Named Data Networking (NDN) Project
2012 - 2013 Annual Report

Principal Investigators

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

Beichuan Zhang
University of Arizona

Gene Tsudik
University of California, Irvine

Kim Claffy and Dmitri Krioukov
University of California, San Diego

Dan Massey and Christos Papadopoulos
Colorado State University

Paul Ohm
University of Colorado

Tarek Abdelzaher
University of Illinois at Urbana-Champaign

Katie Shilton
University of Maryland

Lan Wang
University of Memphis

Edmund Yeh
Northeastern University

Ersin Uzun and Glenn Edens
Palo Alto Research Center (PARC)

Patrick Crowley
Washington University

Contents

FIA: Collaborative Research: Named Data Networking (NDN) 2013 Annual Report

Executive Summary

Today's Internet's *hourglass* architecture centers on a *universal* network layer (i.e., IP) which implements the minimal functionality necessary for global interconnectivity. This thin waist enabled the Internet's explosive growth by allowing both lower and upper layer technologies to innovate independently. However, IP was designed to create a *communication network*, where packets named only communication endpoints. Sustained growth in e-commerce, digital media, social networking, and smartphone applications has led to dominant use of the Internet as a *distribution network*. Distribution networks are fundamentally more general than communication networks, and solving distribution problems via a point-to-point communication protocol is complex and error-prone.

The NDN project proposes an evolution of the IP architecture that generalizes the role of this thin waist, such that packets can name objects other than communication endpoints. The name in an NDN packet can be 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 end-to-end communication problems but also content distribution and control problems. Based on three decades of experience with the strengths and limitations of the current Internet architecture, the design also builds in fundamental security primitives (via signatures on all named data) and self-regulation of network traffic (via flow balance between Interest and Data packets). We recognize that any new architecture must be incrementally deployable over the current Internet, and we explicitly consider factors that will facilitate user choice and competition as the network evolves.

Our research challenge in this project is to instantiate this architectural framework into a prototype platform capable of solving real problems, particularly in application areas poorly served by today's Internet protocol stack. In the third year we advanced our research agenda in five areas: applications, routing, forwarding, security, and fundamental theory to analyze networks. Milestones achieved this year include:

1. a building automation system in two UCLA buildings using industry-standard hardware; four peer-to-peer, serverless applications that highlight NDN's capabilities, including gaming, chat, conferencing, and file sharing applications; techniques to exploit application-specific naming information to optimize data distribution and caching
2. many new libraries to support experimentation with common client functionality, browser-based application development, security requirements, and an NDN-based file system (NDNFS);
3. the Named-data Link State Routing Protocol (NLSR) which also supports hyperbolic routing by disseminating hyperbolic coordinates using its link state announcements;
4. a web-based monitoring tool that retrieves a network status page over the NDN testbed;
5. expansion of the open source NDN simulator, ndnSIM, and support for a now global user community;
6. a new forwarding structure together with a highly scalable distributed forwarding engine, and a novel Pending Interest Table that guarantees packet delivery with a compact storage representation;
7. DoS mitigation mechanisms, and efficient methods for content authentication/verification;
8. a Virtual Interest Packet (VIP) theoretical framework with a proof of its throughput optimality and joint caching/forwarding solution stability.
9. a demonstration (in Beijing) of live NDN video streaming to approximately 1000 clients spread in five continents, together with real time control of the lighting system at UCLA

To provide a more agile development platform for our research, this year we forked a version of the CCNx code base. We created a github repository and published documentation on our renovated web site (www.named-data.net) explaining the technical differences between CCNx and NDNx. This coming year we will continue to integrate research results into this code base, deploy it on the testbed, and support community use. Most participating sites have received a one-year extension to continue development, implementation and testing of NDN software libraries and applications, as well as to continue research on appropriate naming schemes, distributed data synchronization, trust management, routing and forwarding – core issues in the named-data networking architecture. Van Jacobson left PARC in October 2012 and is now a full Adjunct Professor at UCLA where he continues to serve as lead architect for the project.

Chapter 1

Architecture Overview

NDN is an entirely new architecture, but one whose design principles are derived from the successes of today's Internet, reflecting our understanding of the strengths and limitations of the current Internet architecture, and one that can be rolled out through incremental deployment over the current operational Internet.

Today's Internet's *hourglass* architecture centers on a *universal* network layer (i.e., IP) which implements the minimal functionality necessary for global interconnectivity. This thin waist enabled the Internet's explosive growth by allowing both lower and upper layer technologies to innovate independently without unnecessary constraints. However, IP was designed to create a *communication network*, where the only entities that could be named in its packets were communication endpoints. Recent growth in e-commerce, digital media, social networking, and smartphone applications has resulted in the Internet primarily being used as a *distribution network*. Distribution networks are fundamentally more general than communication networks, and solving distribution problems via a point-to-point communication protocol is complex and error prone.

The NDN architecture retains the same hourglass shape, but transforms the thin waist to focus on data directly rather than its location. More specifically, NDN changes the semantic of network communication from *delivering a packet to a given destination address* to *retrieving data identified by a given name* (Figure 1.1). The design is also guided by the following principles.

- *Security must be built into the architecture.* Security measures in the current Internet architecture are an afterthought which do not meet the demands of today's diverse environment. NDN provides a fundamental security building block *right at the thin waist* by signing all named data.
- The *end-to-end principle* [2] underlying the TCP/IP architecture enabled development of robust applications in face of unexpected failures. NDN retains and expands this principle by securing data end-to-end.

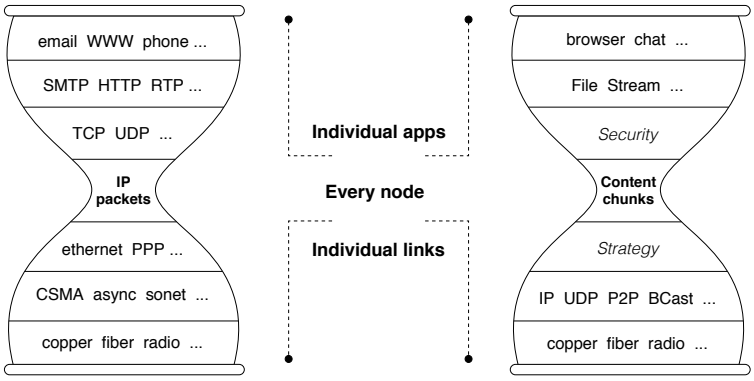

Figure 1.1: Internet and NDN Hourglass Architectures

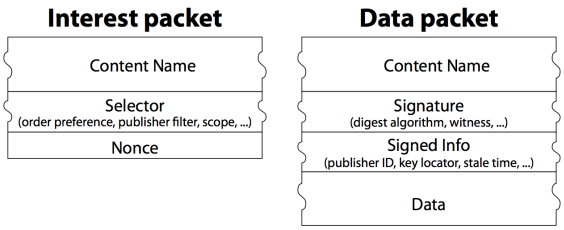

Figure 1.2: Packets In the NDN Architecture.

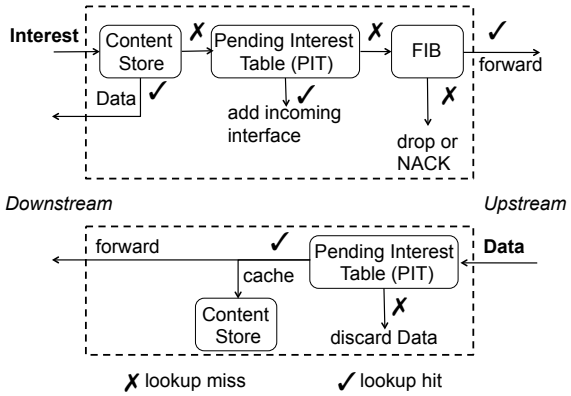

Figure 1.3: Forwarding Process at an NDN Node.

- *Network traffic must be self-regulating.* Flow-balanced data delivery is essential to the stability of large systems. Unlike IP's open-loop packet delivery, NDN designs a flow-balance feedback loop into the thin waist.
- *The architecture should facilitate user choice and competition wherever possible.* Although not considered in the original Internet design, global deployment has taught us that "architecture is not neutral." [1] NDN makes a conscious effort to empower end users and facilitate competition.

Communication in NDN is driven by the receiving ends, *i.e.*, the data consumers, through the exchange of two types of packets: *Interest* and *Data* (see Figure 1.2). Both types of packets carry a name that identifies a piece of data that can be transmitted in one Data packet. To receive data, a consumer puts the name of desired data into an *Interest* packet and sends it to the network. Routers use this name to forward the Interest toward the data producer(s). Once the Interest reaches a node N that has the requested data, node N will return a *Data* packet that contains both the name and the content, together with a signature by the producer's key which binds the two (Figure 1.2). This Data packet follows in reverse the path taken by the Interest to get back to the requesting consumer.

To carry out the above Interest and Data packet forwarding functions, each NDN router maintains three major data structures: a *Forwarding Information Base (FIB)*, a *Pending Interest Table (PIT)*, and a *Content Store* (Fig. 1.3). The FIB is populated by a name-prefix based routing protocol and is used to guide Interests towards data producers. The PIT stores all Interests that have been forwarded but not satisfied yet. It records the Interests name, incoming interface(s) and outgoing interface(s). When a router receives multiple Interests with the same name from downstream consumers, it forwards only the first one upstream towards the data producer. The Content Store is a temporary cache of Data packets that the router has received. Because an NDN Data packet is meaningful independent of where it comes from or where it is forwarded, it can be cached to satisfy future Interests.

When an Interest packet arrives, an NDN router first checks the Content Store for matching data; if it exists the router returns the Data packet on the interface from which the Interest came. Otherwise the router looks up its PIT, and if a matching entry exists, it simply records the Interest's incoming interface in the PIT entry. In the absence of a matching PIT entry, the router will look up its FIB and forward the Interest towards the data producer(s).

When a Data packet arrives, an NDN router finds the matching PIT entry and forwards the data to all downstream interfaces listed in the PIT entry. It then removes that PIT entry, and caches the Data in the Content Store. Data packets always take the reverse path of Interests, and one Interest packet results in one Data packet on each link, providing *flow balance*. Neither Interest nor Data packets carry any host or interface addresses (such as IP addresses); Interest packets are forwarded toward data producers based on the names carried in them, and Data packets are forwarded to consumers based on the PIT state information set up by the Interests at each hop.

1.1 Names

NDN names are *opaque* to the network, *i.e.*, routers do not know the meaning of a name, although they know the boundaries between components in a name. This allows each application to choose the naming scheme that fits its needs and allows the naming schemes to evolve independently from the network. However, the NDN design does assume hierarchically *structured* names, e.g., a video produced by UCLA may have the name `/ucla/videos/demo.mpg`, where '/' delineates name components in text representations, similar to URLs. This hierarchical structure allows applications to represent relationships between data elements. For example, segment 3 of version 1 of the video might be named `/ucla/videos/demo.mpg/1/3`. It also allows name aggregation: In this example `UCLA` could correspond to an autonomous system from which the video originates. Flat names can be accommodated as a special case and useful in local environments, however hierarchical name spaces are essential in scaling the routing system. Naming conventions among data producers and consumers, *e.g.*, to indicate versioning and segmentation, are specific to applications.

To retrieve dynamically generated data, consumers must be able to *deterministically* construct the name for a desired piece of data without having previously seen the name or the data. Either: (1) a deterministic algorithm allows the producer and consumer to arrive at the same name based on information available to both, or (2) *Interest selectors* in conjunction with *longest prefix matching* retrieve the desired data through one or more iterations. Our experience so far suggests that a simple set of selectors can support retrieving data with partially known names. For example, a consumer wanting the first version of the `demo.mpg` video may request `/ucla/videos/demo.mpg/1` with the Interest selector "leftmost child" and receive a data packet named `/ucla/videos/demo.mpg/1/1` corresponding to the first segment. The consumer can then request later segments using a combination of information revealed by the first data packet and the naming convention of the publishing application to request subsequent packets.

Only names used to retrieve data globally require *globally uniqueness*. Individual data names have meaning in specific contexts, ranging from "the light switch in this room" to "all country names in the world". Efficient strategies to fetch data within the intended scope is a new research area. While name space management is not part of the NDN architecture, just as address space management is not part of the IP architecture, naming is the most important part of the NDN design. Naming data enables natural support for functionality such as content distribution, multicast, mobility, and delay-tolerant networking. We are learning through experimentation how applications should choose names that can facilitate *both* application development and network delivery. As we develop and refine our principles for naming, we convert these principles and guidelines into naming conventions and implement them in system libraries to simplify future application development. Fortunately, the opaqueness of names to the network allows architecture development to proceed in parallel with research into namespace structure and navigation in the context of application development.

1.2 Data-Centric Security

In contrast to TCP/IP's approach where security (or lack thereof) is a function of where or how the data is obtained, NDN builds security into data itself by requiring data producers to cryptographically sign every piece of data. The publisher's signature enables determination of data *provenance*, allowing the consumer's trust in data to be decoupled from how (and from where) the data is obtained. It also supports fine-grained trust, allowing consumers to reason about whether a public key owner is an acceptable publisher for a particular piece of data in a specific context. But security based on public key cryptography is typically considered inefficient as well as difficult to deploy and use. Besides efficient digital signatures, NDN needs flexible and usable mechanisms to manage user trust. Keys can be communicated as NDN data, simplifying key distribution. Secure binding of names to data supports a wide range of trust models. If a piece of data is a public key, a binding is effectively a public key certificate. NDN's end-to-end approach to security facilitates trust between publishers and consumers, and gives applications flexibility in customizing their trust model.

NDN's data-centric security has natural applications to content access control and infrastructure security. Applications can control access to data via encryption and distribute (data encryption) keys as encrypted

NDN data, limiting the data security perimeter to the context of a single application. Requiring signatures on network routing and control messages (like any other NDN data) provides a solid foundation for securing routing protocol against, e.g., spoofing and tampering. NDN's inherent multipath routing, together with the adaptive forwarding plane, mitigates prefix hijacking because routers can detect the anomaly caused by a hijack and retrieve data through alternate paths. Since NDN packets reference content rather than devices, it is trickier to maliciously target a particular device. But we are studying other mitigation mechanisms for other methods of attacking NDN networks, such as denial-of-service via Interest flooding.

1.3 Adaptive Routing and Forwarding

NDN routes and forwards packets based on names, which eliminates four problems caused by addresses in the IP architecture: address space exhaustion, NAT traversal, mobility, and address management. There is no address exhaustion problem since the namespace is unbounded. There is no NAT traversal problem since a host does not need to expose its address in order to offer content. Mobility, which requires changing addresses in IP, no longer breaks communication since data names remain the same. Finally, address assignment and management is no longer required in local networks.

NDN can use conventional routing algorithms such as link state and distance vector. Instead of announcing IP prefixes, an NDN router announces *name prefixes* that cover the data that the router is willing to serve. Routers simply treat names as a sequence of opaque components and do component-wise longest prefix match of the name in a packet against the FIB. We have designed and implemented an NDN link state routing protocol and developed efficient data structures and algorithms for fast lookup of variable-length, hierarchical names.

The PIT state at each router supports forwarding across NDN's data plane, recording each pending Interest and the incoming interface(s), and removing the Interest after the matching Data is received or a timeout occurs. This per-hop, per-packet state is a fundamental change from IP's stateless data plane. The state information enables NDN nodes to monitor performance across different interfaces, and adapt to network failures. Via a random nonce in the Interest packet, NDN nodes can identify and discard packets that have returned to the same node, preventing forwarding loops and enabling efficient use of multiple paths toward the same data producer.

The PIT state serves other important purposes. Since it records the set of interfaces over which the Interests for the same data name have arrived, it naturally supports multicast Data delivery. Since each Interest retrieves at most one Data packet, a router can control the traffic load by controlling the number of pending Interests to achieve flow balance. The PIT state can also be help mitigate DDoS attacks. Because the number of PIT entries is an explicit indicator of the router load, placing an upper bound on this number sets the ceiling on the effect of a DDoS attack. PIT entry timeouts offer relatively cheap attack detection; and the arrival interface information in each PIT entry gives information to implement a push-back scheme.

Each NDN node also implements a *forwarding strategy* module, which does not exist in today's IP nodes; the forwarding strategy makes informed decisions about: which Interests to forward to which interfaces, how many unsatisfied Interests to allow, the relative priority of different Interests, load balancing Interest forwarding among multiple interfaces, and choosing alternative paths to avoid detected failures.

1.4 In-Network Storage

Because each NDN Data packet carries a name and a signature, it is meaningful independent of its source or destination. Thus a router can cache the data in its Content Store to satisfy future requests. The Content Store is analogous to buffer memory in IP routers, but IP routers cannot reuse data after forwarding it, while NDN routers can. NDN treats storage and network channels identically in terms of data retrieval. For static files, NDN achieves almost optimal data delivery. Even dynamic content can benefit from caching in the case of multicast (*e.g.*, realtime teleconferencing) or retransmission after a packet loss. In addition to the Content Store, this year we added support for a more persistent and larger-volume in-network storage, called a Repository (Repo for short).

Caching named data raises different privacy concerns from those of IP. In IP, one can examine packet headers, and possibly payload, to learn who is consuming what data. Naming and caching of data in NDN networks may facilitate observation of what data is requested, but without destination addresses it is harder to identify who is requesting it (unless one is directly connected to the requesting host). This aspect of the architecture offers privacy protection at a fundamentally different level than current IP networks.

1.5 From Transport to Distributed Synchronization

The NDN architecture does not have a separate transport layer. It moves the functions of today's transport protocols into applications, their supporting libraries, and the strategy module of the forwarding plane. NDN does not use port numbers; a host knows to which application to deliver packets based on data names, and applications handle data integrity checking, signing, and trust decisions related to their data. To provide reliable delivery across highly dynamic and possibly intermittent connectivity, such as in mobile environments, nodes will discard Interest packets that remain unsatisfied after some threshold of time. The application that originated the initial Interest must retransmit it if it still wants the data. Such functionality is supported by NDN common client libraries.

NDN's flow balance requirement, together with the ability of nodes to control their own traffic load by limiting the number of pending Interests at each hop, means that there is no need for separate end-to-end congestion control, a typical transport layer function in today's networks. If congestion losses occur, caching will mitigate the impact since retransmitted Interests can be satisfied by cached Data packets right before the point of packet losses. Thus NDN avoids the kind of congestion collapse that can occur in today's Internet when a packet is lost near its destination and repeated retransmissions from the original source host(s) consume most of the bandwidth.

Traditional transport services provide point-to-point data delivery and most of today's distributed applications, including peer-to-peer applications, heavily rely on centralized servers. To aid development of robust and efficient distributed applications, we have added a fundamentally new architectural building block that we call *Sync*. Using NDN's basic Interest-Data exchange communication model, Sync uses naming conventions to enable multiple parties to synchronize their dataset. By exchanging individually computed data digests, each party learns about new or missing data quickly and reliably, and retrieves data efficiently via NDN's built-in multicast delivery.

References

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

[2] J. Saltzer, D. Reed, and D. Clark. End-to-end arguments in system design. *ACM Transactions in Computer Systems 2, 4, November, 1984.*

Chapter 2

Research Progress

This chapter summarizes project results during the third year of the project. Teams at various campuses are advancing research in various areas, including forwarding performance, routing, security and privacy, and the fundamental theory for this new type of networking. Our goals for the end of this project include:

1. A specification of the standard formats for the two packet types, Interest and Data, in NDN data delivery. We expect this specification to play a role equivalent to that of RFC791 (Internet Protocol Specification) for NDN networks. The challenge is not the packet format, but the verification and validation of the exact functions that must be supported by the narrow waist.
2. A functional version of each of the necessary supporting modules in an operational NDN network, including libraries to support naming conventions, reliable data delivery, routing protocols and forwarding strategy, trust management, and usable, efficient cryptography for data security[1].
3. A set of applications that operate over an NDN network, including both new ones specifically designed to run on top of NDN, as well as legacy ones in today's Internet.
4. An online documentary of the NDN project process, and a technical report series to capture our thinking along the path of architectural development.

2.1 Progress: Year 3

- Applications (Section 3.1):
  - Deployed an NDN-enabled chilled water and electrical demand sensing system at two UCLA buildings using industry-standard hardware, and designed an initial trust model for data access.
  - Performed a red team attack on the authenticated actuation implementation of NDN-based lighting control on a dedicated testbed.
  - Developed a software toolkit, Minerva, which provides a publish-subscribe environment for smartphone-based data collection using NDN, as a basis for Sync prioritization research.
  - Tested NDN-based live video streaming to China (via Amazon EC2), Japan (via JGN-X), a demo with 1000 international consumers from a single publisher, and began the design and implementation of NDN-based WebRTC video conferencing.
  - Improved web browser support through the NDN-JS Javascript library and cross-platform Firefox plug-in providing "ndn" URL scheme support, and built several web-based applications.
  - Developed peer-to-peer serverless applications on NDN, including ChronoShare, which provides Dropbox-style file-sharing, and Matryoshka, a multiplayer online game using the Unity Game

[1] We see an analogy between the above list and IP's supporting components. Although the IP address allocation system, routing protocols, and DNS are not part of the IP narrow waist, they are nonetheless necessary supporting components to make an operational IP network. The fact that DNS was added *after* the initial IP deployment further underscores the importance of identifying missing components from real deployment.

engine, yielding valuable challenges for the architecture.
  - Created an initial security library implementation to support application development, a C++ client library, and prototyped advanced libraries for vehicular networking, synchronization, and other higher level features.
- Routing (Section 3.2):
  - Designed and implemented the *Named-data Link State Routing Protocol (NLSR)*, a native NDN routing protocol with built-in update authentication and multipath support.
  - Implemented basic hyperbolic routing support in NLSR by disseminating hyperbolic coordinates in NLSR announcements.
  - Developed a methodology to embed a router topology of a given NDN testbed site (an intra-site topology) into the hyperbolic plane, and simulated growth of NDN networks using hyperbolic coordinates.
  - Compared routing algorithms (e.g., link state, distance vector) under IP and NDN with forwarding strategy in simulation, and showed that NDN routing can be simpler and more scalable because routing churn in an NDN network are handled locally by forwarding instead of globally by routing.
  - Started implementation of a new network forwarding daemon, NDNfd, to support new forwarding strategies, forwarding hints, and routing protocols.
- Scalable Forwarding (Section 3.3):
  - Proposed a forwarding structure whose size is dependent upon the information-theoretic differences among rules in a rule set, and designed a scalable distributed forwarding engine.
  - To improve named-based forwarding performance, defined the string differentiation problem based on the behavior of speculative forwarding in the core networks. Studied Patricia-trie and hash-table based methods.
  - Proposed a novel Pending Interest Table design that guarantees packet delivery with a compact and approximate storage representation, allowing use of SRAM and RLDRAM memory.
- Security and Privacy (Section 3.4):
  - Identified and explored (Distributed) Denial-of-Service (DoS/DDoS) vulnerabilities.
  - Designed and evaluated mitigation machanisms agains interest flooding attacks.
  - Designed efficient content verification/authentication mechanisms using Message Authentication Codes (MACs) and aggregated signature objects.
  - Designed and evaluated cache management algorithms that are effective against cache-poisoning attacks.
  - Designed a secure fragmentation algorithm for NDN.
  - Analyzed privacy of NDN networks and designed countermeasures to timing attacks in caches.
- Theory (Section 3.5):
  - Developed the *Virtual Interest Packet (VIP) framework* for high performing NDN networks.
  - Developed two instantiations of the VIP framework, both of which use the VIP count as a parameter of forwarding and caching algorithms.
  - Determined the *VIP network stability region*, the range of Interest packet arrival rates that can be satisfied by a feasible forwarding and caching policy.
  - Proved that the forwarding/caching policy of the first VIP algorithm is *throughput optimal*, i.e., adaptively maximizes VIP throughput.
  - Showed that the second VIP algorithm yields a stable joint forwarding and caching solution in which the cache contents do not cycle in steady-state.
  - Experimentally compared Algorithm 2 to traditional shortest path routing with LFU and LRU caching policies, finding improved performance in terms of user delay and cache hit rate.
- Values in Design and Values from Law and Policy (Chapter 4):
  - Published two papers examining the taxonomy of values concerns in the NDN project.

  - Started paper entitled *A World on NDN*, which examines key components of the NDN architecture and critical departures from TCP/IP, and the implications of these departures for social issues such as free speech, security and privacy, law enforcement, and network neutrality.
  - Condsidered the DMCA notice and takedown provisions of copyright law, in context of how they would apply to NDN; identified how the law could be reframed to better suit the technology and how existing or future NDN features could support underlying legal policy.
- Education (Chapter 5):
  - Established a biweekly NDN Seminar series among all participating sites to share the latest research results and discuss results by other researchers.
  - Contributed educational material to the GENI community by publishing programming project exercises that used the GENI testbed to demonstrate the power of NDN naming and caching in support of sharing large distributed data sets.[2]
  - Continued to incorporate NDN architecture into undergraduate and graduate teaching at NDN project universities; a number of term projects from the graduate seminar courses directly contributed new results to NDN research.

[2] `http://groups.geni.net/geni/attachment/wiki/ScalableMonitoring/ccnx-geni.pdf`

Chapter 3

Architecture Research

This chapter describes the activities and findings in each research area for the third year, as well as milestones for the extension year of the project.

3.1 Applications

Contributors	
PIs	Jeffrey Burke, Deborah Estrin, Van Jacobson & Lixia Zhang (UCLA), Tarek Abdelzaher (UIUC)
Grad Students	Alex Afanasyev, Mevlut Turker Garip, Silas Lam, Yuanjie Li, Ilya Moiseenko, Victor Peres, Wentao Shang, Zhehao Wang, Zhe Wen, Yingdi Yu, Zhenkai Zhu (UCLA); Shiguang Wang (UIUC)
Undergrads	Zening Qu (UCLA)
Staff	Peter Gusev, Alex Horn, Derek Kulinski, Alessandro Marianantoni, Jeff Thompson (UCLA); Hongyan Wang (UIUC)

NDN application research: (1) drives architecture development based on a broad vision for future applications; (2) drives and tests prototype implementations of the architecture using applications for participatory sensing, instrumented environments, and media distribution, among others; (3) verifies and validates performance and functional advantages of NDN in key areas; and (4) demonstrates how NDN's embedding of application names in the routing system promotes efficient *authoring of sophisticated distributed applications*, reducing complexity, opportunities for error, and time and expense of design and deployment.

During the third year of NDN research, the applications group continued to explore the instrumented environment, media distribution, and serverless peer-to-peer applications discussed in last year's annual report. We further explored new architectural primitives, such as "Sync", built initial security libraries to support application developers, and extended the client libraries developed for various languages into the NDN-CCL (Common Client Library) platform effort. We also continued to collaborate with Washington University in St. Louis (WUSTL) in large-scale (thousands of nodes) tests of the media streaming design.

3.1.1 Instrumented environments: Building Automation

We are exploring the use of NDN as a network substrate for Building Automation Systems (BAS) and lighting control. These environments provide interesting and challenging application domains for information-centric networking, which is generally discussed in terms of content retrieval, as opposed to control, actuation, or remote execution. NDN offers solutions for not only these functions but broader industry goals, e.g., to enhance device interoperability, enable data-centric application designs, and provide simplified access to

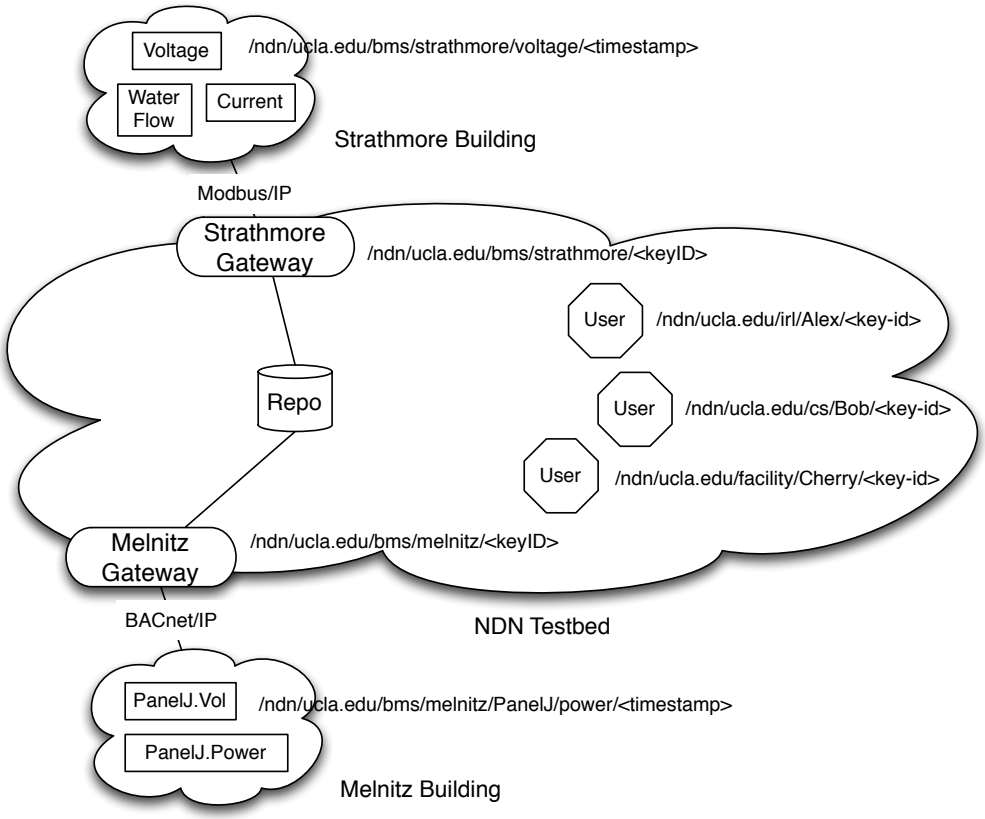

Figure 3.1: Building monitoring system architecture.

networked buildings over commodity networks while providing security against a wide variety of IT and industrial control attacks.

We began to create a scalable, secure data acquisition platform that collects data from different real-world sensors deployed inside buildings on the UCLA campus, and publishes the data into an NDN repository. Our recent development of an NDN Javascript library (NDN-JS) allows us to express NDN lighting control Interests directly from a remote web browser instead of relying on a CGI script to convert HTTP POST commands to NDN control messages. We implemented authenticated Interests in JavaScript that were byte-compatible with our original implementation [4, 3]. Monitoring applications can now fetch data via NDN to provide real-time reports to facility managers via a web browser, using the NDN-JS Javascript library and existing data visualization tools. We transitioned an existing chilled water flow and electrical demand monitoring system, which uses the industry standard Modbus/TCP protocol internally, to publish its data over NDN. With assistance from Siemens and UCLA Facilities, we also designed and installed an electrical demand monitoring system for the space where we deployed NDN lighting control last year. This second system internally uses the BacNet/IP protocol, another industry standard, and its data is now published onto the NDN testbed and accessible via a web client.[1] We initially used a hierarchical namespace to describe the sensor devices' physical location, e.g., `/ndn/ucla.edu/apps/bms/building1/room2/panelA`, similar to the point names we used in UCLA's campus-wide monitoring systems.

The current system design employs identity-based access control that enforces trust relationships, e.g., authorized access, based on the data name hierarchy. Sensor readings are encrypted using symmetric keys before being published into the repository, to prevent eavesdropping or cache leakage. A Key Distribution Service (KDS) publishes encrypted keys for authorized users, appropriately named based on what parts

[1]This work was also supported by an NSF EAGER (CNS-1248049), and the Strathmore building installation was funded by UCLA Facilities Management.

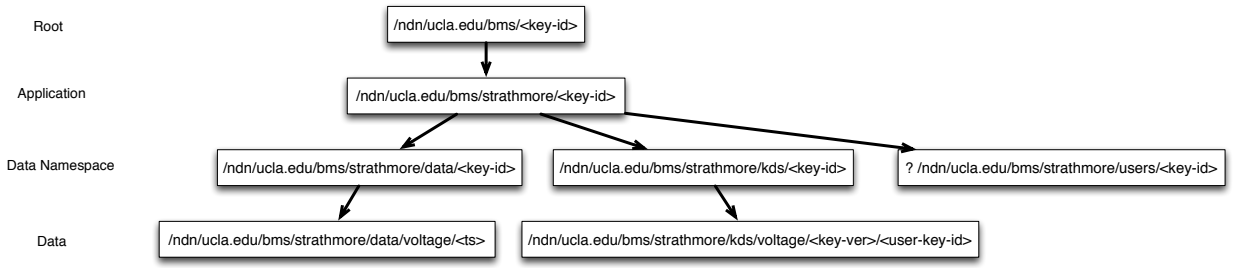

Figure 3.2: Relationship between data and key namespace (proposed).

of the hierarchy the user is allowed to access. Figures 3.1 and 3.2 show the basic architecture and key namespace of the building monitoring system (BMS). The hierarchical structure of the namespace supports delegation of access privileges at different granularities, e.g., shorter prefixes map to many devices in different locations. For example, users of room101 may have access only to that room's devices (`/ndn/ucla.edu/apps/bms/building1/room101`) while the building manager may have control over the entire namespace of `/ndn/ucla.edu/apps/bms/building1`. Traditional BAS implementations rarely include verification or access control, and generally rely on air gaps, VLANs or similar approaches. Therefore, we believe this work will provide a significant contribution, and plan to continue it during the supplement period. The sensor deployment will also motivate and support ongoing explorations of BAS namespace design, application development, and performance analysis. We are also exploring other communication and security models for sensing – such as how to notify remote nodes of rare but critical events at sensors, which may merit a data-push rather than NDN's pull(request)-based approach.

We also organized a red team experiment to validate our lighting control security approach and discover NDN-specific exploits; three graduate students performed the attacks described in [11]. The project used a testbed that consisted of a color LED lighting fixture, and an embedded Linux controller (ARMv7a gumstix w/ Angstrom Linux 2011.03) communicating with an operator machine via NDN over Ethernet. The operator machine was also connected to the commodity Internet, to allow remote access and preview lighting state via webcam. The red team's attacks were inspired by both existing IP attacks and possible NDN-specific attacks: replay and reodering (of authenticated Interests); link-layer ACK attack (attempting to block ACKs to induce retransmits); name-encoding exploit attack (changing name in Interest on the wire); and memory leak exploit attack (using large enough control messages to crash the daemon). Successful attacks revealed CCNx implementation flaws that could be fixed relatively easily.

As a completely different type of instrumented environment, the Department of Architecture and Urban design (funded by a UCLA Transdisciplinary Seed Grant) is designing a bus bench that communicates arrival time and occupancy of the next bus using light color and patterns. We demonstrated the use of NDN to publish and consume the bus transit data through a set of bridge applications that translate XML messages received from UCLA's bus tracking web service into named objects accessed by the bench. The namespace design for this time-series data is simpler than those used for the BMS applications, and includes, e.g.,:

- `/ndn/ucla.edu/apps/transportation/bus/stop/weyburn` [Overall status]
- `/ndn/ucla.edu/apps/transportation/bus/stop/weyburn/eta` [ETA of next bus]
- `/ndn/ucla.edu/apps/transportation/bus/stop/weyburn/occupancy` [Occupancy of next bus]

The bus publisher uses the *FreshnessSeconds* field in the content object to make the content stale at the end of the sampling period. The archiver/publisher/consumer has run with minimal interruption for eleven months. We will test the success of this simpler data publishing architecture as we deploy the bench and refine the lighting control application.

3.1.2 Peer-to-Peer, serverless applications

We have explored NDN support for four *serverless* peer-to-peer applications:

1. We are building on our audioconferencing [16, 15], videostreaming [5], and chat tools [14] to support **multi-party video conferencing** in our new browser-based NDN-WebRTC project [10]. We added high-definition capture support to our audio/video streaming application NDNVideo[2], integrated NDN C++ libraries into a Firefox extension, analyzed the Mozilla WebRTC codebase, and designed a prototype.

2. We ported our **multi-user chatroom** tool [14] to Javascript, to support a wider user base.

3. Our new **ChronoShare** application implements distributed (Dropbox-style) file sharing on NDN, building on a file synchronization prototype that used CCNx SYNC [6]. ChronoShare uses a new sync implementation (ChronoSync) to synchronize file sharing among fixed or mobile devices even during intermittent or ad hoc connectivity. We tested this application during NDN retreats to sync files across laptops in the same room without upstream Internet connectivity [2, 12].

4. We extended last year's work on distributed state and asset synchronization using NDN [8], developing a **multiplayer game** that also incorporates real-world data ingested and published over NDN. We built Matryoshka, a prototype Multiplayer Online Game (MOG) and distributed simulation environment that embeds the concept of region and knowledge into NDN names to leverage and communicate the locality-of-interest inherent in many MOGs.

3.1.3 Information Maximization Utilities

To support applications in resource-constrained environments, UIUC has developed two schemes to maximize information transfer given limited channel resources: a prioritized synchronization scheme that fetches content in an order that minimizes information overlap between information transmitted within a given window; and a content replacement scheme (for repositories) to erase content that overlaps most with surviving objects. To explore and further refine these schemes, UIUC developed the Minerva programming toolkit for smartphone applications that collect and share information. Minverva uses a publish-subscribe paradigm, developed for social sensing applications where different sources (participants sharing sensor data) overlap in the information shared, e.g, different pictures of the same scene or speed measurements on the same street. Minerva's prioritization scheme maximizes information delivery from publishers to subscribers by reducing redundancy, exploiting the non-independent nature of content. Both schemes compute distances between objects based only on content names in a given namespace, which makes appropriate design of the namespace essential to their success. UIUC implemented Minerva-over-NDN on Android phones and tested it in a vehicular testbed. In both smartphone-based experiments and data-driven simulation, the prioritization algorithm outperformed others in terms of an application-level notion of information coverage. BBN researchers implemented and independently evaluated the information-maximizing caching algorithm, confirming its advantages for tactical military networks. Disruption-tolerant and ad hoc networks can also benefit from smarter caching policies that leverage an intelligent namespace design, where data names allow expressing partial redundancy among objects. This redundancy information can be exploited by the cache, to reduce redundancy in the cache, enabling better quality of information (in terms of coverage), higher throughput (in terms of responses per query), and lower latency.

3.1.4 Libraries

This year, we created NDNx, a fork of the CCNx codebase, into which the NDN team will incorporate new features, bug fixes, and prototype code without being tied to the CCNx release schedule or roadmap. UCLA will track updates to CCNx and merge NDN-developed features into the NDNx client libraries, while WUSTL will manage the incorporation of new research results into NDNx's software routing daemon. NDNx is one component of the new "NDN Platform" that gathers together major software components to support project and external development of NDN applications. The platform will follow a 3-4 month release heartbeat managed jointly by UCLA and WUSTL, and will be the recommended "stable" release of

[2] https://github.com/remap/ndnvideo

NDN libraries. It will include NDNx, the common client libraries (see below), web browser API, security, synchronization, file system functions.

1. We gathered NDN language bindings into the **"Common Client Libraries" (NDN-CCL)** and started to standardize the API across three publicly available implementations: Python, Javascript, C++ and C.[3] Using NDN-CCL, applications work with Name, Interest and Data objects that are abstracted from the wire format and have an asynchronous communication API for publishing and consuming data. We plan to provide modular support for different wire formats, and a transport abstraction layer to support different transports, including TCP, UDP, and Ethernet.

   - **NDN-CPP** is a C++ implementation of NDN-CCL, to achieve higher performance that can leverage the Python and Javascript libraries' ease-of-use. While NDN-CPP provides an object-oriented C++ API for advanced applications, its core functionality is implemented in C, and makes few assumptions about memory management or linked library support in order to promote use on a range of platforms.
   - **PyNDN** (developed last year) is a set of C and Python bindings for NDNx, our port of CCNx, to facilitate rapid application development.
   - **NDN-JS** is a pure JavaScript client library, previously called "lwNDN". Advancements this year include: (1) support for all Interest selectors including exclusion filters; (2) a lightweight PIT implementation to support a single connection with an external hub; (3) registering a prefix with a hub to respond to an Interest and send content; (4) automatic connection to the NDN testbed; (5) support for WebSockets to allow pure JavaScript without needing a Java applet; (6) full compliance with the NDN name URI scheme; and (7) compilation ("minification") of the library to compressed JavaScript for more efficient loading in a browser.

   A library that does not yet conform to the NDN-CCL conventions is NDN-C#, a thin wrapper of the CCNx library that exposes core CCNx APIs to Unity, the game engine used in the Matryoshka multiplayer game described above. The library and the C# bindings are built into a Unity plugin. Because the C# APIs are consistent with the C APIs, applications manipulate Name, Interest and Data in the same way as with the C library. To facilitate fast prototyping, NDN-C# provides some APIs that hide many details of CCNx. New components could be added to NDN-C#, e.g., synchronization, octree-based virtual world partitioning, and k-nearest-neighbor discovery.

2. **Web browser library support.** Last year we created a patch for Firefox supporting NDN-based file retrieval. This year we used our Javascript library (NDN-JS) to build a Javascript extension that provides similar functionality without native (platform-specific compiled) code. The cross-platform NDN Firefox add-on implements an 'ndn:' URI scheme, which can be entered into a browser location bar or used in HTML anchor tags. The add-on connects to an external hub using privileged APIs in Firefox to communicate using TCP or UDP directly. Implemented features include:

   - Ability to fetch and combine file segments
   - URI support for Interest selectors, e.g., `ndn:/ucla.edu/maps.html?ndn.ChildSelector=1`
   - establishing permanent links using a self-certifing content digest in the name, e.g., `ndn:/example.com/license.html/%FD%05%0BZ%94%B4l/%C1.M.G%C1<binary-XML-encodedContentDigest>`.
   - Retrieving versions of named content using ChildSelector and Exclusion fields.
   - Interest completion semantics in the UI, enabling the user to enter a name prefix in the address bar and the add-on to replace it with the full name of the fetched content.

   We compared performance in different browsers, with and without signature verification, and compared to other retrieval methods [10]. As expected, cryptographic functions implemented in pure Javascript are quite slow, but given the usefulness of the library we plan to continue its development and take advantage of cryptographic support in the browser native code when it emerges.

   The Javascript library used to build the extension, NDN-JS, provides NDN-based AJAX-style communication to browser applications, and connects to a WebSockets proxy at the (remote) daemon side.

[3] `http://github.com/named-data`

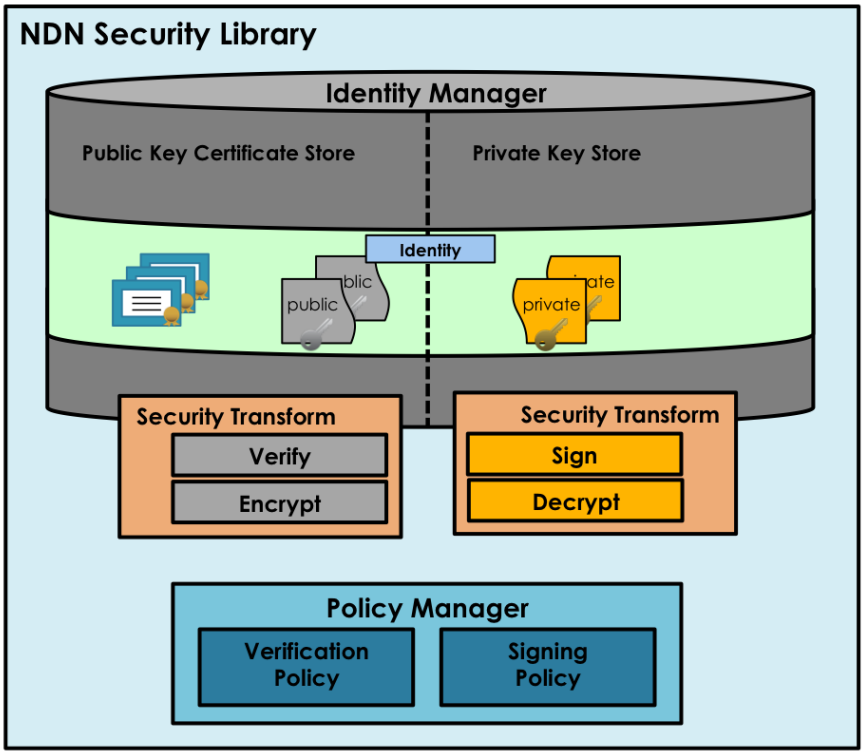

Figure 3.3: Security library architecture.

We intend for NDN-JS to prompt and support research on NDN features, including studying the pros and cons of using NDN over TCP/IP for web applications, and comparing NDN trust management approaches on browsers. Within three months of its release, several experimental applications used it, including the user interface for ChronosShare, a visualization platform for NDN sensor data, and community-created NDN-based Wiki software.[4] We also completed initial support for a standalone version outside the browser using Node.js.

3. **Security Library.** The existing NDN API provides basic data signing and verification, but there is no policy management, so in order to verify a Data packet each application must figure out whether the public key pointed to by the Key Locator is legitimate. UCLA built a minimal security library for common cases in NDN applications, supporting hierarchical trust models that follow a data object's name: e.g., `/ndn/ucla.edu/cs/foo_object` is signed by a key, such as `/ndn/ucla.edu/cs/key`, which is signed by `/ndn/ucla.edu/key`. The current security library consists of two parts:

   - The *Identity Manager* provides key management, including key generation. Identities, expressed as names, are associated with private keys in corresponding public key certificates. An identity name could represent an organization, a user, an application, or some other entity. The Identity Manager consists of two parts, a Private Key Store and a Public Key Certificate Store. The Private Key Store is implemented as a "black box", in which all security transformation related to secret keys, such as signing and decryption, are performed in a manner that cannot be accessed directly by the applications. The Public Key Certificate Store manages certificates of public keys, e.g., certificate publishing and revocation, and provides security transformation related to public keys (such as verifying signatures and encrypting packets).
   - The *Policy Manager* enables developers to specify policies for packet signing and verification in terms of the data and key namespaces.

[4] "CCNx Federated Wiki Mach1", Ward Cunningham, April 2013, `http://www.youtube.com/watch?v=xX22CgG4d18`

4. Building on lessons learned from using CCNx SYNC, we revised and improved **ChronoSync**, a dataset synchronization protocol for distributed applications running in NDN networks [13]. ChronoSync names the state of a dataset by its digest at a given time. Carrying this name, each sync Interest is broadcast to all participants in a synchronization group to solicit Data that reports changes in the dataset. Using the ndnSIM simulation platform [1] we demonstrated that ChronoSync is highly robust to packet loss, link failures, and network partition. We have started to use ChronoSync in other NDN applications, e.g., ChronoShare, Matryoshka, and multi-party chat.

5. We designed **NDNFS, an NDN-oriented file system** to address gaps in the current CCNx repository implementation, including lack of local file system interface and delete capability. NDNFS allows local applications to use standard system calls to create, edit, and remove files and directories. It organizes file segments in the form of NDN Data packets, using a similar hierarchical structures as in the naming scheme, to facilitate direct use of packets to satisfy Interest requests. We implemented the NDNFS prototype for Mac OS X and Linux, using the FUSE library, which bridges the OS system calls to user-defined callback handlers. Files in the NDNFS tree are segmented, packaged as NDN Data packets with signatures pre-computed, and stored in a local database. The current implementation adopts the NoSQL database MongoDB to leverage its simplified query interface and better scalability (compared to SQL databases). We also implemented an NDNFS server that fetches file segments from local databases to satisfy remote Interests. We are still investigating how to mitigate performance bottlenecks such as the required RSA signature (and associated slow *write* operation) for every file segment stored, and the transaction delay of the back-end database.

3.1.5 Standardized Application Nodes for the Testbed

UCLA maintains distributed hardware nodes at six other NDN project sites; these hosts stream audio/video using NDN and run web proxies for the NDN-JS javascript libraries. They are configured with standard CCNx distribution and available NDN applications, and enable remote updates to push new applications and libraries. They connect to the geographically closest NDN hub in the networking testbed. Additionally, UCLA participates in the NDN testbed, and hosts nodes (software routers) in both the Zhang group and the Burke group. UCLA collaborates to provide tools for operators to develop the real-time status report created by the University of Memphis[5]. This applications testbed has helped the applications team better understand deployment issues, such as the impact of MTU limitations in the underlying transport layer for some nodes in the testbed. We are using the hosts as superpeers that publish data/objects and that maintain persistence for clients playing the NDN MOG [8]. We also use them to test ChronoSync-based applications [13], which enables us to provide file persistence to peer-to-peer users of the Dropbox-style ChronoShare application [2]. Other NDN institutions may use this testbed for distributed application testing.

3.1.6 Future plans

1. We will investigate **naming techniques**, including ways to reduce the dimensionality of namespaces, mixing hierarchical and set-based naming [9], and developing meta-data standards (analogous to HTTP Content-Type data).

2. We plan to continue work on primary application areas:

   - Our ***instrumented environments*** area includes two projects: (1) complete a *Building Automation Systems* testbed to explore more sophisticated applications and trust models and publish sensor data from the UCLA campus building management system; and (2) complete the (UIUC) transit data-based bus stop application.
   - Improving performance and scalability of **NDN-based web conferencing**, and adding support for content encryption. We will implement and deploy the NDN-WebRTC application, and explore new video streaming features for bandwidth adaptation and metadata synchronization, including tests over Amazon EC2, GENI, and JGN-X, Japan's research network.

[5] `http://netlab.cs.memphis.edu/script/htm/status.htm`

- We will continue work on our *Matryoshka* **multiplayer online game** to study peer-to-peer synchronization methods suitable for real-time interactivity. We will start by evaluating latency and bandwidth usage at the players' end for various synchronization approaches. We will also explore cheating-prevention using NDN's per-packet signatures.

3. **Libraries.** We will continue to develop and release libraries that promote experimentation with NDN, as well as help identify application needs, design patterns, testing solutions, and protocol design issues.

   - *NDN-CCL: Common Client Libraries.* Based on application and community needs, we will continue to work on updates and standardization of the platform client libraries, including NDNx's C library, PyNDN, NDN-CPP, and NDN-JS, and new architectural primitives such as synchronization. We will explore features such as (1) minimizing round trips of selecting leftmost/rightmost child; (2) autodiscovery for joining an NDN network; (3) key maintenance for signing content, based on security libraries under development; (4) trust model for verifying content; (5) a lightweight Java port primarily for supporting Android developers.
   - *Security API.* We will examine the correctness and efficiency of the security library by applying it in real NDN applications. Anticipated improvements include the support of capability-based and role-based policies, in addition to identity-based policy. We plan to explore more advanced encryption techniques, such as broadcast encryption[6], and to extend future iterations of our applications to focus on application-specific trust management approaches using NDN primitives. In parallel, we will continue to (1) assess trust management practice in existing implementations, such as lighting control, the audio conferencing tool, building automation, routing protocols, and (2) survey the trust management literature over the last twenty years.
   - *Browser support.* Inspired by the uptake of NDN-JS, we will continue supporting the "browser as a platform," such as with the WebRTC work (described above) and continued development of the Firefox add-on, which runs in privileged code and remains live as browser pages open and close. These libraries will enable future development, including (1) Interest forwarding; (2) persistent content store; (3) persistent key management; (4) use of the emerging built-in web cryptography API to improve signature performance; (5) using C++ NDN-CPP instead of JavaScript for core functions to improve performance and provide a more reasonable security model.
   - *In-network storage.* Extending the NDNFS functionality will involve integrating access control and data encryption, in order to enhance security and privacy protection. We plan to implement network-write functionality to enable users to manipulate NDNFS storage remotely.
   - *Advanced and Application-Specific Libraries.* The Advanced API project is exploring an NDN-based Inter Process Communication (IPC) primitive, roughly analogous to sockets. NDN offers more options than IP, but its existing APIs lack higher-level features such as automatic sequencing, reliability during transmission, realistic security capabilities, and the ability to influence forwarding decisions. We hope to design an easy-to-use skeleton for associating a namespace with transmission parameters (sequencing and reliability), security parameters (signing and encryption) and forwarding parameters (scope and actions). We will also continue to explore libraries to support important application domains such as vehicular networking, and UIUC's work in information maximization utilities for limited capability networks. Our objective continues to be collaboratively developing intermediate services that bridge the semantic gap between NDN abstractions and applications.

References

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

[2] Alexander Afanasyev, Zhenkai Zhu, and Lixia Zhang. The story of ChronoShare, or how NDN brought distributed file sharing back. Under submission.

[6] Recently explored by the ICN community in [7].

[3] J. Burke, P. Gasti, N. Nathan, and G. Tsudik. Securing instrumented environments over Content-Centric Networking: the case of lighting control. In *IEEE INFOCOM 2013 NOMEN Workshop*, April 2013.

[4] Jeff Burke, Alex Horn, and Alessandro Marianantoni. Authenticated lighting control using named data networking. Technical Report NDN-0011, NDN Project, October 2012.

[5] Derek Kulinski and Jeff Burke. NDN Video: Live and Prerecorded Streaming over NDN. Technical Report NDN-0007, NDN Project, September 2012.

[6] Jared Lindblom, Ming-Chun Huang, Jeff Burke, and Lixia Zhang. FileSync/NDN: Peer-to-peer file sync over Named Data Networking. Technical Report NDN-0012, NDN Project, March 2013.

[7] S. Misra, R. Tourani, and N. Majd. Secure content delivery in information-centric networks: Design, implementation, and analysis. In *ACM SIGCOMM ICN Workshop*, Aug 2013.

[8] Zening Qu and Jeff Burke. Egal car: A peer-to-peer car racing game synchronized over Named Data Networking. Technical Report NDN-0010, NDN Project, October 2012.

[9] Stuart Sechrest and Michael McClennen. Blending hierarchical and attribute-based file naming. In *12th International Conference on Distributed Computing Systems*. IEEE, 1992.

[10] Wentao Shang, Jeff Thompson, Meki Cherkaoui, Jeff Burke, and Lixia Zhang. NDN.JS: A javascript client library for Named Data Networking. In *INFOCOM 2013 NOMEN Workshop*, April 2013.

[11] Mevlut Turker Garip Victor Peres, Silas Lam. Smart buildings: Security of lighting control using named data networking. June 2013.

[12] Zhenkai Zhu, Alexander Afanasyev, and Lixia Zhang. ChronoShare: a new perspective on effective collaborations in the future Internet. Poster, UCLA Tech Forum 2013, May 2013.

[13] Zhenkai Zhu, Alexander Afanasyev, and Lixia Zhang. Let's ChronoSync: Decentralized dataset state synchronization in Named Data Networking, 2013. under submission.

[14] Zhenkai Zhu, Chaoyi Bian, Alexander Afanasyev, Van Jacobson, and Lixia Zhang. Chronos: Serverless multi-user chat over NDN. Technical Report NDN-0008, NDN Project, October 2012.

[15] Zhenkai Zhu, Jeffrey Burke, Lixia Zhang, Paolo Gasti, Yanbin Lu, and Van Jacobson. A new approach to securing audio conference tools. In *7th Asian Internet Engineering Conference*, AINTEC'11, 2011.

[16] Zhenkai Zhu, Sen Wang, Xu Yang, Van Jacobson, and Lixia Zhang. ACT: An audio conference tool over named data networking. *ACM SIGCOMM ICN Workshop*, August 2011.

3.2 Routing

Contributors	
PIs	Beichuan Zhang (Arizona), Lan Wang (Memphis), Dmitri Krioukov (CAIDA), Lixia Zhang (UCLA)
Grad Students	Cheng Yi, Junxiao Shi, Yifeng Li, Mathias Gibbens, Aras Shravan, Jerald Paul (Arizona); A. K. M. Mahmudul Hoque, Yaoqing Liu (Memphis); Alex Afanasyev (UCLA)
Undergrads	Yi Huang, Troy Bowman (Arizona); Adam Alyyan, Andrew Hood (Memphis)
Staff	Syed Obaid Amin (Memphis); Ken Keys, Marina Fomenkov and Alex Ma (CAIDA)

The goal of NDN's network layer is to provide a name-based packet delivery service with the following properties:

1. *Scalability*: support a large Internet topology and large number of name prefixes.
2. *Security*: provide integrity, provenance, and pertinence of routing messages.
3. *Resiliency*: detect and recover from packet delivery problems quickly.
4. *Efficiency*: exploit multi-path forwarding and data caching for efficient use of network resources.

Our technical approach begins with extending existing routing protocols, such as OSPF (intra-domain) and BGP (inter-domain), to support name-based routing. These extensions include supporting name prefixes in routing updates and route computation, providing multipath capability, and using NDN instead of IP as underlying transport mechanism. We use these extensions to explore longer-term solutions and challenges, including designing a routing protocols that can benefit from NDN's resiliency and security, using NDN's Interest/Data packets to carry routing updates. We tackle the routing scalability problem by exploring both a map-and-encap approach and a hyperbolic routing scheme. Our protocol design uses states in every router's PIT to observe data retrieval performance, and adapt to problems, e.g, by exploring alternative paths. Our research methods include protocol design, simulation studies, implementation, and testbed deployment. The rest of this section will provide more details about the following activities and findings in the routing area during the third year of the NDN project:

- **Dynamic intra-domain routing protocol**: We designed and implemented the *Named-data Link State Routing Protocol (NLSR)*, a native NDN routing protocol that has built-in update authentication and multipath support [5].
- **Hyperbolic routing**: We implemented basic hyperbolic routing in NLSR by disseminating hyperbolic coordinates in link state announcements. We developed a methodology to embed a router topology of a given NDN testbed site (an intra-site topology) into the hyperbolic plane, and simulated the growth of NDN networks using hyperbolic coordinates.
- **Forwarding strategy**: We compared the same routing algorithm (e.g., link state, distance vector) under IP (without strategy) and NDN (with strategy), and showed that NDN routing can be simpler and more scalable since strategy modules can handle routing churn locally.
- **Code development**: We began to develop a new network daemon called NDNfd, which we will use to implement forwarding strategies, forwarding hints, and better support for routing protocols.
- **Network monitoring**: Using the NDN javascript library (NDN-JS), we re-designed our web-based monitoring tool to retreive the status page over the NDN testbed using native NDN messages. We also updated the monitoring scripts to reflect changes in the testbed setup.

3.2.1 Named-data Link State Routing Protocol

In the second year of the project, we developed OSPFN, an OSPF extension that distributes name prefixes and computes routes to name prefixes. OSPFN is a short-term solution to get the NDN testbed operational; it does not support dynamic multipath forwarding and has no mechanism to authenticate routing data.

We are designing a new routing protocol – Named-data Link State Routing protocol (NLSR) – which runs natively, that is, using NDN's Interest and (signed) Data packets to exchange routing messages. NLSR uses names instead of IP addresses to identify routers and links, and routers use the signature of each routing message to verify its origin and authenticity. It is a link-state protocol in the sense that adjacency information and name prefixes are propagated throughout the entire network and each router builds a complete network topology in order to compute routing paths. However, the route computation no longer produces just a single shortest-path; it ranks all policy-compliant next-hops and installs them into the FIB in order, essentially providing a name-based multipath routing table for NDN's forwarding strategy.

While NLSR benefits from the security primitives provided by NDN, we still need a naming scheme for the various components in a routing system as well as a trust model to ensure that routers can originate only their own routing updates. We use a hierarchical naming scheme to reflect the inherently hierarchical relationships between between routers, routing processes and routing data in the case of intra-domain routing. We also devised a hierarchical trust model based on common management structure and operational practice in a single network domain.

We finished an implementation of NLSR using CCNx's SYNC mechanism to distribute link state routing information. We conducted extensive testing using an internal testbed of 12 nodes, discovering several problems with CCNx's sync/repo implementations, including high memory consumption, inability to delete information from the repo, and failure to notify NLSR of routing changes when the update rate is high. These problems have prevented us from moving NLSR to the NDN testbed. One of our main tasks in year 4 is to design and implement a synchronization mechanism for NLSR as part of the new NDNx libraries.

3.2.2 Routing on Hyperbolic Metric Space

In previous work [6, 3] we described how hyperbolic metric spaces underlying complex networks enable efficient greedy forwarding (GF) without any global knowledge of the network topology. Since each node in the network has its coordinates in this hidden metric space, a node can compute the distance between each of its neighbor nodes and the destinations whose coordinates are carried in packets, and forward packets to the neighbor that is the closest to the destination in the hyperbolic space. If we can find a way to map NDN name space to a hyperbolic metric space, this approach can in theory dramatically reduce routing table size. During Year 3, we made progress on both the theory and implementation of hyperbolic routing.

We developed a novel framework of network growth, Popularity×Similarity Optimization (PSO) model [7], in which new connections optimize certain trade-offs between popularity and similarity, instead of simply preferring popular nodes. The framework has a geometric interpretation in which popularity preference emerges from local optimization. In contrast to standard preferential attachment, our optimization framework accurately describes the large-scale evolution of technological (the Internet), social (trust relationships between people) and even biological (Escherichia coli metabolic) networks, predicting the probability of new links with high precision. We found a hyperbolic embedding method for intra-site router topologies. Step 1 is to grow (or embed) an intra-site router topology as a "separate universe," that is, as a separate instance of a network growing according to the PSO model. Step 2 is to glue the resulting graph to the hyperbolic location of the site gateway.

We investigated the question of whether one can map a real network into the hyperbolic plane in a way congruent with the PSO model. We developed a systematic framework called HyperMap [8] that accomplishes this task by replaying the network's geometric growth. When applied to the Autonomous Systems (AS) Internet topology, the HyperMap outperformed our previous methods for hyperbolic network mapping in terms of mapping accuracy, method simplicity, and computational complexity. The theoretical probability of connections between ASes as a function of hyperbolic distances between them matched the connection probability in the real mapped Internet. The method also identified communities of ASes that belong to the same geographic region, outperforming existing methods to predict missing links in networks [8].

We have implemented hyperbolic routing in NLSR using a new link state message to carry the coordinates of each router. Each router chooses the next hop(s) for a name prefix based on the distances between its neighbors and the destination router that originates the name prefix. We used Emulab to evaluate the effectiveness of such a scheme, by running both link-state shortest-path and hyperbolic distance calculations

in parallel on the same network, and observing the next hops calculated by both algorithms at each rank. We also compared the RTT of ping packets under hyperbolic routing and link-state routing. Our preliminary results showed that hyperbolic routing results in considerably higher RTTs than link-state routing when we allow only two next hops per name prefix. The difference is much smaller when we allow three or more next hops per name prefix, but allowing more next hops also leads to more Interest messages, as some next hops may not lead to the destination (unlike in link-state routing).

3.2.3 Forwarding Strategy

IP networks forward packets after consulting a routing table that is established via control plane signaling, but NDN network nodes can use **forwarding strategies** to adapt forwarding decisions based on data-plane performance. For example, by observing the Interest/Data two-way traffic, a router can detect problems caused by link failures, congestion, prefix hijacks, etc, and explore alternative paths to avoid problematic links. During Year 3, we improved our forwarding strategy design to simplify and improve routing protocols, and started exploring strategies for congestion control and local area networks.

Packet forwarding in IP relies solely on the routing plane to handle failures, which means the routing protocol must maintain consistent routing state throughout the network, inducing high overhead in large networks that may slow convergence (e.g., BGP) or limit scalability (e.g., OSPF). NDN's forwarding strategy can handle failures locally and forwarding process is able to deliver packets during the routing convergence period, learning about connectivity changes by probing with extra Interests. probes, to these links. The tradeoff is between the overhead of probing and the timeliness of finding better paths. Periodically probing every interface at every node will introduce significant extra traffic. To minimize probing overhead, we take an approach where probing is triggered when a new path is learned via routing and the new path is better than the current one. The time it takes to learn better forwarding options is around the routing convergence time after link recovery, which is usually fast in most routing protocols. using existing paths.

With forwarding strategy that can handle both link failures and recovery, routing protocols can ignore short-term churns and instead focus on propagating long-term topology and policy information. We conducted simulations to compare the performance of the same routing protocol (e.g., link state, and distance vector) under IP and NDN. The results show that (1) NDN is able to deliver packets successfully during slow convergence, (2) NDN is able to deliver packets successfully when short link failures (which have been reported to account for most network failures [1]) are ignored by routing, (3) ignoring short link failures and using larger timer values can make routing protocols more scalable, in terms of control message and route computation overhead), (4) even a distance vector routing protocol performs reasonably well under NDN because of adaptive forwarding.

We started exploring strategies for local area networks, which we assume operate no protocol to disseminate name prefixes or topology information. We are designing a self-learning strategy to discover content locations and adapt to network changes. The basic idea is that for the first Interest that a node doesn't know where to forward, it sends the Interest to all interfaces. When the data packet returns, the node will use this information to forward future Interests under the same prefix without flooding again. The node may learn and store multiple forwarding options based on returned data, and may re-flood Interests when stored forwarding options have stopped working. Similar to Ethernet's self-learning, NDN's includes loop detection, adaptive forwarding, and multipath. We implemented and tested the basic idea, which revealed a few design issues we are further investigating.

Congestion control is another aspect of strategy in the sense that it controls how fast a node forwards Interests instead of which interface it uses to forward Interests. Congestion control in NDN differs from that in TCP/IP because it is receiver-driven, hop-by-hop, and multi-path. Our earlier results have shown that NDN has the potential to react to congestion more quickly, adjust sending rate more accurately, and utilize multiple paths [4]. During Year 3, we used ndnSIM to implement and evaluate different congestion control schemes in different network scenarios. So far we have compared hop-by-hop vs. end-to-end control, AIMD vs. MIMD, single source vs. multiple sources with different delays, interest NACK vs. without NACK, single flow vs. multiple flows. Based on the analysis of these results we will design and evaluate a detailed congestion control strategy that will also allow fair allocation of bottleneck bandwidth.

3.2.4 NDN forwarding daemon

The open-source CCNx implements most network layer functionality in a daemon program called ccnd. While it has played a critical role in our development so far, we have had to acknowledge more limitations as our research has progressed. First, it is not easy to add new features or change existing ones; the forwarding logic is not modularized, so code must be significantly refactored to support different strategies. Second, bug fixes take unpredictable time, sometimes blocking our research. In order to implement and deploy different strategies and support for routing protocol implementation, we need a more customizable network daemon that we can quickly add new features and push bug fixes. We started developing a new ccnd-based network daemon called NDNFD. NDNFD supports the same APIs and wire formats as ccnd to support legacy installations and applications, but supports efficient experimentation with new forwarding strategies, table structures, and flexible face management to support addition of new link types and protocols (e.g., NDNLP). We have created and implemented the basic framework for NDNFD, and added features, such as Ethernet support, and a simple flooding strategy.

3.2.5 Network Monitoring

We run scripts that monitor the status of NDN testbed by collecting information from link-state routing messages, the ccnd status page of all site gateways, then displaying the results on a web page. During Year 3, we maintained these scripts to keep up with the evolvement of the NDN testbed. We also re-designed the web page using NDN javascript library, so that now the status web page itself can be retrieved as NDN data over the testbed. In addition, we developed and maintained a basic network diagnosis tool, ccnping. The maintenance effort includes fixing bugs, adding more command-line options, and making the output more user friendly.

3.2.6 The NDN Simulator (ndnSIM) and its growing user base

Last year we developed an open source, ns-3 based NDN simulator, ndnSIM [2]. Since its release in summer 2012 (`http://ndnsim.net`) we have witnessed rapid growth of a community using it to study NDN design choices and system properties. We have worked with this community to further enhance ndnSIM's functionality and usability. ndnSIM's modular structure makes the extension easy and simple (Figure 3.4). This year we added the following functions and features to ndnSIM:

- an advanced application API to enable users to run and evaluate real application code on top of ndnSIM with minimal or no modification to the source code;
- support for multiple wire formats, which allows driving simulations using traffic traces from real application runs, as well as experimenting with new packet format proposals.
- simplified and unified network-layer API for handling Interest and Data packet, allowing more flexibility for users to experiment with forwarding strategies.
- "Exclude Filter" support; support for other Interest selectors is coming soon.
- support for overlay-based simulations, i.e., with the addition of TCP-face and UDP-face, one can use ndnSIM to mimic real NDN deployment over today's transport protocols[7].

We set up a mailing list end of 2012 to facilitate discussion; the list has over 150 subscribers from multiple countries (`http://www.lists.cs.ucla.edu/mailman/listinfo/ndnsim`). A year after the ndnSIM release, there have been at least 13 papers with results from using ndnSIM (`http://ndnsim.net/ndnsim-research-papers.html`); only 5 of which were authored by NDN team members. The broader community has also started contributing to ndnSIM development. In addition to bug reports and feature suggestions, there are 15 public folks on github, and several users have contributed code development to ndnSIM (`https://github.com/NDN-Routing/ndnSIM/blob/master/AUTHORS`).

[7]Transport protocol overlay, instead of IP tunnel, is often needed to traverse firewalls.

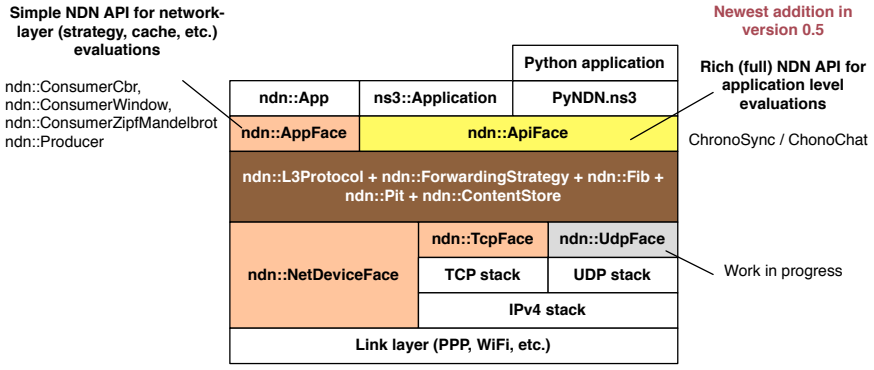

Figure 3.4: ndnSIM modules and new features

3.2.7 Future Plans

- Improve the design and implementation of NLSR for better efficiency, robustness, supporting adjacency discovery and broadcast links. Deploy NLSR on the NDN testbed.
- Finish the performance comparison between hyperbolic routing and link state routing using Emulab. Conduct the same experiments on the NDN testbed.
- Integrate the hyperbolic graph generator into the ndnSIM.
- Develop a full solution to scale routing tables based on the idea of forwarding hints; implement and deploy on NDN testbed.
- Implement forwarding strategies in NDNFD and test them on the testbed.
- Develop NDN congestion control solutions.
- Further expand ndnSIM functionality and its user community to make ndnSIM the common platform for all NDN related simulation experimentations.

References

[1] A. Markopoulou and G. Iannaccone and S. Bhattacharyya and C.-N. Chuahand Y. Ganjali, and C. Diot. Characterization of Failures in an Operational IP Backbone Network. *IEEE/ACM Transactions on Networking (TON)*, 16(4), August 2008.

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

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

[4] C. Yi and A. Afanasyev and I. Moiseenko and L. Wang and B. Zhang and L. Zhang. A case for stateful forwarding plane. *Computer Communications: Information-Centric Networking Special Issue*, 2013.

[5] 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.

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

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

[8] 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.

3.3 Scalable Forwarding

Contributors	
PIs	Patrick Crowley (Wash U)
Grad Students	Hila Ben Abraham, Shakir James, Haowei Yuan (Wash U)
Staff	John Dehart, Jyoti Parwatikar, Tian Song (Wash U)

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.

Our technical approach relies on the Open Network Lab, an NSF-funded testbed where experimenters can design a network topology, configure and program gigabit routers, and control Linux end-hosts. ONL currently contains 14 programmable routers, over 100 Linux hosts, and 1 Gb and 10 Gb links and switches. To support NDN experimentation, we added a new end-host type to ONL that boots with CCNx, installed and properly configured. CCNx routes and statistics can be changed and monitored via the remote laboratory interface (RLI). The RLI supports real-time charts, which can be used to report end-hosts characteristics in real-time, including link bandwidth, packets/s, queue lengths, CPU utilization, and memory utilization [2]. ONL accounts and usage are free. Registration and step-by-step CCNx instructions can be found at http://wiki.arl.wustl.edu/onl/index.php/CCNx_in_ONL.

The rest of this section provides more details about the activities and findings in the forwarding area during the third year of the NDN project.

3.3.1 Scalable Name-Based Forwarding

Named-based packet forwarding represents a core characteristic of the NDN architecture. IP-inspired forwarding methods are not feasible because a) name-based forwarding must support variable-length lookup keys of unbounded length, and b) namespaces for data are substantially larger than the global address prefix rule sets used in today's Internet. We introduce and evaluate an approach that can realistically scale variable-length name forwarding to billions of name prefixes [4]. Our methods are driven by two key insights. First, we show that it is possible to design a forwarding structure whose size is dependent upon the information-theoretic differences between rules in a ruleset, rather than by their length. Second, we show that a distributed forwarding plane can be engineered to support namespaces whose fundamental resource requirements exceed the resources of any single system. We reduce our designs to practice in both software and hardware. We demonstrate that 5.58 MiB memory is needed for the 1 million-name Alexa namespace and 7.32 GiB for a 1 billion-name synthetic set with similar but scaled characteristics. A single node sustains forwarding performance of 1 microsecond per lookup in traditional servers.

To enhance the performance of name-based forwarding, we propose fingerprint-based solutions [3]. The proposed methods improve the lookup performance of Patricia-trie based name forwarding scheme, and the proposed hash-table based designs are demonstrated to support scalable name-based forwarding with less memory and faster lookup speed. The fingerprint-based hash table design needs 1.68 GiB to store 402 million names, and the lookup latency is 0.6 microsecond.

3.3.2 Scalable Pending Interest Table Design

The Pending Interest Table (PIT) is a core component in Named Data Networking. In NDN, packets have unique names rather than IP addresses, and they are forwarded based on name-based lookup results. Unlike

IP forwarding, which performs lookup only in the Forwarding Information Base, NDN forwarding also looks up the Pending Interest Table, which keeps track of the currently unsatisfied Interest requests. Scalable PIT design is challenging because it requires per-packet updates, including memory write operations; and the content name strings stored in the PIT are long, requiring more memory space. As the line speed keeps increasing, e.g., 100 Gbps, traditional hash-table based designs cannot meet these two requirements. Fast memory chips can be used to support per-packet updates, but these chips usually have very limited capacities. Our approach is to reduce the PIT memory requirement for core routers, so that fast memory chips can be employed. We propose a novel Pending Interest Table design that guarantees packet delivery with a compact and approximate storage representation [1]. To achieve this, the PIT stores fixed-length fingerprints instead of name strings. To overcome the classical fingerprint collision problem, the Interest aggregation feature in the core routers is relaxed. The memory requirement and network traffic overhead are analyzed, and the performance of a software implementation of the proposed design is measured. Our results show that 36.72 MiB to 244.44 MiB are required to store the Pending Interest Table at 100 Gbps.

3.3.3 Future plans

- Continue to refine and evaluate 1+ Gbps software NDN forwarding plane prototypes.
- Continue to build and evaluate 10 Gbps hardware-based NDN forwarding plane prototype.
- Release performance evaluation of high-speed forwarding planes, and release source code to users.
- Publish papers and documentation.
- Release final versions to users.

References

[1] Haowei Yuan and Patrick Crowley. Scalable Pending Interest Table Deisgn: From Principles to Practice. submitted to IEEE INFOCOM 2014.

[2] Haowei Yuan and Patrick Crowley. Experimental Evaluation of Content Distribution with NDN and HTTP. In *INFOCOM Mini-Conference*, 2013.

[3] Haowei Yuan and Patrick Crowley and Tian Song. Enhancing Scalable Name-Based Forwarding. submitted to IEEE INFOCOM 2014.

[4] Tian Song and Haowei Yuan and Patrick Crowley. Scalable Name-Based Forwarding: From Millions to Billions. submitted to ACM CoNEXT 2013.

3.4 Security and Privacy

Contributors	
PIs	Gene Tsudik (UCI), Ersin Uzun (PARC), Christos Papadopoulos (CSU)
Grad Students ..	Steven DiBenedetto (CSU), Naveen Nathan, Cesar Ghali, Mishari Almishari (UCI)
Staff	**UCI Postdocs:** Paolo Gasti.
	UCI Visiting Scholars: Alberto Compagno, Mauro Conti, Gergely Acs, Marco Teoli.
	PARC Interns: Alaxendar Afanasyev, Ilya Moiseenko, Abdelberi Chaabane, Christopher Wood.

Rather than securing the communication channel at the application/session layers, as in today's Internet (through SSL/TLS), NDN secures authenticity and integrity of content packets via digital signatures. Named, signed and potentially encrypted content forms a solid foundation for security, but poses three major research challenges: cost-effective fine-grained security operations, usable trust management, and protection of privacy. Our security-related accomplishments in the third year included: design of aggregated signature objects and a secure fragmentation scheme; publishing proposed DDoS mitigation mechanisms: and supporting the security needs of a specific NDN environment: the building automation system. We also continued to explore trust models and key management frameworks, and privacy implications of NDN.

3.4.1 Content Signing/Verification and Secure Fragmentation

This year, we completed the design of aggregated signature objects (ASOs) in NDN [6]. ASOs are signed objects (like content objects) but they include hashes of other content packets as their payload, for efficient aggregated signing of multiple objects. Compared to the existing Merkle-Tree based aggregate signing in NDN, ASOs provide several advantages: (1) aggregate signing on the fly (i.e., as content packets produced) by publishing content objects first and then signing their hashes and publishing them later in an ASO (a Merkle-tree based approach requires buffering all content objects for signing), (2) reduced bandwidth usage and increased signing/verification performance (3) (combined with message authentication codes (MACs)) secure delegation of signing. ASOs make NDN more efficient, friendly to applications that dynamically generate or stream content, and provide a mechanism for resource-constrained devices (e.g., sensors, RFID tags, etc,) to securely delegate the expensive signing operation to more powerful devices.

We also designed a scheme that supports secure content fragmentation with relatively low overhead. NDN does not limit the size of content objects, and secure fragmentation of large objects by intermediate nodes is an open problem, since content can only be signed by its producer. If routers partition signed content into unpredictable-sized fragments, verifying origin authenticity or even integrity of such fragments seems initially impossible. In collaboration with Cisco we developed a solution [12] that support fragmentation of content packets while allowing nodes within the network to verify the authenticity and integrity of fragments.

3.4.2 (Distributed) Denial of Service Exploration

We published several studies that we started in the first two years of the project. We explained why typical IP-based DoS attacks are largely ineffective against NDN, but we described two types of NDN-specific DoS attacks, based on *interest flooding*, and *content/cache poisoning* [10]. We analyzed several flavors of Interest flooding attack, and developed and evaluated the effectiveness of several solutions [2, 8]. Interest Flooding and Content/Cache Poisoning attacks capitalize on two NDN-specific features: PITs and caches. We showed how the stateful nature of NDN routers naturally supports mitigation of flooding attacks [2]. In particular, NDN routers can track unsatisfied (expired) Interests and use this information to limit the number of pending Interests per outgoing interface, incoming interface, and per namespace. The flow balance requirement (one Data packet per Interest packet) allows a router to track and never send more Interests that an interface can satisfy based on average content packet size, expiration values, and bandwidth-delay products. Similarly, when a prefix is under attack, routers along the way (especially closer to the data producer) can limit pending

Interests for that prefix. These routers can propagate such information upstream toward offending interfaces, while limiting the rate of forwarded Interests for the namespace under attack [2, 8].

We explored countermeasures to **content/cache poising attacks**, by allowing prioritization of legitimate content in caches, and we evaluated their effectiveness through simulations [11]. The adversary's goal in a content poisoning attack is to cause routers to forward and cache corrupted or fake data packets[8], preventing consumers from retrieving legitimate content. NDN's signature verification will identify poisoned content, but this verification may be too much overhead for core routers. We explored three countermeasures: (1) a light-weight content ranking algorithm for NDN router caches that statistically ranks Interest packets; (2) mechanisms that allow consumers to specifically express Interests, either by adding a hash of the content as a name suffix (effectively self-certifying names), or indicating the acceptable public key (if available) for the expected content using the *publishers_key_digest* field of an Interest packet; (3) driven by consumer feedback to *verifier* nodes, which attempt to confirm the consumer's findings and try alternate paths accordingly as well as report stale information to upstream verifiers. For this last approach we (UCI) finished an initial system design and are implementing a prototype using the CCNx software.

3.4.3 Exploring NDN Security in Instrumented Environments

Instrumented environments, such as modern building automation systems (BAS), are becoming commonplace and are increasingly interconnected with (and sometimes by) enterprise networks and the Internet. Secure control of devices in such environments is challenging. As the industry moves from proprietary communication media and protocols toward IP, it has become clear that a new Internet architectures such as NDN might be better-suited for instrumented environments. We started by identifying security requirements in a specific instrumented environment lighting control [5]. UCLA and UCI teams constructed a concrete NDN-based security architecture, analyzed its properties and reported on implementation and experimental results. We delivered a fully functioning security prototype that implemented our proposed techniques and recently extended our research into secure sensing applications [4].

3.4.4 Trust Model and Key Management

Every NDN packet carries a signature that binds the name and the content. However, digital signatures are only meaningful and secure when public keys can be fetched and trusted. NDN does not mandate the use of any particular trust model: each application is free to adopt the trust model that best suits it. One trust model we have been exploring assumes rigid relations between keys and the namespaces in which they are valid to publish content. A content object published under namespace *name* must be signed using the key associated with *name* or any of its ancestors. The root of trust in this model is multiple trusted third parties (TTP), e.g., CAs, where the corresponding public keys for those TTPs are securely distributed to clients via out-of-band channels. A CA generates the key-pair $K_{root} = (pk_{root}, sk_{root})$ and distributes pk_{root}. In order to associate pk_P, belonging to producer P, with namespace $name_P$, a TTP publishes, under $name_P$`/key`, a content object containing pk_P. P can delegate a key $sk'_P \neq sk_P$ to sign content in namespace $name_P$`/sub-namespace` by publishing the corresponding public key pk'_P under $name_P$`/sub-namespace/key`. This mechanism allows TTP to delegate certification capabilities to each producer.

Another model we explored this year using the NDN testbed is publishing keys in a common namespace that is efficiently distributed and cached for fast access to verification keys [3]. We used the Repo and SYNC tools in the CCNx/NDN distribution, where the system publishes NDN testbed keys under a strict trust chain starting from a single root. Each NDN testbed site has a site key, signed by the root key, and user keys are signed by corresponding site keys. All keys are published under a common namespace $NDN/keys$ and stored in the repo of each site and continuously synced using the SYNC tool. Verification of keys follows the trust chains until a trusted anchor (either the root key or any cached key) is reached. We built a simple key publication and synchronization application for operators to publish user's keys in the NDN testbed, and also a key verification library (in C/C++) for application developers to leverage NDN's built-in security.

[8] We say that a data packet is corrupted if its signature is invalid, but *fake* if it has a valid signature generated with a wrong (private) key.

3.4.5 Privacy

Privacy issues span four components of the architecture: names, content, caches, and signatures. Last year, in our anonymous named data networking application (ANDaNA)[9], we showed an onion routing application that was more efficient than Tor by requiring fewer onion hops and introducing less delay. This year, we compared privacy aspects of NDN with those of other information-centric architectures and the current Internet. We discussed our results and potential solutions that address each dimension of privacy [7]. We also published a formal framework to evaluate privacy of caches and countermeasures to compromises [1].

3.4.6 Future plans

- Further investigation of DoS/DDoS and architectural resilience issues.
- Security and privacy considerations for synchronous (low-latency) traffic and control applications.
- Design and evaluation of additional new privacy-protection and anonymization techniques.
- Design and publishing of a generic trust and key distribution mechanism.
- Continue developing more efficient cryptographic primitives/tools for content verification.

References

[1] Gergely Acs, Mauro Conti, Paolo Gasti, Cesar Ghali, and Gene Tsudik. Cache privacy in named-data networking. In *IInternational Conference on Distributed Computing Systems (ICDCS)*, 2013.

[2] Alexander Afanasyev, Priya Mahadevan, Ilya Moiseenko, Ersin Uzun, and Lixia Zhang. Interest flooding attack and countermeasures in Named Data Networking. In *IFIP/IEEE International Conferences on Networking (Networking)*, 2013.

[3] Chaoyi Bian, Zhenkai Zhu, Alexander Afanasyev, Ersin Uzun, and Lixia Zhang. Deploying key management on NDN testbed. Technical Report NDN-0009, Revision 2, NDN, February 2013.

[4] J. Burke, P. Gasti, N. Nathan, and G. Tsudik. Secure sensing over named data networking, 2013. Under submission.

[5] J. Burke, P. Gasti, N. Nathan, and G. Tsudik. Securing instrumented environments over content-centric networking. In *IEEE Workshop on Emerging Design Choices in Name-Oriented Networking(NOMEN)*, 2013.

[6] CCNx project. CCNx Signature Generation and Verification. `http://www.ccnx.org/releases/ccnx-0.8.1/doc/technical/SignatureGeneration.html`.

[7] Abdelberi Chaabane, Emiliano De Cristofaro, Muhammed A. Kafaar, and Ersin Uzun. Privacy in content-oriented networking: threats and countermeasures. *ACM Computer Communication Review*, July 2013.

[8] Alberto Compagno, Mauro Conti, Paolo Gasti, and Gene Tsudik. Poseidon: Mitigating interest flooding ddos attacks in named data networking. *Local Computer Networks (LCN)*, 2013.

[9] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun. ANDaNA: Anonymous named data networking application. In *NDSS*, 2012.

[10] Paolo Gasti, Gene Tsudik, Ersin Uzun, and Lixia Zhang. DoS and DDoS in Named-Data Networking. In *International Conference on Computer Communications and Networks (ICCCN)*, 2013.

[11] Cesar Ghali, Gene Tsudik, and Ersin Uzun. Needle in a Haystack: Mitigating Content Poisoning in Named-Data Networking, 2014. Under submission.

[12] A. Narayanan, N. Nathan, D. Oran, and G. Tsudik. Secure fragmentation for content-centric networks, 2013.

3.5 Fundamental theory for NDN

Contributors	
PIs	PIs: Edmund Yeh (Northeastern)
Grad Students	Postdoc: Ying Cui (Northeastern)
Undergrads	Grad Students: Milad Mahdian, Fangxiang Wang (Northeastern)
Staff	Undergrad Students: Kyle Dumont (Northeastern)

3.5.1 Virtual Interest Packets

To realize the true potential of the NDN paradigm, we need a new fundamental theory of networked communication, which poses some key challenges. First, the classical information theory assumption that data streams sent over different source-destination pairs are independent will not hold in NDN. In NDN, data traffic responding to Interests is likely to be statistically dependent, requiring new optimal coding theory and practice for multiple-source multicast. Second, A theory for NDN must capture the essential tradeoff between wires and storage in optimizing communication performance, so that for a given set of link capacities and buffer sizes, we can analyze combinations of transmission and caching strategy to optimize network performance. The essential role of caching data in NDN also raises the importance of optimal asynchronous or delay-tolerant multicast. Two additional issues not prominent in classical theory are the *time value of information* (information cached too long may be useless once reaching receivers) and the central role of mobility in assessing the tradeoff between storage and transmission [3].

The multi-dimensional generalization of classical point-to-point Shannon capacity is the *network capacity region*, consisting of all the source-destination communication rates achievable using any feasible coding scheme [1]. The corresponding *coding* problem is to find low-complexity error correcting codes which allow for reliable communication at the rates given by the capacity region.

This year we made significant progress in developing optimal dynamic forwarding and caching algorithms for maximizing the total consumed bit rate in an NDN network. Several recent contributions on forwarding and caching in content-centric networks [2, 4, 6] have left open the problem of optimal joint forwarding and caching for multi-hop content-oriented networks. To address this challenge, we introduced the *VIP framework*, which uses *Virtual Interest Packets* (VIPs) to capture measured network demand for data objects. The VIP framework uses a *virtual* control plane that operates on VIPs, and an *actual* plane that handles Interest and Data Packets. This framework enables a large class of control and optimization algorithms operating on VIPs in the virtual plane, as well as a large class of mappings that use the VIP flow rates and queue lengths from the virtual plane to specify forwarding and caching in the actual plane. The VIP framework presents a powerful paradigm for designing efficient NDN-based networks with different properties and trade-offs. We developed two instantiations of the framework: the first with a forwarding/caching policy in the virtual plane, which achieves effective load balancing and maximizes the throughput of VIPs; and the second consisting of algorithms that achieve not only load balancing but also stable caching configurations. Experimental results show that the latter set of algorithms have superior performance in terms of user delay and cache hit rate.

Consider a connected multi-hop network consisting of N nodes and L directed links. Let $C_{ab} > 0$ be the transmission capacity (in bits/second) of link (a, b). Let L_n be the cache size (in bits) at node n. We assume a fixed set of K data objects, and that $L_n < K$ for all $n \in \mathcal{N}$. For each data object, assume that there is a unique node $src(k)$ that serves as the content source for the object. Interest Packets (IPs) for a given data object can enter the network at any node, and exit the network upon being satisfied by a matching Data Packet (DP) at the content source for the object, or at the nodes which decide to cache the object. We assume that content sources are fixed, caching points vary, and FIBs already have entries for Data objects, i.e., topology discovery has already occurred. Our goal is to find dynamic forwarding and caching policies that perform well in terms of metrics such as the total number of Interest packets satisfied (i.e. a corresponding Data packet is received by the requesting node), delay in satisfying Interest packets, and cache hit ratios.

The VIP framework for joint dynamic forwarding and caching relies on the essential new device of *virtual Interest packets* (VIPs), which are generated as follows. For each Interest packet for Data object k entering the network, a corresponding VIP for object k is generated. NDN nodes may aggregate duplicate Interests, but not VIPs; the latter parameter captures the empirically *measured demand* for the requested data objects in the network. The key insight is that control algorithms operating in the virtual plane can exploit local information on network demand (as represented by VIPs) that is unavailable in the actual plane due to Interest collapsing and suppression. The flow rates and queue lengths of the VIPs resulting from the control algorithm in the virtual plane are then used to specify the forwarding and caching policies in the actual plane. After being forwarded, the content source and any nodes that have cached object k remove VIPs for that object. Physically, the VIP count can be interpreted as a *potential*. For any given data object, there is a downward "gradient" from entry points of the Interests to the content source and caching nodes.

In the first instantiation of the VIP framework, called Algorithm 1, the VIP count metric determines both the forwarding and caching algorithms in the virtual control plane. The forwarding strategy is given by the application of the backpressure algorithm [5] to the VIP queue state. The caching strategy is given by the solution of a max-weight problem involving the VIP queue size. The VIP flow rates and queue lengths are then mapped to forwarding and caching strategies in the actual plane. We showed that the forwarding/caching strategy is *throughput-optimal* within the virtual plane, in that it maximizes the throughput of VIPs in the network with appropriate transmission rate constraints. Specifically, let the *VIP network stability region* Λ be the closure of the set of all VIP arrival rates $\boldsymbol{\lambda}$ for which there exist some feasible joint forwarding and caching policy that can guarantee stability of VIP queues. Note that if the Interests are not collapsed or suppressed at the PITs, then the Interest Packet (IP) stability region in the actual plane is the same as the VIP stability region in the virtual plane. Since the VIP count of a data object is decremented immediately after caching the object at node n, the strategy in Algorithm 1 exhibits oscillatory caching behavior, whereby cached data objects are immediately removed from the cache again due to the VIP counts of other data objects now being larger. Thus, even though Algorithm 1 is throughput optimal in the virtual plane, its mapping to the actual plane leads to forwarding/caching policies that are difficult to implement. We developed a more practical VIP algorithm (Algorithm 2) that looks for a stable solution in which cache contents do not oscillate. Although the algorithm is not theoretically optimal (optimal caching is in general NP-hard), it provides significant performance gains in simulation experiments. The forwarding policies in the virtual and actual planes under Algorithm 2 are the same as those under Algorithm 1. The caching policy for the actual plane is based on the guidance provided by the VIP flows in the virtual plane.

Suppose that at time slot t, node n receives a Data Packet (DP) for data object k_{new} which is not currently cached. If there is sufficient unused space in the cache of node n to accommodate the DP for k_{new}, then node n proceeds to cache it. Otherwise, the node compares the *cache scores* for k_{new} and the currently cached objects, computed as the average number of VIPs for object k received by node n over a sliding window of size T prior to time slot t. Let $\mathcal{K}_{n,old}$ be the set of objects that are currently cached at node n. If some object k' in the cache has a lower score than object k_{new}, then object k' is evicted and replaced with object k_{new}. Otherwise, the cache is unchanged. If objects have different sizes, the optimal set of objects is chosen so as to maximize the total cache score. This is a knapsack problem for which low complexity heuristics exist. At each time t, the VIP count at node n for object k is decreased due to the caching at node n, which attracts VIPs for each object $k \in \mathcal{K}_{n,new}$ to node n. The DPs for data objects evicted from the cache diffuse to other parts of the network, potentially cached more efficiently where demand for the evicted data object is higher. Before the data object is evicted, VIPs and IPs flow toward the caching point, a sink for the object. After eviction, the VIP count increases since VIPs would not exit at the caching point. As the VIPs increase further, the back pressure forwarding policy would divert them away from the current caching point to other parts of the network.

We ran simulation experiments to compare the joint caching-forwarding VIP algorithm (Algorithm 2) against two different caching algorithms, Least Frequently Used (LFU) and Least Recently Used (LRU), both employed in conjunction with shortest path routing. We tested these three algorithms under different scenarios on two topologies: the Abilene topology[9] and a Service Network topology. Experimental results show conclusively that the VIP joint forwarding/caching algorithm has significantly improved performance in terms of user delay and the rate of cache hits.

[9]See `https://itservices.stanford.edu/service/network/internet2/abilene`.

3.5.2 Future plans

In the final year of the project, we plan to enhance the practical utility of the forwarding and caching algorithms by investigating the following issues:

- reduce the state (number of VIP counters) necessary in carrying out the forwarding and caching
- generalize algorithms to handle dynamic generation of content names
- improve algorithms to further reduce user delay in obtaining requested content
- develop congestion control algorithms for NDN networks with dynamic forwarding and caching, in order to maximize network throughput while enforcing fairness among different content objects.
- develop dynamic forwarding and caching algorithms to maximize the total information collected by NDN-based sensor and mobile networks

References

[1] Thomas M. Cover and Joy A. Thomas. *Elements of information theory.* Wiley-Interscience, New York, NY, USA, 1991.

[2] S. Eum, K. Nakauchi, M. Murata, Y. Shoji, and N. Nishinaga. Catt: Potential based routing with content caching for icn. In *Proceedings of SIGCOMM 2012 ICN*, pages 49–54, Helsinki, Finland, August 2012.

[3] Matthias Grossglauser and David N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Trans. Netw.*, 10(4):477–486, 2002.

[4] Meghana M. Amble, P. Parag, S. Shakkottai, and L. Ying. Content-aware caching and traffic management in content distribution networks. In *Proceedings of IEEE INFOCOM 2011*, pages 2858–2866, Shanghai, China, April 2011.

[5] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling for maximum throughput in multihop radio networks. *IEEE JSAC*, 37(12):1936–1949, Dec 1992.

[6] H. Xie, G. Shi, and P. Wang. Tecc: Towards collaborative in-network caching guided by traffic engineering. In *Proceedings of IEEE INFOCOM 2012:Mini-Conference*, pages 2546–2550, Orlando, Florida, USA, March 2012.

Chapter 4

Values & Policy in NDN Design

Contributors	
PIs	Katie Shilton (Univ. of Maryland), Paul Ohm (Univ. of Colorado Boulder)
Grad Students	Jes Koepfler, Amalia Levy (Univ. of Maryland)
Staff	Charles Duan (Univ. of Colorado)

The NDN project was extended in 2011 to include two new activities: Values in the Design of the NDN Architecture (VD-NDN), led by a team of social scientists, and Values from Law and Policy in NDN (VLP-NDN), led by legal scholars. These two complementary projects support the design, impact, and implementation of NDN by evaluating how the NDN architecture might affect the social and policy environments in which it is deployed. The research objectives for VD-NDN are: 1) understanding social challenges and values engendered by the proposed architecture, 2) investigating design activities that encourage discussion of values during design, 3) operationalizing values such as privacy, anonymity, security and information equity as design principles and features, and 4) understanding NDN's impacts on information policy. VLP-NDN focuses on values derived from law and policy. The objectives are to: identify how today's laws would respond to the NDN design; investigate how law expresses important values that designers should consider; prepare researchers for the inevitable conflicts likely to arise in a future NDN Internet; and understand the political economy of how engineering designs can anticipate, and surmount, political roadblocks on the path to implementation.

We completed our initial goal of mapping relationships between campuses, PIs, and work practices of the NDN team. PI Shilton attended one all-hands meeting with the NDN PIs, one all-PI meeting for the FIA program, and numerous conference calls to take notes and observe. As a result of this activity, Shilton and her graduate assistant developed a taxonomy of values concerns in the NDN project. This year we used this taxonomy to understand social challenges and values in the NDN project [2, 1], We also interviewed project PIs and students about their work on the NDN project, and how their work is impacted by values in NDN design. We began to code interview data to investigate values levers in NDN design.

In collaboration with other NDN team members, the VD-NDN subteam is also writing a technical report entitled *A World on NDN*, which examines key components of the architecture and critical departures from TCP/IP, and the implications of these departures for social issues such as free speech, security and privacy, law enforcement, and network neutrality. Analysis of the social impacts of NDN highlights four departures from TCP/IP: an emphasis on enabling *semantic classification, provenance, publication,* and *decentralized communication.* While all of these ideas can be implemented in the current Internet's application layer, and not every NDN application need use them, NDN enables these values by design, and encourages development of applications that comport with them. By changing the way data is identified, handled, and routed across the web, these changes from TCP/IP will improve opportunities for free speech, produce positive changes for security, privacy and anonymity while further complicating data retention and forgetting, and alter current cyber law enforcement mechanisms. These changes also produce complications for ensuring network

neutrality, requiring future work to identify fair network management mechanisms. NDN's overall impact on network neutrality remains an open question, dependent upon choices still to be made in naming and routing.

The CU-Boulder (VLP-NDN) team did research on the DMCA notice and takedown provisions of copyright law, considering how they could be reframed to better suit NDN technology, and conversely which NDN features could be added to support the underlying legal policy framework. The team conducted an NDN discussion seminar to talk about these issues. The legal team prepared an analysis that recommends implementing features to enable deletion and expiration timers for content in caches and repos, to support DMCA safe harbor protections. They also identified areas where the legal provisions of the DMCA safe harbors did not adapt well to the technical innovations of NDN, e.g., DMCA tends to assume communications from server to server, which is incongruent with NDN's content focus. Also, the safe harbor provisions assume that content is stored at the endpoints of the network and that the network core only relays data; when data is stored in the network core as in NDN, it implies substantial legal burdens due to these safe harbor provisions.

4.1 Future plans

In the final year we will pursue

- an empirical analysis of data naming strategies in target application domains to help us understand how existing practices might impact semantically-meaningful naming practices in NDN.
- examine how these environments' privacy and security requirements can best be reflected in naming practices, routing protocols, and network storage polices can strengthen next-stage NDN design.
- investigate social questions emerging from more general architectural choices, such as how prioritization decisions in data-centric routing protocols could impact network neutrality, can promote not only a more efficient, reliable and trustworthy Internet but also one that more fundamentally enables democracy and equity of information access.

References

[1] Katie Shilton and Jes A. Koepfler and Kenneth R. Fleishmann. Charting Sociotechnical Dimensions of Values for Design Research. 29(5), 2013.

[2] Katie Shilton and Jes A. Koepfler. Making space for values: communication & values levers in a virtual team. In *6th International Conference on Communities and Technologies*, New York, NY, USA, 2013. ACM.

Chapter 5

Education Impacts

Contributors	
PIs	Tarek Abdelzaher (UIUC), Dan Massey & Christos Papadopoulos (CSU), Lan Wang (Memphis), Gene Tsudik (UCI), Beichuan Zhang (Arizona), Van Jacobson, Jeff Burke, Lixia Zhang (UCLA)

The NDN architecture development gives us a unique opportunity to teach students how to think architecturally. Today instructors may strive to teach architectural concepts, but unfortunately students often focus on mechanics, e.g., *what* fields are in an IP header, rather than *why* those fields are in the IP header. We use the comparison between the IP architecture and the NDN architecture as a specific case study to teach students architectural evolution, as well as how individual components fit in. This theoretical view is complemented by a more tangible projects where students use the NDN prototype to explore the systemic impacts of design choices. Students are exposed to NDN architectural concepts in the context of real software development challenges, and use APIs to the core NDN layer to build software that explores these concepts. Implementing projects under both IP and NDN paradigms inform theoretical and practical discussion of networking concepts. Running the NDN variants on our testbeds lead to unexpected network behavior that challenges the testbed student operators. Finally, we record exchanges and discussions from the NDN design and development efforts, capturing the *process* of creating a new architecture to enable such research and teaching in the future.

During this reporting period we launched biweekly NDN seminar series among participating universities. NDN seminar topics range from design discussions to societal impact, a great opportunity for graduate students to broaden their views and vision. 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. UIUC graduate student Shiguang Wang coordinates the seminars, which so far have covered the following topics:

- *Key publishing on NDN testbed*: Alex Afanasyev (UCLA) presented and led discussion of the current proposed design.
- *DeNardis - Hidden Levers of Internet Control*: a paper published in the journal "Information, Communication & Society (iCS)". Katie Shilton of UMD presented the paper and explored potential changes NDN may bring to the picture.
- *Digital Millennium Copyright Act*: Charles Duan of Univ. Colorado led the discussion on the impact of DMCA on NDN and vice versa.
- *Performance Study of CCNx*: Haowei Yuan of Washington University reported their measurement results on the NDN prototype implementation.
- *Naming in Content-Oriented Architectures*: a paper from SIGCOMM ICN Workshop 2011. Yingdi Yu of UCLA presented a contrast and comparison study of this paper with NDN's naming and security

design approach.

- *Blending of hierarchical and non-hierarchical naming*: a 1992 ICDCS paper. Jeff Burke of UCLA presented the paper and articulated different ways to design NDN naming.
- *NLSR: Named-data Link State Routing Protocol*: A K M Mahmudul Hoque of Univ. Memphis presented this new routing protocol developed by NDN routing group.
- *Content Poisoning in NDN*: Steve DiBenedetto of Colorado State presented his proposed solution to collect feedback from the whole team.
- *ChronoShare file sharing*: Alex Afanasyev of UCLA presented this new NDN application design (aka NDN-dropbox) which has been in use among UCLA students.
- *NSA wiretapping and NDN*: Charles Duan of Univ. Colorado led the discussion on the potential impact of NDN architecture on wiretapping.
- *Smart buildings: Security of lighting controls using Named Data Networking*: Victor Perez and Turker Garip of UCLA reported their findings from doing a redteam attack against the prototype implementation of a simple NDN lighting control system.

Curriculum development teaching activities include:

University of Arizona During Year 3 Beichuan Zhang included NDN material in 2 courses: CS 296H/496H, a research seminar for undergraduate students, and CS 525, a graduate level network class. CS296H/496H has one introductory lecture on NDN to stimulate students' interest in network architecture research. CS 525 has 2 lectures that cover NDN's basic concepts, operations, research issues, and progresses. The course also contains materials on other future Internet designs and other ICN designs to provide a more comprehensive view of the research area.

Colorado State Similar to the Arizona model, Colorado State PIs introduced basic NDN concepts in both undergraduate and graduate courses, adding more substantial projects as the courses advanced. Dan Massey included an NDN lecture in the undergraduate computer network course, CS 457 - Computer Networks. Christos Papadopoulos taught NDN in the first year graduate networking course CS557 - Advanced Computer Networks, and advanced graduate course CS 657 - Computer Networks Seminar, as well as the undergraduate networking course. CS 557 dedicated four lectures out of 30 to the NDN architecture. One (of three) course project involved substantial programming development in NDN.

As part of the educational material for the broader community we produced a "Hello World"-style class project for NDN, which shows how to fetch an object over an NDN network. This project familiarizes students with NDN application development and provides a starting point for more complex programming assignments involving NDN naming, congestion control, routing, etc. The code is structured along the familiar lines of other socket programming tutorials, with a client and server program. The project description also includes instructions on how to install CCNx, step by step instructions to run the code, and a set of questions instructors can ask students to answer. We plan to use this project in future networking classes to introduce students to the nuts and bolts of NDN programming, and to build and share more complex class assignments.

We have also contributed educational material to the GENI community. In collaboration with the GENI people we have published an exercise using the GENI testbed[1] that demonstrates the power of NDN naming and caching to support sharing of large distributed data sets. The exercise includes tools and material that climate researchers use, such as tools that manipulate data in NetCDF format, and a set of questions for students to answer. The exercise is fairly extensive, asking students to setup NDN nodes, routing information and data used in the exercise. Students using this 12-page exercise will also familiarize themselves with the GENI educational infrastructure. CSU will use this in a Fall 2013 undergraduate course.

The advanced seminar at CSU, CS 657, was devoted primarily to NDN by both Massey and Papadopoulos. Again similar to the Arizona's graduate course structure, the course focused on reading and discussions with substantial individual projects. The seminar course is offered every other year, with this past year being the off year.

[1] `http://groups.geni.net/geni/attachment/wiki/ScalableMonitoring/ccnx-geni.pdf`

UCI During the 2012/2013 academic year, Gene Tsudik directed the Networked Systems Seminar Series at UCI. As part of this year-long seminar course (taken by 30+ MS and PhD students), several talks focused on next-generation Internet topics. These included:

- Jeff Burke (UCLA), "Named Data Networking",
- Srikanth V. Krishnamurthy, "Recent Research Endeavors in Mobile Computing and Social Network Privacy for Cyberphysical Systems",
- George Kesidis (PSU), "Anomaly Detection Among Packet-Flows", and
- Ken Calvert (UKY), "Refactoring the Internet for the 21st Century".

UCLA Van Jacobson presented a department seminar on "Evolving the Internet into the Future via Named Data Networking" in January 2013 to an overflow auditorium, including 60+ undergraduate students who took the seminar as a special class for CS118, an introductory course to networking.

Lixia Zhang's annual graduate seminar course "CS217B: Advanced Topics in Internet Research" has been devoted to the NDN architecture research discussions since 2010. During this reporting period (September 2012 – August 2013), Zhang taught CS217B in spring 2013. This course also provided in depth studies of other FIA designs, including Mobility-First and XIA. To provide students a broad perspective on network architecture research, the course also covered materials ranging from early work "Internet Indirection Infrastructure" (SIGCOMM 2002) and "A Data-Oriented (and Beyond) Network Architecture" (SIGCOMM 2007), to recent developments such as "Intelligent Design Enables Architectural Evolution" (Hotnets 2011) and "Software-Defined Internet Architecture" (Hotnets 2012). The students compared different network architecture design choices across time and the design spectrum to learn how to *think architecturally*.

In parallel to architecture design discussions, the students also carried out the following course projects:

- developing solutions to mitigate content poisoning attacks against NDN networks,
- developing key management and visualization tools to facilitate NDN's trust management,
- developing an "NDN-dropbox" file sharing application built on top of NDN's distributed synchronization support (this work has been submitted to CoNext 2013), and
- organizing a red team to attack NDN lighting control system prototype (this work has been accepted by IEEE Network Security Workshop 2013).

Memphis Lan Wang gave a talk on NDN in COMP7960, which is a course that all graduate teaching and research assistants must take. The title of the talk was "Adaptive Forwarding in Named Data Networking."

We plan to make all the lecture materials developed from the above efforts publicly available to promote broader incorporation.

References

- NDN Project Education Webpage, `http://named-data.net/education.html`
- CCNx BootCamp Webpage, `http://www.ccnx.org/ccnxbootcamp2011/`
- AsiaFI NDN Hands-on Workshop Webpage, `http://www.asiafi.net/org/ndn/hands-on2012/`

Chapter 6

Global Outreach

1. The NDN project continues to attract attention from the global networking community. The second CCNx Community Meeting took place September 12-13, 2012 at INRIA's Sophia Antipolis Méditerranée Research Center, France (see `http://www.ccnx.org/ccnxcon2012/`). Over 100 people attended and exchanged their research results in moving NDN forward. NDN also continues to draw interest from Asia Pacific region. The Future Internet Forum (AsiaFI, `http://www.asiafi.net/`) organized "AsiaFI 2013 Summer School" in conjunction with SIGCOMM 2013 in Hong Kong (`http://asiafi.net/meeting/2013/summerschool/home.html`), and invited the NDN team to provide an ndnSIM tutorial and NDN research session.

2. An increasing number of research efforts closely aligned with NDN are underway around the world, as suggested by the growth in academic workshop and conference activities. ACM SIGCOMM has hosted three Information Centric Networking Workshops (ICN) in 2011, 2012, and 2013 respectively. IEEE INFOCOM hosted two workshops on Emerging Design Choices in Name-Oriented Networking (NOMEN), in March 2012 and April 2013 respectively. IEEE ICNP 2013 also chose ICN as a major theme. Most NDN project PIs have served on the Technical Program Committees for these workshops. Lixia Zhang served as either one of the main organizers or TPC co-chair for all of these events. Since late 2012 the organizers of ACM ICN Workshops and IEEE NOMEN Workshops have been preparing an ACM ICN conference series. The first ACM Information Centric Networking Conference is currently planned for September 2014; Lixia Zhang will serve as the TPC co-chair.

3. **We presented a live NDN demo at the China-America Frontiers of Engineering Symposium in Beijing.** We used the NDN testbed and approximately 1000 Amazon EC2 instances across five continents (Figure 6.1). The demo itself consisted of three phases. 1) Demonstrate that keys can be signed transitively to form a chain of truest. 2) Stream video to approximately 1000 video clients spread across five continents, using automatic multi-path route switching and requiring only one copy of data on any link. 3) Broadcast a live audio and video from performance studio at UCLA, and demostrate the NDN-based lighting control application.

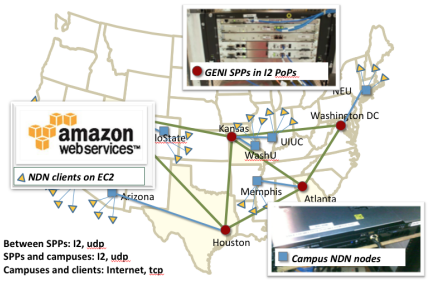

Figure 6.1: NDN Testbed

Chapter 7

Year 3 Publications

1. "Enhancing Scalable Name-Based Forwarding" by Haowei Yuan, Patrick Crowley, and Tian Song. Submitted to IEEE INFOCOM 2014.
2. "Scalable Pending Interest Table Deisgn: From Principles to Practice" by Haowei Yuan and Patrick Crowley. Submitted to IEEE INFOCOM 2014.
3. "Social Values in a Future Internet: Central Values in the Named Data Networking Project" by Katie Shilton. Submitted to *Ethics and Information Technology.*
4. "Scalable Name-Based Forwarding: From Millions to Billions", Tian Song, Haowei Yuan, and Patrick Crowley. Submitted to ACM CoNEXT 2013.
5. "VIP: A Framework for Joint Dynamic Forwarding and Caching in Named Data Networks" by Edmund Yeh, Tracey Ho, Michael Burd, Ying Cui and Derek Leong. Submitted to ACM CoNEXT 2013.
6. "The story of ChronoShare, or how NDN brought distributed file sharing back" by A. Afanaysev, Z. Zhu and L. Zhang. Submitted to ACM CoNEXT 2013.
7. "NDNFS: An NDN-oriented File System" by W. Shang, Z. Wen, Yuanjie Lie, A. Afanasyev, and L.Zhang. Submitted to ACM CoNEXT 2013.
8. "Exploring Covert Channels in Named Data Networking", by M. Conti, P. Gasti and G. Tsudik, In Submission, 2013.
9. "Secure Sensing over Named Data Networking" by J. Burke, P. Gasti, N. Nathan and G. Tsudik. Under submission.
10. "Securing Lighting Control over NDN" by J. Burke, P. Gasti, N. Nathan and G. Tsudik. Under submission.
11. "Needle in a Haystack: Mitigating Content Poising in Named Data Networking" by Cesar Ghali, Gene Tsudik and Ersin Uzun. Under submission.
12. "Making Named Data Networking Resilient Against Cache Poisoning" by Cesar Ghali, Gene Tsudik, and Ersin Uzun. Under submission.
13. "Secure Content Individualization and Dissemination in Content Centric Networks" by Christopher A. Wood and Ersin Uzun. Under Submission.
14. "Optimizing Bi-directional Low-Latency Communication in Named Data Networking" by M. Almishari, N. Nathan, P. Gasti and G. Tsudik. Under submission.
15. "A Lightweight Mechanism for Detection of Cache Pollution Attacks in Named Data Networking" by M. Conti, P. Gasti and M. Teoli. Computer Communications, accepted, to appear.
16. "Let's ChronoSync: Decentralized Dataset State Synchronization in Named Data Networking" by Z. Zhu and A. Afanasyev. Proceedings of IEEE International Conference on Network Protocols (ICNP 2013), October 2013.

17. "Security Evaluation of a Control System Using Named Data Networking" by Victor Perez, Mevlut Turker Garip, Silas Lam, and Lixia Zhang. Eighth Workshop on Secure Network Protocols (NPSec), October 2013.
18. "Poseidon: Mitigating Interest Flooding DDoS Attacks in Named Data Networking" by A. Compagno, M. Conti, P. Gasti and G. Tsudik. IEEE Local Computer Networks (LCN), October 2013.
19. "Secure Fragmentation for Content-Centric Networks" by A. Narayanan, N. Nathan, D. Oran and G. Tsudik. CCNxCon2013, September 2013.
20. "NLSR: Named-data Link State Routing Protocol" by A K M Mahmudul Hoque, Syed Obaid Amin, Adam Alyyan, Lan Wang, Beichuan Zhang, and Lixia Zhang. Proceedings of SIGCOMM ICN workshop, August 2013.
21. "Privacy in content-oriented networking: threats and countermeasures" by Abdelberi Chaabane, Emiliano De Cristofaro, Muhammed A. Kafaar and Ersin Uzun. ACM Computer Communication Review, July 2013. *Invited paper presentation at ACM SIGCOMM 2013*
22. "Video Streaming Over Named Data Networking" by Derek Kulinski, Jeff Burke, and Lixia Zhang. IEEE COMSOC Multimedia Communications Technical Committee E-Letter, July 2013.
23. "Cache Privacy in Name-Data Networking" by G. Acs, M. Conti, C. Ghali, P. Gasti and G. Tsudik. IEEE ICDCS, July 2013.
24. "DoS and DDoS in Named-Data Networking" Paolo Gasti, Gene Tsudik, Ersin Uzun and Lixia Zhang. International Conference on Computer Communications and Networks (ICCCN), 2013.
25. "MINERVA: Information-Centric Programming for Social Sensing" by S. Wang, S. Lu, S. Li, H. Liu, M. Uddin, and T. Abdelzaher. International Conference on Computer Communications and Networks (ICCCN), Nassau, Bahamas, July 2013.
26. "Making Space for Values: Communication & Values Levers in a Virtual Team" by Katie Shilton and Jes Koepfler. *Proceedings of 6th International Conference on Communities & Technologies (C&T 2013)*, June 2013.
27. "Interest flooding attack and countermeasures in Named Data Networking" by Alexander Afanasyev, Priya Mahadevan, Ilya Moiseenko, Ersin Uzun and Lixia Zhang. IFIP/IEEE International Conferences on Networking (Networking), May 2013
28. "ChronoShare: a new perspective on effective collaborations in the future Internet" by Z. Zhu, A. Afanasyev, and L. Zhang. UCLA Tech Forum Poster, May 2013.
29. "Vehicular Inter-Networking via Named Data" by G. Grassi, D. Pesavento, L. Wang, G. Pau, R. Vuyyuru, R. Wakikawa, and L. Zhang. ACM HotMobile 2013 Poster.
30. "Securing Instrumented Environments over Content-Centric Networking: the Case of Lighting Control" by J. Burke, P. Gasti, N. Nathan, and G. Tsudik. Proc. IEEE INFOCOMM 2013 NOMEN Workshop, April 2013.
31. "NDN-JS: A JavaScript Client Library for Named Data Networking" by W. Shang, J. Thompson, M. Cherkaoui, J. Burke, and L. Zhang. Proc. IEEE INFOCOMM 2013 NOMEN Workshop, April 2013.
32. "Experimental Evaluation of Content Distribution with NDN and HTTP" by Haowei Yuan and Patrick Crowley. Proc. of INFOCOM 2013 Mini-Conference.
33. "Information-maximizing Caching in Ad Hoc Networks with Named Data Networking" by W. Dron, A. Leung, M. Uddin, S. Wang, T. Abdelzaher, R. Govindan, and J. Hancock. Proc. 2nd IEEE International Workshop on Network Science (NSW), West Point, NY, April, 2013.
34. "A Case for Stateful Data Plane" by Cheng Yi, Alexander Afanasyev, Lan Wang, Beichuan Zhang, Lixia Zhang. Computer Communications, vol. 36, no. 7, pp. 779791, April 2013.
35. "Mitigation of Cache Pollution Attacks in NDN" by M. Teoli. MSc Thesis, University of Padova (Italy), 2013.
36. "Countermeasures for Interest Flooding Attacks in NDN" by A. Compagno. MSc Thesis, University of Padova (Italy), 2013.

37. "Charting Sociotechnical Dimensions of Values for Design Research" by Katie Shilton, Jes Koepfler, and Kenneth Fleischmann. *The Information Society, 29*(5), November 2012.
38. "Popularity versus similarity in growing networks" by Fragkiskos Papadopoulos, Maksim Kitsak, M. Ángeles Serrano, Marián Boguñá, and Dmitri Krioukov. *Nature*, 489:537–540, 2012.
39. "Replaying the geometric growth of complex networks and application to the AS internet" by Fragkiskos Papadopoulos, Constantinos Psomas, and Dmitri Krioukov. *ACM SIGMETRICS Perf E R*, 40(3):104, 2012.

NDN Technical Reports

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

- "The Development and Experimentation with NDN.JS, a JavaScript Client Library for Named Data Networking" by Wentao Shang, Jeff Thompson, Jeff Burke, and Lixia Zhang. NDN Technical Report NDN-0014, August 2013.
- "A New Perspective on Mobility Support" by Zhenkai Zhu, Alexander Afanasyev, and Lixia Zhang. NDN Technical Report NDN-0013, July 2013.
- "FileSync/NDN: Peer-to-Peer File Sync over Named Data Networking" by J. Lindblom, Ming-Chun Huang, J. Burke, Lixia Zhang. NDN Technical Report NDN-0012, March 2013.
- "Authenticated Lighting Control Using Named Data Networking" by J. Burke, A. Horn, and A. Marianantoni. NDN Technical Report NDN-0011, October 2012.
- "Egal Car: A Peer-to-Peer Car Racing Game Synchronized Over Named Data Networking" by Z. Qu and J. Burke. NDN Technical Report NDN-0010, October 2012.
- "Deploying Key Management on NDN Testbed" by Chaoyi Bian, Zhenkai Zhu, Alexander Afanasyev, Ersin Uzun, and Lixia Zhang. NDN Technical Report NDN-0009, Revision 2, February 2013.
- "Chronos: Serverless Multi-User Chat Over NDN" by Z. Zhu, C. Bian, A. Afanasyev, V. Jacobson, and L. Zhang. NDN Technical Report NDN-0008, October 2012.
- "NDN Video: Live and Prerecorded Streaming over NDN" by Derek Kulinski and Jeff Burke. NDN Technical Report NDN-0007, September 2012.
- "NDNLP: A Link Protocol for NDN" by Junxiao Shi and Beichuan Zhang. NDN Technical Report NDN-0006, July 2012.
- "ndnSIM: NDN simulator for NS-3" by Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. NDN Technical Report NDN-0005, Revision 2, October 2012.
- "OSPFN: An OSPF Based Routing Protocol for Named Data Networking" by Lan Wang, A K M Mahmudul Hoque, Cheng Yi, Adam Alyyan, and Beichuan Zhang. NDN Technical Report NDN-0003, July 2012.
- "A Case for Stateful Forwarding Plane" by Cheng Yi, Alexander Afanasyev, Ilya Moiseenko, Lan Wang, Beichuan Zhang, and Lixia Zhang. NDN Technical Report NDN-0002, July 2012.
- "Named Data Networking" by the NDN project team. NDN Technical Report NDN-0001, October 2010.