. Wang, B. Zhang, and L. Zhang. APT: A practical tunneling architecture for routing scalability. Technical Report 080004, UCLA Comp. Sc. Dep., 2008.

[27] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá. Hyperbolic geometry of complex networks. *Physical Review E*, 82:036106, 2010.

[28] Dave Meyer, Lixia Zhang, and Kevin Fall. Report from the IAB workshop on routing and addressing. *RFC 4984*, September 2007.

[29] Ilya Moiseenko and Lixia Zhang. Consumer-Producer API for Named Data Networking. Technical Report NDN-0017, NDN, 2014.

[30] Ashok Narayanan and David Oran. NDN and IP routing: Can it scale? Proposed Information-Centric Networking Research Group (ICNRG), Side meeting at IETF-82, November 2011.

[31] NDN Project Team. Control command. Online http://redmine.named-data.net/projects/nfd/wiki/ControlCommand, 2014.

[32] NDN Project Team. Ndn technical memo: Naming conventions. Technical Report NDN-0022, NDN, 2014.

[33] NDN Project Team. NFD management protocol. Online: http://redmine.named-data.net/projects/nfd/wiki/Management, 2014.

[34] Bug #2592: Ineffective duplicate suppression for satisfied interest. http://redmine.named-data.net/issues/2592.

[35] Ndn packet format specification. http://named-data.net/doc/ndn-tlv/.

[36] Alexander et al.) NFD Team (Afanasyev. Nfd developer's guide. Technical Report NDN-0021, NDN, February 2015.

[37] NLSR documentation and code. http://named-data.net/doc/NLSR/current/.

[38] M. O'Dell. 8+8: An alternate addressing architecture for IPv6. Internet draft, 1996.

[39] Fragkiskos Papadopoulos, Maksim Kitsak, M. Ángeles Serrano, Marián Boguñá, and Dmitri Krioukov. Popularity versus similarity in growing networks. *Nature*, 489:537–540, 2012.

[40] Fragkiskos Papadopoulos, Constantinos Psomas, and Dmitri Krioukov. Replaying the geometric growth of complex networks and application to the AS internet. *ACM SIGMETRICS Perf E R*, 40(3):104, 2012.

[41] Fragkiskos Papadopoulos, Constantinos Psomas, and Dmitri Krioukov. Network mapping by replaying hyperbolic growth. *IEEE ACM Transactions on Networking*, 2014.

[42] Scalable Networks. QualNet simulator. http://www.scalable-networks.com/products/qualnet/.

[43] Won So, Ashok Narayanan, and David Oran. Named data networking on a router: Fast and DoS-resistant forwarding with hash tables. In *In ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'13)*, 2013.

[44] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson. Measuring isp topologies with rocketfuel. *IEEE/ACM Trans. Netw.*, 12(1):2–16, February 2004.

[45] R Steward. Stream control transmission protocol. RFC4 960, 2007.

[46] Marcel Waldvogel, George Varghese, Jon Turner, and Bernhard Plattner. Scalable high speed IP routing lookups. In *ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, Cannes, France, 1997.

[47] Cheng Yi, Alexander Afanasyev, Ilya Moiseenko, Lan Wang, Beichuan Zhang, and Lixia Zhang. A case for stateful forwarding plane. *Computer Communications: Information-Centric Networking Special Issue*, 36(7):779–791, April 2013.

[48] Haowei Yuan and Patrick Crowley. Reliably scalable name prefix lookup. In *To appear in ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, 2015.

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

Chapter 5

Evaluation

Contributors	
PIs	Beichuan Zhang (Arizona), Van Jacobson & Lixia Zhang (UCLA), Lan Wang (Memphis), Christos Papadopoulos (Colorad State University), Patrick Crowley (Washington University)
Grad Students	Junxiao Shi, Jerald Abraham, Yi Huang (Arizona); Ilya Moiseenko, Yingdi Yu, Wentao Shang (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 (11/2014 – present) (Memphis)
	Postdoc: Alex Afanasyev (UCLA)

5.1 NDN Testbed: Deployment, Management, Expansion

A year ago, the NDN Testbed consisted of 9 gateway nodes at NDN PI sites and 7 at collaborating institutions. This past year, 8 new sites joined the testbed, which now includes nodes in the US (12), China (3), South Korea (1), Japan (1), France (3), Switzerland (1), Spain (1), Italy (1) and Norway (1). Two more new sites (in .us and .at) have agreed to join which will bring the total to 26 nodes in 10 countries.

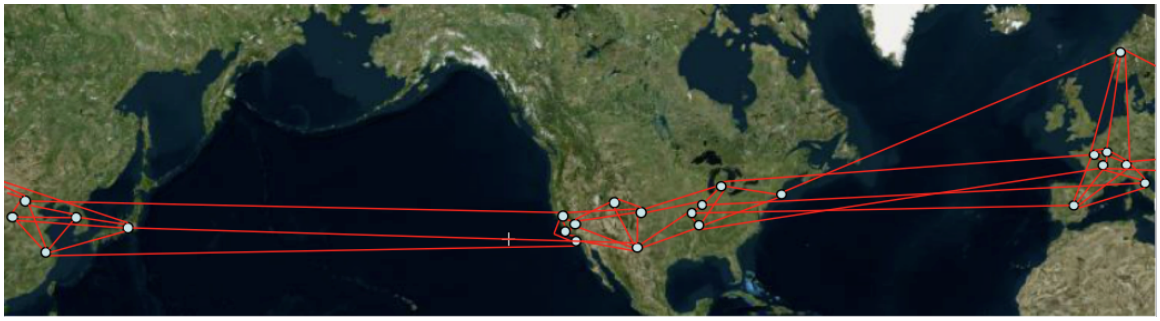
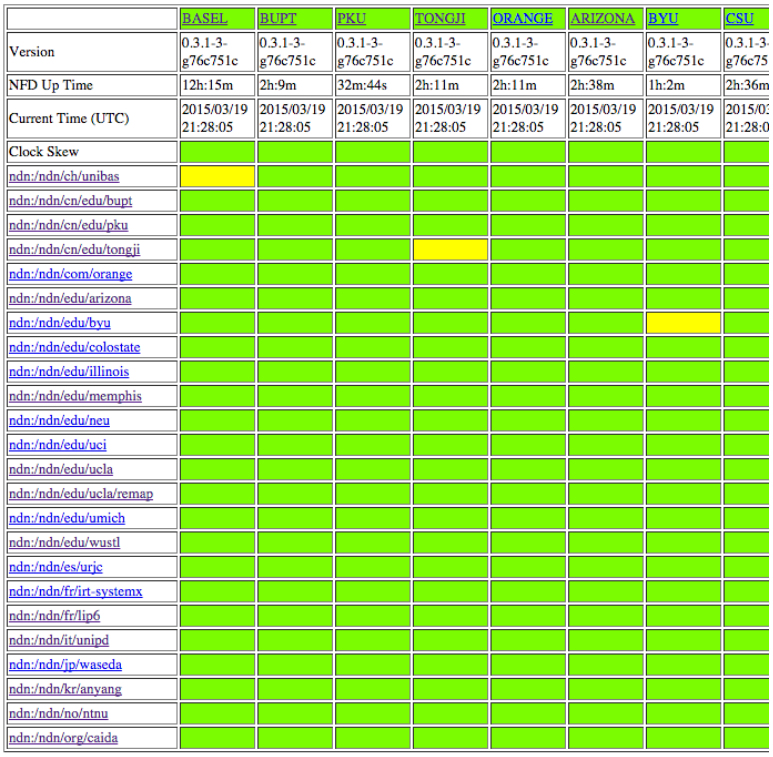


Figure 5.1: The NDN Testbed spans the globe.

The NDN team at Washington University operates and manages all testbed nodes and all integration testing and deployment activities. We enhanced our testbed monitoring tools, including redeveloping the real-time network usage map to use a pure NDN paradigm. A central server on the Washington Univer-

sity campus sends NDN Interests to client daemons that run on each testbed node. These daemons collect usage data which they return as content in data packets sent back to the server. The map tool displays this usage data on a web page at `http://ndnmap.arl.wustl.edu`. The testbed team also updates a status table every five minutes, (`http://www.arl.wustl.edu/~jdd/ndnstatus/ndn_prefix/tbs_ndnx.html`) which shows the status of each node and the state of its forwarding table with respect to the default prefixes for each node in the testbed.



	BASEL	BUPT	PKU	TONGJI	ORANGE	ARIZONA	BYU	CSU
Version	0.3.1-3-g76c751c	0.3.1-3-g76c751c	0.3.1-3-g76c751c	0.3.1-3-g76c751c	0.3.1-3-g76c751c	0.3.1-3-g76c751c	0.3.1-3-g76c751c	0.3.1-3-g76c75
NFD Up Time	12h:15m	2h:9m	32m:44s	2h:11m	2h:11m	2h:38m	1h:2m	2h:36m
Current Time (UTC)	2015/03/19 21:28:05	2015/03/19 21:28:05	2015/03/19 21:28:05	2015/03/19 21:28:05	2015/03/19 21:28:05	2015/03/19 21:28:05	2015/03/19 21:28:05	2015/0 21:28:0
Clock Skew								
ndn:/ndn/ch/unibas								
ndn:/ndn/cn/edu/bupt								
ndn:/ndn/cn/edu/pku								
ndn:/ndn/cn/edu/tongji								
ndn:/ndn/com/orange								
ndn:/ndn/edu/arizona								
ndn:/ndn/edu/byu								
ndn:/ndn/edu/colostate								
ndn:/ndn/edu/illinois								
ndn:/ndn/edu/memphis								
ndn:/ndn/edu/neu								
ndn:/ndn/edu/uci								
ndn:/ndn/edu/ucla								
ndn:/ndn/edu/ucla/remap								
ndn:/ndn/edu/umich								
ndn:/ndn/edu/wustl								
ndn:/ndn/es/urjc								
ndn:/ndn/fr/irt-systemx								
ndn:/ndn/fr/lip6								
ndn:/ndn/it/unipd								
ndn:/ndn/jp/waseda								
ndn:/ndn/kr/anyang								
ndn:/ndn/no/ntnu								
ndn:/ndn/org/caida								

Figure 5.2: A portion of the NDN Status page.

We use the testbed for external technical demonstrations, and for advancing the research agenda of the NDN-NP project. Unit testing, simulation and emulation are effective methods for debugging and functional verification, but most end-to-end evaluations need real-world deployment with real application traffic to inspire confidence. We use the testbed to debug and deploy end-user applications, such as the *ndnrtc* real-time videoconferencing application, which we have used across the NDN testbed during NDN working meetings. The testbed has also been an essential component in driving other elements of NDN research forward because it provides a means for real world testing that an emulated environment cannot adequately provide. For example, in testing NLSR, testbed usage triggered several bugs [4, 2, 3, 1] that we would not otherwise have discovered. Similarly, deployment and operation of the NDN Forwarding daemon (nfd) on the testbed allowed validation of its design and implementation, bug discovery, and recognition of the need for new features, including a per-face byte counter, and remote prefix registration.

5.2 Development of ndnSIM Version 2.0

We developed an open source, NS-3 based NDN simulator, ndnSIM, during the second year of the NDN project. Since our release of ndnSIM 1.0 [5] in summer 2012, the global NDN research community has used it to study NDN design choices and system properties. This year we upgraded ndnSIM to incorporate

the new NDN forwarding daemon (NFD) developments, the new NDN packet format, and other research advances, and to consolidate efforts of code development. We released ndnSIM 2.0 [9] in January 2015.

ndnSIM 2.0 integrates the new NDN forwarder (NFD) and ndn-cxx library (NDN C++ library with eXperimental eXtensions) codebases with minor modifications. The NDN protocol stack (ndn::L3Protocol) remains a core component of the simulator, but packet processing now directly uses the NFD source code, with the ndnSIM protocol stack serving as a redirection point to send packets to/from the simulation environment (Figure 5.3). This convergence enables researchers to simulate different NDN forwarding strategies under various deployment scenarios before putting the code into real deployment. ndnSIM 2.0 now also supports per-namespace forwarding strategy, interest selectors, and and crypto operations (as provided in ndn-cxx). The simulator uses C++ classes in a modular way to model behavior of each NDN entity: Application and Network Device Face, NFD's Forwarding Interest Table (FIB), Pending Interest Table (PIT) and Contest Store (CS), etc. It provides a more extensive collection of interfaces and support for detailed tracing of components and traffic flow.

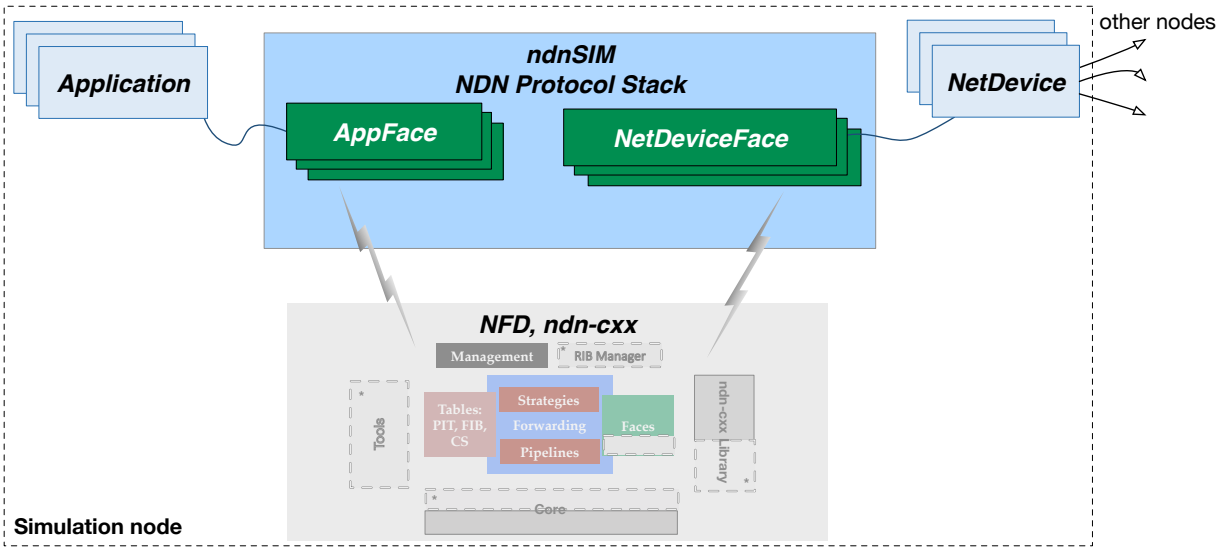


Figure 5.3: Structural diagram of ndnSIM 2.0

As of May 2015, the ndnSIM mailing list has over 290 subscribers, and according to Google Scholar, [5] has been referenced by over 130 papers. It supports a range of NDN research, including evaluating application behavior, routing protocol scalability, forwarding strategies, caching algorithm performance, security and cryptography schemes. It also supports simulation of NDN deployment on mobile, vehicular, peer-to-peer, ad-hoc and IoT environments.

5.3 Mini-NDN

Active development of NLSR has revealed the need for automatic integration testing, to avoid having to manually update testbed nodes and check whether they correctly compute and install routes. We also needed an automated way to evaluate the performance of routing algorithms. In the past we used Emulab, an emulation testbed shared by researchers, to experimentally compare link state and hyperbolic routing. But Emulab requires manual preparation of machine images for each update to any NDN library or NLSR, tedious configuration of distributed experiments, and navigating contention for Emulab resources. We are designing the Mini-NDN [10] environment with the goal of making testing and evaluation fast and effortless.

Early development of Mini-NDN started with extending Mini-CCNx [6], an emulation tool built for CCN based on Mininet [8]. Mini-NDN runs an experiment on a single machine, using a container to hold each

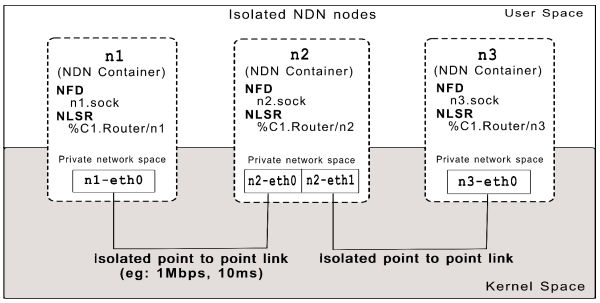


Figure 5.4: Mini-NDN's container based emulation

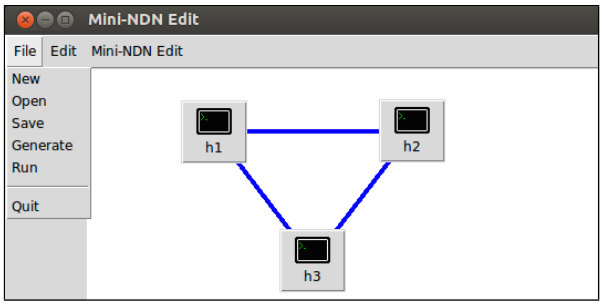


Figure 5.5: Mini-NDN's GUI

node in the network topology with its own resources (Figure 5.4). NFD and NLSR run on each Mini-NDN node. We modified the Mini-CCNx configuration file to suit our needs, such as specifying hyperbolic coordinates for NLSR. We also modified the GUI to automatically generate configuration files for NFD and NLSR (Figure 5.5). We now program NLSR experiments using Mini-NDN, which is much easier than using Emulab since we have a central point to control nodes via the GUI or the python API provided by Mininet. The Mini-NDN code is available on GitHub [10] and we use Gerrit for code review.

Performance results using Mini-NDN [7] closely approximated those of Emulab. The biggest advantage of Mini-NDN is that we can test the latest version of NLSR immediately, which has allowed us to catch and fix bugs and inconsistencies much more quickly than we could with Emulab. Using Mini-NDN as a rapid testing environment has dramatically improved and accelerated the development of NLSR.

Although our use of Mini-NDN thus far has been largely NLSR-related, researchers can program any experiment they like or test and evaluate any NDN application within the Mini-NDN framework. We plan to integrate Mini-NDN with Gerrit and Jenkins such that it can automatically run integration tests and post results to the Gerrit web interface.

References

[1] Bug #1944: Allow trailing slash for udp and tcp faceuris. `http://redmine.named-data.net/issues/1944`.

[2] Bug #2005: A router's synclogichandler should not fetch lsas for itself. `http://redmine.named-data.net/issues/2005`.

[3] Bug #2323: Sequencingmanager::splittsequenceno does not copy all bits for name lsa. `http://redmine.named-data.net/issues/2323`.

[4] Bug #2733: Rebuild adjacency lsa when face destroyed belongs to active node. `http://redmine.named-data.net/issues/2733`.

[5] Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. ndnSIM: NDN simulator for NS-3. Technical Report NDN-0005, NDN Project, July 2012 (revised October 2012).

[6] Carlos M.S. Cabral, Christian Esteve Rothenberg, and Maurício Ferreira Magalhães. Mini-ccnx: Fast prototyping for named data networking. In *3rd ACM SIGCOMM Workshop on ICN*, 2013.

[7] Hyperbolic routing evaluation results. `http://netwisdom.cs.memphis.edu/mini-ndn/`.

[8] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In *9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, 2010.

[9] Spyridon Mastorakis, Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. ndnSIM 2.0: A new version of the NDN simulator. Technical Report NDN-0028, NDN Project, January 2015.

[10] Mini-NDN GitHub. `https://github.com/named-data/mini-ndn`.

Chapter 6

Impact: Education

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

6.1 Education Philosophy and Objectives

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

The students of today will become the network architects of tomorrow. Network architectures will change with technology advances over time and today's students will determine how they change. As networked computing systems such as the Internet grow rapidly in complexity along with our dependence on them, the ability to rigorously understand the fundamentals of network communication architectures becomes more important. While we hope NDN succeeds as a network architecture, our education objectives are broader: to teach students "architectural thinking", a type of *computational thinking* that encompasses system principles, invariants, and design trade-offs, to enable students to make informed decisions on how network architecture should evolve. We encourage students to challenge architectural choices and explore alternatives.

6.2 Biweekly NDN 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. 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. This year's seminars covered the following topics:

- Jeff Burke - BOSS: Building Operating System Services
- Jeff Burke - Open mHealth "Straw man" Design for Discussion
- Obaid Amin - Hyperbolic Routing Performance Evaluation
- Jongdeog Lee - Information Maximization
- Ilya Moiseenko - Producer and Consumer API
- Alex Afanasyev - ndnSIM 2.0: A new version of the NDN simulator for NS-3

- Yanbiao Li - NDN remote registration
- Alex Afanasyev - SNAMP: Secure Namespace Mapping to Scale NDN Forwarding

6.3 Education Efforts

6.3.1 Arizona

The University of Arizona continued to include NDN material in both undergrad and graduate courses. In Fall 2014 PI Beichuan Zhang gave one introductory lecture on NDN to CS296H/496H, a research seminar for undergraduate honor students. The goal was to stimulate student interest in network architecture research. He also discussed NDN in two lectures of CS 525, a graduate-level network course. These lectures covered basic concepts of NDN operations and research issues. The course also contains materials on other future Internet designs and other ICN designs to provide a more comprehensive view of the research area. In Spring 2015 PI Beichuan Zhang spent one lecture on NDN in CS 425, the undergrad network course. The PI and students also helped prepare and deliver the NDN tutorial at ICN 2015.

6.3.2 Colorado State University

CSU continued to teach NDN and assign NDN-related programming assignments in both the graduate and undergraduate networking courses. Its undergraduate class in 2014 used NDN assignments. PI Papadopoulos taught the graduate class in Spring 2015, which used NFD and the ndn-cxx C++ library. The transition from the old CCNx codebase required us to prepare skeleton NDN programs [1] to help students get their bearings with the new library. Both classes had strong graduate student participation in terms of teaching NDN and NDN-related topics such as content poisoning. Both classes used templates developed at CSU and published on the NDN website. CSU also led the development of tutorials at ICN 2014 and GENI 21.

6.3.3 Memphis

During Spring 2015, PI Lan Wang used six lectures to introduce NDN concepts to students in her COMP4/6310 (Wireless and Mobile Computing) course. In the first three lectures, she gave an overview of the NDN architecture and various NDN applications. Her lab staff showed the students how to install the NDN platform, as well as how to use the jNDN library to write simple NDN programs. In the next three lectures, she covered several wireless NDN applications, including lighting control, building management and vehicular networking, to give students concrete new design patterns in NDN. The students also worked on a course project to design and implement a shared white-board application in NDN.

6.3.4 UCLA

Lixia Zhang taught a graduate course "CS217A: Internet Architecture & Protocols" during Fall 2014 quarter, which compared NDN with today's Internet TCP/IP architecture. Students from the NDN team offered additional tutorials on the installation of an NDN prototype implementation. About one third of the students in the class took optional term projects on NDN research topics, including:

- NDN Radio: similar to iTunes FM radio stations that are available for streaming, the goal was to grab audio streams from a website such as `http://radio.com/station-directory/` and re-publish them over NDN. This project highlighted differences in application architecture iTunes connects clients to a single (or a few) IP address, whereas in NDN Radio, clients pick and choose among multiple producers by leveraging NDN network layer capabilities of issuing interest packets with different name prefixes.
- Adaptation of a REST-based communication model to a data dissemination approach inspired by NDN.
- Using an NDN traffic generator to produce a baseline performance measurement of NFD, NDN's Forwarding Daemon on the NDN testbed.

- Designing an extension of the NDN Link Protocol (NDNLP) for failure detection, so that NFD can quickly try an alternative path. We are incorporating the design into NDNLP version 2.

Postdoc and graduate students from the NDN team served as points of contact for each topic. All four projects contributed to the NDN developments. The first two topics became the students' master projects, which will be finished during spring 2015 to fulfill the degree requirements.

During the spring 2015 quarter Lixia Zhang is teaching CS217B on "Advanced Topics in Internet Research", a graduate seminar course focusing on NDN architecture design and application development. In addition to covering the literature on NDN design, the course also aims to trace the basic NDN architecture ideas back from a series of research papers spanning the last few decades (e.g. [3, 6, 4]). Students also learn about other architectural designs under NSF's FIA program (XIA [5], Mobility First [2]). In parallel to class discussions, students are also actively pursuing research topics as listed at `http://www.cs.ucla.edu/classes/cs217/ProjectList`.

References

[1] CS 557 NDN Hello World.

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

[3] 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.

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

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

[6] 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*.

Chapter 7

Impact: Expansion of NDN Community

The NDN project continues to attract attention from the global networking community. The PIs participated in conferences and speaking engagements, as listed below, and engaged 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 codebase.

7.1 NDN Tutorials

The NDN platform provides a rich software environment for researchers and developers to experiment with both the architecture research and future Internet applications. We created tutorials to ease newcomers into experimenting with the new forwarder, NFD, extending NDN libraries, as well as writing NDN applications. We created tutorials that target developers familiar with network programming, but are new to NDN. The materials consist of two pieces: a written description that walks, nearly line by line, through example applications and "solution" code for those wishing to experiment with the NDN platform.

7.1.1 Tutorial at ACM ICN 2014

We presented an NDN tutorial during the ICN 2014 conference in Paris, France. The tutorial was a joint effort from virtually all members of the NDN team. All material used in the tutorial is available on the main NDN page, including a few video presentations. We begin by walking developers through a "hello world" example that includes producer and consumer applications using PyNDN2 [4]. The developer learns about Interests/Data, their various fields and attributes, name prefix registration, how to run NFD, and a typical NDN software design pattern. Next, the developer extends their producer and consumer into more interesting applications. Specifically, the tutorial covers logical content to data segmentation (i.e. chunking) and Interest pipelining. The tutorial materials also show developers how to extend the new forwarder, NFD, by writing custom forwarding strategies. We walk developers through two strategies: a random load balancer and a weighted load balancer. Again, we follow an evolutionary instruction model where the developer extends their random load balancer into the weighted load balancer. However, there is a secondary benefit of these specific examples, demonstrating a stateless and stateful forwarding strategy, respectively.

Our tutorial materials continue the NDN project's commitment to producing free software and are publicly available on Github. Our initial tutorial targets were the first NDN community meeting, NDNcomm 2014, and the ICN 2014 conference's NDN tutorial session. We bundled and distributed the NDNcomm and ICN tutorial materials as part of special NFD code repositories [3] and [2], respectively. We encourage tutorial improvements from the community and have accepted contributions.

Figure 7.1: Professor Lan Wang (University of Memphis) speaking at NDNComm 2014.

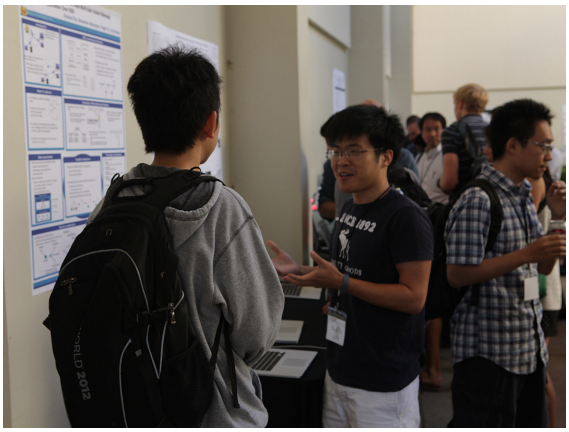


Figure 7.2: UCLA Graduate Student Yingdi Yu explaining his poster at NDNComm 2014.

7.1.2 GENI 21 Tutorial

We produced a streamlined version of our ICN tutorial materials for the 21st GENI Engineering Conference (GEC 21). This version of the tutorial consisted of running the previously mentioned applications and forwarding strategies over an NDN testbed-like topology. However, this time we had the challenge of ensuring our audience could follow along and run the scenarios themselves. As such, we also had to develop several deployment and configuration scripts, including an Ubuntu VM image with the standard NDN software pre-installed and a GENI RSPEC, so attendees could easily setup the scenarios on their individual GENI slices. Consequently, we distribute the GEC 21 tutorial materials as a separate repository [1].

7.2 First NDN Community meeting

The team organized and executed the first NDN Community Meeting (held September 3-5, 2014 at UCLA), which brought over 80 researchers from academia and industry together to discuss the architecture, future research, and important applications. Attendees included students, faculty, and staff from the NDN campuses, as well as other universities conducting NDN research and industry including Ericsson, Cisco, Huawei, Panasonic Research, PARC, Intel, and others. The full agenda and meeting materials are hosted on the Named Data Networking First Community Meeting website, `http://www.caida.org/workshops/ndn/1409/`.

7.3 Establishment of the NDN Consortium

In a significant milestone for NDN, we launched the NDN Consortium in September 2014 to promote a vibrant open source ecosystem of research and experimentation around NDN by providing developer support tools, organizing community meetings, generating outreach activities to orchestrate the effort from the broader community in advancing NDN research and development. The consortium will facilitate community efforts on NDN architecture development and promote its usage to solve application problems that do not have good TCP/IP-based solutions. In the first year, activities included a short research review meeting, adjacent to the (currently annual) NDN Community meeting [6], to provide Consortium members early access to research results. That said, the NDN project aims for public dissemination of research results, and opportunities for joint research, including academic sites hosting industry consortium members and vice-versa to engage in research. The NDN Consortium continues to attract organizations that seek an open innovation model and codebase that will foster and accelerate research on information-centric networking, and can demonstrate the benefits of ICN over current solutions. To that end, we have committed to using a GPL license for the core NDN protocol implementations, to create a layer where everyone will share copyrights and patents, everyone

Industrial	Academic
Alcatel-Lucent	Anyang University, Korea
Brocade	Colorado State University
Cisco Systems	Waseda University, Japan
Fujitsu Laboratories of America	Memphis University
Huawei/FutureWei Technologies, Inc.	Tongji University, China
Intel Corporation	University of Arizona
Panasonic Corporation	University of Basel
Verisign, Inc.	University of California, Los Angeles
The MITRE Corporation	University of California, San Diego
	University of Illinois
	University of Michigan
	Washington University
	Tsinghua University, China
	University of Basel, Switzerland
	Pierre-and-Marie-Curie University, France

will be on equal terms, and users will have rights to share, modify, and distribute what they learn. We fully expect and embrace differentiation at other layers in the stack by commercializers targeting productization, similar to how the TCP/IP and Linux ecosystems have generated vast technological innovation and broad impacts.

On 12 February 2015 we held our first NDN Consortium Board Meeting at UCLA with teleconferencing facility for remote participants. PI Jeffrey Burke described the initial structure of the Consortium Board and how it would govern the consortium through the management team in place for the NSF NDN-NP award. We will evolve governance structure as we explore the option of turning the Consortium into a 501(c)(6)[7] sometime in 2016. The Board provided positive feedback as well as a list of priorities they would like to see the Consortium support: increased dissemination of research direction and results; opportunities for consortium members and community engagement; sustainable, increased funding for the NDN research effort; formalizing the open source code development process; and annual NDNcomm meetings (NDNcomm 2015 will be 28-29 September at UCLA). More information on the *Named Data Networking Consortium* is at `http://named-data.net/consortium`. Table 7.3 lists the NDN Consortium members as of May 2015.

7.4 The First ACM Information Centric Networking Conference

After three successful ACM SIGCOMM Information Centric Networking Workshops (ICN 2011-2013) and two IEEE INFOCOM workshops on ICN-related topics (NOMEN 2012-2013), which NDN team members helped organize, ACM launched the first Information Centric Networking Conference in September 2014 in Paris, France. Most NDN PIs served on the Technical Program Committee for ACM ICN 2014. Lixia Zhang served as TPC co-chair. The NDN team presented 2 (of 17) papers, 3 (of 8) posters, and a demo. We also presented a half-day tutorial introducing NDN and its software architecture. The high number of ICN 2014 submissions and registered attendees led to ACM's decision to sponsor the next ICN conference in San Francisco in September 2015, a strong indication of the growing popularity of this field.

7.5 Reaching Out: NDN Presentations

NDN team member Jeff Burke gave a keynote at IFIP 2014 to continue the team's outreach effort to the networking community. The talk, entitled "Why architecture matters to everyone: Creativity on the Future Internet" introduced NDN, outlined the team's approach, and suggested a path for formally evaluating the architecture's impact on application developers. common starting usability [5].

NDN team member Lixia Zhang gave a LINCS[1] seminar presentation to a full room in June 2014 on "The Art of Packet Format Design". The talk explained how the new NDN packet format design took into account the successes and lessons from both today's Internet protocol packet format designs as well as the lessons learned from their evolution over the last 30+ years.

Lixia Zhang gave an invited talk at VeriSign in August 2014 on "IoT Networking via NDN", which explained why an information-centric architecture such as NDN, provides a better fit for IoT applications than TCP/IP, and how NDN's data naming simplifies overall system design, data security, access control, and resource discovery. The talk was well attended and a few VeriSign researchers indicated their interest in investigating into this area.

Christos Papadopoulos gave invited talks at about seven venues including Internet2 meetings, climate workshops, and scientist groups. These talks introduced NDN and outlined how the new architecture simplifies Big Data applications in such domains (Section 2.4). Examples include translation from existing, ad-hoc namespaces to NDN hierarchical namespaces, distributed publishing for location independence, data discovery, and robust failover.

Presentations

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

1. Tarek Abdelzaher, "Foundations of Information Distillation for Self-aware Computing," Invited Speaker, Dagstuhl Seminar on Model-driven Self-aware Computing, Schloss Dagstuhl, Germany, January 2015.
2. Tarek Abdelzaher, "The Power of Analytics," Keynote Talk, International Conference on Smart Computing (SmartComp), Hong Kong, November 2014.
3. Tarek Abdelzaher, "Notes on Cyber Physical Systems Education," National Academy of Sciences, Washington DC, October 2014.
4. Jeff Burke, "Named Data Networking, an Architecture for the Future Internet." World Presidents Organization, Southern California Chapter. Invited speaker as part of the WPO Visit to UCLA "Changing Los Angeles and the World: Great Innovation in Our Own Backyard." November 20, 2014.
5. Jeff Burke, "ICN at the Edge." Invited panel participation. ACM Information Centric Networking, September 24, 2014, Paris, France.
6. Jeff Burke, "Why Architecture Matters to Everyone: Creativity on the Future Internet." Cornell Tech, Cornell Connected Media Symposium, August 28, 2014.
7. Jeff Burke, "Why architecture matters to everyone: Creativity on the Future Internet", keynote, IFIP Networking 2014, Trondheim, Norway, June 3, 2014.
8. Jeff Burke, "The Conditions of Algorithmic Life, Occams Hourglass, Mellon Research Initiative in Digital Cultures", invited talk, UC Davis, May 15-16, 2014.
9. Jeff Burke, "Named Data Networking and the Internet of Everything", invited talk, Huawei Corporate-level Science & Technology Workshop, Huawei headquarters, Shenzen, China, May 13, 2014.
10. Patrick Crowley, University of Memphis Computer Science Colloquium Series, "Named Data Networking," Memphis, TN. October 24, 2014.
11. Patrick Crowley, IEEE 15th Intl Conference on High Performance Switching and Routing, "Security, Privacy & Future Networks", Vancouver, Canada. July 3, 2014.
12. Christos Papadopoulos, Cathie Olchanowsky, Susmit Shannigraphi, Steve DiBenedetto, David Randall, Kelly Wittmeyer, Don Dazlich, and Mark Branson, invited talk, NDN Community Meeting, "Supporting Climate Applications over Named Data Networking," September 2014.
13. Christos Papadopoulos, Cathie Olchanowsky, Susmit Shannigraphi, Steve DiBenedetto, David Randall, Kelly Wittmeyer, Don Dazlich, and Mark Branson, invited talk, Cooperative Institute for Research in

[1]Laboratory of Information, Networking and Communication Sciences, Paris, France.

the Atmosphere (CIRA), "Supporting Climate Applications over Named Data Networking," September 9, 2014, Fort Collins, CO.

14. Christos Papadopoulos, Cathie Olchanowsky, Susmit Shannigraphi, Steve DiBenedetto, David Randall, Kelly Wittmeyer, Don Dazlich, and Mark Branson, invited talk, FTW Workshop, "Supporting Climate Applications over Named Data Networking," July 16, 2014, Boulder, CO.
15. Christos Papadopoulos, Cathie Olchanowsky, Susmit Shannigraphi, Steve DiBenedetto, David Randall, Kelly Wittmeyer, Don Dazlich, and Mark Branson, invited paper, LANMAN Workshop, "Supporting Climate Applications over Named Data Networking," May 23, 2014 Reno, NV.
16. Christos Papadopoulos, Cathie Olchanowsky, Susmit Shannigraphi, Steve DiBenedetto, David Randall, Kelly Wittmeyer, Don Dazlich, and Mark Branson, invited talk, NSF CC-NIE PI Workshop, "Supporting Climate Applications over Named Data Networking," May 1, 2014 Arlington, VA.
17. Lan Wang, Invited Talk: "NDN Architectural Development and Routing Design," Donaghey College of Information Science and Technology, University of Arkansas, Little Rock, Nov. 2014
18. Lan Wang, "NLSR: Named-data Link State Routing Protocol," NDN Community Meeting, Sept. 2014
19. Lan Wang, "Named Data Networking," Computer Science Camp for Girls, University of Memphis, July 2014
20. Lan Wang, Invited Talk: "NDN Architectural Development and Routing Design," Chinese Academy of Sciences, Computer Network Information Center, June 2014
21. Lixia Zhang, "Named Data as the Center of Internet Architecture", IEEE INFOCOM Panel on "What should the future Internet be centered: Information, service, mobility, or user?", April 2015.
22. Lixia Zhang, "Tackling the Challenge of Developing A New Internet Architecture", invited talk, Peking University, April 23 2015.
23. Lixia Zhang, "Toward an Interconnected World via Named Data Networking", invited talk, Atos Scientific Community Meeting, Paris, France, February 2015.
24. Lixia Zhang, invited talk, "IoT Networking via NDN", VeriSign, August 2014.
25. Lixia Zhang, "Named Data Networking", presented to the Large Scale Networking (LSN) Coordinating Group, August 2014.
26. Beichuan Zhang, Keynote speech, "Information-Centric Networking for Internet of Things", First International Conference on Informative & Cybernetics for Computational Social Systems, Oct. , 2014.
27. Beichuan Zhang, invited talk, "Named Data Networking", Xi An Jiaotong Un., China, June 2014.

References

[1] GEC 21 NDN Tutorial Materials.

[2] ICN 2014 NDN Tutorial Materials.

[3] NDNcomm 2014 Tutorial Materials.

[4] PyNDN2.

[5] Thomas R. G. Green and Marian Petre. Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *Journal of visual languages and computing*, 7(2):131–174, 1996.

[6] kc claffy, Josh Polterock, Alexander Afanasyev, Jeff Burke, and Lixia Zhang. The First Named Data Networking Community Meeting (NDNcomm). *ACM SIGCOMM Computer Communication Review (CCR)*, April 2015.

[7] John Francis Reilly, Carter C. Hull, and Barbara A. Braig Allen. Irc 501(c)(6) organizations. Internal Revenue Service. `http://www.irs.gov/pub/irs-tege/eotopick03.pdf`.

Chapter 8

Publications

Listed below are publications by NDN-NP team members during the first year of the NP project (1 May 2014 - 30 April 2015).

1. "InfoMax: An Information Maximizing Transport Layer Protocol for Named Data Networks," by Jongdeog Lee, Akash Kapoor, Md Tanvir Al Amin, Zeyuan Zhang, Radhika Goyal, Tarek Abdelzaher, and Zhahao Wang. To appear, 24th International Conference on Computer Communications and Networks (ICCCN), Las Vegas, NV, August 2015.
2. "Navigo: Interest Forwarding by Geolocations in Vehicular Named Data Networking," by Giulio Grassi, Davide Pesavento, Giovanni Pau, Lixia Zhang, Serge Fdida. to appear, IEEE World of Wireless Mobile and Multimedia Networks (WoWMoM), June 2015.
3. "Reliably Scalable Name Prefix Lookup", by Haowei Yuan and Patrick Crowley. to appear, ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS 2015), May 7-8, 2015, Oakland, California, USA.
4. "Synchronizing Namespaces with Invertible Bloom Filters" by Wenliang Fu, Hila Ben Abraham, and Patrick Crowley to appear, ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS 2015), May 7-8, 2015, Oakland, California, USA.
5. "Named Data Networking in Climate Research and HEP Applications", by Susmit Shannigrahi, Artur Barczuk, Christos Papadopoulos, Alex Sim, Inder Monga, Harvey Newman, John Wu, and Edmund Yeh. 21st International Conference on Computing in High Energy and Nuclear Physics (CHEP2015), Okinawa, Japan, April 2015.
6. "SNAMP: Secure Namespace Mapping to Scale NDN Forwarding", by Alexander Afanasyev, Cheng Yi, Lan Wang, Beichuan Zhang, and Lixia Zhang. 18th IEEE Global Internet Symposium, April 2015.
7. "RepoSync: Combined Action-Based and Data-Based Synchronization Model in Named Data Network," by Weiqi Shi and Alexander Afanasyev. 18th IEEE Global Internet Symposium, April 2015.
8. "The First Named Data Networking Community Meeting (NDNcomm)", by Kim Claffy, Joshua Polterock, Alexander Afanasyev, Jeff Burke, Lixia Zhang. ACM SIGCOMM Computer Communication Review (CCR), April 2015.
9. "VIP: Joint traffic engineering and caching in Named Data Networks", by Edmund Yeh, Tracey Ho, Ying Cui, Michael Burd, Ran Liu, Derek Leong. 2015 International Conference on Computing, Networking and Communications (ICNC), February 2015
10. "On the Role of Routing in Named Data Networking", by Cheng Yi, Jerald Abraham, Alexander Afanasyev, Lan Wang, Beichuan Zhang, Lixia Zhang. 1st ACM Conference on Information-Centric Networking, Paris, France, September 2014.
11. "VIP: A Framework for Joint Dynamic Forwarding and Caching in Named Data Networks", by Edmund Yeh, Tracey Ho, Ying Cui, Michael Burd, Ran Liu, Derek Leong. 1st ACM Conference on Information-Centric Networking, Paris, France, September 2014.

12. "Consumer-Producer API for Named Data Networking" by Ilya Moiseenko and Lixia Zhang. Poster, 1st ACM Conference on Information-Centric Networking, Paris, France, September 2014.
13. "Kite: A Mobility Support Scheme for NDN" by Yu Zhang, Hongli Zhang, Lixia Zhang. Poster, 1st ACM Conference on Information-Centric Networking, Paris, France, September 2014.
14. "iSync: A High Performance and Scalable Data Synchronization Protocol for Named Data Networking" by Wenliang Fu, Hila Ben Abraham, Patrick Crowley. Poster, 1st ACM Conference on Information-Centric Networking, Paris, France, September 2014.
15. "Named Data Networking" by L. Zhang, A. Afanasyev, J. Burke, claffy, L. Wang, V. Jacobson, P. Crowley, C. Papadopoulos, B. Zhang. ACM SIGCOMM Computer Communication Review (CCR), July 2014.
16. "Securing Building Management Systems Using Named Data Networking" by Wentao Shang, Qiuhan Ding, Alessandro Marianantoni, Jeff Burke, and Lixia Zhang. IEEE Network, May/June 2014.
17. "The Information Funnel: Exploiting Named Data for Information-maximizing Data Collection" by Shiguang Wang, Tarek Abdelzaher, Santhosh Gajendran, Ajith Herga, Sachin Kulkarni, Shen Li, Hengchang Liu, Chethan Suresh, Abhishek Sreenath, Hongwei Wang, William Dron, Alice Leung, Ramesh Govindan, John Hancock. In Proc. 10th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), Marina Del Rey, CA, May 2014.
18. "Scalable Pending Interest Table Design: From Principles to Practice" by Haowei Yuan and Patrick Crowley. INFOCOM 2014, Toronto, Canada, April-May 2014.
19. "VANET via Named Data Networking" by Giulio Grassi, Davide Pesavento, Giovanni Pau, Rama Vuyyuru, Ryuji Wakikawa, Lixia Zhang. IEEE INFOCOM 2014 Workshop on Name Oriented Mobility (NOM), Toronto, Canada, April-May 2014.

NDN Technical Reports

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

- "Schematizing and Automating Trust in Named Data Networking" by Yingdi Yu, Alexander Afanasyev, David Clark, kc claffy, Van Jacobson, and Lixia Zhang. NDN Technical Report NDN-0030, Revision 2: June 2, 2015.
- "NFD Developers Guide" by Alexander Afanasyev, Junxiao Shi, Beichuan Zhang, Lixia Zhang, Ilya Moiseenko, Yingdi Yu, Wentao Shang, Yanbiao Li, Spyridon Mastorakis, Yi Huang, Jerald Paul Abraham, Steve DiBenedetto, Chengyu Fan, Christos Papadopoulos, Davide Pesavento, Giulio Grassi, Giovanni Pau, Hang Zhang, Tian Song, Haowei Yuan, Hila Ben Abraham, Patrick Crowley, Syed Obaid Amin, Vince Lehman, and Lan Wang. NDN Technical Report NDN-0021. Revision 4, May 12, 2015.
- "Packet Fragmentation in NDN: Why NDN Uses Hop-By-Hop Fragmentation (NDN Memo)" by Alexander Afanasyev, Junxiao Shi, Lan Wang, Beichuan Zhang, and Lixia Zhang. NDN Memo, Technical Report NDN-0032, Revision 1, May 12, 2015.
- "NDNlive and NDNtube: Live and Prerecorded Video Streaming over NDN" by Lijing Wang, Ilya Moiseenko, and Lixia Zhang. NDN Technical Report NDN-0031, Revision 1: April 30, 2015.
- "Public Key Management in Named Data Networking" by Yingdi Yu. NDN, Technical Report NDN-0029, Revision 1: April 20, 2015.
- "Athena: A Configurable Validation Framework For NDN Applications" by Yingdi Yu NDN, Technical Report NDN-0030, Revision 1: April 20, 2015.
- "ndnSIM 2.0: A new version of the NDN simulator for ns-3" by Spyridon Mastorakis, Alexander Afanasyev, Ilya Moiseenko and Lixia Zhang NDN Technical Report NDN-0028, Revision 1: January 15, 2015.

- "NDNFS: An NDN-friendly File System" by Wentao Shang, Zhe Wen, Qiuhan Ding, Alexander Afanasyev, and Lixia Zhang. NDN Technical Report NDN-0027, Revision 1: October 27, 2014.
- "Fetching content in Namded Data Networking" by Ilya Moiseenko. NDN Technical Report NDN-0025, Revision 1: September 25, 2014.
- "NDN Common Client Libraries" by Jeff Thompson and Jeff Burke. NDN Technical Report NDN-0024, Revision 1: September 5, 2014.
- "An Endorsement-based Key Management System for Decentralized NDN Chat Application" by Yingdi Yu, Alexander Afanasyev, Zhenkai Zhu, and Lixia Zhang. NDN Technical Report NDN-0023, Revision 1: July 22, 2014.
- "NDN Technical Memo: Naming Conventions" by NDN Project Team. NDN Technical Report NDN-0022, Revision 1: July 21, 2014.
- "NFD Developers Guide" by Alexander Afanasyev, Junxiao Shi, Beichuan Zhang, Lixia Zhang, Ilya Moiseenko, Yingdi Yu, Wentao Shang, Yi Huang, Jerald Paul Abraham, Steve DiBenedetto, Chengyu Fan, Christos Papadopoulos, Davide Pesavento, Giulio Grassi, Giovanni Pau, Hang Zhang, Tian Song, Haowei Yuan, Hila Ben Abraham, Patrick Crowley, Syed Obaid Amin, Vince Lehman, and Lan Wang. NDN Technical Report NDN-0021, Revision 1: July 1, 2014.
- "Kite: A Mobility Support Scheme for NDN" by Yu Zhang, Hongli Zhang, and Lixia Zhang. NDN Technical Report NDN-0020, Revision 1: June 3, 2014.