Named Data Networking (NDN) Project 2013 - 2014 Report

Principal Investigators

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

Beichuan Zhang
University of Arizona

Kim Claffy and Dmitri Krioukov
University of California, San Diego

Christos Papadopoulos
Colorado State University

Tarek Abdelzaher
University of Illinois at Urbana-Champaign

Lan Wang
University of Memphis

Edmund Yeh
Northeastern University

Patrick Crowley
Washington University

Contents

FIA: Collaborative Research: Named Data Networking (NDN) 2014 Report

Executive Summary

This report summarizes the NDN project team's fourth year of research achievements supported by FIA supplemental funding. Chapter 1 reviews the NDN architectural model. Chapter 2 gives an overview of the NDN project objectives and the milestones achieved in our fourth year effort. Chapter 3 describes activities and findings in our five main research areas. Chapter 4 details one of our major achievement in 2014, the design of a new NDN packet format and the development of a new NDN Forwarding Daemon (NFD). areas. Chapters 5 and 6 review efforts in education and community outreach, respectively.

The heart of the Internet architecture is a simple, universal network layer (IP) which implements all the functionality necessary for global interconnectivity. This *thin waist* was the key enabler of the Internet's explosive growth but one of its design choices is also the root cause of today's many persistently unsolved problems: IP was designed to create a *communication network*, therefore the only entities that can be named in its packets are 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.

NDN retains the Internet's hourglass architecture but evolves the thin waist to enable the creation of completely general distribution networks. The core element of this evolution is removing the restriction that packets can only name communication endpoints. As far as the network is concerned, the name in an NDN packet can 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 communication problems but also digital distribution and control problems. Our research challenge is to turn the above vision into an architectural framework capable of solving real problems, particularly in application areas poorly served by today's Internet protocol stack. Solving real problems forces us to fill in architectural details, and most importantly, verifies and shapes the architectural direction.

We achieved our major milestones for the fourth year of the project. Highlights include:

- The design of a new NDN packet format, which uses a type-length-value (TLV) pattern to support efficient processing and flexibility for future extension.
- The design, implementation, public release, and testbed deployment of a new NDN Forwarding Daemon (NFD). The basic NFD design goals are modularity and extensibility to enable experimentation with new protocol features, algorithms, and applications.
- The re-design and implementation of NLSR, NDN Link-State Routing protocol, to run with the new packet format and NFD and use ChronoSync for routing database synchronization.
- The development of an NDN security library which supports efficient signing and validation of NDN packets as well as key management to facilitate secure application development.
- Continued advancement in NDN application developments, including advances in building automation systems, NDN-IoT toolkit for smart home R&D, the development of ndnRTC, a real-time conferencing application over NDN, and application of NDN to climate data distribution.
- The establishment of the NDN Consortium to promote a vibrant open source ecosystem of research and experimentation around Named Data Networking, and the organization of the first NDN Community meeting to orchestrate the effort from broader community in advancing NDN research and development.

These achievements establishes a platform that facilitates new research experimentations. Our 4-year effort has produced a range of new applications, a rich set of libraries, a functioning testbed spanning three continents, a deepened understanding of the NDN architecture as well as its remaining challenges, and most importantly, a team with skills and experience in future architecture research. These achievements will guide the project's next phase, NDN-NP, and lead us to a new level of understanding in naming design, distributed data synchronization, usable trust management, routing and forwarding—core issues in the Named-Data Networking architecture.

Chapter 1

Architecture Overview

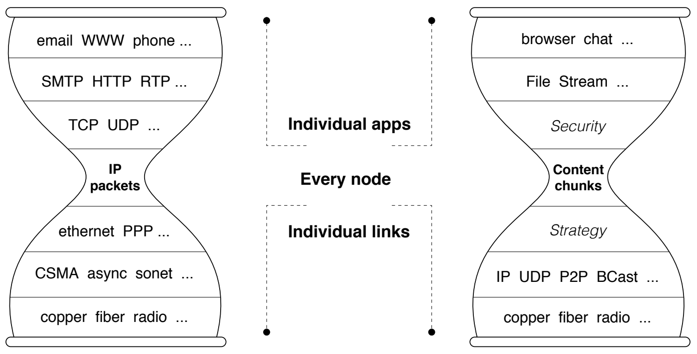


Figure 1.1: Internet and NDN Hourglass Architectures

NDN is an entirely new architecture, but its 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. The NDN architecture retains the same hourglass shape, but transforms the thin waist to focus on data directly instead of 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 functionality in the TCP/IP Internet is layered with inherent complexity atop a fundamentally too-trusting architecture. NDN provides a fundamental security building block *at the thin waist* by signing all named data.
- The *end-to-end principle* [6] 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.
- *Network traffic must self-regulate.* 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." [2] 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 to the network. Routers use this name to forward the

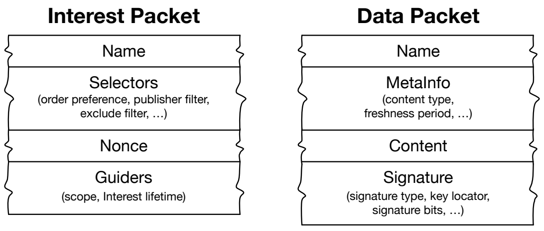

Interest Packet
Name
Selectors
(order preference, publisher filter, exclude filter, ...)
Nonce
Guiders
(scope, Interest lifetime)
Data Packet
Name
MetaInfo
(content type, freshness period, ...)
Content
Signature
(signature type, key locator, signature bits, ...)

Figure 1.2: Packets In the NDN Architecture.

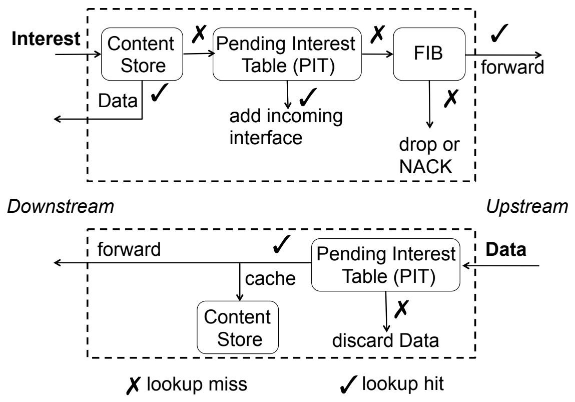


Figure 1.3: Forwarding Process at an NDN Node.

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 execute Interest and Data packet forwarding functions, each NDN router maintains three 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 guides Interests toward data producers. The PIT stores all Interests that are not yet satisfied, recording the Interest's 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 toward 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 to, 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 in its PIT, and if a matching entry exists, it records the Interest's incoming interface in the PIT entry. In the absence of a matching PIT entry, the router will look in its FIB and forward the Interest toward 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 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 decoupling allows each application to choose the naming scheme that fits its needs and allows the naming schemes to evolve independently from the network. NDN by design assumes hierarchically *structured* names, e.g., a video produced by UCLA may be named `/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, e.g., `UCLA` could correspond to the autonomous system originating the video. 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 but opaque to the network.

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. This construction can be enabled either by a deterministic algorithm shared by the producer and consumer to arrive at the same name based on information available to both, or by the combined effort of NDN's *Interest selectors* and the stack's *longest prefix matching* to retrieve the desired data through one or more iterations. Our experience so far suggests that a simple set of selectors is quite powerful for 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", in order to receive a data packet named `/ucla/videos/demo.mpg/1/1` corresponding to the first segment. The consumer can then later request segments using a combination of information revealed by the first data packet and the naming convention of the publishing application to request subsequent packets.

Data that may be retrieved globally must have a *globally unique* name, however names used for local communications may be heavily based on local context, and require only local routing (or local broadcast) to find corresponding data. In fact, individual data names can be meaningful in various specific scopes and contexts, ranging from "the light switch in this room" to "all country names in the world". How to develop efficient strategies to fetch data within the intended scope is a new research area.

Name space management is not part of the NDN architecture, just as address space management is not part of the IP architecture. However, 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 with its name [5]. 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.

Historically, security based on public key cryptography has been considered inefficient, unusable and difficult to deploy. Besides efficient digital signatures, NDN needs flexible and usable mechanisms to manage user trust. Since keys can be communicated as NDN data, key distribution is simplified. 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. Finally, 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 decryption) 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 routing protocol security, e.g., protecting against spoofing and tampering. NDN's inherent multipath routing, together with the adaptive forwarding plane (Section 1.3), mitigates prefix hijacking because routers can detect the anomaly caused by a hijack and retrieve data through alternate paths.

Two possible attacks against an NDN network are cache poisoning and denial of service via Interest flooding. We have addressed the latter in our recent work [1, 3], and we are actively pursuing effective solutions to the former.

1.3 Adaptive Routing and Forwarding

NDN routes and forwards packets based on names, which eliminates four problems caused by addresses in today's IP architecture: address space exhaustion, NAT traversal, mobility, and address management. There is no address exhaustion problem since the namespace is unbounded, which also eliminates the need for name translation. 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, which is especially empowering for embedded sensor networks.

NDN can make use of 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 sequences 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 (Section 3.2.1), and developed efficient data structures and algorithms for fast lookup of variable-length, hierarchical names (Section 3.3).

PIT state at each router supports forwarding across NDN's data plane. Routers record each pending Interest and its incoming interface(s), and remove Interests after matching Data is received or a timeout occurs. This per-hop, per-packet state is a fundamental change from IP's stateless data plane. This state information enables NDN nodes to monitor packet delivery performance across different interfaces, and adapt to network failures, all at the time scale of a round-trip time. 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. This allows NDN nodes to use multiple paths toward the same data producer.

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 used to mitigate DDoS attacks. Because the number of PIT entries is an explicit indicator on router load, limiting its size also constraints the effect of a DDoS attack. PIT entry timeouts offer relatively cheap attack detection [7]; and arrival interface information in each PIT entry can support a push-back scheme [4].

Each NDN node also implements a *forwarding strategy* module, which does not exist in today's IP nodes. The role of this forwarding strategy module is to inform decisions about which Interests to forward to which interfaces, how many unsatisfied Interests to allow, relative priority of different Interests, how to load balance Interest forwarding among 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, NDN supports a more persistent and larger-volume in-network storage, called a Repository (Repo in short). While a Content Store provides opportunistic caching to optimize performance, a repo provides managed storage to meet specific application needs.

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. The naming and caching of data in NDN networks may facilitate observation of what data is being requested, but it is harder to identify who is requesting data (unless one is directly connected to the requesting host). This aspect of the architecture offers privacy protection at a fundamentally different level than the IP architecture.

1.5 From Transport to Distributed Synchronization

The NDN architecture does not require 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 ad hoc 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 (Section 1.3), 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 [8].

References

[1] Alexander Afanasyev, Priya Mahadevan, Ilya Moiseenko, Ersin Uzun, and Lixia Zhang. Interest flooding attack and countermeasures in Named Data Networking. In *Proceedings of IFIP Networking*, May 2013.

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

[3] Paolo Gasti, Gene Tsudik, Ersin Uzun, and Lixia Zhang. DoS & DDoS in named-data networking. In *Proceedings of International Conference on Computer Communications and Networks*, July 2013.

[4] John Ioannidis and Steven M. Bellovin. Router-based defense against ddos attacks. In *Proceedings of Internet Society Symposium on Network and Distributed System Security*, 2002.

[5] Van Jacobson, Diana Smetters, James Thornton, Michael Plass, Nicholas Briggs, and Rebecca Braynard. Networking named content. In *Proceedings of the ACM CoNext*, 2009.

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

[7] Vyas Sekar, Nick Duffield, and Oliver Spatscheck. Lads: Large-scale automated ddos detection system. In *Proceedings of USENIX 2006.*

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

Chapter 2

Research Plan and Progress

In this chapter we present the major outcomes we are working to deliver and the plan we are executing to move the NDN architecture from ideas to reality and validation. We summarize the execution results during the third year of the project and the milestones for the remaining duration of the report.

2.1 Intended Outcome of the NDN Project

As shown in Figure 1.1, the NDN architecture retains the same hourglass shape as the IP architecture, with the narrow waist being its centerpiece. However the minimal functionality of NDN's narrow waist, as we described in Chapter 1, is fundamentally different from IP's. NDN's minimal functionality includes support for consumer-driven data delivery, built-in data security, stateful data plane, and in-network storage. These two building blocks can together support scalable data dissemination, flow balancing, multiple path data retrieval, as well as facilitating mobile and delay-tolerant communication.

We aim at the following major outcome:

1. A specification of the standard formats for the two NDN packet types, Interest and Data. We expect this specification to play a role equivalent to that of RFC791 (Internet Protocol Specification) for NDN networks. The challenge is not the format details, but verification and validation of the minimal functions that must be supported by this new narrow waist. Over the past year we have produced a preliminary NDN packet format specification [3], which is now used in NDN pilot applications running on the NDN testbed.
2. A functional version of the necessary supporting modules in an operational NDN network, including: libraries for naming conventions, reliable delivery, and security utility that reside above the NDN layer; routing protocols and forwarding strategy module that reside at the narrow waist NDN layer; trust management; and usable, efficient cryptography for data security. We see an analogy between the above list and IP with its 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 in an operational IP network. The fact that DNS was added *after* the initial IP deployment further underscores the importance of identifying missing components during real deployment. Our most significant milestone this year was completion of a new modular and extensible NDN Forwarding Daemon (NFD), which will provide a platform for the broader community to investigate NDN router data structure and forwarding strategy designs.
3. A set of applications that operate over NDN, including entirely new applications as well as NDN versions of legacy applications deployed on 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.

2.2 Path and Progress

The NDN protocol specification is our fundamental goal, and we seek to achieve it through experimenting with real applications in an operational environment. Development of pilot applications (Section 3.1) deepens our understanding of how applications can choose names to facilitate both application development and network delivery. An application-driven approach also organically reveals general principles and guidelines for naming in NDN networks, and suggests how to translate these principles and guidelines into naming conventions in system libraries (Section 3.1.4). Our efforts in library implementation requires us to operationalize what we have learned in a form that supports consistent reuse, simplifying future application development and accelerating progress.

The opaqueness of names to the network means that design and development of the NDN architecture can proceed in parallel with our research into name structure, name discovery and namespace navigation, all studied in the context of real application development. Similar to previous years, in parallel with our investigation into application development, we progressed on NDN routing protocols, data forwarding strategies, and testbed operations. Deployment of these components has provided a live NDN environment to ask and answer design questions, and verify and demonstrate NDN capability in reality. A major achievement in 2014 was the design and implementation of a new NDN Forwarding Daemon (NFD). The NDN testbed previously relied on the CCNx implementation developed by PARC, which lacks modularity and makes experimentation difficult. NFD is highly modular and extensible, has been deployed on the NDN testbed and released to public.

To support larger-scale experiments than our current testbed can support, we developed an ns-3 based NDN simulator, ndnSIM [1], to answer questions regarding large scale NDN network properties. Since ndnSIM's public released during summer 2012, its functionality has been substantially expanded through active usage by ourselves and by a global ndnSIM user community.

2.3 Progress: Year 4

- Applications (Section 3.1): We made progress on important fronts through course-based explorations and a variety of internships, in addition to ongoing research of graduate students and staff:
  - To complement enterprise building automation and management work, developed a series of device-focused smart home application and library designs, with experimental prototypes for device control and sensor data acquisition created for the Raspberry PI.
  - Developed and prototyped an NDN Internet of Things tool kit for the Raspberry PI.
  - Designed, developed, and performed experimental deployment of a practical, end-to-end real-time audio and video conferencing application (ndnrtc) using the NDN-CCL libraries and industry-standard WebRTC media components.
  - Continued to explore different approaches to peer-to-peer multiplayer online gaming, resulting in a working demonstration for the ACM ICN conference.
  - Transitioned building management system prototype and other NDN applications to use new NDN-TLV packet format and work with new NFD, library, and repository implementations.
  - Expanded the NDN Common Client Libraries (NDN-CCL), providing support for NDN application development across C++, Python, Javascript, and Java, and some C#.
  - Transitioned research results into the NDN-CCL, including support for the new NDN-TLV packet format, native Python, a prototype SYNC implementation, security library functionality, and application-side content cache.
  - Organized regular NDN Platform releases to aid researchers working with NDN.
  - Continued work on web browser support through expansion of the NDN-JS Javascript library, now used by multiple practical applications.
- Routing (Section 3.2):
  - Dynamic intra-domain routing protocol: We re-implemented the *Named-data Link State Routing*

Protocol (NLSR) [2] to work with NFD. NLSR supports both link state and hyperbolic routing (Section 3.2.2), synchronization using ChronoSync, and a hierarchical trust model for routing within a single administrative domain. We have deployed it on the NDN testbed.

  - Hyperbolic routing:
    - We embedded the updated NDN testbed topology into the hyperbolic plane using newer Internet topology data, and our simpler and more efficient hyperbolic network mapping algorithm.
    - We implemented the basic hyperbolic routing in NLSR by disseminating hyperbolic coordinates in link state announcements. We have been conducting Emulab experiments to evaluate the feasibility of hyperbolic routing in NDN by comparing it with link-state routing under various conditions.
    - We developed and released a stand-alone software package for hyperbolic network generation and greedy routing simulation.
  - Dynamic Interest Limiting for NDN Congestion Control: We designed a new scheme to detect link-layer packet loss and dynamically adjust Interest limit to effectively control network congestion. This is the first hop-by-hop NDN congestion control scheme that does not assume the knowledge of underlying link bandwidth and can work better on overlay links. Our evaluation using ndnSIM has shown significant performance improvement over existing approaches.
- Scalable Forwarding (Section 3.3):
  - Refined and evaluated lookup performance of FIB design with 1 billion synthetic names. Investigated incremental updates on proposed data structures and evaluated insertion latency.
  - Refined the proposed fingerprint-only PIT design to support the case with multiple core routers and investigated in supporting all prefix matching with this PIT design.
  - Started to develop an NDN forwarding engine on a general-purpose multi-core platform.
- Security and Privacy (Section 3.4):
  - We designed and implemented a **security library** to facilitate experimentation with trust models and help application developers enable security support. Within the security library, we achieved several goals, including:
    - Secured private key storage and management on local machine;
    - PIB service: which allows different libraries/applications to share and manage public key information on local machine;
    - Basic automated packet signing through "identity, key, certificate" abstraction;
    - A new extensible NDN public key certificate format.
    - Generalize packet validation process and design a validation framework.
    - Design a policy language to express a variety of trust models.
  - We designed, implemented, and deployed a **testbed certificate system** to facilitate the process of deploying public key certificates on the NDN testbed. The system consists of two parts: a set of tools for users to generate and manage keys/certificates, and for testbed operators to issue certificates; and a web-based certificate requesting system, which provides a user-friendly interface, email-based user authentication, efficient notification to operators/users who process certificate requests.
  - We experimented with a **Web-of-Trust model** in ChronoChat, a server-less multi-party chat application. For this application, we adapted the Web-of-Trust model into an endorsement-based authentication system and also designed a key distribution system to support it.
- Theory (Section 3.5):
  - We published the description of VIP framework and algorithms for joint forwarding and caching in NDN networks and demonstrated their superior performance relative to other algorithms as one of the first algorithmic contributions to the NDN literature with a solid theoretical foundation.

  - We developed joint congestion control, forwarding, and caching algorithms for NDN based on the VIP framework, pioneered the study of congestion control in information-centric networks where utilities (fairness) are now associated with content objects, rather than source-destination pairs. We also developed fair congestion control schemes to obtain a tradeoff between the utility gained by admitting more demand into the network layer, and the incurred average network delay.
  - We studied throughput and delay behavior in large-scale information-centric wireless networks, obtained the optimal tradeoff between throughput and delay in information-centric wireless networks using shortest path interest routing and optimized cache placement. We also extended these results to hybrid wireless scenarios where wireless nodes can communicate not only among themselves but also to a set of base stations.
- NFD and testbed deployment (Section 4):
  - We produced a new NDN packet format specification, which is now used by all NDN applications running on the NDN testbed.
  - We designed, implemented, and deployed a new NDN Forwarding Daemon (NFD) on the NDN testbed.
  - We expanded the NDN testbed to three continents and multiple collaboration sites.
- Education (Chapter 5):
  - Continued biweekly NDN Seminars among all participating sites to share the latest research results and discuss results by other researchers.
  - Created screencasts to demonstrate features of NDN such as distributed publishing, enumeration, discovery, retrieval and failover.
  - Continued to incorporated NDN architecture into undergraduate and graduate teaching at NDN project participation institutions; a number of term projects from the graduate seminar courses directly contributed new results to NDN research.
- Global expansion of NDN effort (Chapter 6):
  - We organized the first NDN Community meeting.
  - We established the NDN Consortium.
  - We contributed to the organization of, and participated in the first ACM Information Centric Networking Conference.

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] 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.

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

Chapter 3

Architecture Research

The NDN design introduced in the Architecture Overview (Chapter 1) represents a novel architectural blueprint with both unique opportunities and many challenges. This chapter describes activities and findings in each research area during the fourth year.

3.1 Applications

Contributors

PIs	Jeffrey Burke, Van Jacobson & Lixia Zhang (UCLA), Tarek Abdelzaher (UIUC)
Grad Students	Shuo Chen, Ilya Moiseenko, Mengchen Pei, Wentao Shang, Yingdi Yu (UCLA); Jongdeog Lee, Shiguang Wang (UIUC)
Undergrads	Akash Kapoor, Yang Sheng (UIUC)
Staff	Peter Gusev, Alex Horn, Jeff Thompson, Zhehao Wang, (UCLA); Hongyan Wang (UIUC)
	Postdoc: Alex Afanasyev

3.1.1 Summary of Objectives

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 functional and performance 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.

3.1.2 Technical Approach

During the fourth year of NDN research, the applications group continued to explore the instrumented environment, media distribution, and serverless peer-to-peer applications. In the process we confirmed the value of developing pilot applications to drive NDN architecture design and development.

3.1.3 Progress - Applications

This section summarizes the progress of each application area, emerging design patterns, and impact on our architecture research. The application areas include instrumented environments, peer-to-peer, serverless applications, NDN in the browser, and NDN support for climate data application.

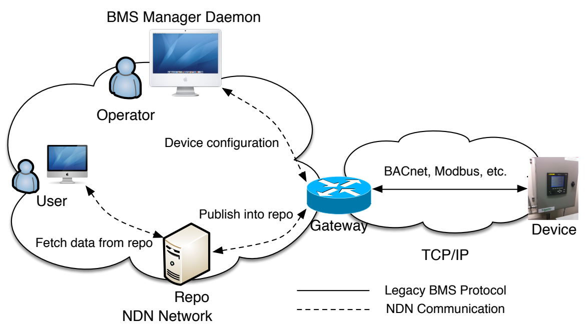


Figure 3.2: Building management system prototype deployment conceptual diagram.

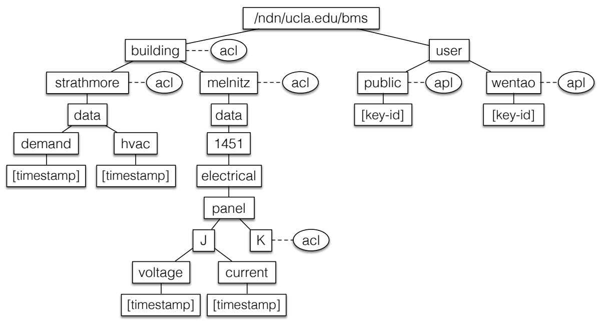


Figure 3.3: Building management system namespace.

Instrumented environments: Building Automation Systems & Smart Home/Internet of Things

Information-centric networking is generally discussed in terms of content retrieval, as opposed to control, actuation, or remote execution. We believe NDN offers design solutions for not only these functions but broader BAS industry goals, e.g., to enhance device interoperability, enable data-centric application designs, and provide simplified access to networked buildings over commodity networks while providing security against a wide variety of IT and industrial control attacks. To explore NDN as a network substrate for Building Automation Systems (BAS), we connect BAS devices to the NDN testbed and incorporate them into applications that leverage unique features enabled by the architecture. We reached a milestone with the publication of a IEEE Network Magazine article, "Securing Building Management Systems using Named Data Networking" [8], which summarizes a design and prototype for building management systems over NDN. Another milestone was our initial repository design for building management data using a graph database and the imminent field deployment (October 2014) of an *ambient informatics* project with the Architecture Department at UCLA, which will use NDN for internal communication of transportation data and lighting control [4]. We also transitioned the existing building management code to use the new forwarder (NFD), repo, and libraries.

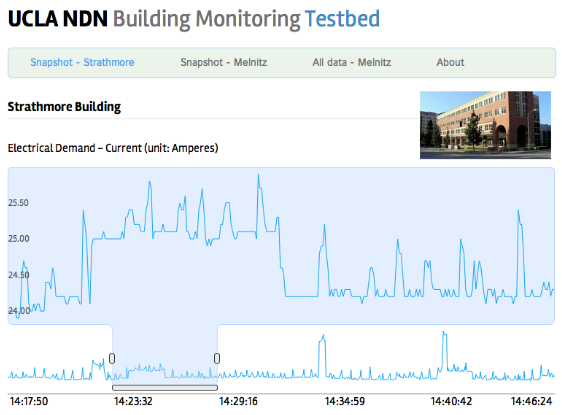
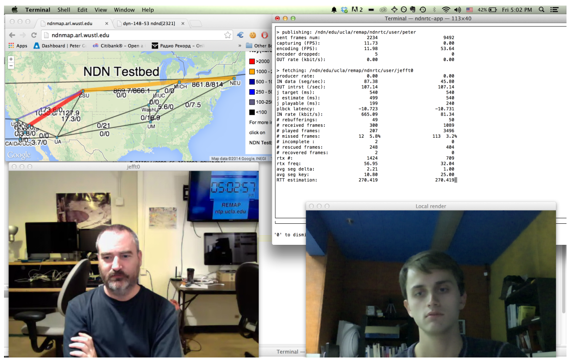


Figure 3.1: Interface for deployed building management system, now ported to NDN-TLV packet format

This year, we also started work to support device-level IoT applications for NDN, for use in smart home R&D. Significant results include: 1) a basic NDN-IoT tool kit and demonstration applications for NDN on the Raspberry PI, with support for basic sensor and actuator I/O and HDMI-CEC consumer device control (e.g., for televisions and home electronics); and 2) an initial design for discovery and bootstrapping (of trust and services) for devices on a local network [1]. The primary goal of the NDN-IoT toolkit for the Raspberry PI is to provide a platform for users to create a home network of NDN-capable Raspberry Pi computers. The platform is easy to install, run, and customize, and uses NDN's built-in security features to help users protect their network from outside access.

The NDN-IoT toolkit is primarily distributed as an SD card[1] image based on the Raspbian Linux distri-

[1]Secure Digital Card is an ultra small flash memory card designed to provide high-capacity memory in a small size, and



```
> publishing: /ndn/edu/ucla/remap/ndnrtc/user/peter
sent frames num:                    2234                   9492
capturing (FPS):                   11.73                   0.00
encoding (FPS):                    11.98                  53.64
encoder dropped:                       5                      0
OUT rate (kbit/s):                  0.00                   0.00

> fetching: /ndn/edu/ucla/remap/ndnrtc/user/jefft0
producer rate:                      0.00                   0.00
IN data (seg/sec):                 87.38                  45.80
OUT intrst (/sec):                107.14                 107.14
j target (ms):                       540                    540
j estimate (ms):                     499                    540
j playable (ms):                     199                    240
plbck latency:                   -10.723                -10.731
IN rate (kbit/s):                 665.09                  81.34
# rebufferings:                       49                     50
# received frames:                   300                   1089
# played frames:                     207                   3496
# missed frames:                 12  5.8%              113  3.2%
# incomplete :                         2                      0
# rescued frames:                    248                    404
# recovered frames:                    2                      0
rtx #:                              1424                    709
rtx freq:                          56.95                  32.04
avg seg delta:                      2.21                   1.00
avg seg key:                       10.80                  25.00
RTT estimation:                  270.419                270.419

'0' to dismi
```

Figure 3.4: ndnrtc conference screenshot.

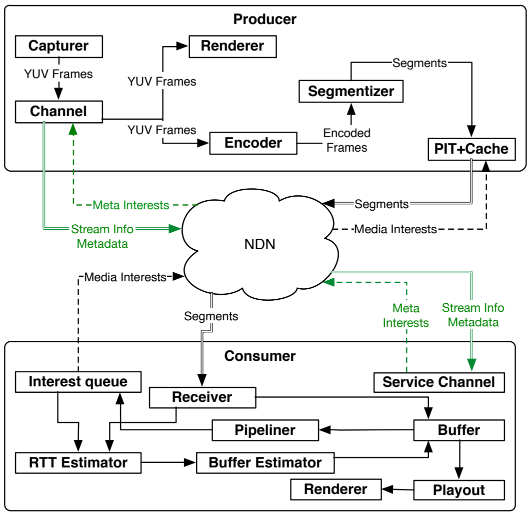


Figure 3.5: ndnrtc Application Architecture.

bution, which users can flash onto their Raspberry Pi SD cards. The image contains scripts for configuring and running the NDN forwarder, as well as instructions and tutorials for creating nodes and adding them to a home network. Currently, users can create or modify network behavior using the ndn_pi module, developed with the Python binding of the NDN Common Client Libraries (`http://named-data.net/doc/ndn-ccl-api/`). There are also examples of simple networks that are ready to run. All source code for the ndn_pi module is available online at `http://github.com/remap/ndn-pi`. The SD card image includes several other libraries for more advanced NDN application development, including C++ bindings of the Common Client Libraries.

Peer-to-Peer, Serverless Applications

Real-time video conferencing We made significant progress in the development and evaluation of real-time video conferencing over NDN through the *ndnrtc* application, which enables many-to-many conferencing over NDN [3]. Figure 3.4 shows a sample screenshot; Figure 3.5 shows a diagram of the application architecture. This application requires low-latency packet delivery and consumer-side retransmission of Interests, and has proved a useful driver for evaluating the NFD forwarder with real world traffic. It has also motivated design dialogue around the relationship between application configuration and forwarder strategy. Further, it has driven the development of the NDN-CPP C++ library and motivated support for features such as application-side cache management. We expect *ndnrtc* to gain real-world use among the project team in the Fall of 2014, and motivate traffic congestion control research. It also proved a fertile ground for external collaboration: during summer 2014 we hosted an intern from Keio university who collaborated on incorporating Forward Error Correction (FEC), and we are currently working with researchers from Panasonic Corporation on adaptive rate control algorithms for real-time video of NDN, with a paper forthcoming [16]. The **ndnrtc** NDN Real Time Communication Library & Application is available at `https://github.com/remap/ndnrtc`.

Multi-player online game We continued research on peer-to-peer virtual environments, investigating a variety of namespace designs, and implementing a functional peer-to-peer multiplayer game in the Unity 3D graphics engine. The current game design uses SYNC for namespace synchronization, and static octree

used extensively in portable devices.

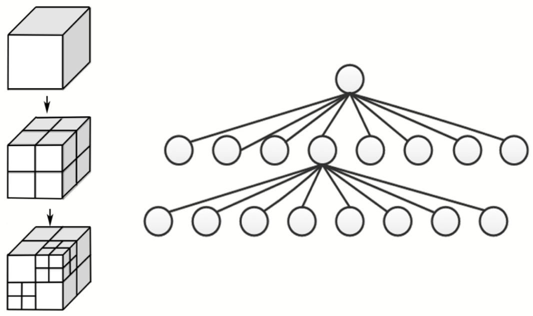


Figure 3.6: Octree hierarchical partitioning of the game world.

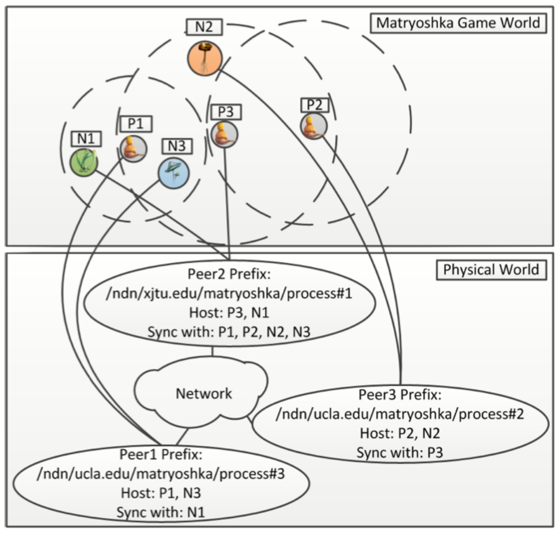


Figure 3.7: Multi-player on-line game discovery and communication problem.

partitioning of the game world (Figure 3.6) to solve peer-to-peer discovery and updating of game objects (Figure 3.7) via Interest/Data exchange [15].

Matryoshka is a multiplayer online game, which adopts a serverless, pure peer-to-peer architecture. Synchronization in a distributed gaming environment presents a key challenge: each player only needs to know objects near itself in the game world, yet players whose areas of interest intersect must reach consistent conclusions about objects in the intersected area. The current design statically and recursively partitions the virtual environment into octants, thus providing a shared namespace for every peer running the game. Then, we apply a two-step synchronization to deal with the two questions which each peer asks the network: "Which players are in my vicinity?" in a broadcast discovery namespace, and "What are those players doing?" in a multicast update namespace. The game application is implemented using Unity3D game engine, and ndn-dot-net, a C# adaptation of the NDN common client library. A presentation of this game has been accepted by ACM ICN 2014 in the demonstration session, where we will showcase its player discovery and position update utilities [14].

ChronoChat This year we continued work on ChronoChat, a serverless multi-party chat application. Since the chatroom has no centralized management, peer authentication must be distributed, which motivated our investigation of Web-of-Trust models for NDN applications. For ChronoChat specifically, we designed an endorsement-based authentication mechanism to allow peers in a chatroom to manage chatroom membership and authenticate each other's identity, thus only authorized peers can speak in a chatroom and have their chat messages authenticated [17]. In addition, we also developed a chatroom discovery mechanism, so that interested users can learn all ongoing chatrooms automatically [18]. This research continued our progress on discovery and bootstrapping challenges across different applications.

Web applications / NDN in the browser

Support for web applications has continued primarily through expansion of the Javascript library, NDN-JS. In part because of its adoption by developers outside of the application group, it has continued to evolve to support experimentation on web browser support for NDN and rapid prototyping of user interfaces. The library team has incorporated continued enhancements provided by the community for such features as Node.js standalone Javascript application support. The software development team integrated websockets proxy support, enabling direct NDN communication from browsers, into the new NFD forwarder.

The applications team also finished a port of ChronoChat, originally coded in C++, to Javascript, and created several small tools such as the namespace browser NDNExplore. Colorado State developed the NDN-Browserkit prototype, an NDN toolkit for client side browser apps, packaged as a Node.js npm (Node Package Manager) module.

Finally, the applications team designed the *ndnrtc* real-time conferencing library for integration into browsers, and began building a web-based conferencing tool, which should complete in 2015 as part of the NDN-NP project. Integration of the NDN code with a complex multi-threaded platform such as the Mozilla browser motivated some of our approaches to threading and shared pointers in the NDN-CPP library.

NDN for Climate Data

In a companion project titled "Supporting Climate Applications over Named Data Networking (NDN)" (NSF CC-NIE Integration Award#13410999) PI Papadopoulos at Colorado State University has been investigating NDN as a technology to support applications in the climate domain. In this project, a team of networking researchers in the Computer Science and Atmospheric Sciences departments at Colorado State University are working jointly to integrate NDN with the Global Cloud Resolving Model (GCRM). GCRM simulates the global atmosphere using a grid which is fine enough to crudely resolve the larger individual clouds. This driver application typically generates terabyte to petabyte scale datasets. GCRMs are currently under development at several modeling centers in the U.S. and abroad.

The CSU team has built modules to translate GCRM datasets into an namespace appropriate to run over an NDN network. The translators take into account filenames, directory structures, metadata inside data files, and information provided by scientists to construct namespaces suitable for capturing desirable information in the names while imposing a NDN-compliant hierarchical name structure. The tasks for the translator range in difficulty, from relatively straightforward, such as CMIP5 data that already obeys a strict hierarchical structure, to hard, such as climate model output data with many user-defined parameters.

We have deployed a small testbed made of five 10GB-connected NDN nodes: two at CSU (CS and Atmospheric Sciences), one each at Denver, Sacramento, and LBNL. A sixth node will be deployed at the supercomputing center in Wyoming. Existing nodes are seeded with about 50TB of CMIP5 data, with names that have been translated to NDN names. We have also developed a simple browsing application to publish, discover, enumerate, and retrieve datasets. Finally, we presented our work in several venues, ranging from networking (Internet2 and ESnet) workshops to climate research groups.

3.1.4 Progress - Libraries

NDN Software Platform

This year, we launched the NDN Platform, a starting point for experimention with NDN. It collects individual software components into a coherent, usable, and well-documented platform for exploring NDN in practical applications. The platform has the following objectives:

1. Gather stable versions of core components needed for NDN experimentation.
2. Use a release heartbeat to stimulate interoperability testing and discussion.
3. Improve access to and consistency of NDN software projects.
4. Employ an open and lightweight process, minimizing central management but identifying stewards of each component
5. Ensure that the managed nodes on the testbed run the Platform.

NDN Platform 0.3 The platform had the following components in its third release in August 2014:

- **NFD** - NDN Forwarding Daemon, version 0.2.0
- **ndn-cxx** library, version 0.2.0: The NDN C++ library with eXperimental eXtensions (CXX)

    - **ndnsec** security tools to manage security identities and certificates
- **NDN-CCL** - NDN Common Client libraries suite, version 0.3
    - **NDN-CPP** C++ / C library
    - **PyNDN2** Python library
    - **NDN-JS** JavaScript library (with Node.js support)
    - **jNDN** Java library (preliminary)
- **NLSR** - Named Data Link State Routing Protocol, version 0.1.0
- **repo-ng** - next generation of NDN repository, version 0.1.0
- **ndn-tlv-ping** - ping application for NDN, version 0.2.0
- **ndn-traffic-generator** - traffic generator for NDN, version 0.2.0
- **ndndump** - packet capture and analysis tool for NDN, version 0.5
- Preliminary binary package support on Ubuntu, MacOS X, others.

The NDN Platform Codebase is located at `http://named-data.net/codebase/platform/`.

NDN-CXX: NDN C++ library with eXperimental eXtensions

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

To optimize developer productivity while experimenting with their applications, ndn-cxx encourages and extensively uses the well-known Boost libraries [2], leveraging more than 7000 person years of high-quality code and modern design patterns. The library includes many utility classes and helpers to simplify common operations in NDN applications, which have been discovered during the last few years of NDN application development. Finally, ndn-cxx tries to stay as simple as possible, be easy to understand by new developers, and easy to maintain and extend. To meet these objectives, ndn-cxx is implemented purely in C++, adhering to object-oriented programming principles, with extensive documentation. A ndn-cxx developer guide is under development which describes the basic concepts, and shows examples and common patterns in NDN applications.

Usage in applications ndn-cxx was released in August 2014, together with the new NDN Forwarding Daemon (NFD). It is currently used by the following projects:

- **NFD** - NDN Forwarding Daemon
- **NLSR** - Named-data Link-State Routing protocol
- **repo-ng** - a new implementation of NDN repository
- **ChronoChat** - Multi-user NDN chat application
- **ChronoSync** - Sync library for multiuser realtime applications for NDN
- **ndn-tlv-ping** - Ping Application For NDN
- **ndn-traffic-generator** - Traffic Generator For NDN.

NDN security library is part of ndn-cxx, which is described in Section 3.4.

NDN-CCL: Common Client Libraries

The NDN Platform release also includes the NDN Common Client Libraries (CCL) [12] that we have been developing over the last few years. NDN-CCL provide a common application programming interface (API) across several languages. Currently, the CCL is implemented in C++, Python, JavaScript and Java. Significant effort continued on the development of these libraries to support application research and experimentation. We extended them to support both the existing forwarder (NDNx, a port of CCNx) and the new forwarder (NFD), to allow comparison and simultaneous experimentation. (This support enabled a critical transitional period; we will slowly phase out use of the NDNx codebase and deprecate library support.)

We updated the NDN-CCL libraries to support the following expressed needs of the research and community: 1) preliminary trust management based on the architecture group's reference implementation in ndn-cxx; 2) ECDSA signatures; 3) a language-independent, industry-standard approach to message description (Google protocol buffers), which generates TLV or binary XML format messages in Data and Interests across all CCL languages; 4) configuration interaction with the NFD forwarder; 5) a preliminary generalized API for the SYNC protocol, across multiple languages; 6) a pure Python library implementation that runs on a variety of platforms; 7) an application-side in-memory repository; 8) the major components of the Java library implementation, in preparation for work on Android mobile platforms; and 9) initial incorporation of unit testing and continuous integration. These library developments required ongoing dialogue between the architecture and applications groups, which continued to benefit the design evolution of the NDN protocol itself, including naming conventions and protocol structure decisions.

Usage in applications. The NDN-CCL is used by the following applications and projects:

- **CCNx Federated Wiki**, an NDN port of the Smallest Federated wiki (NDN-JS)
- **Chronochat-js**, a javascript implementation of the ChronoChat application (NDN-JS)
- **Matryoshka**, an experimental multi-player online game using NDN and the Unity3D game engine. (jndn as the basis of the .NET port of CCL used in this project.)
- **ndn-bms**, building management system prototype (part of NDN-NP project) (PyNDN, NDN-JS)
- **ndn-lighting**, lighting control application using NDN (PyNDN, NDN-JS)
- **ndn-protocol**, a firefox browser plug-in supporting an ndn:/ retrieval scheme (NDN-JS)
- **NDNEx**, an NDN-based mobile health application (part of NDN-NP research project). (jndn)
- **ndnfs** and **ChronoShare**, NDN file sharing platforms (PyNDN)
- **NDNoT**, the Named Data Network of Things toolkit for the Raspberry PI (PyNDN, NDN-JS)
- **ndnrjs**, a javascript implementation of an NDN repository (NDN-JS)
- **ndnrtc**, a peer-to-peer multiparty audio, video, and chat application (NDN-CPP, NDN-JS)
- **ndnstatus**, the NDN routing status web page (PyNDN, NDN-JS)
- **NDNVideo**, a video playout application for NDN (PyNDN)
- **OpenPTrack-NDN** an open source person tracking system that will add NDN support in Fall 2014. (NDN-CPP)

Advanced API Project

Over the last year we started a new API development that provides higher-level abstractions than that provided by the existing libraries. The existing libraries provides a basic programming interface called Interest / Data API, which lacks features commonly desired by applications, such as content segmentation, packet reassembly, reliable data fetching, flow control, realistic security capabilities, and the ability to influence forwarding decisions. Consequently application developers spent considerable time re-implementing these functions. This observation motivated us to develop a new API to fit NDN's information distribution model. We expect this new API to play the same role in NDN as sockets in TCP/IP.

We dedicated time and many thoughtful discussions this year to identifying commonalities in the design of various NDN applications, and to design common programming abstractions to accommodate them. We proposed two new programming abstractions – consumer context and producer context – specifically tailored for NDN's distribution model to allow developers to build applications quickly and consistently. We have built a prototype of this Consumer-Producer API as described in DN Technical Report NDN-0017 [5]. The current state of this research project was presented at the First NDN Community Meeting [7], and a brief description of the design, "Consumer-Producer API for Named Data Networking", was accepted by ACM ICN 2014 Conference as a poster presentation [6].

Information maximization

Researchers at UIUC investigated the exploitation of hierarchical data names to achieve information-utility maximizing data transport. A novel transport-layer mechanism, called the information funnel, was developed that maximizes a measure of delivered information utility. Named-data networking is especially suited for utility-maximizing transport because the network is aware of hierarchical data names, allowing intelligent sampling of sets of named objects such that redundancy is minimized and information utility is increased. NDN gives rise to a notion of distance between named objects that is a function of only the topology of the name tree. This distance, in turn, can expose similarities between named objects that can be leveraged for minimizing redundancy among objects stored in cache or transmitted over bottlenecks, thereby maximizing their aggregate utility. With appropriate naming conventions, objects with higher similarity will share a longer prefix in the name tree. For example, different versions of the same object might share all parts of the name except a version number. Hence, partial redundancy can be removed by preferentially dropping objects that share more of their names with others (in the case of versioned objects, this results in retaining the latest version). Experiments at UIUC demonstrated that with a proper hierarchical name space design, the information funnel prioritizes transmission and storage of data objects to maximize information utility, with very weak assumptions on the utility function. This prioritization is achieved merely by comparing data name prefixes, without knowing application-level name semantics, which makes it a generalizable transport-layer mechanism across a wide range of applications. Evaluation results show that the information funnel improves the utility of the collected data objects compared to other lossy protocols.

The information funnel is motivated by the advent of the age of data overload, where applications will increasingly need a new transport protocol, different from TCP and UDP, that offers a representative sampling of information as opposed to reliably conveying all bits (TCP) or randomly dropping packets (UDP). The current protocol stack does not offer an efficient and general way to implement such transport because the network is aware of bits or abstract datagrams only, which carry no information on application-level data boundaries. In contrast, in an NDN-based architecture, the network is aware of application-relevant named objects, which can be meaningfully sampled. Hence, the information funnel allows applications to retrieve a representative sampling of information based on an information-flow maximizing protocol. The protocol resulted in a paper published in DCoSS 2014 [13].

3.1.5 New Architectural Findings from Application Development

Collaboration between the application, architecture, and NDN testbed teams has pointed to important areas of further research that generalize some of the experiences of recent design and implementation effort.

- Deployment on the testbed of applications such as ndnrtc and ChronoChat have demonstrated the critical relationship between application packet delivery requirements and per-namespace forwarding strategy in the forwarder. What strategies are needed to support different applications, and how those strategies are selected and configured will be an important part of future research.
- ndnrtc and other applications using real-time data suggest that consumer retransmission strategies designed to maximize throughput and latency performance amid varying network conditions are an important general area of consideration. We can draw from previous work in TCP flow control for this, and potentially provide library support for data buffering and interest pipelining based on application-level deadlines and performance targets.
- Although NDN is intrinsically "pull-based", the internal approach of many applications is to "publish-

and-forget", so that they do not have to handle incoming interests for data concurrently with other aspects of the application. Even for dynamic data, an application may wish to publish it persistently for the lifetime of the process, so that the data does not need to be regenerated due to loss, or for requests for historical data. Our libraries have evolved to support in-memory, application store to meet such needs. An emerging area of investigation is the relationship between per application storage and the local content store, which could also provide this functionality in a centralized and efficient manner on a given host.

- Naming convention design for versioning, segmenting, timestamping, and the general inclusion of metadata associated with data objects have evolved over the course of this year and will continue to be a focal area of research. The project team recently released a technical memo describing the current approach to such conventions, hoping to inspire community dialogue [11]. How to best translate multi-dimension object descriptions into NDN names has been a challenge in the design of a number of NDN applications, a challenge we plan to address in the coming year.

3.1.6 Values in Design

Our collaboration with Values in Design researchers continued this year, resulting in the "World on NDN" technical report [9] and a related upcoming submission [10]. This paper, collaboratively authored with Katie Shilton at the University of Maryland with applications and architecture researchers of the NDN project team, discusses implications for free speech, anonymity and obscurity, data retention and forgetting, privacy, content regulation, law enforcement, and network neutrality. We look forward to continuing this collaboration through Shilton's upcoming NeTS-Small grant, which will enable further engagement with the NDN namespace design and trust management.

References

[1] Adeola Bannis. Ndn internet of things toolkit for raspberry pi (poster). In *NDN Community Meeting*, Los Angeles, CA, September 2014.

[2] Boost Project. Boost Libraries. Online: `http://www.boost.org/`, 1998.

[3] Peter Gusev, Zhehao Wang, and Jeff Burke. Ndn real-time conferencing library (poster). In *NDN Community Meeting*, Los Angeles, CA, September 2014.

[4] Alex Horn, Adeola Bannis, Jeff Burke, Dana Cuff, and Jason Payne. Ambient informatics - ndn bus bench (poster). In *NDN Community Meeting*, Los Angeles, CA, September 2014.

[5] Ilya Moiseenko and Lixia Zhang. Consumer-producer api for named data networking. Technical Report NDN-0017, Revision 1, NDN, February 2014.

[6] Ilya Moiseenko and Lixia Zhang. Consumer-producer api for named data networking. In *Proc. 1st ACM Conference on Information-Centric Networking (ICN-2014)*, Paris, France, September 2014.

[7] Ilya Moiseenko and Lixia Zhang. Consumer-producer api for named data networking (poster). In *NDN Community Meeting*, Los Angeles, CA, September 2014.

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

[9] Katie Shilton, Jeff Burke, Kimberly Claffy, Charles Duan, and Lixia Zhang. A world on ndn: Affordances & implications of the named data networking future internet architecture. Technical Report NDN-0018, Revision 1, NDN, April 2014.

[10] Katie Shilton, Jeffrey Burke, Kimberly Claffy, and Lixia Zhang. Anticipating policy and social implications of named data networking. *Communications of the ACM (submitted).*

[11] NDN Project Team. Ndn technical memo: Naming conventions. Technical Report NDN-0022, Revision 1, NDN, July 2014.

[12] Jeff Thompson and Jeff Burke. Ndn common client libraries. Technical Report NDN-0024, Revision 1, NDN, September 2014.

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

[14] Zhehao Wang, Zening Qu, and Jeff Burke. Matryoshka: Design of an ndn multiplayer online game (demo). In *Proc. 1st ACM Conference on Information-Centric Networking (ICN-2014)*, Paris, France, September 2014.

[15] Zhehao Wang, Zening Qu, and Jeff Burke. Project matryoshka: Ndn multiplayer online game (poster). In *NDN Community Meeting*, Los Angeles, CA, September 2014.

[16] Takahiro Yoneda, Ryota Ohnishi, Eiichi Muramoto, and Jeff Burke. Consumer-driven adaptive rate control for real-time video streaming in named data networking. *IEICE Trans. Fundamentals Commun. Electron. Inf. & Systems. (Tentatively accepted for publication.)*, 2014.

[17] Yingdi Yu, Alexander Afanasyev, Zhenkai Zhu, , and Lixia Zhang. An endorsement-based key management system for decentralized ndn chat application. Technical Report NDN-0023, Revision 1, NDN, July 2014.

[18] Zhenkai Zhu, Alex Afanasyev, Yingdi Yu, and Lixia Zhang. Chronochat: a server-less multi-user instant message application over ndn (poster). In *NDN Community Meeting*, Los Angeles, CA, September 2014.

Demonstrations, and Posters

1. Adeola Bannis. Poster presentation, demo, and conference talk, "NDN Internet of Things Toolkit for Raspberry Pi." NDN Community Meeting. September 3-5, 2014, Los Angeles CA.
2. Alex Horn, Adeola Bannis, Jeff Burke, Dana Cuff, Jason Payne. Poster presentation and demo, NDN Community Meeting. "Ambient Informatics - NDN Bus Bench." September 3-5, 2014, Los Angeles CA.
3. Giulio Grassi. Poster presentation, NDN Community Meeting. "Using GeoFaces to route Interests and Data in Vehicular Networks." September 3-5, 2014, Los Angeles CA.
4. Peter Gusev, Jeff Burke, Zhehao Wang. Poster presentation, demo and conference talk, NDN Community Meeting. "NDN Real-Time Conferencing Library." September 3-5, 2014, Los Angeles CA.
5. Ilya Moiseenko and Lixia Zhang. Poster presentation and lightning talk, NDN Community Meeting. "Consumer-Producer API for Named Data Networking." September 3-5, 2014, Los Angeles CA.
6. Jeff Thompson and Jeff Burke. Poster presentation and demo, NDN Community Meeting. "NDN Common Client Libraries API." September 3-5, 2014, Los Angeles CA.
7. Zhehao Wang. Poster presentation and demo, NDN Community Meeting. "Project Matryoshka: NDN Multiplayer Online Game." September 3-5, 2014, Los Angeles CA.
8. Zhenkai Zhu, Alexander Afanasyev, Yingdi Yu, Lixia Zhang. Poster presentation and demo, NDN Community Meeting. "ChronoChat: a Server-less Multi-User Instant Message Application Over NDN." September 3-5, 2014, Los Angeles CA.

3.2 Routing

Contributors	
PIs	Beichuan Zhang (Arizona), Lan Wang (Memphis), Dmitri Krioukov (CAIDA/NU), Lixia Zhang (UCLA)
Grad Students	Cheng Yi, Junxiao Shi, Yifeng Li (Arizona); A. K. M. Mahmudul Hoque, Minsheng Zhang (Memphis); (UCLA)
Undergrads	Nathan Yee (Arizona); Adam Alyyan, Marc Badrian, Ashlesh Gawande, Vince S. Lehman, Nic Smith (Memphis)
Staff	Chiara Orsini, Rodrigo Aldecoa, Ken Keys, Marina Fomenkov and Alex Ma (CAIDA) **Postdoc:** Alex Afanasyev (UCLA), Syed Obaid Amin (Memphis)

The goal of NDN's network layer is to provide a name-based packet delivery service for applications to build upon. To meet the requirements of the future Internet, the network layer must have 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.

Over the last year the routing group's effort focused on the following three areas:

- **Dynamic intra-domain routing protocol**: We re-implemented the *Named-data Link State Routing Protocol (NLSR)* [3] to work with the newly developed NFD. NLSR supports both link state and hyperbolic routing (Section 3.2.2), uses ChronoSync for routing database synchronization among routers, and uses a hierarchical trust model for routing security within a single administrative domain. NLSR has been deployed on the NDN testbed.
- **Hyperbolic routing**: We embedded an updated NDN testbed topology into the hyperbolic plane using more recent Internet topology data, and our new, simpler and more efficient, hyperbolic network mapping algorithm. We also implemented basic hyperbolic routing in NLSR by disseminating hyperbolic coordinates in link state announcements. We have been conducting experiments on Emulab to evaluate the feasibility of hyperbolic routing in NDN by comparing it with link-state routing under various conditions. We also developed and released a stand-alone software package for hyperbolic network generation and greedy routing simulation.
- **Dynamic Interest Limiting for NDN Congestion Control**: We designed a new scheme to detect link-layer packet losses and dynamically adjust the Interest packet forwarding rate to control network congestion. This scheme is the first hop-by-hop NDN congestion control approach that does not assume knowledge of underlying link bandwidth and can work well over tunneling links. Our evaluation using ndnSIM showed significant performance improvement over other proposed approaches [8, 6, 2, 7] .

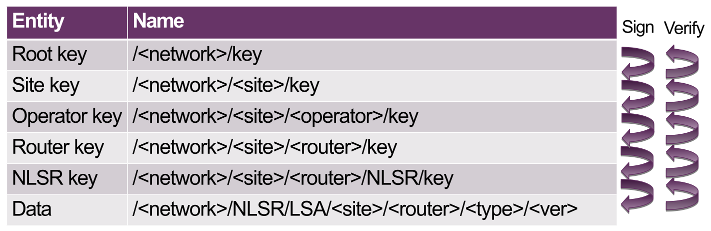
3.2.1 Named-data Link State Routing Protocol (NLSR)

Named-data Link State Routing (NLSR) [3] runs directly on top of NDN, i.e., it uses NDN's Interest and Data packets to exchange routing updates. NLSR uses names instead of IP addresses to identify the various components of a routing system, and can use any underlying communication mechanism that NDN uses (e.g., Ethernet, IP tunnels, TCP/UDP tunnels) for routing message exchanges. In Year 3 we implemented a preliminary version of NLSR in C that runs over CCNx and discovered several problems with CCNx sync/repo, 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. During Year 4, as the new NDN forwarding daemon NFD was being developed, we re-implemented NLSR in C++ using the new ndn-cxx developer library to work with NFD. Below we summarize the main design decisions and features in NLSR.

We designed a naming scheme that associates various entities in the routing system with each other (see Table 3.1). In the case of intra-domain routing, the relationship between routers, routing processes, and routing data is inherently hierarchical. Therefore, we decided to use a hierarchical naming scheme to reflect this hierarchy of operation. More specifically, a router name has its network name as the first name component and site name as the second component. The NLSR process running on a router has the router name as its prefix, and similarly the name of a routing update generated by an NLSR process has an association with the NLSR process name.

Component	**Name**
Router	/⟨network⟩/⟨site⟩/⟨router⟩
NLSR	/⟨network⟩/⟨site⟩/⟨router⟩/NLSR
NLSR data	/⟨network⟩/NLSR/LSA/⟨site⟩/⟨router⟩/⟨type⟩/⟨version⟩
NLSR key	/⟨network⟩/⟨site⟩/⟨router⟩/NLSR/key
Router key	/⟨network⟩/⟨site⟩/⟨router⟩/key

Table 3.1: Naming Scheme in NLSR (not all names are shown.)



Entity	Name
Root key	/<network>/key
Site key	/<network>/<site>/key
Operator key	/<network>/<site>/<operator>/key
Router key	/<network>/<site>/<router>/key
NLSR key	/<network>/<site>/<router>/NLSR/key
Data	/<network>/NLSR/LSA/<site>/<router>/<type>/<ver>

Figure 3.8: Routing Security Trust Model

We devised a hierarchical trust model based on the common management structure and operational practice in a single network domain. Since NLSR routing updates are carried in NDN data packets and every NDN data packet carries a signature, a router can verify that a routing message is created by the origin router and that the message is not altered during dissemination. While NLSR benefits from the security primitives provided by NDN, we still need a trust model to ensure that the signer of the routing message is indeed the origin router. Figure 3.8 shows our trust model with a root key owned by the administrator of the network. When an NLSR router receives a Data packet, it uses the trust model to map the data name to the correct NLSR key name and compares this name with the key name carried in the Data packet. If the names match, it fetches the NLSR key and verifies the signature on the Data packet. It then verifies the NLSR key using the router key, operator key, and site key until NLSR reaches the self-signed root key (preconfigured). If at any step the signature is found to be invalid or the key cannot be located, NLSR determines that the data is signed by an unauthorized key. If the final verification step does not reach the root key, the data is considered illegitimate.

NLSR routers distribute link-state advertisements (LSA) for adjacency information and for name prefixes reachable through that router (the LSA formats are shown in Figure 3.9). Routers use the Adjacency LSAs to build a network topology, calculate routing paths, and determine next-hops for names, and use the Name LSAs to track which name prefixes are reachable through routers in the network. These LSAs are stored in a link-state database (LSDB), which is synchronized with other routers using ChronoSync [9]. Every node periodically sends a digest of their LSDB to other nodes in the network using NDN Interest packets. When a node produces a new LSA and its digest changes, it will reply to others' Interests with the name of the new LSA. Other nodes can then fetch the new LSA data. LSDB synchronization is shown in Figure 3.10.

To detect link and process failures, NLSR sends periodic `INFO` Interest messages to each neighboring router. If an `INFO` Interest times out, NLSR will resend `INFO` Interests in case the Interest was lost. If there is no response from the neighbor after a few tries, the adjacency with the neighbor is considered `INACTIVE`. NLSR will then send a new adjacency LSA to inform others of the topology change. The NLSR process will continue to send `INFO` Interests to detect the recovery of the neighbor. When the adjacency recovers, NLSR will receive a response to its `INFO` Interest and change the adjacency status to `ACTIVE`. It will again send a new adjacency LSA for the topology change.

One important feature of NDN is multi-path forwarding. NDN routers can forward Interests to multiple faces to find the best path by taking advantage of the forwarding state for each Interest, which helps detect loops and store measured round-trip delay between Interest and corresponding Data packets. To support multi-path, NLSR uses a modified Dijkstra's algorithm to produce a ranked list of policy-compliant next-

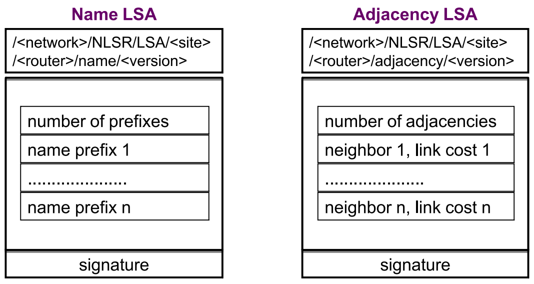
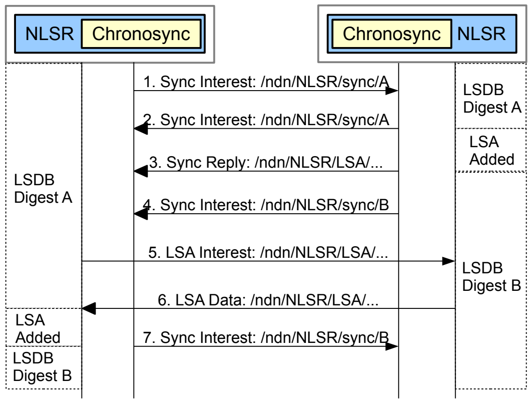


Figure 3.9: LSA Formats



NLSR
Chronosync
Chronosync
NLSR
1. Sync Interest: /ndn/NLSR/sync/A
2. Sync Interest: /ndn/NLSR/sync/A
3. Sync Reply: /ndn/NLSR/LSA/...
4. Sync Interest: /ndn/NLSR/sync/B
5. LSA Interest: /ndn/NLSR/LSA/...
6. LSA Data: /ndn/NLSR/LSA/...
7. Sync Interest: /ndn/NLSR/sync/B
LSDB Digest A
LSA Added
LSDB Digest B
LSDB Digest A
LSA Added
LSDB Digest B

Figure 3.10: LSDB Synchronization

hops and inserts them into the FIB. This essentially provides a name-based multi-path routing table, which can be utilized by NDN's forwarding strategy.

NLSR 0.1.0 was released in August 2014 (`http://named-data.net/doc/NLSR/0.1.0`). It is currently running and being evaluated on the NDN testbed, which includes 16 nodes and 29 links.

3.2.2 Hyperbolic Routing

Since the NDN name space is in principle unbounded, a scalable routing solution is necessary. Hyperbolic routing is a potential solution to this problem if one can map a given real network topology to hyperbolic space [4, 1]. Hyperbolic routing is a geometric routing scheme that relies on hyperbolic coordinates to send packets efficiently through a network. Assuming a mechanism for retrieving the coordinates of a name prefix exists, each Interest can carry the coordinates of the name prefix and routers and use greedy routing to forward the Interest, i.e., choose the next hop(s) for the Interest based on the distances between a router's neighbors and the desired name prefix. This scheme is highly scalable as there is no need to maintain a routing table or FIB and there are no dynamic routing updates except to learn neighbors' coordinates.

The viability of hyperbolic routing in NDN depends on several factors. First, is the underlying topology hyperbolic? Second, can the forwarding strategy at each NDN router effectively find the best path in case the hyperbolic coordinates do not give the best path? Third, when an Interest reaches a local minimum, which can happen due to either greedy routing or a failure, can the forwarding strategy find alternative paths quickly? Finally, how sensitive is the routing performance with respect to the density of the topology and the number of faces allowed in multi-path forwarding?

To explore above questions, we developed *HyperMap* [5], a simple method to map a given real network to its hyperbolic space using a recent geometric theory of complex networks modeled as random geometric graphs in hyperbolic spaces. The method replays the network's geometric growth, estimating at each time step the hyperbolic coordinates of new nodes in a growing network by maximizing the likelihood of the network snapshot in the model. We applied HyperMap to embed the AS-level Internet topology derived from CAIDA's Ark measurements into its hyperbolic space and thus obtained the hyperbolic coordinates of the ASes participating in the NDN testbed.

We then measured the performance of the modified greedy forwarding algorithm in the NDN testbed. This algorithm excludes the current node from any distance comparisons and finds the neighbor closest to the destination. The packet is dropped if this neighbor is the same as the packet's previously visited node. We used the following performance metrics: (i) the success ratio which is the percentage of the successful paths that reach their destinations; and (ii) the average stretch of three types. Stretch 1 is the standard hop stretch measured on the actual topology, that is the ratio between the hop lengths of greedy paths and the corresponding shortest paths in the graph. Stretch 2, measured in the underlying hyperbolic space, is the ratio of the length of a successful greedy path to the actual hyperbolic distance between the source and the destination. Stretch 3, also measured in the underlying hyperbolic space, is the ratio of the length of the shortest path to the actual hyperbolic distance between the source and the destination. The lower these two hyperbolic stretches, the closer the greedy and shortest paths stay to the hyperbolic geodesics, and the more congruent the network topology is with the underlying geometry. (`http://www.caida.org/research/routing/greedy_forwarding_ndn/`)

If the hyperbolic coordinates of testbed sites are set to the hyperbolic coordinates of their corresponding ASes, then the greedy routing success ratio (i.e., the ratio of source-destination pairs than can successfully communicate via shortest greedy routing paths) is 82% vs. 71% in our previous embedding. In an ideal hyperbolic topology constructed for the testbed by using the same node coordinates and setting up links between the sites according to the growing hyperbolic network model, we obtain 100% success ratio both in the full graph of the testbed, and in all graphs obtained removing any single link or node. This result indicates that such failures would induce no routing updates or path recompilations in such testbeds.

We also implemented hyperbolic routing in NLSR by propagating router coordinates to the network through link state advertisements. Using a set of emulated experiments on a network of 10 nodes, we compared the performance of hyperbolic routing (HR) to that of link state (LS) routing, which calculates the optimal paths, under various conditions including different network topologies, forwarding strategies, number of multi-path faces, and failure scenarios. If it performs as well as link state routing, then hyperbolic routing can be considered a viable NDN routing protocol. We tested across two separate topologies to emulate the conditions of an ideal network scenario, as well as a setup that mimics the actual NDN testbed and the ideal topologies described above. We employed both best-route and ncc forwarding strategies to determine which is more suitable to hyperbolic routing. The ncc forwarding strategy in NFD is the same as the forwarding strategy in CCNx, while the best-route strategy uses the best path calculated by the routing scheme as long as the path returns data. The performance metrics are Round-Trip Time (RTT), packet loss ratio, number of messages generated, and failure response time. These tests revealed three findings: (1) the forwarding strategy greatly impacts the performance of hyperbolic routing. In our tests, the ncc strategy performed better (in terms of RTTs) than the best-route strategy because ncc utilizes multiple paths more efficiently; (2) in the "ideal" topology, RTTs under HR and LS are close regardless of the number of multi-path faces used. In the real testbed topology, using more multi-path faces improves the RTT under HR; and (3) the accuracy of the hyperbolic coordinates affects performance. Surprisingly, resulting RTTs in the ideal topology are sometimes much higher than those in the real NDN testbed topology. The HR coordinates may not be ideal, which would lead to suboptimal routes. Our results are at `http://netwisdom.cs.memphis.edu/pvthome.html`.

Finally, we implemented *hggraphs*, a C++ library that provides a collection of functions and data structures for a) generating synthetic graphs embedded in hyperbolic metric spaces, and b) computing properties of the graphs associated with the hyperbolic geometry (e.g. hyperbolic distance between two vertices). The *hggraphs* library supports the development of the hyperbolic routing in the NDN environment in two ways: a) it enables the implementation of tools to assess the effectiveness of the greedy routing approach in synthetic networks of variable size; b) it facilitates the creation of new ndnSIM scenarios in which we can extend the default forwarding strategy to simulate the hyperbolic routing on an NDN network. The library is available at `http://named-data.github.io/Hyperbolic-Graph-Generator/` and contains two additional tools: (i) the Hyperbolic Graph Generator - a command line utility that generates synthetic topologies embedded in a hyperbolic metric space and saves the graph in an output file; and (ii) the greedy routing tester - a command line tool that loads in memory a graph generated by the Hyperbolic Graph Generator and returns the greedy routing success ratio, i.e. the percentage of paths successfully connecting two random vertices built according to the greedy forwarding approach. We also developed *HyperbolicRoutingScenario*, a new ndnSIM scenario that generates a synthetic network topology, embeds into an hyperbolic space and simulates the hyperbolic greedy forwarding on each vertex to calculate the routing success ratio. This scenario relies on the ndnSIM simulator and leverages the utility functions provided by the hggraphs library. The software is currently under test.

3.2.3 Dynamic Interest Limiting for NDN Congestion Control

Data packets carry content, are generally much larger than Interests and more likely to cause congestion. An NDN node can control the amount of Data packets received from an interface by limiting the number of Interests departing through that interface. The per-interface Interest limit can be coupled with adaptive forwarding to optimize network resource usage, e.g., diverting overflowing Interests to alternative interfaces. We have explored the advantages of NDN's adaptive forwarding with respect to congestion control, but have not yet discovered a practical and effective congestion control scheme.

Interest rate limiting induces a tradeoff: wasted bandwidth (if the limit is too small) vs. congestion (if too big). The proposed methods to compute the Interest limit so far [8, 6, 2, 7] all have serious limitations. A common false assumption of these solutions is that key network and traffic properties are known: size and/or retrieving delay of Data packets, available bandwidth of each link. Currently, NDN is typically deployed as overlay on top of IP or Ethernet, where NDN nodes are oblivious about the bandwidth of underlying links – one NDN hop can be many IP/Ethernet hops, with a bottleneck link several hops away from the NDN node. The bottleneck link can be shared by many flows, making the actually available bandwidth dynamic.

To address these obstacles in the existing congestion control proposals, we designed Dynamic Interest Limiting (DIL). A static Interest limiting mechanism does not work well in NDN due to the dynamics in Data size, round-trip time and underlying link bandwidth; thus routers need long output queues to accommodate the potential bursty traffic caused by such dynamics. However, excessive packet buffering may cause bufferbloat, inducing high latency and jitter for applications and degradation of throughput. Traditional AQM mechanisms such as RED and CoDel are not efficient in NDN since they will simply drop Data without explicit notification. We propose to adjust the Interest limit for each interface dynamically using an AIMD algorithm similar to TCP. The Interest limit increases when Data is received, and decreases when congestion is detected. To manage the queue length of the upstream node, a novel Random Early NACK (REN) mechanism can send explicit congestion notification to the downstream node.

One limitation of REN is that it cannot be used to detect congestion in NDN overlay scenarios, since NDN nodes may not be able to monitor the queues of underlying IP routers. We propose Link-layer Congestion Detection (LCD) for such scenarios. Each NDN node adds a link-layer header to each NDN packet, such that the node at the other end of the link can detect congestion by counting the *gaps* in the received sequence numbers. REN and LCD work together to detect congestion more reliably than either alone.

Another feature of DIL is to enforce fairness at routers. The Fair Interest Limiting (FIL) component of DIL can fairly divide the total Interest limit among multiple namespaces. With FIL, we no longer rely on end hosts to provide fairness, hence eliminating the potential damage caused by ill-behaved consumers.

Figure 3.11 provides an overview of DIL, where solid arrows represent flow of regular packets while dashed arrows represent flow of rejected packets. One DIL module is installed on each interface. When a packet arrives at an interface, the LCD module will remove the link-layer header and store the sequence number, and then pass the remaining packet to the REN module. REN only checks incoming Interests, and will reject them if the output queue of the interface keeps growing. If a packet is accepted, it will be further passed to the forwarding strategy module. When forwarding Interests, the strategy module will query the DIL module for availability of interfaces. An Interest is forwarded to an interface only if the limit for the flow has not been reached. Otherwise, the strategy module will immediately try alternative interfaces. The LCD module will append link-layer headers to the packets before they are actually forwarded.

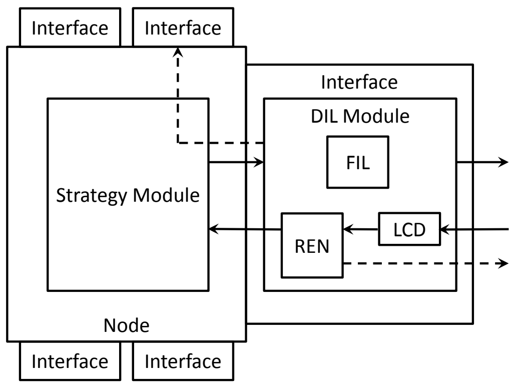


Figure 3.11: Node model for DIL.

We evaluated DIL under different scenarios, including with and without caching, various end-to-end delays, various topologies and different sizes of buffers. On a small linear topology, DIL finished 11.3% faster than HIS [7] in transferring $10M$ bytes of Data while always maintaining an average queue under 3 packets after the initial stage. In the more general case with caching, DIL achieves shorter application delay and completion time. In a large-scale experiment using Sprint topology and many concurrent traffic flows, DIL finished faster than TCP in all cases.

References

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

[2] Giovanna Carofiglio, Massimo Gallo, and Luca Muscariello. Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks. In *Proceedings of ACM SIGCOMM ICN Workshop*, 2012.

[3] 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.

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

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

[6] N. Rozhnova and S. Fdida. An effective hop-by-hop Interest shaping mechanism for CCN communications. In *Proceedings of IEEE INFOCOMM NOMEN Workshop*, 2012.

[7] Yaogong Wang, Natalya Rozhnova, Ashok Narayanan, David Oran, and Injong Rhee. An improved hop-by-hop interest shaper for congestion control in named data networking. In *Proceedings of ACM SIGCOMM ICN Workshop*, 2013.

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

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

3.3 Scalable Forwarding

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

The goal of our forwarding research 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 of arbitrary length. In addition, we provide feedback to the architecture and routing research teams when 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.

During the fourth year of the NDN project, we refined our previously proposed data structures and algorithms for the Forwarding Information Base and Pending Interest Table, and we started implementing an NDN forwarding engine on a general-purpose multi-core platform. This section provides describes our activities and findings in the forwarding area.

3.3.1 Scalable Name-Based Forwarding

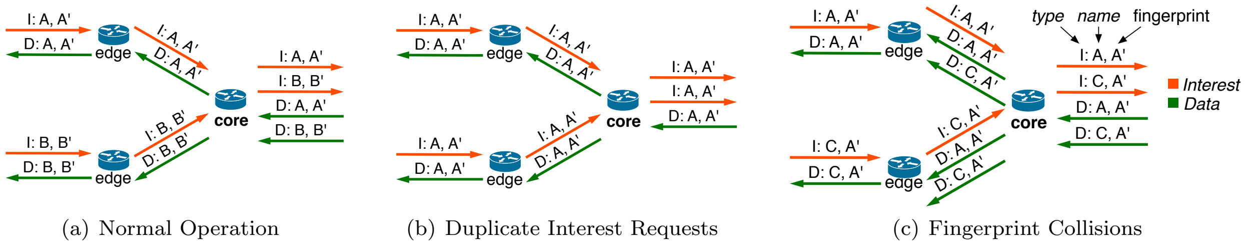
Name-based packet forwarding represents a core characteristic of the NDN architecture. IP-inspired forwarding methods are not feasible because a) name-based forwarding must support variable-length lookup keys of unbounded length, and b) namespaces for data are substantially larger than the global address prefix rule sets used in today's Internet. In our previous work, we have investigated the information-theoretic difference approach and proposed trie-based solutions that can scale variable-length name forwarding to billions of name prefixes [1]. We have also proposed fingerprint-based solutions to improve the name-based forwarding performance further, where both trie-based and hash table-based data structures have been developed [2].

In the past year, we have continued to refine and evaluate the proposed design. To demonstrate the scalability of the proposed data structures, we have evaluated the lookup performance with a forwarding table that contains 1 billion synthetic names. The hash table-based implementations outperform the trie-based implementations because of less number of memory references. The proposed fingerprint-based hash table requires only 3.2 GB to store 1 billion names, and the measured lookup latency of the software-based implementation is 0.69 microseconds on the machine equipped with 2.53 GHz Intel Xeon E5540 processor, 8 MB of L3 cache, and 12 GB of DDR3 memory.

The proposed succinct data structures are typically generated from data structures that maintain the entire forwarding information. We have investigated incremental updates on the succinct data structures and evaluated the insertion latency of data structures that hold the complete forwarding information. The insertion latency was measured on a machine equipped with 1.7 GHz AMD Opteron 6164 HE processor and 96 GB of memory because the data structures that hold the complete information have large memory requirements. The measured insertion latency of the fingerprint-based hash table is 1.92 microseconds.

3.3.2 Scalable Pending Interest Table Design

The Pending Interest Table (PIT) is another core component of the NDN architecture. Scalable PIT design is challenging because it requires per-packet updates, including memory write operations; and the entire content name strings are stored in the PIT, requiring more memory. Hence, high-speed memory devices, such as SRAM and RLDRAM, are favored, but they have limited capacity. We have proposed the fingerprint-only PIT design [3] that stores fingerprints rather than content name strings to reduce the memory requirements. To guarantee packet delivery when fingerprint collisions occur, we relax the Interest aggregation feature in the PIT so that all Interest packets are forwarded. In addition, the colliding PIT entry will not be deleted until it expires to provide sufficient time for potential Data packets to arrive. Because the PIT cannot differentiate between duplicate Interest requests and hash collisions, we propose a network-wide solution to reduce the chances of receiving duplicate Interest requests in the core routers. In our design, only core routers store the fingerprints and relax the Interest aggregation feature. Edge routers perform normal operations and still aggregate Interest packets.



(a) Normal Operation (b) Duplicate Interest Requests (c) Fingerprint Collisions

Figure 3.12: PIT Operations

Essentially, our design is based on the ideas that storing fingerprints saves memory space, and that edge routers can aggregate most duplicate Interest packets. In the case with only one core router, three operational situations are shown in Figure 3.12: a) Normal Operation, where different Interest names map to different fingerprints, e.g., name A maps to fingerprint A', and name B maps to B'. This scenario operates the same way as with storing name strings, and introduces no traffic overhead. b) Duplicate Interest Requests, which are not aggregated, increasing traffic overhead. If duplicate requests arrive at the core router from different faces, the fingerprint stays in the PIT longer, inducing memory overhead. c) Fingerprint Collisions, where two content names share the same fingerprint. The two corresponding Interest packets consume one PIT entry, rather than two, reducing PIT memory consumption. But the colliding PIT entry stays longer in the PIT, which introduces memory overhead compared with the normal case. Also, both Data packets arrive at both faces, inducing data traffic overhead. To evaluate the design, we implemented it in software and analyzed memory and network traffic overhead in the case of single core router software [3].

During the fourth year, we refined the PIT design to support the case with multiple core routers. The challenge with multiple core routers is that each core router knows only local fingerprint collisions and does not have knowledge of remote fingerprint collisions, thus packet delivery cannot be guaranteed when an Interest packet has a remote fingerprint collision but no local collision. We have proposed a solution to address this issue. Each data packet carries an additional bit to indicate if it has encountered any PIT collision along the path. This way, remote fingerprint collision information is propagated to downstream routers. After receiving remote collision information, downstream routers mark the corresponding PIT entries as collided and retain them until expiration. This additional mechanism increases the effective fingerprint collision rates, thus increasing memory requirements and network traffic overhead. We will continue to analyze and optimize the memory requirements and network traffic overhead in the multiple core router case.

We have also investigated issues with supporting all prefix matching with the proposed PIT design that performs only exact matching with fingerprints of the full names. Because edge routers store the complete content names and support all prefix matching, the Data packets can carry additional information about matched prefixes from edge routers to core routers. After receiving the information, the core routers can compute the corresponding fingerprints and consume as many pending Interests as possible. As part of the NDN-NP project, we will continue to explore and evaluate the detailed design.

3.3.3 Forwarding Engine Development

The best way to demonstrate the efficiency of the proposed FIB and PIT design is to build an NDN forwarding engine that employs these methods, and evaluate its performance with real network traffic. Implementing this forwarding engine also exploits performance optimization opportunities at the system level. We have started to implement an NDN forwarding engine on a general-purpose multi-core platform. The forwarding engine takes advantage of memory-efficient data structures and algorithms, parallel processing power in the multi-core platform, and fast packet I/O frameworks such as netmap [4] and Intel DPDK [5].

The NDN forwarding engine consists of five major components: Fast Packet I/O, Face Table, Forwarding Information Base, Pending Interest Table, and Content Store. The Fast Packet I/O component bypasses the operating system protocol stack so that packets can be efficiently processed in the user-space. The Face Table maps an interface to a face ID used internally in the forwarding engine, and is implemented as a hash table. The current implementation uses MAC addresses as Face Table lookup keys, as the forwarding engine updates the MAC addresses for each packet. Our design does not preclude using other information, such as IP+Port pairs, as lookup keys. The FIB supports multi-threading naturally because it is mostly used for lookup. The PIT and CS, which require frequent updates, needs to be implemented in a thread-safe manner. The PIT and CS can be implemented centrally and shared by all the cores, or in a distributed fashion where each core has its dedicated PIT and CS [6].

We have implemented the single-threaded NDN forwarding engine based on the netmap framework. As part of the NDN-NP project, we will continue to develop and evaluate the multi-core forwarding engine.

References

[1] "Scalable Name-Based Forwarding: From Millions to Billions", Tian Song, Haowei Yuan, Beichuan Zhang, and Patrick Crowley. Submitted to IEEE INFOCOM 2015.

[2] "Enhancing Scalable Name-Based Forwarding", Haowei Yuan, Patrick Crowley, and Tian Song. Submitted to IEEE INFOCOM 2015.

[3] "Scalable Pending Interest Table Deisgn: From Principles to Practice", Haowei Yuan and Patrick Crowley. In IEEE INFOCOM 2014.

[4] "netmap: A Novel Framework for Fast Packet I/O", Luigi Rizzo. In USENIX ATC'12.

[5] "Intel DPDK: Data Plane Development Kit", http://www.dpdk.org.

[6] "Named Data Networking on A Router: Fast and DoS-Resistant Forwarding with Hash Tables", Won So, Ashok Narayanan, and David Oran. In ANCS 2013.

3.4 Security

Contributors	
PIs	Van Jacobson (UCLA), Christos Papadopoulos (CSU), Lixia Zhang (UCLA)
Grad Students . .	Steven DiBenedetto (CSU), Yingdi Yu (UCLA)
Staff	**UCLA Postdocs:** Alaxendar Afanasyev

The implementations of some early NDN applications used default keys stored in a file to sign data, and either performed data verification themselves or otherwise skipped it entirely. Over the past year we built an experimental security library to facilitate the development of secure applications.

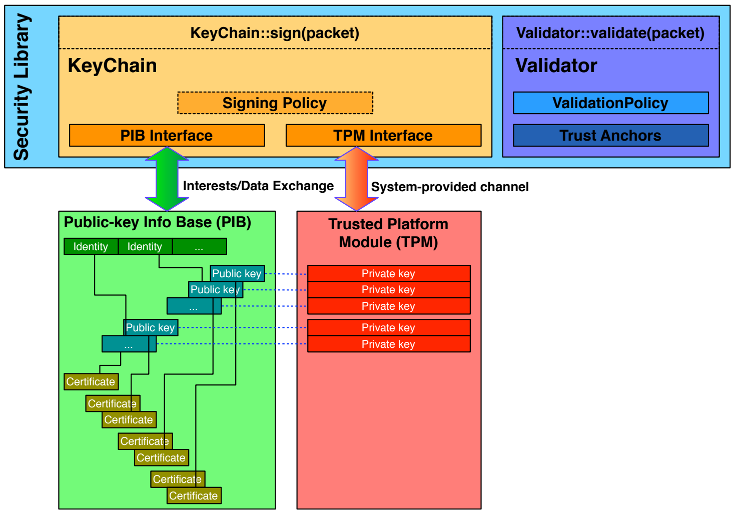


Figure 3.13: Framework of security library.

3.4.1 Security Library

Our experimental security library (Figure 3.13) provides support for efficient signing and validation of packets, so that application developers do not have to handle security operations by themselves. The library also includes configuration files that guide developers toward best practices. NDN applications that use the security library support include NFD, NLSR, ChronoChat, and a toolset for testbed certificate deployment. The reference implementation for the security library is part of **ndn-cxx**. Support for primary and stable features is also included in **NDN-CCL**.

Key Management

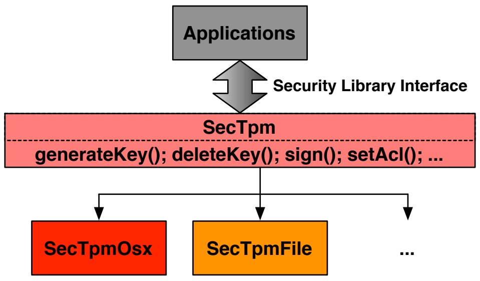

Applications
Security Library Interface
SecTpm
generateKey(); deleteKey(); sign(); setAcl(); ...
SecTpmOsx
SecTpmFile
...

Figure 3.14: The TPM Interfaces in the security library.

Protect private keys One of the primary goals for the security library is the ability to manage keys on a localhost, which requires hiding private key operations, e.g., access control and packet signing, from applications. The old API library stored private keys in a file encrypted with a password, which had three drawbacks: (1) applications must maintain a mapping from key name to file name; (2) applications must explicitly handle passwords for key decryption; and (3) private keys are exposed to applications, making them vulnerable to compromises. We introduced the Trusted Platform Module (TPM) into the security library to address these problems.

The system-provided TPM service (e.g., KeyChain in Mac OS X) stores private keys securely, and performs all key-related operations as a black box. The security library provides an NDN-friendly

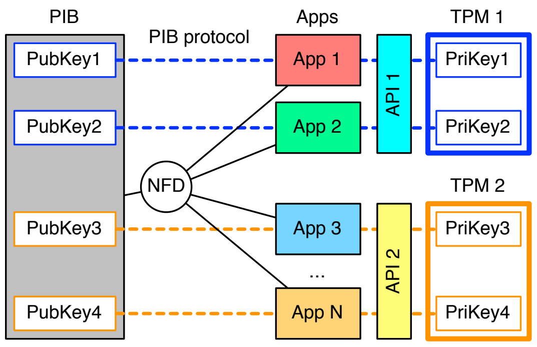


Figure 3.15: Serving public key information through PIB protocol to applications using different API libraries.

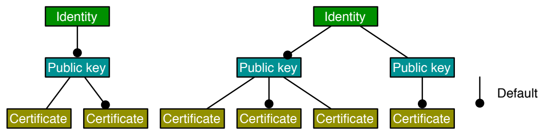


Figure 3.16: Organization of public key info in PIB.

interface to the TPM service, so that applications only need to specify the name of the signing key and the packet to sign, and the TPM service will return the appropriate signature according to the signing algorithm of the key. The system-provided TPM service also provides application-level access control, so that application developers do not need to handle passwords directly.

For the Mac OS X system, we provided a reference TPM interface SecTpmOsx in the security library to utilize OS X's TPM service. For other systems lacking the TPM service, the security library provides a built-in file-based pseudo TPM SecTpmFile. Since the file-based TPM is a lightweight placeholder for a real TPM, it simply relies on file system permission (rather than passwords) to provide basic user-friendly protection to private keys.

Another design goal of the TPM interface is to decouple the TPM implementation from the application programming interface (API), so that applications on different platforms do not have to know details of the underlying TPM. The TPM interface enables researchers to contribute to the security library by making their own TPM implementation (Figure 3.14).

Sharing public key information NDN applications on a local host may have shared access to certain keys and their meta-data, e.g., identities represented by keys, corresponding certificate names to put in KeyLocator fields of packets. This information can help applications decide which signing key to use, thus facilitating automated signing. To share this information within a local host, especially among applications developed in different programing languages. we created a *Public-key Information Base (PIB) service* that allows applications to query stored information through NDN Interest/Data exchanges. Our PIB query protocol [1] facilitates development of libraries in different language (Figure 3.15). The PIB maintains all information about a key, including identity it represents, the key name that uniquely identifies it, crypto algorithms of the key, certificates issued by different parties, etc. Each identity has a default key, and each key has a default certificate (Figure 3.16). The PIB simplifies the signing interface because applications only need to know the name of the signing identity, and can leave key and certificate resolution to the PIB service. Decoupling identity from exact key/certificate is our first step toward automated signing. It allows application developers to express signing policy in terms of trust relationships between identities; based on these signing policies, the security library can automatically determine the signing identity. Thus the library can determine the signing key from the name of the packet to be signed.

Validation Framework

Public key certificate format. This year we defined a new public key certificate format [4], which preserves all required fields of the X.509 certificate but allows researchers to extend the usage of certificates.

Unlike the old certificate format [5], the new format enables a certificate to carry all information in a single data packet, facilitating certificate fetching for packet validation.

Generalized validation process Packet validation involves several procedures, e.g., policy checking, validity period checking, and signature verification. All applications use some of these procedures. A standard extensible packet validation framework can allow researchers and developers to focus on designing trust models rather than implementation details. This year we investigated commonalities of packet validation in NDN applications, and defined a packet validation framework (Figure 3.17). The framework provides a policy module and hooks for developers to extend the validation process. In the policy module, one can specify rules that a packet should obey using the policy language described below.

Three hooks are located: 1) before fetching a key, 2) after fetching a key, and 3) after a certificate is validated. The first two hooks can be used to extend the key fetching process, allowing developers to fetch keys in different fashions. The last hooks can be used to extend the key authentication (e.g., some trust model may require more than one valid certificate to authenticate a key). We have included this framework in the most recently released codebase.

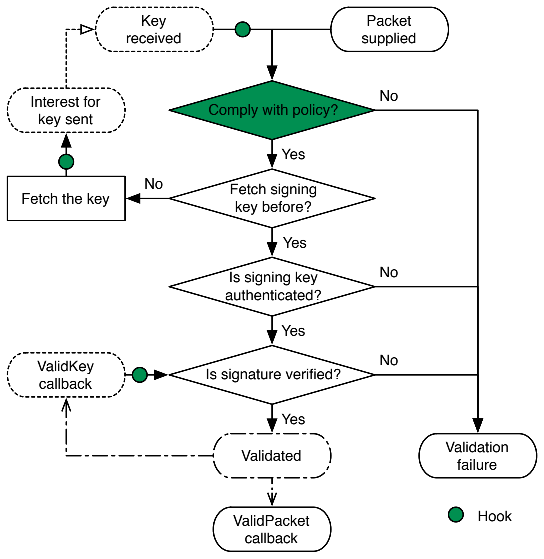


Figure 3.17: Processing flow in the validation framework.

Policy language Although many validation procedures are common across applications, trust models and implementations vary. To save researchers and developers the effort of implementing their trust model, we defined a policy language to allow expression of a trust model via configuration [2]. The configuration consists of a set of rules that a packet should obey, e.g., the relationship between a data packet name, signing key name, and type of packet signature. To describe the relationship between a packet name and the signing key name, we defined "Name Pattern" [3] to extract name components to use in name matching. The policy language also supports all signature types defined in the NDN TLV specification, and allows users to specify trust anchor management models. For example, users can specify the key bits or file path of a trusted key, and/or specify a trust anchor directory and have a policy module periodically update trust anchors in that directory, to support trust anchor rollover. Finally, the security library provides a policy parser to convert logic into policy language for use in this framework.

3.4.2 Testbed Certificate Deployment

We built a testbed certificate management system to help testbed users easily and securely obtain public key certificates. This system assumes a hierarchical trust model, thus it allows NDN site operators to apply for a public key certificates for their own site. With their site certificate, operators can issue public key certificates to users at their site. The system provides a web interface for site operators and users to submit requests for public key certificates, authenticate requester identities via email exchange, and then automatically determine the namespace allocated to the user and provide instructions for the user to generate a public key certificate signing request. The system also notifies a site operator when it receives a signing request for a certificate belonging to its site. If the operator approves the request, the system provide an automation tool to help the operator issue, publish, and notify the requestor of its new certificate. This certification system has been deployed over the NDN testbed (`http://redmine.named-data.net/projects/nfd/wiki/Ndncert`).

3.4.3 Web-of-Trust

Some NDN applications run between entities that all belong to a hierarchically structured namespace, and we have experimented with a trust model that strictly follows their naming hierarchy. There are other applications where the entities are not under the same hierarchical namespace, thus a hierarchical trust model does not apply well. For example, in ChronoChat, chatroom users may come from anywhere and their names do not share a common prefix, and the previous ChronoChat implementation could not enforce any membership control in a chatroom. Such distributed applications must manage trust in a distributed way. This year we explored a Web-of-Trust endorsement model to solve this problem by using ChronoChat [6] as a test case. A user in a chatroom expresses trust in another user via an endorsement, which associates the endorsee's public key with a trust scope expressed in an NDN name.

With the endorsement-based mechanism, a new user can join a chatroom with a membership endorsement (invitation) from an existing member in the chatroom. The new user responds by endorsing the membership of its inviter, in order to derive the membership of other users through the inviter. A user can stay in a chatroom as long as at least one existing member still endorses his membership; users with more endorsements are more unlikely to be removed from a chatroom. Membership endorsements constitute a graph that interconnects chatroom members, and bi-directional membership endorsements allow users to derive trust in membership from their own trust anchors (inviters), removing the need for a third-party trust anchor. Finally, unlike traditional WoT systems, which store certificates and revocations at centralized servers, ChronoChat uses its built-in synchronization mechanism, ChronoSync, to distribute and revoke endorsements, eliminating any third-party dependency. This endorsement-based trust model has been implemented in the latest version of ChronoChat.

3.4.4 Content Poisoning Mitigation

Malicious objects named identically to legitimate ones can slow or even prevent desired content retrieval. Consumers can detect poisoning of caches by verifying the signature on each object, but they have limited ability to evade such poisoning attacks, because Interests describe desired content rather than where to retrieve it. Routers cannot scalably retrieve keys and verify signatures to detect poisoned content objects. We designed a mitigation system that enables consumers to report poisoned content to their immediate upstream nodes. If the node can confirm the report, it can try alternate forwarding paths, as well as propagate the information further upstream. The propagation enables the network to purge cached copies, and ultimately evict the malicious source from the path. To prevent harm from false reports, our system requires originators of reports to cryptographically authenticate them, although routers may whitelist well-behaved downstream nodes.

References

[1] "Public-key Information Service", `http://redmine.named-data.net/projects/ndn-cxx/wiki/PublicKey_Info_Base`.

[2] "Trust Policy Language", `http://redmine.named-data.net/projects/ndn-cxx/wiki/CommandValidatorConf`.

[3] "Name Pattern", `http://redmine.named-data.net/projects/ndn-cxx/wiki/Regex`.

[4] "Public Key Certificate Format", `http://redmine.named-data.net/projects/ndn-cxx/wiki/Certificate`.

[5] "Deploying Key Management on NDN Testbed", Chaoyi Bian, Zhenkai Zhu, Alexander Afanasyev, Ersin Uzun, and Lixia Zhang. NDN, Technical Report NDN-0009.

[6] "An Endorsement-based Key Management System for Decentralized NDN Chat Application", Yingdi Yu, Alexander Afanasyev, Zhenkai Zhu, and Lixia Zhang. NDN, Technical Report NDN-0023.

3.5 Fundamental theory for NDN

Contributors	
PIs	Edmund Yeh (Northeastern)
Grad Students	Milad Mahdian, Fangxiang Wang, Ran Liu (Northeastern)
Undergrads	Kyle Dumont, Sean Kerr (Northeastern)
Staff	**Postdoc:** Ying Cui (Northeastern)

3.5.1 Objectives

The basic objective of the theory module of the NDN project is to start the development of a fundamental theory for NDN networks, which properly incorporates the role of caching. Since the approach of NDN to networking is fundamentally different from traditional connection-based approaches, certain basic assumptions must be reconsidered. As a first step, the relevant network performance metrics should be redefined. Instead of seeking to maximize source-destination communication rates, we focus on maximizing the total amount of information satisfied per unit time for the interest sources (nodes generating interest packets) within the network. As an initial estimate, this can be captured by the total consumed bit rate, i.e., the total bit rate arriving at nodes requesting content.

A theory for NDN must capture the essential tradeoff between wires and storage in optimizing communication performance. Given an appropriate performance metric, we pose the following fundamental questions as in classical information theory: for a given set of link capacities, storage capacities, interest and data generation rates, what are the achievable capacity region and complexity costs for feasible joint routing, forwarding, scheduling, caching and coding schemes? Initial NDN implementations yield achievable schemes corresponding to subsets of the capacity region. Our long-term goal is to characterize the entire region.

Guided by the performance of these algorithms and combining with appropriate performance bounds, we seek to characterize the macroscopic scaling laws governing NDN network capacity, in a manner similar to [1]. This provides a useful starting point for studying the scalability of the NDN architecture. Next, we will incorporate issues beyond network capacity, such as latency and fairness. Not all points in the network capacity region are equally desirable, and we will investigate network management policies which achieve a given set of performance objectives in the optimal manner. These optimal policies are expected to differ significantly from those established in the classical network theory literature.

3.5.2 Progress for Theory Activities

Joint Forwarding and Caching for NDN

We have continued progress in developing optimal dynamic forwarding and caching algorithms for maximizing total consumed bit rate in an NDN network. Assuming the prevalence of caches, we seek to optimally utilize both bandwidth and storage for efficient content distribution. This highlights the need for joint design of traffic engineering and caching strategies, in order to optimize network performance given current and future traffic demands. Unlike many existing works on centralized algorithms for static caching, our goal is to develop distributed, dynamic algorithms that can address caching and forwarding under changing content, user demands, and network conditions.

To address this fundamental problem, we have introduced the *VIP framework* for the design of high performing NDN networks [4]. The VIP framework relies on the new device of Virtual Interest Packets (VIPs), which captures the measured demand for the respective data objects in the network. The central idea of the VIP framework is to employ a *virtual* control plane which operates on VIPs, and an *actual* plane which handles IPs and DPs. Within the virtual plane, we develop control algorithms operating on VIPs, aimed at yielding desirable performance in terms of network metrics of concern. 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.

To illustrate the utility of the VIP framework, we have developed two instantiations of the framework, called Algorithms 1 and 2, which both use the VIP count as a metric for determining both the forwarding and caching algorithms. We have determined the VIP network stability region, the set of all IP arrival rates that can be satisfied by some feasible forwarding and caching policy in an NDN network. We have proved that the forwarding/caching policy in Algorithm 1 is throughput optimal, in the sense of adaptively maximizing the VIP throughput, and therefore the user demand rate satisfied by an NDN network. We have shown that Algorithm 2 achieves not only load balancing but also stable caching configurations. We have run numerical experiments to compare the joint caching-forwarding VIP algorithm (Algorithm 2) against several baseline routing and caching policies. 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 [4].

Throughput and Delay Scaling for NDN Wireless Networks

We have completed our investigation of throughput and delay scaling laws for information-centric wireless networks with nodes are uniformly distributed at random in the network area. Each node has a limited-capacity Content Store, which it uses to cache contents. We considered a content-centric traffic model with a general content popularity distribution, where users leverage multihop communication to retrieve the requested content from the closest cache. We derived the throughput-delay tradeoff of the proposed network paradigm for a general content popularity distribution, and solved the problem of joint optimization of caching and forwarding strategies. We evaluated network performance for a Zipf content popularity distribution, while letting the number of content types and network size both go to infinity. We considered contents with different sizes and hybrid network scenarios, and verified our theoretical results through extensive simulations [2].

Fair Congestion Control for NDN

Finally, we investigated the problem of NDN congestion control using the VIP framework [3]. When IP (VIP) arrival rates are outside the VIP stability region, in order to stabilize the VIP network, a controller limits the number of VIPs (and IPs) admitted into the network layer. Newly arriving IPs first enter transport layer storage reservoirs before being admitted to the network layer. The goal of congestion control is to admit a portion of the VIPs to achieve a given fairness criterion, which may be realized by choosing the admitted VIPs to maximize a sum of utility functions which are increasing and concave in the admitted VIP rate. Unlike traditional network settings, the utilities (fairness) are associated with content objects rather than source-destination pairs. We have developed a joint congestion control, forwarding and caching algorithm that yields a VIP throughput vector which can be arbitrarily chose to the optimal solution of the utility maximization problem, while keeping all VIP queues stable. We have also developed fair congestion control schemes to obtain a tradeoff between the utility gained by admitting more demand into the network layer, and the incurred average network delay [3]

References

[1] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Trans. on Information Theory*, 46(2):388–404, Mar. 2000.

[2] Milad Mahdian and Edmund Yeh. Asymptotic behavior of wireless networks with named data networking architecture. Technical report, Northeastern University, 8 2014.

[3] Edmund Yeh, Ying Cui, and Ran Liu. Joint congestion control, forwarding, and caching in named data networks. Technical report, Northeastern University, 9 2014.

[4] Edmund Yeh, Tracey Ho, Ying Cui, Michael Burd, Ran Liu, and Derek Leong. Vip: A framework for joint dynamic forwarding and caching in named data networks. In *Proceedings of ACM Conference on Information-Centric Networking (ICN)*, 9 2014.

Chapter 4

NFD Development and NDN Testbed

Contributors

PIs	Beichuan Zhang (Arizona), Van Jacobson & Lixia Zhang (UCLA), Lan Wang (Memphis), Christos Papadopoulos (Colorad State University), Patrick Crowley (Washington University)
Grad Students . .	Junxiao Shi, Jerald Abraham, Yi Huang (Arizona); Ilya Moiseenko, Yingdi Yu, Wentao Shang (UCLA), Steve DiBenedetto, Chengyu Fan (Colorado State), Haowei Yuan, Hila Ben Abraham (Washington Univerisity)
Undergrads	Vince S. Lehman (Memphis)
Staff	John DeHart, Jyoti Parwatikar (Washington University)
	Postdoc: Alex Afanasyev (UCLA); Syed Obaid Amin (Memphis)

Our major testbed accomplishment this year was deploying our new NDN Forwarding Daemon (NFD) on the NDN testbed. In previous years we used PARC's CCNx package for both network layer research and application prototyping, complemented by our ndnSIM simulator. As the NDN research advanced, it became clear that the CCNx package was impeding progress, due to the difficulties in understanding and modifying the codebase. We developed a new NDN packet format and a new NDN forwarder that can meets NDN research needs more effectively.

First, our research has shown that a Type-Length-Value (TLV) based NDN packet format is superior to CCNx's binary XML packet format. A TLV-based packet format can provide adequate flexibility to meet our research needs without overcomplicating packet processing. Second, the CCNx code does not provide enough modularity to support experimentation with alternative designs and new features, in particular different forwarding strategies. Third, as the global NDN community grows, a codebase that is open and free to all users and developers is essential to the sustainability of long-term NDN development.

Our goal is to develop NFD as an open and free software under GPL 3.0 license, which supports a new TLV packet format and prioritizes code modularity and extensibility to facilitate research. It is an ambitious goal considering the scope, the technical depth, and the sheer amount of work. We worked closely as a team over multiple conference calls per week for ten months, involving six PIs and dozens of students, postdocs, and staff members. We succeeded in releasing the first specification of the TLV-based packet format in late 2013, and NFD v0.2 implementation in August 2014. The NDN testbed is now running NFD natively and is running NFD-based routing protocols globally. While the packet format, NFD codebase, and testbed continue to evolve, they now form a solid foundation for future NDN development.

4.1 New Packet Format Design

In the past, our research and development efforts used a packet format specification that was originally defined as part of the CCNx project (`http://www.ccnx.org/releases/latest/doc/technical/index.html`). As our understanding of NDN design deepened through experimentation over the last three years, and based on inputs from the broader NDN research community, we realized the need for several important changes. In 2013, we started designing a new packet format [?] that uses TLV encoding instead of binary XML for efficient processing by network elements. This effort also gave us the opportunity to rethink the structure of the packets and reexamine various fields. The resulting Interest and Data packet fields are illustrated in Figure 4.1. Note that naming conventions and the use of special markers inside NameComponents are not part of the packet specification. In July 2014 we released an initial draft of naming conventions as an NDN technical memo [?] to propose and hopefully inspire discussion of an initial standardization for common use cases, e.g., segmentation, versioning, time stamping, and sequencing. Changes to the packet format include:

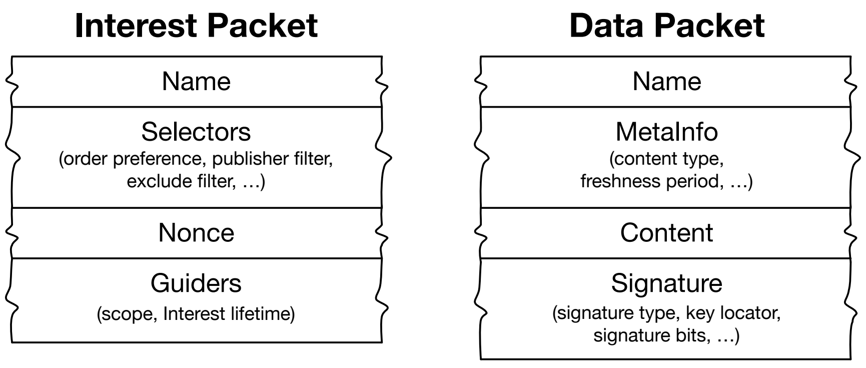

Interest Packet
Name
Selectors
(order preference, publisher filter, exclude filter, ...)
Nonce
Guiders
(scope, Interest lifetime)
Data Packet
Name
MetaInfo
(content type, freshness period, ...)
Content
Signature
(signature type, key locator, signature bits, ...)

Figure 4.1: New NDN Interest and Data packet structure

- Changes in Interest packet structure (Figure 4.1):
  - The `nonce` is now required, since it is needed to detect forwarding loops.
  - `PublisherPublicKeyDigest` is replaced by PublisherPublicKeyLocator, to accommodate either a key digest or a key name by which the key can be retrieved.
  - `AnswerOriginKind` is simplified from a 4-bit to a 1-bit MustBeFresh.
  - `FaceID` no longer exists.
  - `InterestLifetime` is measured in milliseconds
  - The `Exclude` selector no longer supports the Bloom Filter option.
  - A new default semantics of data staleness: NDN-TLV Interest with no selectors will bring any data that matches the name, and only when `MustBeFresh` selector is enabled will it retrieve fresh data. The FreshnessPeriod is a field in the Data packet and is set by the content producer. In the previous binary XML encoded Interest packet format, the default behavior was to bring "fresh" data and return "stale" data only when `AnswerOriginKind` is set to 3.
- Changes in Data packet structure (Figure 4.1):
  - `SignedInfo` has been renamed to MetaInfo and its content has changed.
  - `PublisherPublicKeyDigest` and ExtOpt have been removed.
  - `Timestamp` has been removed
  - `KeyLocator` is now inside the Signature (SignatureInfo) block
  - Three content types, ENCR, GONE, and NACK have been removed. They were listed in CCNx as placeholders for some potential usages, but we decided that they are either not needed (ENCR, GONE), or will required detailed specification (NACK) in some future version of the packet

format.
  - `FreshnessSeconds` has been renamed to `FreshnessPeriod` and is expressed in milliseconds
- New definition of Data packet signature
  - `Signature` block is now at the end of Data packet to facilitate packet processing.
  - `KeyLocator` has been moved to be part of the `SignatureInfo` block, if it is applicable for the specific signature type.
    The rationale for the move is to make Signature (a tuple of `SignatureInfo` and `SignatureValue` TLVs) self-contained and self-sufficient.
  - Signature type (or signing method information) is now expressed as an assigned integer value (with no assumed default), rather than OID.
  - Support for hash-only "signature", which can be used when the producer has a slow CPU or limited power, or for applications that put stronger signatures inside the Data payload.
  - New signature type for Elliptic Curve Digital Signature Algorithm (ECDSA) has been added.

The NDN team has fully adopted the new packet format, and all librariers and NFD support it. We devised a transition mechanism, used from January 2014 to July 2014, when we had initial support for the new packet format on the NDN Testbed with use of the `ndnd-tlv` package [?]. This package provides translation between the binary XML format and the TLV format, allowing applications to experiment with the new format and features, while still interoperating with legacy applications (Figure 4.2). It was used on the NDN Testbed from January 2014 to July 2014 and played an important role in helping applications migrate to the TLV packet format.

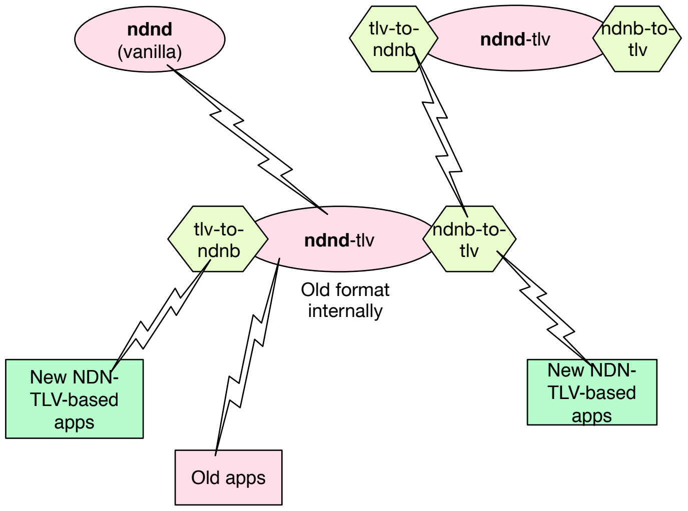


Figure 4.2: ndnd-tlv package and experimentation opportunities

4.2 New NDN Forwarder: NFD

4.2.1 Development Overview

NFD (`http://named-data.net/doc/NFD`) is a network forwarder that implements and evolves together with the NDN protocol. NFD is an open and free software package licensed under GPL 3.0 license and is the centerpiece of our commitment to making NDN's core technology open and free to all Internet users and developers. For more information about the licensing details and limitation, refer to (`https://github.com/named-data/NFD/blob/master/COPYING.md`).

The primary objective of NFD is to facilitate community research. As our understanding of the NDN architecture matured over the course of the last few years, we needed an platform that implements the latest architectural components to support easy experimentation with new protocol features, algorithms, and applications. Therefore, the NFD design emphasizes code *modularity* and *extensibility*.

To support the broader community in experimenting with the NDN architecture as well as in contributing the NDN development, we adopted a common open-source project model for NFD development. We set up and used Redmine for issue tracking, Jenkins for code review, and Travis-CI for automated testing. Bug reporting, feature discussions, code reviews, unit tests and integration tests are all part of the development pipeline. The NFD development team involved 6 PIs, a dozen students, and a few postdocs and staff members. We also involved our collaborators from University Pierre & Marie Curie, Sorbonne University and Beijing Institute of Technology to contribute code to NFD. All participating students gained tremendously from the process in making system design tradeoffs, learning large system development skills and effective remote collaborations.

NFD v0.1 was released internally in spring 2014 for testing and development purposes. The first public release, NFD v0.2, was in August 2014. It is now part of the NDN platform and we will aim for the same release cycle, i.e., every 3 months. To simplify and promote the adoption of NFD and NDN technology in general, in addition to providing full access to the source code, we also provide official binary packages for a set of supported platforms, including Ubuntu Linux 12.04, Ubuntu Linux 14.04, and Mac OSX with MacPorts. NFD is known to run on other platforms including RedHat, Gentoo, FreeBSD, Raspberry Pi, and OpenWRT/DD-WRT. The public release of the forwarder and other related tools was accompanied by the release of a set of extensive documentation on NFD's homepage (`http://named-data.net/doc/NFD/current/`). These documents provide detailed explanation on how to compile, configure, use, monitor, and debug NFD. In particular, to help developers improve NFD and extend it for their own research, we wrote the "NFD Developer's Guide" (Technical Report NDN-0021 [**?**]) which explains NFD's internals including the overall design, major modules, their implementations, and their interactions.

After its public release, we expect NFD will attract usage and contributions from the broader community. To support the NDN-NP project, we intend to continue NFD development to keep pace with the protocol specification, while maintaining a modular, stable, and lean framework to facilitate researchers experimentation with new features, some of which may make their way into future version of the protocol specification. We will leverage the software development system and our project management experience in future NFD and related NDN software development.

4.2.2 Major Components and Functionality

The main functionality of NFD is to forward Interest and Data packets: it abstracts lower-level network transport mechanisms into NDN Faces, maintains basic data structures such as the Content Store (CS), PIT, and FIB, and implements the packet processing logic. NFD supports multiple forwarding strategies, and implements a management interface for applications to configure, control, and monitor NFD. Figure 4.3 illustrates the major components of NFD:

- The **Face System** is separated into three logical abstractions: protocol factories, channels, and faces. A **protocol factory** creates channels or faces of specific protocols, e.g., the TCP protocol factory creates TCP faces. A **channel** represents an NFD-side endpoint for unicast communications (i.e., "listening" socket or socket from which connection will be established). NFD refers to these endpoints using the concept of FaceURI, which defines protocol and protocol-specific parameters of the endpoint. A **Face** is an abstraction which implements communication primitives to send and receive Interest and Data packets. Depending on the nature of communication, a Face can have different properties, such as local vs. remote, unicast vs. multicast. The implementation is heavily based on the Boost.Asio library and uses asynchronous operations to avoid blocking the rest of the daemon while performing potentially lengthy network operations. The current release supports the following underlying platforms: TCP, UDP, Unix Socket, raw Ethernet, and WebSocket. Thanks to its modular design, it is easy to add new types of faces into NFD.

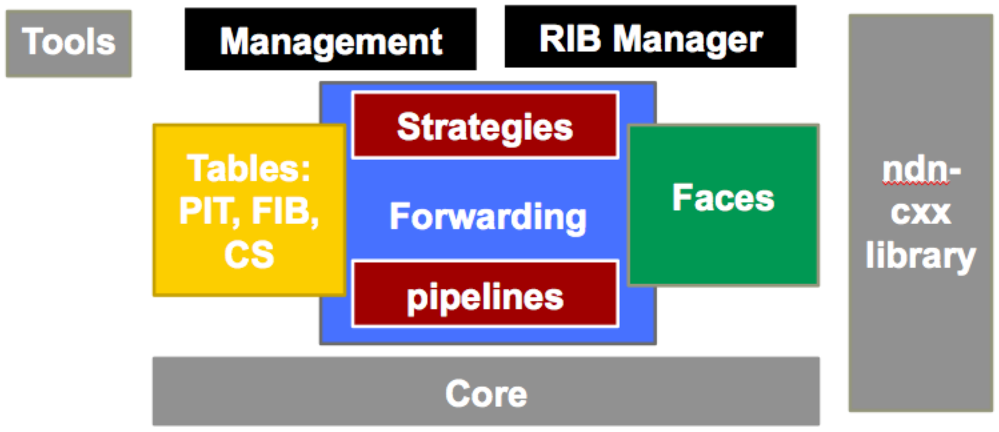


Figure 4.3: Overview of NFD modules

- **Tables**
  NFD implements five tables to store information. The Forwarding Information Base (FIB) is the name-based forwarding table, which is consulted in order to forward an Interest. Its content can be manually configured and/or managed by routing protocols. The Content Store (CS) caches passing Data packets opportunistically. It currently supports a simple cache replacement policy. The Pending Interest Table (PIT) records Interests that have not been satisfied by returned Data or timed out yet. It also helps detect loops and facilitate strategy decisions. Compared with CCNx, NFD introduces two new tables to support namespace-specific strategies. The StrategyChoice Table records the forwarding strategy chosen for a namespace, while the Measurements Table stores past performance results to inform forwarding decisions. FIB, PIT, StrategyChoice, and Measurements have similar index structures. To improve performance and reduce memory usage, a common index called NameTree can be shared among these four tables. NameTree defines a set of common APIs, which can be implemented by different data structures and algorithms, making it easy to experiment with different designs.
- **Forwarding**
  Packet processing in NFD has two dimensions: **forwarding pipelines** and **forwarding strategies**. Temporally, packet processing is broken into a series of steps called pipelines, shared by all strategies. Between pipelines, a router may invoke a namespace-specific strategy to make forwarding decisions before entering the next pipeline. For example, when a router receives a new Interest, it pushes the Internet into the *onIncomingInterest* pipeline, which will detect loops, consult the PIT/CS/FIB etc. Depending on the lookup result, the router may invoke the *onOutgoingData* pipeline for a CS match, or a forwarding strategy for a FIB match, etc. The strategy can be set for each namespace by applications, and it can use the Measurements table to store strategy-specific state information. Therefore by breaking packet processing into small pipelines and placing strategy hooks in between, we enabled the support of per-namespace strategy. The current release supports the following strategies: broadcast, best-route, client control, and the strategy used in ccnx. We plan to make strategy support more flexible and add more strategies in the next step.
- **Management**
  The NFD Management Protocol defines an Interest/Data API to control, configure, and query NFD. In particular, users can:
  - create, destroy Faces, and enable/disable local control features on local faces (Face Manager)
  - add and remove FIB entries (FIB Manager)
  - manipulate selection of the forwarding strategy for namespaces (Strategy Choice Manager)
  - retrieve status and traffic statistics

  Each manager is an interface for some part of the lower layers of NFD. Adding a new manager is a fairly straightforward task; one only needs to determine what part(s) of NFD should be exported to an

Interest/Data API and create an appropriate command Interest interpreter. All management actions that change NFD state require the use of *control commands*, a form of signed Interests. Management actions that just query the current state of NFD do not need to be authenticated. In the future if data access control is desired, some data can be encrypted.

- **RIB Management**
  Different parties may update the RIB in different ways, including various routing protocols, application prefix registrations, and command-line manipulation by sysadmins. The RIB management module processes all these requests to generate a consistent forwarding table, and syncs it with NFD's FIB, which contains only the minimal information needed for forwarding decisions. Strictly speaking RIB management is part of the NFD management module, but due to its operational importance and more complex processing of routing flags, we implement it as a separate module.
- **Tools**
  We implemented a set of tools for managing and experimenting with NFD. These tools include *nfd-start/nfd-stop* scripts, *nfdc* to manipulate NFD states at run-time via the management interface, *nfd-status* and *nfd-status-http-server* to retrieve NFD status and publish it over HTTP, *autoconfig* to automatically detect and connect to an NDN gateway, *ndnping* to test connectivity, and *ndn-traffic-generator* to generate traffic with different characteristics for testing purposes. These tools played an important role in testing and deploying NFD.
- **Core**
  This module provides some common services within NFD. They include hash functions, logging facility with different log levels, configuration file processing, and DNS resolution.
- **NDN-CXX Library**
  NFD and its tools uses the NDN-CXX library.

4.3 NDN Testbed: Deployment, Management, Expansion

The past year has seen substantial changes and progress in the nature and organization of the NDN testbed. Most significantly, the testbed has been transitioned from using CCNx to NFD as the underlying forwarding substrate. The NFD development effort was ambitious but triumphant, illustrated by the NDN testbed, which now runs NFD natively and NFD-based routing protocols globally.

The Testbed has also transitioned from an internal testbed for the NDN project to an open testbed for the broader community. In 2014 we formalized the procedure for a site to join the testbed and opened it up to interested researchers. Currently, there are 9 gateway nodes at NDN PI sites, and 7 collaborating sites, including 3 in Asia and 2 in Europe. All nodes in the NDN testbed are operated and managed by the testbed NOC at Washington University which operates, manages, tests, and upgrades all testbed nodes. NDN teams at various campuses developed various supporting software including NFD, NLSR, the traffic generator, video streaming application, and other monitoring tools.

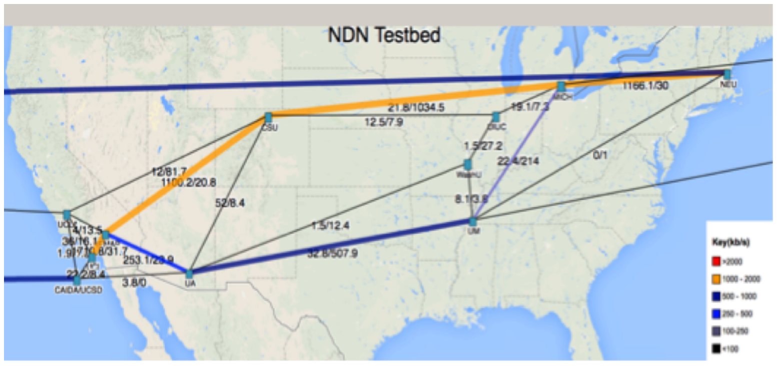


Figure 4.4: The NDN Map reflects the new NFD-based global testbed.

Two developments from Washington University enabled the NDN transition and its monitoring. First, the Open Network Lab (ONL) enabled a network-scale integration test of the entire testbed before, during and after the testbed transition to NFD-

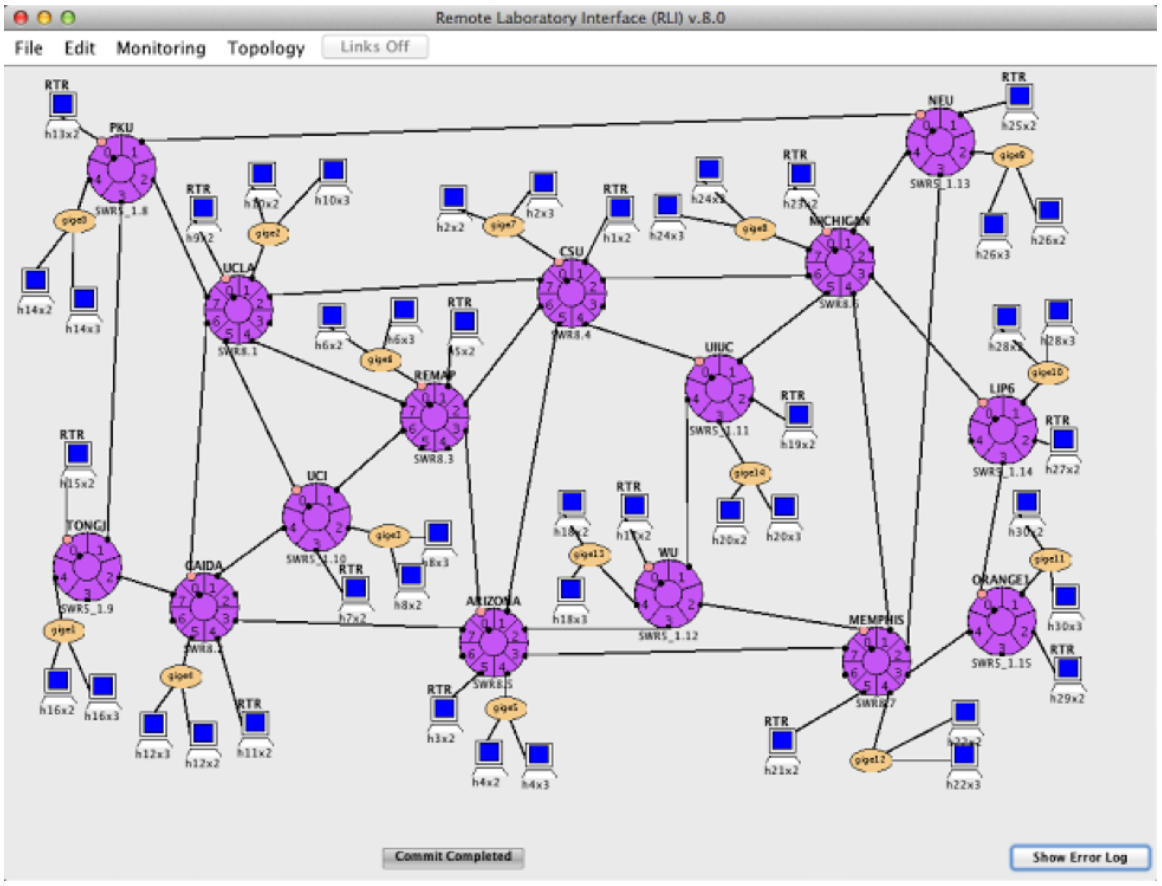


Figure 4.5: The NDN Testbed topology in ONL enables integration testing prior to testbed deployment.

based nodes. ONL was able to shake out some bugs in routing protocols that could only be found with a large-scale deployment, as well as bugs in NFD during stress tests. Figure 4.5 shows the graphical-user interface displaying the testbed in ONL.

Second, the NDN map tool originally designed for CCNx has been redesigned and implemented to work with NFD-based nodes. The NDN map makes it easy for anyone to see both the organization and the real-time usage of the global NDN testbed. It is helpful as a visual aid during wide-area network demonstrations of NDN. Figure 4.4 includes a view drawn from http://ndnmap.arl.wustl.edu.

Chapter 5

Impact: Education

Contributors

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

5.1 Education Philosophy and Objectives

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

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

5.2 Biweekly NDN Seminars

During this reporting period we continued our biweekly NDN seminar series among participating universities. Shiguang Wang, a UIUC graduate student and A K M Mahmudul Hoque, a Memphis graduate student, collected topics and created the schedule. Seminar covered the following topics:

- 2/24/14: Ilya Moiseenko, Consumer-Producer API for Named Data Networking
- 3/17/14: Peter Gusev, NDN Real Time Conferencing Library
- 4/14/14: Jeff Burke, NP proposal "network environments" (Open mHealth and Enterprise Building Automation, mobile media application cluster)
- 6/6/14: Jeff Burke, Introduction to "Enterprise Building Automation and Management network" environment
- 6/30/14: Jeff Burke, Enterprise Building Automation and Management network

5.3 Education Efforts

This section describes the education efforts carried out during the reporting period by NDN team members at various campuses.

University of Arizona During Year 4 Beichuan Zhang took sabbatical leave and visited Tsinghua University. His class material for CS 525, however, was still used in the Fall 2014 semester at the University of Arizona. It has 2 lectures that cover NDN's basic concepts, operations, research issues, and progresses. The course also contains material on other future internet designs and other ICN designs to provide a more comprehensive view of the research area. While at Tsinghua University, Beichuan Zhang developed a new advanced course that focuses on NDN. The course is titled Internet Routing Architectures and Protocols: A Comparative Study of IP and NDN. Half of the course is lectures, covering IP architecture and major routing protocols and content distribution solutions, NDN architecture and its approaches to routing, mobility, security and content distribution. The other half of the course is class discussions, in which students compare IP and NDN solutions to various problems and debate their pros and cons. At the end of the course each student submits a report on a self-selected NDN-related research topic. Ten students formally registered the course and four other students audited the course. The approach of teaching IP and NDN at the same time while making comparison seems to be effective. The submitted class reports reflected that most students gained good understanding of NDN architecture with good technical depth. This course will be further developed at the University of Arizona into an advanced graduate course.

Colorado State: CSU continued to teach NDN and assign NDN-related programming assignments in both the graduate and undergraduate networking courses. PI Papadopoulos taught both classes in Fall 2013 and Spring 2014. As a demonstration of how NDN makes applicatipon development easy, the graduate class assigned two projects with identical specs: an implementation a simple version of BitTorrent; one, however, was implemented using IP and the other using NDN. The students were asked to comment on the difficulty of the implementations. The difference was startling, with students finding it far easier and more natural to implement the project in NDN.

CSU is currently working on updating the old GENI exercises to use the new NDN codebase. In the process we will make the exercises richer, demonstrating more features of NDN. Finally, CSU produced a few screencasts to demonstrate important features of NDN, which are described separately below.

UCLA: Lixia Zhang taught a graduate course "CS217A: Internet Architecture & Protocols" during Winter 2014 quarter, which covered both today's Internet TCP/IP architecture as well as a brief introduction to Named Data Networking (NDN), which was covered in three lectures. In addition, the students were also encouraged to take optional term projects on several NDN research topics, including:

- Installation of NDN NFD on Raspberry Pie to enable exploration of home IoT.
- Simulation experimentation with a new mobility support approach: Interest forwarding via NDN's Pending Interest Table (PIT).
- Dynamic web publishing
- Implementation of a graph-database based NDN repository
- Design of a new naming structure to facilitate data discovery from k-nearest neighbor nodes.
- Implementation of NDNS (NDN-DNS)

Either a faculty member or a graduate student from the NDN team served as the contact person for each topic. The first topic led to a master project with a successful demonstration. The work started with the second topic eventually led to a paper submission to the first ACM ICN conference. An undergraduate student took on the third topic and he has been involved in NDN research ever since. Another undergraduate student from Zhang's undergraduate teaching also got involved in NDN research by helping with the NDNS implementation and the development of an NDN wikipedia page.

CS217A had about 30 enrolled students, however over 50 people attended three NDN introductory lectures, including graduate students, visiting students and scholars. During the 2014-2015 academic year, Lixia Zhang will teach both CS217A in fall 2014, hoping to exceed the success from last time, and CS217B on "Advanced Topics in Internet Research" in spring 2015, a graduate seminar course that to NDN architecture research and NDN application development since 2010.

NDN/CCN Tutorial Transcript: We transcribed, edited, and published the text of a 3+ hour tutorial given by Van Jacobson on Content-Centric Networking for the Future Internet Summer School (FISS 2009), hosted by the University of Bremen in Germany. This 35-page approximate transcript is a goldmine for understanding the deep motivations and implications of the NDN architecture, and is available online along with the video at `http://named-data.net/2014/04/15/ndn-humans/`. We have referred students and others wishing to engage deeply with research on the architecture to this new reference.

Memphis: Lan Wang gave a lecture on NDN in COMP4410/6410 (Computer Security) to explain how provenance will be a built-in feature in the new Internet architecture.

5.4 Educational Screencasts

We developed a number of screencasts demonstrating several features of NDN such as data discovery, distributed publication, exclusion, retrieval, enumeration of a name prefix, and automatic failover. This screencasts show how NDN works and provide a visual experience for new users. Moreover, they capture features that will benefit other research areas, such as climate research. These screencasts have been displayed at several NDN workshops and meetings. The videos can be found here - `http://www.cs.colostate.edu/~susmit/ndn_screencasts/`.

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

Impact: Expansion of NDN Community

The NDN project continues to attract attention from the global networking community. The PIs have participated in numerous conferences and speaking engagements, as listed below, and engaged a variety of interns and visiting researchers from university and industry. These formal and informal efforts have helped to disseminate the research results and core ideas of the project, as well as practical information about the NDN codebase, to the community.

6.1 First NDN Community meeting

The team organized and executed the first NDN Community Meeting (held September 3-5, 2014 at UCLA), which brought over 80 researchers from academia and industry together to discuss the architecture, future research, and important applications. Attendees included students, faculty, and staff from the NDN campuses, as well as other universities conducting NDN research and industry including Ericsson, Cisco, Huawei, Panasonic Research, PARC, Intel, and others.

The **full agenda** can be found on the Named Data Networking First Community Meeting website, `http://www.caida.org/workshops/ndn/1409/`.

Figure 6.1: Professor Lan Wang (University of Memphis) speaking at NDNComm 2014.

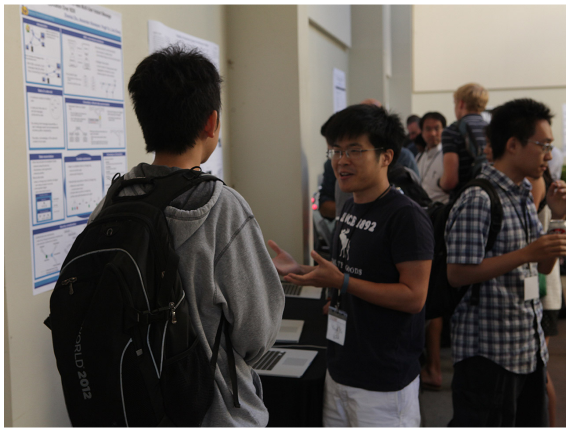


Figure 6.2: UCLA Graduate Student Yingdi Yu explaining his poster at NDNComm 2014.

6.2 Establishment of NDN Consortium

In a significant milestone for NDN, the project team launched the NDN Consortium to promote a vibrant open source ecosystem of research and experimentation around NDN by providing developer support tools, organizing community meetings, generating outreach activities, and hosting working groups for both industry verticals and cross-cutting activities.

In addition to the eight NDN participating campuses which made the founding members of the consortium, seven universities and six companies signed up with the first three weeks of the consortium announcement. They include(to date)

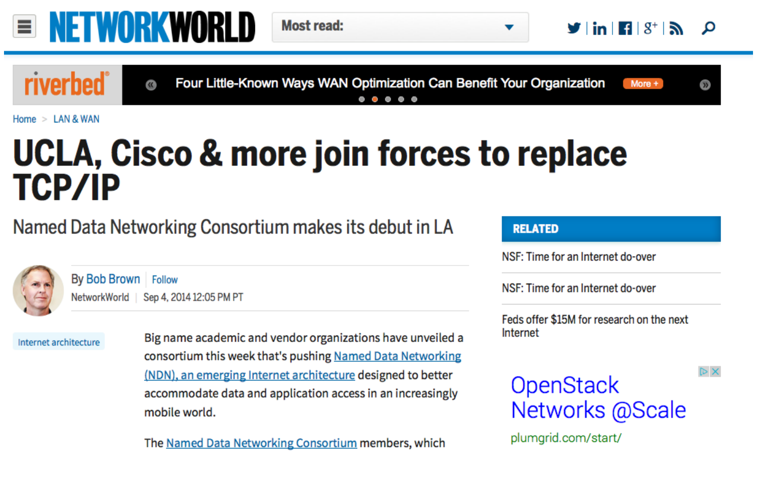
- Anyang University, Korea
- Tongji University, China
- Tsinghua University, China
- University of Basel, Switzerland;
- Pierre-and-Marie-Curie University, France
- Waseda University, Japan.
- Alcatel-Lucent
- Cisco Systems
- Huawei Technologies
- The MITRE Corporation
- Panasonic Corporation (in progress)
- Verisign, Inc.

We expect that the consortium and events like the community meeting will build engagement and participation in the research, bring new ideas and applications into the mix, and work towards and open and accessible core architecture per our original Intellectual Property Statement.

More information on the **Named Data Networking Consortium** can be found at `http://named-data.net/consortium`.

6.3 The First ACM Information Centric Networking Conference

After three successful ACM SIGCOMM Information Centric Networking Workshops (ICN 2011-2013) and two IEEE INFOCOM workshops on Emerging Design Choices in Name-Oriented Networking (NOMEN 2012-2013), where NDN team members played important roles in organizing the activities, ACM launched the first Information Centric Networking Conference to be held in September 2014 in Paris, France. Most NDN PIs served on the Technical Program Committee for ACM ICN 2014. Lixia Zhang served as TPC co-chair. The NDN team will present 2 papers (of 17), 3 posters (of 8), and a demo at the conference. We will also present a half-day tutorial on "An Introduction to NDN and its Software Architecture", a joint effort by NDN PIs, supporting staff member, postdoc,



UCLA, Cisco & more join forces to replace TCP/IP

Named Data Networking Consortium makes its debut in LA

By Bob Brown | Follow
NetworkWorld | Sep 4, 2014 12:05 PM PT

Big name academic and vendor organizations have unveiled a consortium this week that's pushing Named Data Networking (NDN), an emerging Internet architecture designed to better accommodate data and application access in an increasingly mobile world.

The Named Data Networking Consortium members, which

NSF: Time for an Internet do-over

NSF: Time for an Internet do-over

Feds offer $15M for research on the next Internet

Figure 6.3: Network World article on consortium, Sept 4, 2014.

and graduate student. The high number of ICN 2014 submissions and registered attendees already led to ACM's decision to sponsor the next ICN conference in San Francisco, California in September 2015, a strong indication of the growing popularity of information centric networking research.

6.4 Reaching Out: NDN Presentations

Lan Wang and Beichuan Zhang participated in INFOCOM 2014 Panel on "Whether NDN: Doubts, Tough Questions, Progress, and Challenges". They presented NDN architecture design principles and progress.

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

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

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

Christos Papadopoulos gave invited talks at about seven Internet 2 conferences, climate workshops and scientist groups. These talks introduced NDN and outlined how the new architecture significantly simplifies Big Data applications in such domains. Examples include translation from existing, ad-hoc namespaces to NDN hierarchical namespaces, distributed publishing for location independence, data discovery and enumeration of all datasets under a common prefix, and instant failover through multiple NDN repositories.

Presentations

1. Tarek Abdelzaher, Invited Talk, Syracuse University, "Social Sensing: Making Reliable Observations from Unreliable Data," Syracuse, NY, October 2013.
2. Tarek Abdelzager, Panelist, 1st International Workshop on Sensing and Big Data Mining, "Bridging Big Data and Sensing: What is Missing?" Rome, Italy, November 2013.
3. Tarek Abdelzaher, Invited Speaker, Global Innovation Festival, Daegu Gyeongbuk Institute of Science and Technology (DGIST), "Data Analytics for Human-centric Cyber-physical Systems," Daegu, Korea, November 2013.
4. Tarek Abdelzaher, Award Talk, IEEE Real-time Systems Symposium, "Cyber-physical Systems in Social Spaces: A Data Reliability Perspective," Vancouver, Canada, December 2013. Note: The IEEE RTSS Award Talk is an invited talk given by the recipient of the IEEE Outstanding Technical Achievement and Leadership Award in Real-time Systems, the year following the award.
5. Tarek Abdelzaher, Invited Talk, Hong Kong Polytechnic University, "Social Sensing: Making Reliable Observations from Unreliable Data," Hong Kong, December 2013.
6. Tarek Abdelzaher, Invited Talk, UIUC (Department of Agricultural and Biological Engineering), "Social Sensing: Making Reliable Observations from Unreliable Data," Urbana, IL, January 2013.
7. Tarek Abdelzaher, Keynote Talk, Danish Academy of Technical Sciences (Big Data Seminar), "Social Sensing: Making Reliable Observations from Unreliable Data," Keynote Talk, Aalborg, Denmark, February 2014.

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

8. Jeff Burke, keynote, IFIP Networking 2014, "Why architecture matters to everyone: Creativity on the Future Internet", Trondheim, Norway, June 3, 2014.
9. Jeff Burke, invited talk. "The Conditions of Algorithmic Life, Occams Hourglass, Mellon Research Initiative in Digital Cultures", UC Davis, May 15-16, 2014.
10. Jeff Burke, invited talk. Huawei Corporate-level Science & Technology Workshop, "Named Data Networking and the Internet of Everything", Huawei headquarters, Shenzen, China, May 13, 2014.
11. Jeff Burke, invited talk. Big Conference, "Big Data and the Big Network", University of California, Irvine, April 11, 2014.
12. Jeff Burke, invited talk. Huawei Science & Technology Workshop, "Named Data Networking and the Internet of Everything", Futurewei, Santa Clara, March 11, 2014.
13. Jeff Burke, invited talk. Woodbury University, "Stories to Systems to Architectures", January 13, 2014.
14. Jeff Burke, invited talk. Cinegrid, "Named Data Networking Video Streaming and Conferencing", December 10, 2013.
15. Jeff Burke. Packet Video Conference, co-chair, Information-Centeric Networking Special Session. Cisco, San Jose, December 12, 2013.
16. Jeff Burke, invited talk. National Speakers Conference, "The Next Generation of Storytelling: The Future of the Entertainment Industry in a Mobile World", October 25, 2013.
17. Christos Papadopoulos, Cathie Olchanowsky, Susmit Shannigraphi, Steve DiBenedetto, David Randall, Kelly Wittmeyer, Don Dazlich, and Mark Branson, invited talk, NSF CC-NIE PI Workshop, "Supporting Climate Applications over Named Data Networking," May 1, 2014 Arlington, VA.
18. Christos Papadopoulos, Cathie Olchanowsky, and David Randall, invited talk, Internet 2 Summit, "Supporting Climate Applications over Named Data Networking," April 7, 2014 Denver, CO.
19. Christos Papadopoulos, Cathie Olchanowsky, Susmit Shannigraphi, Steve DiBenedetto, David Randall, Kelly Wittmeyer, Don Dazlich, and Mark Branson, invited paper, LANMAN Workshop, "Supporting Climate Applications over Named Data Networking," May 23, 2014 Reno, NV.
20. Christos Papadopoulos, Cathie Olchanowsky, Susmit Shannigraphi, Steve DiBenedetto, David Randall, Kelly Wittmeyer, Don Dazlich, and Mark Branson, invited talk, FTW Workshop, "Supporting Climate Applications over Named Data Networking," July 16, 2014, Boulder, CO.
21. Christos Papadopoulos, Cathie Olchanowsky, Susmit Shannigraphi, Steve DiBenedetto, David Randall, Kelly Wittmeyer, Don Dazlich, and Mark Branson, invited talk, NDN Community Meeting, "Supporting Climate Applications over Named Data Networking," September 2014.
22. Christos Papadopoulos, Cathie Olchanowsky, Susmit Shannigraphi, Steve DiBenedetto, David Randall, Kelly Wittmeyer, Don Dazlich, and Mark Branson, invited talk, Cooperative Institure for Research in the Atmosphere (CIRA), "Supporting Climate Applications over Named Data Networking," September 9, 2014, Fort Collins, CO.
23. Beichuan Zhang, invited talk, "Adaptive Forwarding in Named Data Networking", University of Science and Technology, China, October 19, 2013.
24. Beichuan Zhang, invited talk, "Update on NDN Spec", ICNRG meeting IETF 88, November 2013.
25. Beichuan Zhang, invited talk, "Named Data Networking: Architecture and Challenges", The First t HotICN workshop, Beijing, December 7, 2013.
26. Beichuan Zhang, invited talk, "Named Data Networking", Beijing Institute of Technology, Dec. 2013.
27. Beichuan Zhang, invited talk, "Named Data Networking", The 3rd Future Network Development and Innovation Forum, December 17, 2013.
28. Beichuan Zhang, invited talk, "Named Data Networking", Xi An Jiaotong Un., China, June 2014.

Chapter 7

Publications

Listed below are our publications during Year 4 of the NDN project.

1. "On the Role of Routing in Named Data Networking", by Cheng Yi, Jerald Abraham, Alexander Afanasyev, Lan Wang, Beichuan Zhang, Lixia Zhang. 1st ACM Conference on Information-Centric Networking, Paris, France, September 2014.
2. "VIP: A Framework for Joint Dynamic Forwarding and Caching in Named Data Networks", by Edmund Yeh, Tracey Ho, Ying Cui, Michael Burd, Ran Liu, Derek Leong. 1st ACM Conference on Information-Centric Networking, Paris, France, September 2014.
3. "Consumer-Producer API for Named Data Networking" by Ilya Moiseenko and Lixia Zhang. Poster, 1st ACM Conference on Information-Centric Networking, Paris, France, September 2014.
4. "Kite: A Mobility Support Scheme for NDN" by Yu Zhang, Hongli Zhang, Lixia Zhang. Poster, 1st ACM Conference on Information-Centric Networking, Paris, France, September 2014.
5. "iSync: A High Performance and Scalable Data Synchronization Protocol for Named Data Networking" by Wenliang Fu, Hila Ben Abraham, Patrick Crowley. Poster, 1st ACM Conference on Information-Centric Networking, Paris, France, September 2014.
6. "Named Data Networking" by L. Zhang, A. Afanasyev, J. Burke, claffy, L. Wang, V. Jacobson, P. Crowley, C. Papadopoulos, B. Zhang. ACM SIGCOMM Computer Communication Review (CCR), July 2014.
7. "The Information Funnel: Exploiting Named Data for Information-maximizing Data Collection" by Shiguang Wang, Tarek Abdelzaher, Santhosh Gajendran, Ajith Herga, Sachin Kulkarni, Shen Li, Hengchang Liu, Chethan Suresh, Abhishek Sreenath, Hongwei Wang, William Dron, Alice Leung, Ramesh Govindan, John Hancock. In Proc. 10th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), Marina Del Rey, CA, May 2014.
8. "Scalable Pending Interest Table Design: From Principles to Practice" by Haowei Yuan and Patrick Crowley. INFOCOM 2014, Toronto, Canada, April-May 2014.
9. "VANET via Named Data Networking" by Giulio Grassi, Davide Pesavento, Giovanni Pau, Rama Vuyyuru, Ryuji Wakikawa, Lixia Zhang. IEEE INFOCOM 2014 Workshop on Name Oriented Mobility (NOM), Toronto, Canada, April-May 2014.
10. "Lets ChronoSync: Decentralized Dataset State Synchronization in Named Data Networking" by Zhenkai Zhu and Alexander Afanasyev. Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP 2013), Goettingen, Germany, October 2013.
11. "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.

12. “Performance measurement of the CCNx synchronization protocol” by Hila Ben Abraham and Patrick Crowley. In Proceedings of the 9th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), San Jose, CA, October 2013.

NDN Technical Reports

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

- “NDN Common Client Libraries” by Jeff Thompson and Jeff Burke. NDN Technical Report NDN-0024, Revision 1: September 5, 2014.
- “An Endorsement-based Key Management System for Decentralized NDN Chat Application” by Yingdi Yu, Alexander Afanasyev, Zhenkai Zhu, and Lixia Zhang. NDN Technical Report NDN-0023, Revision 1: July 22, 2014.
- “NDN Technical Memo: Naming Conventions” by NDN Project Team. NDN Technical Report NDN-0022, Revision 1: July 21, 2014.
- “NFD Developers Guide” by Alexander Afanasyev, Junxiao Shi, Beichuan Zhang, Lixia Zhang, Ilya Moiseenko, Yingdi Yu, Wentao Shang, Yi Huang, Jerald Paul Abraham, Steve DiBenedetto, Chengyu Fan, Christos Papadopoulos, Davide Pesavento, Giulio Grassi, Giovanni Pau, Hang Zhang, Tian Song, Haowei Yuan, Hila Ben Abraham, Patrick Crowley, Syed Obaid Amin, Vince Lehman, and Lan Wang. NDN Technical Report NDN-0021, Revision 1: July 1, 2014.
- “Kite: A Mobility Support Scheme for NDN” by Yu Zhang, Hongli Zhang, and Lixia Zhang. NDN Technical Report NDN-0020, Revision 1: June 3, 2014.
- “Named Data Networking” by Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. NDN Technical Report NDN-0019, Revision 1: 10 April 2014.
- “A World on NDN: Affordances & Implications of the Named Data Networking Future Internet Architecture” by Katie Shilton, Jeff Burke, kc claffy, Charles Duan, and Lixia Zhang. NDN Technical Report NDN-0018, Revision 1: 11 April 2014.
- “Consumer-Producer API for Named Data Networking” by Ilya Moiseenko and Lixia Zhang. NDN Technical Report NDN-0017, February 2014
- “On the Role of Routing in Named Data Networking” by Cheng Yi, Jerald Abraham, Alexander Afanasyev, Lan Wang, Beichuan Zhang, Lixia Zhang. NDN Technical Report NDN-0016, December 2013
- “NDNBlue: NDN over Bluetooth” by Arjun Attam, Ilya Moiseenko. NDN Technical Report NDN-0015, November 2013
- “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.
- "Scaling NDN Routing: Old Tale, New Design" by Alexander Afanasyev, Cheng Yi, Lan Wang, Beichuan Zhang, and Lixia Zhang. NDN Technical Report NDN-0004, Revision 1: July 18, 2013.
- "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.