A Brief Introduction to Named Data Networking

Alex Afanasyev
FIU
aa@cs.fiu.edu

Jeff Burke
UCLA
jburke@remap.ucla.edu

Tamer Refaei
The MITRE Corporation †
mrefaei@mitre.org

Lan Wang
Univ. Memphis
lanwang@memphis.edu

Beichuan Zhang
Univ. Arizona
bzhang@cs.arizona.edu

Lixia Zhang
UCLA
lixia@cs.ucla.edu

***Abstract*—As a proposed Internet architecture, Named Data Networking (NDN) is designed to network the world of computing devices by naming data instead of data containers as IP does today. With this change, NDN brings a number of benefits to network communication, including but not limited to built-in multicast, in-network caching, multipath forwarding, and securing data directly. NDN also enables resilient communication in intermittently connected and mobile ad hoc environments, which is difficult to achieve by today's TCP/IP architecture. This paper offers a brief introduction to NDN's basic concepts and operations, together with an extensive reference list for the design and development of NDN for readers interested in further exploration of the subject.**

***Index Terms*—Network architecture, Named Data Networking**

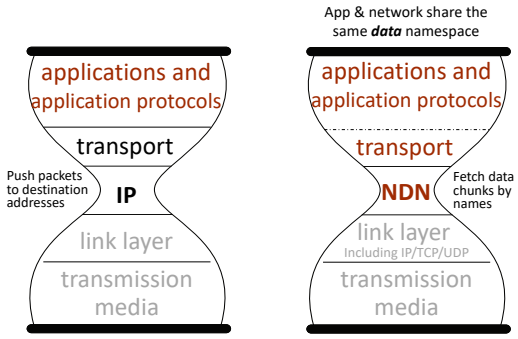

Fig. 1. Comparing IP and NDN protocol hourglasses

I. Introduction

In a nutshell, all a network does is ship data bits. It is the job of the network architecture to define how this data shipping is realized. A network architecture design makes two fundamental decisions: (1) what namespaces are used for data delivery, and (2) what are the specific mechanisms for the delivery.

There are two main options for the first design decision: (1) name the locations to ship the bits to; (2) name the bits themselves. The second design decision depends on the first: if named by location, data bits to be sent from host A to host B may either be delivered along an established path between the two communicating endpoints (virtual circuit), or travel through the network as independent pieces to reach B (datagrams). If one names the data bits directly instead of by location, then a network delivers named bits (named data packets) to their requesters.[1]

Today's TCP/IP protocol architecture picked the first option of network namespace design, *naming locations*, the same communication model used by circuit-switched telephone networks. A telephone network assigns a phone number to each telephone set wired at a specific location and sets up a circuit between two calling parties. From telephony to IP networking, phone numbers are replaced by IP addresses, circuit switching by packet switching with datagram delivery, but the same location-based, point-to-point communication model remains.

Named Data Networking (NDN) [1], [2] takes the second option of network namespace design, *naming bits*. As a proposed Internet architecture, NDN is designed to network the world of computing devices, ranging from IoT sensors to cloud servers, by naming data bits. As we show in Figure 1, named (and secured as will be shown later) data chunks make the centerpiece in the NDN network architecture, and the NDN network layer uses application data names to communicate. This design empowers the network to retrieve named data by any means necessary, treating networking, storage, and computing resources in the same manner (Section IV) and enables one to secure data directly (Section V).

This paper provides a brief introduction to NDN's basic concepts and operations, together with an extensive reference list that points to the major results from the NDN design and development efforts over the last eight years. Although space limitations require that we must leave a detailed description of the overall NDN design to another paper, we hope that this paper offers the reader an overall picture of NDN's basic principles, concepts, operations, and properties. Interested readers are encouraged to explore the references to learn more details.

Taking a different approach from the earlier work on NDN

This work is partially supported by US National Science Foundation under award CNS-1719403, CNS-1629769, CNS-1629009 and CNS-1629922.

†Author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by the author. It is approved for Public Release; Distribution Unlimited. Case Number 18-xxx

[1]One might speculate whether an architecture design could support naming both locations and data. Although nothing seems wrong about this proposal on the surface, naming locations or naming data each leads to fundamentally different network designs. We leave this explanation to another paper.

introductions [2], in this paper we introduce the basic concepts in the NDN design through comparisons with two protocols that most people should be familiar with: HTTP and TCP/IP protocol stack (Sections II and III). We then sketch NDN's basic approaches to network forwarding (Section IV), secure communications (Section V), and dataset synchronization (Sync), which plays the role of transport services in the NDN protocol stack (Section VI). We wrap up the paper with discussions on several topics related to the NDN design (Section VII), including (i) NDN's applicability to battlefield applications, (ii) where the basic ideas in the NDN design came from, and (iii) the role of applications in the NDN architecture development.

II. NDN as a Data Retrieval Protocol

From 10,000 feet, NDN's basic idea might be seen as shifting HTTP's semantics of *request*, for a named data object, and *response*, containing the requested object, to the network layer. Being a network layer protocol, NDN's requests and responses work at a network packet granularity (Figure 2) —each request, carried in an NDN *Interest* packet, contains the name of the requested data and fetches one NDN *Data* packet back.[2] If the size of an application data object is large, the object is segmented, with the segment number being a component in the data packet name.[3] Both packet types carry the data name; neither contains an address nor any information about the requester.

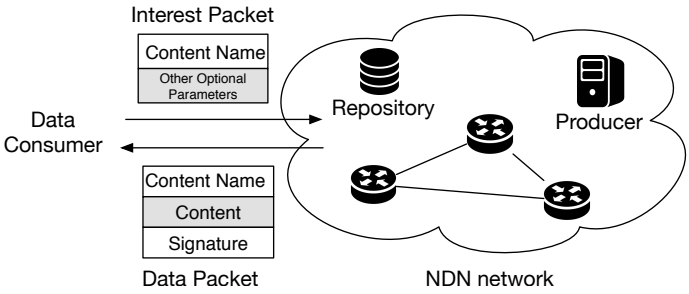

Fig. 2. In an NDN network, one Interest packet can fetch one Data packet back from either the original data producer, or a router cache, or from a dedicated data repository (see Section IV)

Most of today's Internet applications are built on web protocols that request content by names. NDN adopts this request-reply communication model and directly uses application data names at the network layer to make network services best match application communication patterns. However, in contrast with URLs used in HTTP requests, which are used by applications only, NDN data names are also used by network layer to fetch data, as explained in Section IV, and to define security policies and automate data authentication and confidentiality control (Section V). Consequently, namespace design is the most important step, and often the most challenging one, in all NDN application and protocol designs, although little has been written about the lessons that have been accumulated from the NDN namespace design exercises.[4] One of our ongoing efforts is systematically documenting our experience to guide future NDN development efforts.

Another important difference between HTTP as an application protocol and NDN as a network layer protocol is what each is responsible to accomplish. HTTP runs over a *transport connection*, e.g., TCP or QUIC, which reliably delivers packets between a requestor and a data source. Thus, a web application needs only to send the request and wait for either a reply or a connection error. On the other hand, NDN delivers packet over a *network*, which may be a locally scoped IoT network, an ad hoc network made of mobile devices, or a global scale Internet, thus an NDN Interest packet may travel multiple hops to fetch the requested data, and the data packet needs to make back to the requester while the Interest carries *no* requester information. We explain how NDN achieved this through its stateful forwarding plane in Section IV.

In addition to being *network layer* packets, NDN Data packets also differ from HTTP data objects in two other important aspects. First, while an HTTP response message is implicitly bound to the requesting URL by the underlying TCP connection, every NDN Data packet explicitly carries the data name in addition to the requested content, together with a signature that cryptographically binds the name to the content at the time of data creation; the content may also be encrypted whenever needed. Second, while the same URL may retrieve different content,[5] NDN Data packets are immutable: each name uniquely identifies an NDN Data packet[6]; when a producer changes the content of a data packet, it needs to generate a new packet with a new name to distinguish the different versions of the content.

III. Hourglass-Shaped NDN Protocol Stack

Network researchers are intimately familiar with the existing TCP/IP protocol architecture. We leverage this familiarity to help readers grasp the NDN concepts by describing the NDN protocol stack in comparison to the TCP/IP protocol stack as shown in Figure 1. We highlight the similarities and, more importantly, several fundamental differences between the two stacks.

The most prominent similarity is that the NDN protocol stack retains the same hourglass shape as TCP/IP. Just like IP, the NDN network protocol performs datagram delivery and runs over any transport media that can carry datagrams. Media might include existing layer 2 protocols most often used to interconnect nearby devices as well as TCP/UDP/IP tunnels (which we use today to connect NDN-enabled devices that are remote to one another).

Despite a similar structure, Figure 1 also shows that NDN changes the narrow waist of the hourglass from IP packets

[2]In the rest of the paper, we may shorten Interest packets to *Interests*, and Data packets to simply *Data*.

[3]If a data fragment is still bigger than the link MTU it needs to cross, NDN performs hop-by-hop fragmentation and reassumbly [3].

[4]There exist a few scattered discussions on the NDN namespace design, see Section III in [4], Section 2 in [5], and Section IV in [6] for a few examples.

[5]For example, the content of http://cnn.com changes over time.

[6]An NDN Data packet may have multiple names.

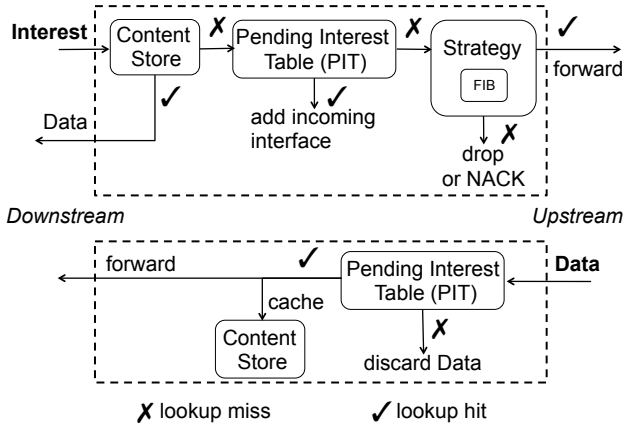

Fig. 3. Forwarding Process at an NDN Node: *upstream* indicates the direction of data producer and *downstream* is to data consumer.

carrying source and destination addresses to named, secured NDN data chunks. This conceptually simple change leads to several profound differences between the NDN and IP:

- The NDN network layer has no addresses; instead, it uses application-defined namespaces.
- Consequently, NDN names data instead of data locations.
- In NDN, consumers fetch data instead of senders pushing packets to destinations.

The direct use of application namespace for network communication can greatly simplify NDN-based systems: there is no more need for IP address allocation or DNS services to translate names used by applications to addresses used by IP for delivery. Next, we describe NDN packet delivery, which illustrates another significant change from IP: NDN's use of a *stateful* forwarding plane.

IV. NDN as a Forwarding Engine

By default, an Interest packet carries the name of the requested data only and does not contain any information about the requester.[7] Each NDN forwarder uses the name in an Interest to determine to which interface or interfaces this Interest should be forwarded.

A. Stateful Forwarding Plane

As shown in Figure 3, each NDN node's forwarding module contains three basic components: a Content Store (CS), a Pending Interest Table (PIT), and a Forwarding Strategy that includes the Forwarding Information Base (FIB), which is populted by routing protocols or other means such as *self-learning* [8]. Upon receiving an Interest IN, the forwarder first checks the CS, then the PIT: if a matching Data is found from the CS, the data is returned; otherwise, if a matching PIT entry is found, IN's incoming interface is added to the PIT. If there is no match in the CS or the PIT, the forwarder records IN's incoming and outgoing interfaces in the PIT, together with a timestamp, then the Forwarding Strategy decides the output interface(s) based on the FIB and observed performance, as we explain below.

Once an Interest reaches a node which has a data packet D with the matching name, D is forwarded over the reverse path of the Interest, hop-by-hop, to reach *all* the requesters, regardless how many there may be; D can also be cached in the CS of each hop to serve future requests for the same data. As one can see, this Interest-Data exchange creates a closed feedback loop at each hop, allowing each forwarder to measure data retrieval performance, to report forwarding problems through hop-by-hop NACK, and to perform effective congestion control.

We refer interested readers to [9], [10] for more details about the NDN forwarding functions, but make three brief observations. First, NDN has a *stateful* forwarding plane for *datagram* delivery: the state is per-packet, per hop. Second, this stateful forwarding plane, whose size scales proportionally with the bandwidth times round trip delay, can potentially put a high demand on NDN routers' memory. However, the PIT enables a number of capabilities, including multicast delivery, loop-free multipath forwarding, in-network congestion control, efficient loss recovery, instant detection and robust recovery from path failures, that the Internet has long sought after [9]. We argue that a new network architecture design must take full advantage of technology advances, such as new types of fast and large-capacity memory, to best satisfy applications' needs. Lastly, an NDN forwarder can buffer both Interests (in PIT) and Data (in CS), a distinct feature that makes NDN natively capable of communicating through intermittent connectivity in an adverse environment, as we discuss further in Section VII-A.

B. Routing in NDN Networks

An NDN network runs routing protocol(s) to propagate the reachability of data names, similar to an IP network running routing protocols to propagate the reachability of IP addresses. However, there exist several important differences between routing in IP and NDN networks.

First, an NDN routing protocol is an NDN application, and routing updates are named and secured NDN data packets. Thus, NDN routing security is natively built-in, while enhancing IP routing with security has been a multi-year effort and still far from done.

Second, NDN supports multipath forwarding by allowing each FIB entry to have multiple next hops without worrying about Interest looping.[8] In contrast, due to concerns about packet looping, an IP FIB entry has only one next hop.

Third, an NDN network of a small size may not run any routing protocol, but instead, it can use self-learning to discover data availability [8].

Finally, as we explain in [11], NDN's stateful forwarding plane fundamentally changes the requirements and importance

[7] An Interest may contain the requester's information if needed by applications; an example is the case of *signed Interests* [7], where the signature contains the signer's identity.

[8] Each NDN Interest carries a nonce, enabling PIT to detect any looping Interest and discard it.

of a routing protocol, as the FIB is only one of, but not the sole input factor in forwarding decisions.

One commonly heard concern about name-based NDN routing is scalability, as IP has a finite address space but NDN's namespace is unbounded. Interested readers may look up [12] and [13], which report preliminary results in addressing NDN routing scalability.

C. The Power of Using Names for Network Communication

The Internet has enabled an abundance of diverse applications that have changed our lives by performing a conceptually straightforward task: delivering IP packets from any host to any other host. Routing and forwarding provide the basic power of the network layer to do this—they determine how to get from one host to another on a global scale.

By naming data and doing routing/forwarding on names, NDN magnifies that power. As of now, for an app on a mobile phone to get desired data, it must first figure out the destination IP address to send its data request to, which is a nontrivial task—applications work with semantically meaningful names and know nothing about addresses or network topology. Today, we let the phone look up DNS to find the address of a cloud server and let the cloud handle the app's need, even when a neighbor's storage server (e.g., a Synology box[9]) may be hosting all the photos taken from the block party a few days ago, and the desktop with a GPU in the city library a few blocks away may well be able to offer photo processing functions.

With NDN, the neighbor's Synology box could locally announce the names of its collected contents (the photos), so can the library desktop announce its computing services. By fetching data with appropriate names, NDN no longer tells the difference between wires, storage, or even processing, since the requested data can come from any of them—an original camera, a nearby storage, or a processing unit if a user requests annotated pictures [14]–[16]. The same capability extends to a larger scale with proper handling of routing scalability. NDN blends networking, storage, and processing into one integrated system.

V. Communication Security

Despite advances in cryptography, secure protocols and system-level defenses, security remains the biggest challenge in today's Internet. NDN addresses this challenge in fundamentally different ways than today's practices. NDN enables one to *secure data directly* by having data producers cryptographically sign each data packet to bind together the data name, producer and content. One can also encrypt the data whenever data confidentiality is needed [17]. These security properties—*authenticity* and *confidentiality*—stay with the data itself, independent from the data containers and communication channels. Named, secured data packets provide a basic building block directly at the narrow waist of the NDN protocol stack to secure NDN communications. In addition, in-network storage/caching, multipath forwarding, and flow balance are key functionalities offered by the NDN architecture that provide great resilience against *Denial of Service* attacks (e.g. bandwidth-depletion, reflection, and black-holing attacks) that the current TCP/IP architecture has struggled to cope with [18].

A major issue in using cryptographic protection is the management and availability of keys and certificates, which are needed for end users to verify the received data and decrypt encrypted content. In NDN, keys, certificates, trust and access policies are all named, secured data packets. In addition, NDN allows one to make use of structured, semantically meaningful names to define relations between data names and cryptographic key names, to develop effective solutions for trust management and security policies [19]. Furthermore, NDN can also provide usable key management solutions through defining naming conventions [17].

Cryptographic protections are rooted in the establishment of trust anchors. Instead of blindly trusting a large number of commercial certificate authorities, NDN advocates an approach to trust anchor establishment that resembles the Simple Public Key Infrastructure/Simple Distributed Security Infrastructure (SPKI/SDSI) model [20]. The NDN design lets the authority of each networked system (an organization, a smart home, etc.) establish its own trust anchor(s) and have its own local means to securely install the trust anchors into all devices under its control. We refer interested reader to [21] for more details.

VI. The Need for Dataset Synchronization

Conceptually, an ongoing TCP connection can be viewed as synchronizing the dataset at the two ends: either end produces data that is then reliably delivered to the other end. However, TCP only works for point-to-point data synchronization and supports only synchronous communication (i.e., both ends must be online at the same time).

The concept of NDN Sync, or Sync in short, was born from observed common needs in developing NDN applications—dataset synchronization that is *multiparty* and *asynchronous*. This can be seen as a generalization of TCP. Unlike TCP, Sync does not add an additional header in front of application generated packets, and is implemented in system libraries to support application data delivery needs.

As an illustrative use case for Sync, consider an NDN-based chat application in a battlefield scenario, as shown in Figure 4, which enables the soldiers and commanders to communicate securely through simple messages. Sync can significantly reduce the development effort of such an app: the app can simply hand over user messages to Sync, which then takes the responsibility to deliver the message to all the users in the same chatroom as soon as possible over heterogeneous, lossy, and intermittently connected communication channels. Other tasks that a chatroom app must perform, such as defining security policies and managing the chatroom membership, can also be simplified by making use of NDN libraries and data-centric properties.

[9]Synology is one of the many inexpensive NAS (network-attached-storage) products on the market today.

Over the years multiple Sync protocol designs have been developed. A companion submission titled "A Brief Introduction to NDN Dataset Synchronization (NDN Sync)" provides a summary on what has been achieved in the development of Sync. Interested readers can also find further details from [4], [22]–[25].

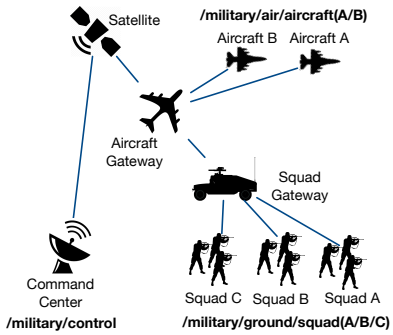

Fig. 4. A battlefield scenario, where the various wireless communication channels offer lossy and intermitted connectivities.

VII. Discussion

A. NDN for the Battlefield

Tactical applications tend to be peer-to-peer in nature, providing global, zoned or localized situational awareness. They need to operate in a bandwidth-limited, highly mobile, and constantly disconnected environment. The combination of three basic ingredients in NDN design, i.e., naming data, securing data directly, and stateful forwarding, leads to a secure and resilient network delivery service with a number of highly desirable properties to support tactical applications.

- *Naming data automatically enables host multihoming*: Interests and Data may come and go through any of the multiple wireless interfaces a mobile device may have.
- *Facilitating ad hoc networking through rendezvous by application names*: One can simply request data by names, which are built into applications, from other encountered devices, with no additional configuration needed.
- *Enabling delay/disruption tolerant networking*: As mentioned in Section IV, each NDN node has built-in storage for pending Interests (PIT) and Data (CS), so it can carry them around until it meets the next node to forward them further.
- *Supporting sharing among devices*: caching and multicast allow devices to share their data, leading to more efficient utilization of the bandwidth in the network and to the outside, which is typically low in a tactical network.
- *Allowing quick recovery from losses*: Suppose a Data packet gets lost on its way to the consumer. As long as a node on the path has cached the data before it is dropped, the consumer can retransmit its Interest to find the previously cached data. This way the Interest does not need to travel all the way to the producer, enabling fast recovery from losses.

An NDN node can freely communicate with any node it encounters because authenticity and access control are built into the data and independent from the data containers or communication channels. One can find further explanation in [9], [21], [26], [27]. Next we articulate how the NDN design achieves these advantages.

B. "Study the past if you would define the future" – Confucius

No bible teaches us how to design a network architecture; instead, we learn from experimentation that feeds iterative design. The NDN design, in particular, is the consolidation of research insights accumulated step-by-step over decades. The NDN design adheres to the well-understood Internet design principles. Similar to an IP network, an NDN network performs *datagram delivery* to keep the design simple. Its stateful forwarding plane keeps the state at a per-datagram per-hop granularity, offering significantly enhanced robustness and resiliency to packet losses and component failures as compared to IP [9], [28]. Moreover, NDN extends the end-to-end principle to include *end-to-end data security*: each producer seals the binding between the name and content of every data packet it generates with a signature, and all consumers can verify data authenticity regardless of where the data comes from. This works well in the presence of an increasing number of CDN boxes for content scalability, and increasing device mobility and capacity where mobiles serve as effective data carriers to enable communications even when there is never a direct communication channel between data producers and consumers.

One can trace back each of the main ideas in the NDN design to earlier research results. As introduced earlier, the Interest for data communication patterns mimics web applications. The idea of a receiver-driven data delivery model is adopted from IP multicast [29], [30]. The idea of having packets carry application defined data units or Application Level Framing (ALF) was discussed in [31] back in 1990; the idea of enabling resilient communication through multicast and ALF came from [32] a few years after [31]. In-network state for IP was used in the receiver-driven reservation protocol RSVP [33]. Forwarding based on application names in an overlay network with storage appeared in Adaptive Web Caching in 1998 [34], while location-independent data naming and data-centric communication were proposed in 2001 [35]. Finally, DNS Security Extensions, whose design started in mid 90's, secure all response data directly.

Much of IP's success is due to the minimalism of its network layer and the weak demands it places on Layer 2, and NDN inherits these properties. At the same time, NDN recognizes the characteristics of today's applications. Consequently NDN both shares similarities with the TCP/IP architecture and differs from it in fundamental ways.

C. Application-Driven Architecture Development

If a new architecture is useful in solving real problems and is easy to deploy, we are confident that it would be readily accepted, just like what happened to IP deployment 30+ years

ago. To ensure that NDN solves real problems, the NDN design has adopted an application-driven approach for the architecture design and development, which targets (i) useful apps and (ii) running code that verifies/validates the design.

We believe that NDN can best prove its usefulness in greenfields where IP has not made progress due to intrinsic limitations. Trying out emerging applications that do not and cannot have good IP-based solutions—such as vehicle networking, serverless distributed applications, participatory and user-centric mobile health, and AR/VR in wireless ad-hoc environments—can validate beneficial built-in properties of the NDN architecture. It also drives engineering of architecture components to simplify development and deployment of such applications. In particular, the application prototypes we built over the years [5], [36], [37] demanded auto-configuration of initial forwarding state and security. While we are still refining these solutions, a number of useful tools and specifications have already emerged over the years [38], including autoconfig tools [39]–[41] to establish connectivity to NDN hubs in non-native NDN environments, automatic prefix propagation [42], prefix re-advertise [43], Wi-Fi Direct data discovery [44] and self-discovery [8] protocols to simplify FIB management in local environments without the use of heavy-weight routing protocols, trust bootstrapping protocols [21], NDNCERT [45] to automate certificate management, trust schema [19] to automate data validation, NAC [17] to automate management of encryption keys, and many others.

VIII. Conclusion

Today's Internet runs over the TCP/IP protocol stack whose specifications [46] were published 37 years ago to the date of this conference. Over this time period, computing and communication technologies have advanced rapidly. These advances have enabled new applications, which in turn place new demands on the underlying technologies, including TCP/IP. Consequently, the networking architecture must evolve to take advantages of technology advances and to meet the demands of new applications, which range from distributing vast amount of video to a global audience that is increasingly connected through mobile rather than wireline networks, to securing billions of IoT devices, and to supplying 3D maps to soldiers in adverse battlefield environments.

The NDN architecture is built upon the insights in the basic network design principles and lessons learned from the operational Internet which have been accumulated over the last few decades. The research efforts into the NDN architecture development led to various NDN application prototypes, forwarding strategies, congestion control mechanisms, routing protocols, data synchronization, security solutions, and an experimental platform running over an operational testbed.

The application-driven approach described in the previous section verifies and validates various design decisions, discovers missing pieces, and exposes design tradeoffs. Through it, we have learned (for example) the importance of, as well as the challenges in, the namespace design. We have learned the power from utilizing naming conventions and the necessity of automating security supporting mechanisms. The more recent effort in applying NDN to support mobile ad hoc communications has led to further insight into NDN Sync design spaces.

Although the NDN architecture design is still in its research stage and a number of issues remain open, efforts of applying NDN to solve real world problems have begun. Similar to the IP rollout process, where a number of important issues were identified and resolved through large scale use,[10] we expect that the NDN architecture and implementation will continue to mature through what is learned from new application and deployment efforts.

We believe that the NDN design brings not only new research topics in the overall architecture and protocol development, but also new opportunities to take NDN prototypes to trial deployment. We invite everyone to join this effort, and hope the references provided in this paper can serve as a starting point for interested readers to further explore this new exciting direction.

References

[1] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking Named Content," in *Proc. of CoNEXT*, 2009.

[2] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *ACM Computer Communication Review*, July 2014.

[3] A. Afanasyev, J. Shi, L. Wang, B. Zhang, and L. Zhang, "Packet fragmentation in NDN: Why NDN uses hop-by-hop fragmentation (NDN Memo)," NDN, Technical Report NDN-0032, May 2015.

[4] Z. Zhu and A. Afanasyev, "Let's ChronoSync: Decentralized dataset state synchronization in Named Data Networking," in *Proc. of IEEE ICNP*, 2013.

[5] H. Zhang, Z. Wang, C. Scherb, C. Marxer, J. Burke, L. Zhang, and C. Tschudin, "Sharing mhealth data via named data networking," in *Proc. of ACM ICN*, 2016.

[6] W. Shang, Z. Wang, A. Afanasyev, J. Burke, and L. Zhang, "Breaking out of the cloud: Local trust management and rendezvous in Named Data Networking of Things," in *Proc. of ACM/IEEE IoTDI*, 2017.

[7] NDN Project Team, "Signed Interest," https://named-data.net/doc/ndn-cxx/current/specs/signed-interest.html, 2018.

[8] J. Shi, E. Newberry, and B. Zhang, "On Broadcast-based Self-Learning in Named Data Networking," in *Proc. of IFIP Networking*, 2017.

[9] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications: ICN Special Issue*, vol. 36, no. 7, pp. 779–791, April 2013.

[10] K. Schneider, C. Yi, B. Zhang, and L. Zhang, "A practical congestion control scheme for Named Data Networking," in *Proc. of ACM ICN*, 2016.

[11] C. Yi, J. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "On the role of routing in Named Data Networking," in *Proc. of ACM ICN*, 2014.

[12] A. Afanasyev, C. Yi, L. Wang, B. Zhang, and L. Zhang, "SNAMP: Secure namespace mapping to scale NDN forwarding," in *Proc. of IEEE Global Internet Symposium*, 2015.

[13] V. Lehman, A. Gawande, B. Zhang, L. Zhang, R. Aldecoa, D. Krioukov, and L. Wang, "An experimental investigation of hyperbolic routing with a smart forwarding plane in NDN," in *Proc. of IEEE/ACM IWQoS*, 2016.

[14] C. Marxer, C. Scherb, and C. Tschudin, "Access-controlled in-network processing of named data," in *Proceedings of ACM ICN*, 2016.

[15] M. Król and I. Psaras, "NFaaS: Named Function As a Service," in *Proceedings of ACM ICN*, 2017.

[16] J. Burke, "Browsing an Augmented Reality with Named Data Networking," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, July 2017.

[10] Examples include the development of DNS services, TCP congestion control, and BGP the inter-domain routing protocol.

[17] Y. Yu, A. Afanasyev, and L. Zhang, "Name-based access control," NDN, Technical Report NDN-0034, Jan. 2016.

[18] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "DoS and DDoS in named data networking," in *Proc. of ICCCN*, 2013.

[19] Y. Yu, A. Afanasyev, D. Clark, kc claffy, V. Jacobson, and L. Zhang, "Schematizing trust in Named Data Networking," in *Proc. of ACM ICN*, 2015.

[20] R. L. Rivest and B. Lampson, "SDSI—a simple distributed security infrastructure." Crypto, 1996.

[21] Z. Zhang, Y. Yu, H. Zhang, E. Newberry, S. Mastorakis, Y. Li, A. Afanasyev, and L. Zhang, "An overview of security support in Named Data Networking," NDN, Technical Report NDN-0057, Apr. 2018.

[22] M. Zhang, V. Lehman, and L. Wang, "Scalable Name-based Data Synchronization for Named Data Networking," in *Proc. of IEEE INFOCOM*, 2017.

[23] W. Fu, H. Ben Abraham, and P. Crowley, "Synchronizing namespaces with invertible bloom filters," in *Proc. of ACM/IEEE ANCS*, 2015.

[24] W. Shang, A. Afanasyev, and L. Zhang, "VectorSync: Distributed dataset synchronization over Named Data Networking," NDN, Technical Report NDN-0056, Mar. 2018.

[25] W. Shang, Y. Yu, L. Wang, A. Afanasyev, and L. Zhang, "A survey of distributed dataset synchronization in Named Data Networking," NDN, Technical Report NDN-0053, May 2017.

[26] C. Gibson, P. Bermell-Garcia, K. Chan, B. Ko, A. Afanasyev, and L. Zhang, "Opportunities and challenges for Named Data Networking to increase the agility of military coalitions," in *Proc. of DAIS*, 2017.

[27] H. Zhang, Y. Li, Z. Zhang, A. Afanasyev, and L. Zhang, "NDN Host Model," *ACM Computer Communication Review*, July 2018.

[28] S. Vusirikala, S. Mastorakis, A. Afanasyev, and L. Zhang, "Hop-by-hop best effort link layer reliability in Named Data Networking," NDN, Technical Report NDN-0041, Aug. 2016.

[29] S. E. Deering, "Multicast routing in internetworks and extended lans," ser. SIGCOMM '88, 1988.

[30] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," ser. SIGCOMM '96, 1996.

[31] D. Clark and D. Tennenhouse, "Architectural considerations for a new generation of protocols," in *Proc. of SIGCOMM*, 1990.

[32] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM Transactions on Networking*, 1997.

[33] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "Rsvp: A new resource reservaüon protocol," *IEEE network*, 1993.

[34] S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd, and V. Jacobson, "Adaptive web caching: towards a new global caching architecture," *Computer Networks and ISDN systems*, 1998.

[35] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building efficient wireless sensor networks with low-level naming," in *Proceedings of SOSP*, 2001.

[36] A. Afanasyev, Z. Zhu, Y. Yu, L. Wang, and L. Zhang, "The story of ChronoShare, or how NDN brought distributed secure file sharing back," in *Proc. of IEEE MASS Workshop on Content-Centric Networks*, 2015.

[37] P. Gusev, Z. Wang, J. Burke, L. Zhang, E. Muramoto, R. Ohnishi, and T. Yoneda, "Real-time streaming data delivery over Named Data Networking," *IEICE Transactions*, May 2016.

[38] NDN Project Team, "Named Data Networking project specifications," https://named-data.net/project/specifications/, 2018.

[39] ——, "NDN hub discovery procedure," http://named-data.net/doc/NFD/current/manpages/ndn-autoconfig.html, 2018.

[40] G. Liu and A. Afanasyev, "NDN-FCH (find closest hub)," https://github.com/named-data/ndn-fch, 2018.

[41] NDN Project Team, "Local hub prefix discovery," https://named-data.net/doc/NFD/current/local-prefix-discovery.html, 2018.

[42] Y. Li, A. Afanasyev, J. Shi, H. Zhang, Z. Zhang, T. Li, E. Lu, B. Zhang, L. Wang, and L. Zhang, "NDN automatic prefix propagation," NDN, Technical Report NDN-0045, March 2018.

[43] NDN Project Team, "Readvertise end-host routes into NLSR," https://redmine.named-data.net/issues/3818, 2017.

[44] A. Gong, "NDN over WiFi Direct protocol specification," https://redmine.named-data.net/projects/nfd-android/wiki/NDN_Over_WiFi_Direct_Protocol_Specification, December 2016.

[45] Z. Zhang, Y. Yu, A. Afanasyev, and L. Zhang, "NDN certificate management protocol (NDNCERT)," NDN, Technical Report NDN-0050, Apr. 2017.

[46] "Internet Protocol," J. Postel, Ed., September 1981.