NDN, Technical Report NDN-0019, 2014. http://named-data.net/techreports.html
Rev.2: June 8, 2014 [Rev.1 April 10, 2014]

Named Data Networking

Lixia Zhang
Alexander Afanasyev
Jeffrey Burke
Van Jacobson
UCLA

kc claffy
CAIDA
UC, San Diego

Patrick Crowley
Washington University,
St. Louis

Christos Papadopoulos
Colorado State Univ.

Lan Wang
Univ. of Memphis

Beichuan Zhang
Univ. of Arizona

ABSTRACT

Named Data Networking (NDN) is one of five projects funded by the U.S. National Science Foundation under its Future Internet Architecture Program. NDN has its roots in an earlier project, Content-Centric Networking (CCN), which Van Jacobson first publicly presented in 2006.[1] The NDN project investigates Jacobson's proposed evolution from today's host-centric network architecture (IP) to a *data-centric* network architecture (NDN). This conceptually simple shift has far-reaching implications for how we design, develop, deploy, and use networks and applications. We describe the motivation and vision of this new architecture, and its basic components and operations. We also provide a snapshot of its current design, development status, and research challenges. More information about the project, including prototype implementations, publications, and annual reports, is available on named-data.net.

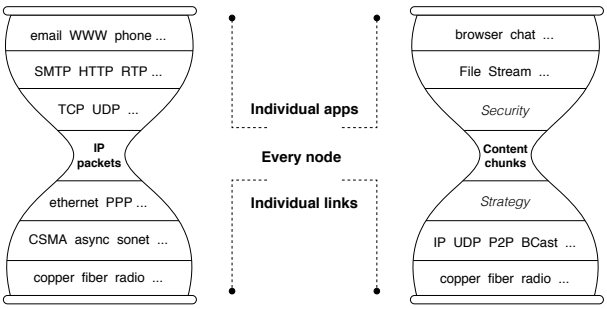

Figure 1: The main building blocks of the NDN architecture are named content chunks, in contrast to the IP architecture's fundamental unit of communication, which is an end-to-end channel between two end endpoints identified by IP addresses.

1. VISION: A NEW NARROW WAIST

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

The Named Data Networking (NDN) project proposed an evolution of the IP architecture that generalizes the role of this thin waist, such that packets can name objects other than communication endpoints (Figure 1). More specifically, NDN changes the semantics of network service from *delivering the packet to a given destination address* to *fetching data identified by a given name*. The name in an NDN packet can name anything – an endpoint, a data chunk in a movie or a book, a command to turn on some lights, *etc.* This conceptually simple change allows NDN networks to use almost all of the Internet's well-tested engineering properties to solve a much broader range of problems including not only end-to-end communications but also content distribution and control problems. Based on three decades of experience with the strengths and limitations of the current Internet architecture, the design also builds in security primitives (via signatures on all named data) and self-regulation of network traffic (via flow balance between Interest and Data packets). The architecture includes functionality designed to be conducive to user choice and competition as the network evolves, such as multipath forwarding and in-network storage.

NDN is one instance of a more general network research direction called *information-centric networking* (ICN), under which different architecture designs have emerged [29]. The Internet Research Task Force (IRTF) established an ICN research working group in 2012[2]. In this paper we provide a brief (and necessarily incomplete) snapshot of the current state of the NDN architecture research project, which includes sixteen NSF-funded principal investigators at twelve campuses, and growing interest from the academic and industrial research communities. A more complete description of recent activities is in the third annual project report [20] and on the NDN web site (named-data.net).

[1] "A New Way to Look at Networking", `https://www.youtube.com/watch?v=oCZMoY3q2uM`

[2] `http://trac.tools.ietf.org/group/irtf/trac/wiki/icnrg`

2. NDN ARCHITECTURE

Communication in NDN is driven by receivers i.e., data consumers, through the exchange of two types of packets: *Interest* and *Data*. Both types of packets carry a name that identifies a piece of data that can be transmitted in one Data packet. A consumer puts the name of a desired piece of data into an *Interest* packet and sends it to the network. Routers use this name to forward the Interest toward the data producer(s). Once the Interest reaches a node that has the requested data, the node 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 2). This Data packet follows in reverse the path taken by the Interest to get back to the requesting consumer.

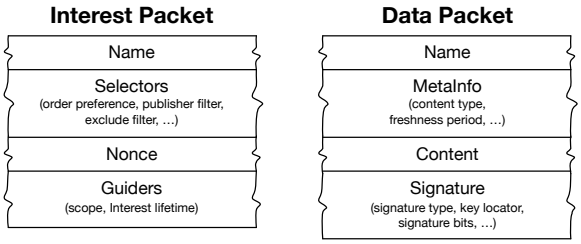

Figure 2: Packets in the NDN Architecture.

To carry out the Interest and Data packet forwarding functions, each NDN router maintains three data structures: a *Pending Interest Table (PIT)*, a *Forwarding Information Base (FIB)*, and a *Content Store (CS)* (Figure 3), as well as a *Forwarding Strategy* module (not shown in the figure) that determines whether, when and where to forward each Interest packet. The PIT stores all the Interests that a router has forwarded but not satisfied yet. Each PIT entry records the data name carried in the Internet, together with its incoming and outgoing interface(s). When an Interest packet arrives, an NDN router first checks the Content Store for matching data; if it exists the router returns the Data packet on the interface from which the Interest came. Otherwise the router looks up the name in its PIT, and if a matching entry exists, it simply records the incoming interface of this Interest in the PIT entry. In the absence of a matching PIT entry, the router will forward the Interest toward the data producer(s) based on information in the FIB as well as the router's adaptive Forwarding Strategy. When a router receives Interests for the same name from multiple downstream nodes, it forwards only the first one upstream toward the data producer(s). The FIB itself is populated by a name-prefix based routing protocol, and can have multiple output interfaces for each prefix.

The Forwarding Strategy may decide to drop an Interest in certain situations, e.g., if all upstream links are congested or the Interest is suspected to be part of a DoS attack. For each Interest, the Forwarding Strategy retrieves the longest-prefix matched entry from the *FIB*, and decides when and where to forward the Interest.[3] The Content Store is a temporary cache of Data packets the router has received. Because an NDN Data packet is meaningful independent of

[3]While an IP router may be able to reach a network prefix via multiple interfaces, it uses only one except in special cases where multiple best paths have identical cost.

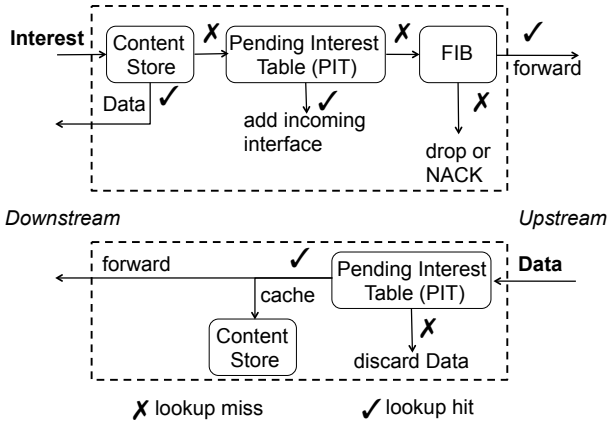

Figure 3: Forwarding Process at an NDN Node.

where it comes from or where it is forwarded, it can be cached to satisfy future Interests.

When a Data packet arrives, an NDN router finds the matching PIT entry and forwards the data to all downstream interfaces listed in that 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, in the absence of packet losses, one Interest packet results in one Data packet on each link, providing *flow balance*. To fetch large content objects that comprise multiple packets, Interests provide a similar role in controlling traffic flow as TCP ACKs in today's Internet: a fine-grained feedback loop controlled by the consumer of the data (see Section 2.1). Neither Interest nor Data packets carry any host or interface addresses; routers forward Interest packets toward data producers based on the names carried in the packets, and forward Data packets to consumers based on the PIT state information set up by the Interests at each hop. This Interest/Data packet exchange symmetry induces a hop-by-hop control loop (not to be confused with symmetric routing, or with routing at all!), and eliminates the need for any notion of source or destination nodes in data delivery, unlike in IP's end-to-end packet delivery model.

2.1 Names

Although routers recognize boundaries between components in a name, they attribute no meaning to names, *i.e.*, NDN names are *opaque* to the network. This design decision allows each application to choose the naming scheme that fits its needs, and naming can thus evolve independently from the network. The NDN design assumes hierarchically structured names, *e.g.*, a video produced by UCLA may have the name `/ucla/videos/demo.mpg`, where '/' delineates name components in text representations, similar to URLs. This hierarchical structure allows applications to represent the context and relationships of data elements. For example, segment 3 of version 1 of a UCLA demo video might be named `/ucla/videos/demo.mpg/1/3`. It also allows name aggregation, e.g., `/ucla` could correspond to an autonomous system originating the video. Flat names can be accommodated as a special case, likely useful in local environments, however hierarchical namespaces are essential both in scaling the routing system and in providing necessary context for the data. (Even advocates of flat routing acknowledge that flat names scale by introducing some hierarchy [2].)

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

Data that may be retrieved globally must have *globally unique* names, but names used for local communications may require only local routing (or local broadcast) to find matching data. Individual data names can be meaningful in various scopes and contexts, ranging from "the light switch in this room" to "all country names in the world".

Namespace 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 NDN application designs. Naming data enables support for functionality such as content distribution, multicast, mobility, and delay-tolerant networking.

Enabling application developers, and sometimes users, to design their own namespaces for data exchange has several benefits: increasing the closeness of mapping between an application's data and its use of the network; reducing the need for secondary notation (record-keeping to map application configuration to network configuration); and expanding the range of abstractions available to the developers.[4]

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 and guidelines for naming, we convert them into naming conventions and implement them in system libraries to simplify future application development (see [19] for one example for intended use with the current codebase [22]). 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.

2.2 Data-Centric Security

In contrast to TCP/IP, which leaves responsibility for security (or lack thereof) to the endpoints, NDN secures the data itself by requiring data producers to cryptographically sign every Data packet [15]. The publisher's signature ensures integrity and enables determination of data provenance, allowing a consumer's trust in data to be decoupled from how or where it is obtained. It also supports fine-grained trust, allowing consumers to reason about whether a public key owner is an acceptable publisher for a specific piece of data in a specific context. The second primary research thrust is designing and developing usable mechanisms to manage user trust. We have experimented with both a hierarchical trust model where a key namespace authorizes use of keys (a data packet carrying a public key is effectively a certificate, since it is signed by a third party) to sign specific data [5], and web-of-trust to enable secure communication without requiring pre-agreed trust anchors [36].

NDN's data-centric security has natural applications to content access control and infrastructure security. Applications can control access to data via encryption and distribute (data encryption) keys as encrypted NDN data, limiting the data security perimeter to the context of a single application. Requiring signatures on network routing and control messages (like any other NDN data) provides a solid foundation for securing routing protocols against, e.g., spoofing and tampering. NDN's use of multipath forwarding, together with the adaptive forwarding strategy module, mitigates prefix hijacking because routers can detect anomalies caused by hijacks and retrieve data through alternate paths [31]. Since NDN packets reference content rather than devices, it is trickier to maliciously target a particular device, although mitigation mechanisms will be needed against other NDN-specific attacks, e.g., Interest flooding DoS [4].

2.3 Routing and Forwarding

NDN routes and forwards packets based on names, which eliminates three problems caused by addresses in the IP architecture: address space exhaustion, NAT traversal, and address management. There is no address exhaustion problem since the namespace is unbounded. There is no NAT traversal problem since NDN does away with addresses, public or private. Finally, address assignment and management is no longer required in local networks.

NDN can use conventional routing algorithms such as link state and distance vector. Instead of announcing IP prefixes, an NDN router announces *name prefixes* that cover the data the router is willing to serve. The routing protocol propagates these announcements across the network, informing each router's construction of its own FIB. Conventional routing protocols, such as OSPF and BGP, can be adapted to route on name prefixes by treating names as a sequence of opaque components and doing component-wise longest prefix match of a name in an Interest packet against the FIB table.

The PIT state at each router supports forwarding across NDN's data plane, recording each pending Interest and the incoming interface(s), and removing the Interest after the matching Data is received or a timeout occurs. This per-hop, per-packet state differs from IP's stateless data plane. Based on information in the FIB and performance measurements, an adaptive *forwarding strategy* module in each router makes informed decisions about: which Interests to forward to which interfaces, how many unsatisfied Interests to allow in the PIT, the relative priority of different Interests, load-balancing Interest forwarding among multiple interfaces, and choosing alternative paths to avoid detected failures [32, 31]. If a router decides that the Interest cannot be satisfied, e.g., the upstream link is down, there is no forwarding entry in the FIB, or extreme congestion occurs, the router can send a NACK to its downstream neighbor(s) that transmitted the Interest [31]. Such a NACK may trigger the receiving router to forward the Interest to other interfaces to explore alternate paths. The PIT state enables routers to identify and discard looping packets, allowing them to freely use multiple paths toward the same data producer.

[4]These examples of increased usability are based on Green and Petre's "cognitive dimensions" framework [12].

The PIT state serves other valuable purposes. First, since it records the set of interfaces over which the Interests for the same data name have arrived, it naturally supports multicast data delivery. Second, 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. Third, the number of PIT entries is an indicator of router load; constraining its size limits the effect of a DDoS attack. Finally, PIT entry timeouts offer relatively cheap attack detection, and the arrival interface information in each PIT entry could support a push-back scheme.

2.4 In-Network Storage

Because each NDN Data packet carries a name and a signature, it is meaningful independent of who requested or from where it is retrieved. Thus, a router can cache received Data packets in its Content Store and use them to satisfy future requests. The Content Store is analogous to buffer memory in IP routers, but IP routers cannot reuse a packet after forwarding it to its destination, 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, the architecture now supports a more persistent and larger-volume in-network storage, called a Repository (Repo for short). This type of storage can support services similar to that of today's Content Delivery Networks (CDNs), without having to engineer them as an application layer overlay using creative protocol tricks (e.g., DNS manipulation) to make them work.

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

Some researchers have particularly emphasized in-network caching as the basic gain of ICN architectures, e.g., [10]. Although NDN can support more powerful CDN architectures than TCP/IP can, NDN also provides many other functions (securing data, flow balance, stateful data plane which leads to a number of gains on its own) that present even more significant and important advantages.

2.5 Transport Function

The NDN architecture does not have a separate transport layer. It moves the functions of today's transport protocols (demultiplexing, reliable delivery, and congestion control) into applications, supporting libraries, and the strategy module of the forwarding plane. Transport-layer information such as port and sequence numbers are unnecessary; all information required for transport is in the Data names. For example, the name `/ucla/videos/demo.mpg/1/3` specifies where to forward Interests for that name (`/ucla/`), which application should receive them (`/video/`), and any application-specific information (version `1`, segment `3`).

When an application requires reliable delivery, the application itself or its supporting library will monitor the status of outstanding Interests and retransmit them when needed, e.g., after a timeout. 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, can provide effective congestion control throughout the network. If congestion losses occur, caching mitigates the impact since retransmitted Interests can be satisfied by cached Data packets right before the point of packet losses. Thus, NDN can avoid 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.

3. NDN ARCHITECTURE DEVELOPMENT

An NDN protocol specification requires standard formats for the two basic packet types (Interest and Data) and description of the functions supported by the network layer, i.e., the new narrow waist. Building an operational NDN network also requires software libraries to support naming, high performance forwarding and routing, forwarding strategy, and trust management. Similar to IP's supporting components (address allocation, routing protocols, DNS), these libraries are not part of the core architecture but intrinsically support it, and all involve daunting research challenges. This section describes the project's application-driven, experimental approach to designing and developing the architecture, including examples that illustrate its capabilities, and open research challenges.

3.1 Application Research

The project's approach is to design and build a variety of applications on NDN to drive the development and deployment of the architecture and its supporting modules, to test prototype implementations, and to encourage community use, experimentation, and feedback into the design. Application-driven development also allows verification and validation of performance and functional advantages of NDN, such as how routing on names promotes efficient authoring of sophisticated distributed applications, by reducing complexity, opportunities for error, and time and expense of design and deployment. A few years of designing and developing prototype applications on NDN has revealed five key areas of application research that map to important features of the architecture: (1) namespaces; (2) trust models; (3) in-network storage; (4) data synchronization; (5) rendezvous, discovery, and bootstrapping. These challenges arise within and across applications. Namespace design must also recognize the interplay between application-specific requirements for data distribution and organization of trust-related information, together with those imposed for efficient routing/forwarding. Similar challenges exist in name discovery, bootstrapping, and mobility support. This commitment to application development paid off early in the project: it uncovered the unanticipated importance of both a per-node repository for persistent storage, and synchronization as a general building block for applications. A few examples of early applications illustrate NDN's benefits and challenges.

Video Streaming. One of the first NDN applications was a functional video streaming application that demonstrated the practical benefits of NDN-based media delivery, which inherently supports caching and multicast. *NDNVideo* [17] streams live and pre-recorded HD video via NDN, and has

been tested and demonstrated over both UDP and Ethernet transport. In its most recent live demonstration, 1000 clients across Amazon Web Services and the NDN testbed consumed video from a single NDNVideo publisher, with only the "plain vanilla" NDN forwarder on intermediate nodes [9]. The NDNVideo application does not require direct communication between publisher and consumer, enabling publisher-independent scalability through NDN's use of in-network storage. Applications that perform on-the-fly assembly of content or selection of video sections, i.e., frame-level random access requirements, are supported directly through namespace design.

Real-time Conferencing. The *ChronoChat* [36] multi-user text chat application has provided a platform to explore data synchronization techniques that can support a peer-to-peer (i.e., no centralized server) chat service. ChronoChat has also motivated experimentation with non-hierarchical trust models (section 3.3), and development of library support for encryption-based access control. Combining experience from ChronoChat, NDNVideo, and early work on an NDN *Audio Conference Tool* [37], is inspiring development of *ndnrtc*, a videoconferencing application incorporating the WebRTC codebase. This tool will enable investigation of NDN-specific approaches to congestion control, rate adaptation, and playout synchronization for real-time communication.

Building Automation Systems. Enterprise building automation and management systems (BAS/BMS) are an ideal driver for NDN research, since a carefully designed namespace and trust model can support authenticated control of sensors [7]. One of the largest NDN application research efforts thus far has been a collaboration with UCLA Facilities Management, which operates a network with over 150K points of sensing and control, and has facilitated both the installation of dedicated, industry-standard electrical demand monitoring system and access to data from existing systems for NDN research [23]. BAS/BMS applications pose different requirements for data naming and trust than multimedia applications. For example, the current NDN-based BMS design publishes data in three namespaces: one for application data access that follows the physical building system configuration, another for device discovery and bootstrapping, and a trust management namespace for keys that embodies the institutional roles and relationships of the principals. Another challenge is to explore how namespace and storage design can support data aggregation and mining from many heterogeneous sensors and other devices.

Vehicular Networking. Vehicular networking is another domain where the NDN architecture offers advantages, enabling location-based content retrieval and new trust models to support ad-hoc, opportunistic communication [11]. Experimentation with vehicular applications has also led to updates to the NDN protocol stack itself, including support for other media (e.g., 3G/LTE, DSRC/WAVE, WiFi, WiMAX) and network-layer support for *data muling*, where vehicular NDN nodes cache data packets heard over a broadcast channel that do not have matching pending Interest in their PIT, in order to later provide them to other vehicles or pass them to infrastructure.

Other Applications. The available NDN software platform [22] has enabled students and others to explore NDN-based distributed file systems, multi-user games, and network management tools. Over the next few years, work will continue on the applications above, as well as new explorations of climate modeling and mobile health environments as drivers of NDN architecture research and development.

New architecture component: Sync. As a direct result of trying to build robust, efficient and truly distributed (i.e., serverless) peer-to-peer NDN applications, the architecture now supports a new building block called *Sync* [35]. Using NDN's basic Interest-Data exchange communication model, Sync uses naming conventions to enable multiple parties to synchronize their datasets. By exchanging individually computed data digests, each party learns about new or missing data quickly and reliably, and then can retrieve data efficiently via NDN's built-in multicast delivery.

3.2 NDN Routing and Forwarding

The NDN forwarding plane poses two major challenges: forwarding strategy and scalable forwarding. In parallel, the project team has developed prototype NDN-based routing protocols to support near-term and medium-term usage on the testbed, while also examining more radical new routing directions that NDN's adaptive forwarding plane can enable. In several cases, routing protocol design revealed missing features in current software libraries.

Forwarding Strategy Design. The forwarding strategy module at each node is the key to NDN's resiliency and efficiency. Through effective use of NDN's multipath capability, an adaptive forwarding strategy can send consumer Interests along the best performing paths, avoid congestion and failures, balance load across paths, and detect and react to attacks such as prefix hijacking and DDoS [31]. But forwarding strategy design is a brand new research area, with many open questions about how to design simple, effective strategies for different contexts and devices.

Forwarding Engine Design. The forwarding engine must support wire-speed operations, including fast table lookup of variable-length names, efficient data structures to store millions to billions of names, and fast packet processing. Project team members have proposed a highly scalable forwarding structure and engine [34, 33]. Simulation prototypes support multi-million entry FIBs stored in less than 10MB, with FIB lookup speeds on the order of microseconds. Additionally, industrial teams from Cisco [26] and Alcatel-Lucent [27] have developed feasible prototype routers.

Routing Protocol Design. The first NDN routing protocol, intended to rapidly prototype name-based forwarding on the testbed while more adventurous routing research proceeded in parallel, was an OSPF extension (OSPFN [28]) that defined a new type of opaque link state advertisement to carry name prefixes and compute name-based FIB. But this simple adaptation of an IP-based routing protocol imposes exactly the type of burden NDN is designed to avoid, including managing GRE tunnels, managing underlying IP addresses, and hacks to support multi-hop forwarding since OSPF supports only single-path and equal-cost multipath forwarding. The current NDN routing protocol is NDN-

based link-state routing (NLSR) [14], which uses names to identify networks, routers, processes, data, and keys. NLSR can use any underlying communication channel (e.g., Ethernet, IP tunnels, TCP/UDP tunnels) to exchange routing messages. Specifically, routers use Interest packets to retrieve routing updates carried in Data packets, which are signed by the origin router to allow verification of authenticity. Most importantly, NLSR creates name-based, multipath FIBs in each router to support NDN's forwarding plane.

Designing the NLSR protocol required considering the same dimensions as any other NDN application: (a) *how to name* routers, links, routing updates, etc.; (b) how to *distribute cryptographic keys* and how to *derive trust* in these keys; (c) *routing update* dissemination, which requires pulling rather than (OSPF's) pushing updates; and (d) how to produce and rank multiple next-hops for each name prefix to facilitate NDN's *multipath forwarding*.

Exploring New Routing Paradigms. Today's IP routing architecture requires dissemination of topology and policy information, route computation, and sometimes extended convergence times as routers detect and route around failures. In NDN, the forwarding plane itself performs fast fault detection and recovery, reducing the role of routing to bootstrapping forwarding and disseminating long-term topology or policy information [30]. This decoupling allows study of more radical, scalable routing approaches that are not possible in IP networks. For example, NLSR now supports a type of hyperbolic routing [16, 6] by disseminating hyperbolic coordinates in link state advertisements. The Internet topology at the AS level is a scale-free, strongly clustered small world [18], which has a deep connection with hyperbolic geometry of latent spaces effectively underlying the topology [16]. Assuming the router topology and name space have a hyperbolic structure, we can use hyperbolic coordinates of each name prefix as well as neighbors' coordinates to calculate the next-hop using greedy forwardingforwarding – each router forwards the Interest packet to its neighbor router closest to the destination name. Comparing the performance of hyperbolic routing over NDN with link-state routing protocols is ongoing.[5] Other possible approaches to routing, e.g., small worlds, pseudo-potential gradient, and epidemic percolation, may be worth exploring for NDN.

3.3 Trust Management

To verify a data packet's signature, an application can fetch the appropriate key, identified in the packet's *key locator* field, just like any other content. But trust management, i.e., how to determine the authenticity of a given key for a particular packet in a given application, is a primary research challenge. Consistent with an experimental approach, NDN trust management research is driven by application development and use: solving specific problems first and then identifying common patterns.

For example, the security needs of NLSR required development of a simple hierarchical trust model in which keys are published with names that reflect their trust relationship. A root key is owned by the network domain's administrator, and below the root are site keys, each owned by a single site's administrator, signed by the root key and published in the next level of the hierarchy. Each site key then signs the site's operator keys, which in turn sign router keys, which in turn sign the key of the NLSR process on that router. Finally, the NLSR key signs the routing data originated by NLSR. In this trust model, the namespace matches the hierarchy of trust delegation, i.e., (conceptually) `/root/site/operator/router/process`. Publishing keys with a particular name in the hierarchy authorizes them to sign specific data packets and limits their scope. Other applications where real-world trust tends to follow a hierarchical pattern, such as in our building management systems (BMS) [23], may use two separate hierarchies for building operators and for application data, to facilitate fine control over who has access to which data. More flexible and expressive trust relations, such as with our chat application [36], have motivated experimentation with a web-of-trust model. A current chatroom participant can introduce a newcomer to others by signing the newcomer's key. Future applications will implement a cross-certifying model (SDSI) [13, 3], which provides more redundancy of verification, allowing data and key names to be independent, which more easily accommodates a variety of real-world trust relationships.

[5]http://netwisdom.cs.memphis.edu/hrhome.html.

4. NDN COMMUNITY & DEPLOYMENT

The success of a new architecture requires broad community involvement and uptake. NDN has gained momentum already, with participation by academia and industry. But incentivizing incremental deployment requires demonstrating that NDN can solve real-world problems where TCP/IP-based solutions are either problematic or non-existent. The NDN team also maintains an open-source implementation of the NDN protocol stack, a simulator, and a testbed to facilitate testing and broader community participation.

Like IP, NDN is a universal overlay: NDN can run over anything that can forward datagrams (Ethernet, WiFi, Bluetooth, cellular, IP, TCP, etc.), and anything can run over NDN, including IP. Instead of trying to replace or change the deployed IP infrastructure, NDN can simply run over it. NDN can also leverage Internet's well-tested engineering solutions that have taken decades to evolve, such as conventions, policies, and administrative practices for naming and routing. Thus NDN's advantages in content distribution, application-friendly communication and naming, robust security, support for mobility and broadcast can be realized incrementally and relatively painlessly.

For enterprise applications (e.g., automated building control), NDN-based solutions can bring immediate value through local deployment. Wide area applications such as the serverless chatroom can operate over IP tunnels. As NDN-based applications are deployed, we envision islands of NDN nodes emerging, using a rendezvous solution to interconnect by tunneling over non-NDN clouds. Once NDN applications gain wide reception, ISP deployment of NDN routers will improve performance and efficiency for themselves and their customers, providing a natural incentive for infrastructure growth. The IP architecture provided similar overlay capabilities and incremental benefit incentives in its own deployment history

Open source software support. Freely available software libraries and tools are essential to scalable rollout of the NDN architecture. The NDN project originally used PARC's open-source package CCNx [8] as its codebase. To provide a more agile development platform for research, in 2013 the

NDN team forked a version of CCNx, and in early 2014 implemented a new NDN forwarder, NFD [21], from scratch. NFD supports the newly developed NDN packet format and is designed with modularity and extensibility to facilitate diverse experimentation. The NDN platform software releases [22] include supported packages of critical components for building and testing NDN networks and applications. Providing NDN support in popular and easy to use languages such as Python and Javascript has further promoted development activities in the community [24]. The team plans to continue development and support of the NDN codebase, including in-browser support for web-style content publishing via NDN, to help the NDN project as well as the broader community to write innovative, experimental NDN applications.[6]

The project team also maintains an open-source, NS-3 based simulator, ndnSIM [1], which provides a common platform to help researchers evaluate aspects of NDN system performance in large networks. The ndnSIM mailing list hosts active discussions on ndnSIM usage and development among over 100 members from a dozen countries[7].

Operational NDN testbed. Vital to our experimental approach to network research and development is a large-scale testbed to evaluate both applications and core architecture components. During the first year of the project, the NDN team established and instrumented a local testbed at Washington University, with programmable routers and a wide-area overlay testbed connecting all institutions participating in the NSF-funded NDN project. The local testbed supported baseline assessment of NDN prototype implementations, and the wide-area testbed supported testing of NDN components, including video streaming, conferencing tools, and routing protocols. Monitoring scripts and visualization tools facilitate testbed management[8]. While the NDN team encourages researchers to create their own testbeds, it also accepts requests from external sites to connect to the NDN project testbed[9].

5. OPEN QUESTIONS

Ververidis *et al.* [29] provides a survey of many existing ICN projects and identifies several significant debates in the community, most notably related to strategies for scalable trust management and naming itself.

Like other ICN designs, NDN achieves data authenticity, confidentiality and integrity through the use of cryptography. Keys are used to bind names to data via signatures, and to protect data (or names) via encryption. Because these keys are themselves named data, all features of the NDN architecture can be leveraged to address common challenges of key management, such as distribution and revocation; this is an active area of research. Development and integration of high-performance cryptographic algorithms is also essential. The most important challenge, however, is robust and usable trust management, which allows content consumers to determine acceptable signing keys in a given context.

[6]The NDN team is currently working on establishing an intellectual property consortium to navigate the complex issues related to patents on NDN-related technologies.

[7]`http://www.lists.cs.ucla.edu/mailman/listinfo/ndnsim`

[8]`http://ndnmap.arl.wustl.edu/`

[9]`http://named-data.net/ndn-testbed/`

Further, most proposed ICN architectures rely on self-certifying names, which enable any node to verify that the name in a packet matches its content. However, self-certifying names require that each application must determine whether the content is what was desired, which creates additional security risks if a secondary binding to application names is used. NDN offers a different approach, taking per-packet data names directly from applications, then securely binding those names to content. Namespace design is thus a critical area of research as it brings together considerations of an application's data, communication, and storage models with the routing and security implications of names in NDN. In addition to scalable forwarding based on these names, an important challenge for NDN research is to develop and evaluate namespace and related protocol designs for current and envisioned application architectures, and to find reusable approaches that can be applied to common use cases.

Finally, in collaboration with social scientists, we are exploring the social impacts of NDN, in particular four aspects that contrast most with today's TCP/IP architecture: support for semantic classification, provenance, publication, and decentralized communication. We believe these features will enhance opportunities for free speech, security, privacy and anonymity while raising new challenges regarding data retention and content regulation [25].

6. LOOKING FORWARD

The application-driven, experimental approach to NDN research has enabled progress, and provided new depth to the original vision for NDN [15]. But the team has only scratched the surface of research on namespace structure and navigation, trust models and management mechanisms, scalable forwarding and routing, forwarding strategy design, distributed data synchronization, and rendezvous, discovery, and bootstrapping. We have a demonstrated reasonable naming approaches for a set of pilot NDN applications, and a clearer picture on remaining challenges. We implemented an NDN routing protocol that supports traditional link-state as well as hyperbolic routing. We sketched an initial design approach for Sync, a new type of transport that supports synchronization of data across a collection. Sync fills the gap between NDN network layer's simple Interest-Data exchanges and the need of distributed applications to synchronize their data set.

Fortunately, other research efforts closely aligned with NDN are underway around the world, as suggested by the growth in academic workshop and conference activities. The NDN team has taken steps that will hopefully bring broader community participation to the next phase of the project. In particular, the NDN web site publishes regular software updates, testbed documentation, technical and annual reports, a FAQ, and blog entries, and archives the public ndn-interest mailing list for users interested in technical discussions.

Note: In addition to the authors, other PIs of the original NDN FIA project were: Tarek Abdelzaher (UIUC), Daniel Massey (CSU) Gene Tsudik (UCI), Ersin Uzun and Jim Thornton (PARC), and Edmund Yeh (Northeastern). NSF grants funding this project included: CNS-1040868 (UCLA), CNS-1039646 (UCSD), CNS-1039585 (CSU), CNS-1039615 (ASU), CNS-1205562 (Northeastern), CNS-1040380 (UIUC), CNS-1040643 (Wash. U.), CNS-1040802 (UC Irvine), CNS-1040036 (Memphis), and CNS-1040822 (PARC).

7. REFERENCES

[1] NS-3-based NDN simulator. `http://ndnsim.net`.

[2] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker. Naming in content-oriented architectures. In *ACM SIGCOMM Workshop on Information-Centric Networking (ICN)*, 2011.

[3] M. Abadi. On SDSI's linked local name spaces. *Journal of Computer Security*, 6(1-2):3–21, Oct. 1998.

[4] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang. Interest flooding attack and countermeasures in Named Data Networking. In *Proc. of IFIP Networking*, May 2013.

[5] C. Bian, Z. Zhu, A. Afanasyev, E. Uzun, and L. Zhang. Deploying key management on NDN testbed. Technical Report NDN-0009, Rev.2, Feb 2013.

[6] M. Boguñá, F. Papadopoulos, and D. Krioukov. Sustaining the Internet with Hyperbolic Mapping. *Nature Comms*, 1:62, 2010.

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

[8] CCNx. Ccnx software. `http://www.ccnx.org`.

[9] P. Crowley. Named Data Networking (Demo). In *China-America Frontiers of Engineering Symposium*, Frontiers of Engineering, 2013.

[10] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker. Less pain, most of the gain: Incrementally deployable ICN. *SIGCOMM Comput. Commun. Rev.*, 43(4), Aug. 2013.

[11] G. Grassi, D. Pesavento, G. Pau, R. Vuyyuru, R. Wakikawa, and L. Zhang. VANET via Named Data Networking. In *IEEE INFOCOM NOMEN Workshop*, Apr. 2014.

[12] T. R. G. Green and M. Petre. Usability analysis of visual programming environments: a "Cognitive dimensions" framework. *Journal of Visual Languages and Computing*, 7(2), 1996.

[13] J. Y. Halpern and R. van der Meyden. A logic for SDSI's linked local name spaces. In *IEEE Computer Security Foundations Workshop*, 1999.

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

[15] V. Jacobson, , D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *CoNEXT*, 2009.

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

[17] D. Kulinski and J. Burke. NDN Video: Live and Prerecorded Streaming over NDN. Technical Report NDN-0007, Sept 2012.

[18] P. Mahadevan, D. Krioukov, M. Fomenkov, B. Huffaker, X. Dimitropoulos, kc claffy, and A. Vahdat. The Internet AS-level topology: Three data sources and one definitive metric. *Comput Commun Rev*, 36(1), 2006.

[19] I. Moiseenko and L. Zhang. Consumer-Producer API for NDN. Technical Report NDN-0017, Feb 2014.

[20] NDN Team. Named Data Networking (NDN) Project 2012 - 2013 Annual Report, Sept 2013.

[21] NDN team. NDN Forwarding Daemon, 2014. `http://named-data.net/doc/NFD/current/`.

[22] NDN team. NDN Platform, 2014. `http://named-data.net/codebase/platform/`.

[23] W. Shang, Q. Ding, A. Marianantoni, J. Burke, and L. Zhang. Securing building management systems using named data networking. *IEEE Network Special Issue on Information-Centric Networking*, Apr 2014.

[24] W. Shang, J. Thompson, M. Cherkaoui, J. Burke, and L. Zhang. NDN.JS: A JavaScript Client Library for Named Data Networking. In *IEEE INFOCOM 2013 NOMEN Workshop*, Apr 2013.

[25] K. Shilton, J. Burke, kc claffy, C. Duan, and L. Zhang. A World on NDN: Affordances and Implications of NDN. Technical Report NDN-0018, April 2014.

[26] W. So, A. Narayanan, and D. Oran. Named data networking on a router: Fast and DoS-resistant forwarding with hash tables. In *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, Oct 2013.

[27] M. Varvello, D. Perino, and J. Esteban. Caesar: A content router for high speed forwarding. In *ACM SIGCOMM Workshop on ICN*, 2012.

[28] L. Wang, A. K. M. M. Hoque, C. Yi, A. Alyyan, and B. Zhang. OSPFN: An OSPF-based routing protocol for NDN. Technical Report NDN-0003, July 2012.

[29] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, and G. Polyzos. A survey of information-centric networking research. *IEEE Communications Surveys Tutorials*, 2013.

[30] C. Yi, J. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang. On the role of routing in Named Data Networking. Technical Report NDN-0016, Dec 2013.

[31] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang. A case for stateful forwarding plane. *Computer Communications: ICN Special Issue*, 36(7):779–791, April 2013.

[32] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang. Adaptive Forwarding in Named Data Networking. *ACM SIGCOMM CCR*, 42(3), 2012.

[33] H. Yuan and P. Crowley. Scalable pending interest table design: From principles to practice. IEEE INFOCOM, 2014.

[34] H. Yuan, T. Song, and P. Crowley. Scalable NDN forwarding: Concepts, issues and principles. In *ICCCN*, 2012.

[35] Z. Zhu and A. Afanasyev. Let's ChronoSync: Decentralized dataset state synchronization in NDN. In *ICNP*, 2013.

[36] Z. Zhu, C. Bian, A. Afanasyev, V. Jacobson, and L. Zhang. Chronos: Serverless multi-user chat over NDN. Technical Report NDN-0008, October 2012.

[37] Z. Zhu, J. Burke, L. Zhang, P. Gasti, Y. Lu, and V. Jacobson. A new approach to securing audio conference tools. In *Asian Internet Engineering Conference*, AINTEC, 2011.