Named Data Networking of Things (Invited Paper)

Wentao Shang*, Adeola Bannis†, Teng Liang‡, Zhehao Wang§, Yingdi Yu*,
Alexander Afanasyev*, Jeff Thompson§, Jeff Burke§, Beichuan Zhang‡, and Lixia Zhang*
*UCLA Internet Research Laboratory
†Carnegie Mellon University, Silicon Valley
‡Dept. of Computer Science, University of Arizona
§UCLA Center for Research in Engineering, Media and Performance

***Abstract*—The Internet of Things (IoT) is a vision for interconnecting all of the world's "things"—from vehicles to diet scales, smart homes and electrical grids—through a common set of networking technologies. Realizing this vision using a host-to-host communication paradigm, such as that of the Internet Protocol (IP), is challenging in the context of highly heterogeneous, constrained devices that connect intermittently to one or more networks, often using multiple interfaces; communicate within various security regimes; and require both local and global communication capability. Using IP and similar protocols as the narrow waist of interoperability for IoT requires managing data exchange and security in terms that are largely orthogonal to application semantics, while simultaneously needing to minimize resource usage. This paper explores how Named Data Networking (NDN), a proposed future Internet architecture, addresses the root causes of these challenges and can help achieve the IoT vision in a more secure, straightforward, and innovation-friendly manner. NDN's data-centric communication model aligns network and application semantics, enabling developers to work with "things" and their data directly, and for IoT networks to be deployed and configured easily. To substantiate the high-level discussion, we give examples of ongoing design and implementation work in IoT over NDN and compare the architecture to well-known existing protocols and frameworks. Finally, we discuss short- and long-term scenarios for employing NDN to enable the Internet of Things.**

I. Introduction

The Internet of Things (IoT) vision, taken broadly, proposes to interconnect *things* of all kinds by leveraging the proliferation of increasingly small and affordable embedded devices for processing, sensing, actuation, and wireless communication. The global realization of this vision will easily exceed the scale of devices and data objects found in the current Internet by orders of magnitude [1].

The roll-out of IoT faces two fundamental and often conflated challenges. The first is how to enable all different types of digital *devices* that provide IoT functionality to communicate locally and globally. The second is how to consistently, securely communicate the data associated with the *things* themselves, once connectivity is achieved. The latter is the heart of the IoT vision, providing access to everything from door lock status and lighting levels in home automation to the flow of water measured by a municipal meter in a smart city, an individual's blood-glucose level, and the soil pH measured across a field by a truck-mounted sensor. Once this data can be retrieved in a secure and consistent manner, a whole host of exciting applications and research opportunities become achievable [2].

Current IoT frameworks, discussed in Section V, focus on interconnecting devices, primarily addressing the first challenge. Building up from the host-to-host communication paradigm of the Internet Protocol (IP), these frameworks conflate the embedded devices with their associated real-world *things* at the network level. They tend to emphasize solutions for device-to-device connectivity and then meet the applications' need of accessing the associated real-world data through a series of mappings. To fetch data about a *thing* itself, a typical application process, or a stack on its behalf, may have to traverse a long series of mappings among interface addresses, devices, channels, and subnetworks, each of which must be secured. Such mappings add complexity and brittleness to what are often simple communications of sensor data, actuation commands, and configuration operations. For example, consider a light (a *thing*). To control its intensity in a contemporary building control system, an application must be able to get packets to the appropriate VLAN and IP subnet, as well as know the lighting gateway device's IP address and protocol, before dealing with the light itself via an application-level identifier. While consumer devices have made this easier, often allowing web-based control over devices on home wireless networks, they do so by making assumptions about such mappings—for example, that all devices are on the same subnet—or rely on cloud services to achieve what is essentially local communication.

In this paper, we discuss how Named Data Networking (NDN) can be applied to improve and simplify such IoT communication. NDN is a proposed future Internet architecture that is a prominent example within the broader field of Information Centric Networking (ICN).[1] NDN fundamentally shifts the network communication model from host-centric to data-centric. Instead of sending packets between source and destination devices identified by numeric IP addresses, NDN disseminates *named data* at the network level, forwarding directly on names that carry application semantics. Moreover, each packet in NDN is secured at the time of production, allowing data replication anywhere in the network and preserving security properties of the data over its lifetime. NDN

[1]For more information on NDN, see [3], [4] and the project website, http://named-data.net. Given space constraints, we do not compare NDN with other ICN architectures here; see [5] for a recent survey.

enables applications to name the *things* of IoT, like the light in the above example, and have the network forward related data using those names directly. It enables both IoT applications and network infrastructure to operate with simpler, more consistent semantics, less brittleness, and increased security.

II. Example Use Cases

To illustrate the challenges of IoT and benefits of NDN, in the rest of the paper we use three (of many) representative use cases.

To create **smart homes**, IoT technologies have been widely integrated to monitor and control the environment of individual homes. Products like smart thermostats, smoke detectors, security cameras [6]; wireless light switches [7]; and intelligent door locks [8] have been introduced to the consumer market in recent years. In many cases, these devices allow remote monitoring and control using smartphones or computers from anywhere in the world.

New **personal health and wellness** applications leverage very small systems on chips (SoCs) that can fit into wearable devices, applying machine learning to reveal personally meaningful patterns in the data that they gather. For example, smart wristbands and watches [9], [10], [11] have long been available on the consumer electronics market. Such personal health devices embed multiple sensors to monitor a user's physical activity, body temperature, and heart rate, which are reported to the user's smartphone via Bluetooth LE or similar wireless technologies. In many cases, personal data is uploaded to cloud-hosted data warehouses for further analysis by software and/or human professionals.

The **precision agriculture industry** is applying IoT technology to improve productivity and yield. Applications in this domain demonstrate tight integration of IoT and machine learning technologies, where the IoT sensors generate a large amount of data that is fed into analytics frameworks for processing. For example, Fujitsu has partnered with Microsoft to use IoT devices to control greenhouses for improving the quality and yield of lettuce production [12]. Another interesting example is the HealthyCow24 solution from SCR Dairy [13], [14] that allows farmers to install motion sensors and microphones on cows to monitor their health and activity. HealthyCow24 applies machine learning algorithms to detect when a cow needs medical attention, and to select the best breeding opportunities to improve milk production.

Many such application examples use a model of request-response for named data, an approach that is similar to what NDN offers at the network layer. For controlling smart buildings, the Building Operating System Services project [15], uses hierarchical human-readable names to address devices. Open mHealth [16] makes consistently described data the "thin waist" of interoperability within an open ecosystem for health and wellness applications. To manage data from the mandatory RFID tags on Australian cattle, in [17] researchers employed the Global Sensor Network middleware [18], which implements a request-response pattern using REST over HTTP.

III. IoT Challenges

The IoT vision is broad and faces many challenges. This paper focuses on the significant networking challenges unmet by current IP-dependent solutions and other approaches that focus on host-to-host communication. IoT devices are fundamentally different from devices successfully networked with IP in the past such as mainframes, desktops, and laptops. They often have a constrained power budget, in many cases are mobile, have limited computational resources, operate in adverse environments—embedded in buildings and objects, buried, or immersed—and perhaps with intermittent connectivity. IoT networking are likely to employ multiple communication technologies simultaneously, each with different scope, security properties, and costs, such as cellular, WiFi, and low-power radios, USB, and serial links. They also may have minimal or non-existent user interfaces, limiting how users may participate in bootstrapping and configuration. These properties influence the following specific networking challenges. Additional in-depth discussion of challenges for IoT over TCP/IP, in comparison with NDN, can be found in [19].

A. Complex Solutions to Simple Communication Needs

Many envisioned IoT applications center around simple operations of *fetching data* and *controlling actuation*, with analytics in between. However, two innate characteristics of IoT, described above, lead to complex solutions for even basic communication over IP: communication technology diversity and resource constraints.

For example, diverse *things* make up a given smart home, agricultural field deployment, or body area network. Each device that provides *things* with connectivity may have various radios, wired interfaces, and serial links, all with their own addressing scheme, IP subnet, and mapping between network-layer packets and application-layer messages. To provide access to the *things* by integrating all of these elements requires either 1) local, application-level middleware to manage interoperability, or 2) pushing all data to cloud services, which we discuss in the next sections below.

IoT middleware and framework developers have the unenviable task of handling the collection and republication of data from a wide variety of devices and services. They have tended to standardize around simple, request-response APIs for named data that resembles examples throughout this paper. However, to provide this, they must configure and maintain mappings from interfaces to devices, and further to the intended named data and control points relating to the *things* themselves. Essentially, an overlay must be created that can deal with a wide variety of underlying communication technologies, all using the TCP/IP suite of protocols, or modified versions that fit the resource constraints of IoT. Simply managing IP and port address assignments is not enough; in more complex scenarios, a Layer 2 configuration must be created in parallel to ensure traffic flows among heterogeneous subnets. Typically, such configuration is done outside the middleware and creates a complex set of interrelated but independently managed elements. Security requirements further complicate the picture.

Maintaining such an overlay, however lightweight, on top of host-to-host communication is made even more challenging when available devices, interfaces, and channels change dynamically. Managing changes is likely to be resource-intensive, conflicting with the second characteristic of IoT environments discussed above. Further, in each of our use cases described in Section II, networks of *things* are put together by people who are not network experts. Complexity impacts usability, and thus the innovation attempted in new products in these target markets.

In summary, a significant amount of effort is expended in current systems just to get devices to be able to communicate by using the existing protocol suite, before one can even work with the *things* themselves. If this first step could be simplified and made robust to dynamically changing network environments, it would be a fundamental enabler for an Internet of Things, rather than a set of vertically integrated application ecosystems. NDN's proposal is to start with a secure request-response primitive as the thin waist of networking, as described in Section IV, building up IoT functionality using this primitive, as discussed in Section VI.

B. Limitations of Channel- and Session-based Security

The umbrella of IoT includes security-sensitive applications, with implications from personal privacy (e.g., health monitoring) and safety (home security) to vandalism (precision agriculture) and corporate espionage. While unsecured IoT networks are not an option, the security approaches used in common Internet applications are not a good fit for IoT. As discussed above, many emerging solutions coordinate device communication through cloud services to also centralize the security problem, although this does not realize the vision of interconnected ecosystems of loosely coupled devices that make up IoT—including the model of "fog computing" models [20] that distribute computation to the edge. Even those that emphasize local communication typically employ session- and channel-based semantics, such as in DTLS and its variants [21], that emerge from the TCP/IP paradigm of host-to-host communications [22]. These approaches are brittle in IoT environments made up of heterogeneous devices, overlapping networks with different administrative domains, and intermittent connectivity via multiple communication channels per device.

A simple example that illustrates the above concerns is HealthyCow24, which communicates over several interface types, including low-power low-rate wireless, Ethernet, USB, and RS232 serial port communication. Performing similar data acquisition and control tasks over each of the interfaces requires different networking stacks: power-aware delay-tolerant communication over low-power wireless, IP for Ethernet, file-based access for USB, custom protocol for RS232. Each stack has different security solutions, if any at all.

Further, securing a channel or session between devices does not tackle how to express identity (of a *thing*) beyond the addresses of devices, manage the provenance of data, express trust relationships among communicating elements, or handle key distribution itself—all of which are required for a robust Internet of Things [23].

Section IV-B explains how NDN secures data independently from communication channels, and Sections VI-C and VI-D explain how this building block can be used to manage trust and achieve data confidentiality.

C. Poor Integration of Local Communication

Local communication is at the heart of many IoT applications that require cooperation of colocated devices, a key part of the IoT vision [1]. The network layer is responsible for providing efficient support for direct communication between nearby IoT nodes, possibly leveraging local broadcast media, such as wireless, LANs, and multi-drop serial. Unfortunately, today's IP-based solutions face significant limitations in achieving application-level support, network-layer efficiency, and secure accessibility of local networks.

First, application support for local communication often requires bootstrapping from DNS and middleware to bridge the gap between application-level names and network-layer addresses. Second, applications and middleware that are built on IP's host-to-host communication model do not leverage the broadcast nature of wireless media typically used for IoT communications. Systems must either be carefully configured to leverage multicast over a constrained radio link, or more likely to use a unicast paradigm over broadcast channels, resulting in brittleness in the configuration if any one host changes its participation in those unicast communications. Finally, typical IoT environments involve multiple overlapping local communication domains—e.g., one for each wireless and wired medium—which require orchestration of Layer 2 bridges and configuration of IP subnets. Scoping of communication must be done either at the application layer or through the use of VLANs, subnets, and similar techniques.

While some IoT solutions using a host-to-host approach to communication may successfully tackle the above challenge for a particular network or device type, e.g., a local subnetwork of Zigbee devices or a consumer product that accesses cloud services by an end-user's WiFi, the vision of an integrated, interoperable ecosystem that involves local devices and computation as well as cloud services appears difficult to realize over current protocols like IP.

Section VI-B gives an example of how NDN can leverage local connectivity for bootstrapping, and Section VI-H discusses how NDN helps achieve the integration between local and global communications.

IV. Applying the NDN architecture to the Internet of Things

In this section, we describe the core NDN architecture, with a focus on how it can be applied to IoT environments to address the aforementioned challenges.

A. Basic Protocol: Named Data Retrieval

Named Data Networking (NDN) [3], [4] makes request-response for secured, application-named *Data* packets the

fundamental model of the network architecture—the "thin waist" of communication. Consumers request data by sending *Interest* packets that include names (or name prefixes) of the desired data, and the NDN network uses the names to retrieve the requested data. NDN names follow a hierarchical structure. Components can define the scope of the data ("`/LivingRoom`"), in order to properly forward requests. They can describe application-specific semantics ("`../Temperature`"). And, they can supply unique identifiers for specific versions or instances ("`.../201601121334`"). For example, in the smart home environment, a heating, ventilation, and air conditioning (HVAC) controller may issue requests—Interest packets in NDN—for "`/LivingRoom/Temperature`" data, requesting the current temperature measurement from the thermometer located in the same room. One of the thermometers in the room can respond to this request with a signed Data packet whose name extends the name in the Interest with a timestamp component "`201601121334`" for the specific data reading. After HVAC receives the response, it can verify that the data was created by an authorized thermometer and, if needed, take appropriate actions to adjust the room temperature.

The pattern of retrieving named data naturally matches the semantics of IoT applications. Section VI-A discusses specific approaches to naming for IoT in more detail. In monitoring and measurement applications, clients can use the Interest-Data exchange primitive to retrieve named sensor data over the NDN network. In actuation applications, controllers can use Interests to express the actuation commands, with the Interest names identifying the object and what needs to be done to the object, e.g., "`/LivingRoom/Lighting/OFF`". This way, without explicitly identifying the specific device, the command can be executed by the proper actuation unit, e.g., a floor lamp, that corresponds to the *thing* identified in the name. Such a *command Interest* carries information to authenticate the issuer of the command, and the response, a Data packet confirming the execution of the command, includes identification and authentication information of the specific actuation unit. In Section VII-D we discuss in more detail our previous work on authenticated actuation [24] and more recent work on security frameworks for constrained devices.

B. Data-centric Security

Security amidst heterogeneity is a critical challenge for IoT networks and applications. NDN's approach is to directly secure named data at the network layer. This ensures that receivers can validate a Data packet independently of where and how they obtained it and that only authorized parties can access the data. Combined with the expressive power of hierarchically structured names, this builds up security mechanisms that overcome many of the challenges of applying typical techniques from the current IP Internet.

In NDN, each Data packet is signed at the time of production, cryptographically binding the name and the payload of the data.[2] Information about the signing key, i.e. the name of signing key certificate, is recorded in the *KeyLocator* field[3] of the Data packet, establishing data provenance and allowing reconstruction of the authentication chain to verify the validity of the data. Section VI-A highlights how the power of hierarchically-structured names can be leveraged to easily enable complex trust relationships with automated signing and verification. To handle different application scenarios, the NDN team is currently exploring various cryptographic mechanisms, such as asymmetric RSA and ECDSA signatures, and more lightweight approaches appropriate for typical IoT messages, such as HMACs and hash chains.

This approach of focusing on *securing the data* rather than securing a channel or session provides a building block for meeting IoT security requirements that is independent of the specific communication technology used to carry bits about a thing to and from a device. By securing the named data directly, NDN enables IoT data to traverse boundaries between heterogeneous network environments without losing security properties. It is also possible to store data in an application-transparent manner in in-network caches and persistent data storage. NDN allows IoT applications to freely distribute data to any place in the network without requiring them to trust any intermediate node to keep data intact and confidential.

In Sections VI-C and Section VI-D, we discuss in more detail the data-centric approach of NDN to security: mechanisms to manage trust and to sign and authenticate data, and mechanisms to encrypt data and grant access permission to it. We also discuss our previous work in securing building management systems via NDN [26].

C. Name-based Forwarding

The NDN network forwards Interest packets based on the names they carry. This fundamental distinction between NDN and IP architectures is what enables NDN-based IoT applications to operate directly on packets that describe *things* and their data.

At each hop, an NDN forwarder first checks an Interest for locally cached Data in its *Content Store* that either matches the Interest name exactly or takes the Interest name as its prefix; it uses the matched Data packet to satisfy the Interest. If no match is found, the forwarder checks its *Pending Interest Table (PIT)*, which keeps track of recently-received Interests and their incoming interfaces. If the same Interest has already been forwarded and recorded in the PIT, the forwarder aggregates identical Interests. Otherwise, the forwarder records the Interest in the PIT, looks up the Interest name in its forwarding table (FIB) using longest prefix match, and propagates the Interest according to the *forwarding strategy*. A matching Data packet is returned to the consumer(s) that requested it by following the "bread crumbs" left in the PITs of the forwarders along the path. The forwarders purge unanswered PIT entries based on the *lifetime* field in each Interest. This

[2]For sensitive applications, the payload and parts of the name can also be encrypted at the time of production.

[3]See [25] for more information on the NDN packet format.

"soft state" mechanism effectively prevents NDN nodes from being overwhelmed by a large amount of unsatisfied Interests. In order to achieve consumer-driven flow balance, the architecture requires that one Interest brings back at most one Data.

NDN's name-based stateful forwarding can be used to realize other important features for IoT, such as a delay-tolerant style of communication and fast local recovery from losses, as well as hop-by-hop congestion control [27]. Specifically: 1) There is no need to configure network-dependent addresses for each interface of every device. 2) Nodes advertise and discover application names directly at the network layer, avoiding the necessity of additional indirection from names to interface identifiers. 3) The stateful forwarding plane allows fine-grained control and adaptation of forwarding decisions at each node, adapting to network connectivity changes. 4) Opportunistic in-network caching facilitates efficient data dissemination in dynamic communication environments with intermittent connectivity and link diversity.

D. In-network Storage

Securing data directly enables even simple NDN applications to use the benefits of in-network storage. NDN routers can opportunistically cache the Data packets they forward, enabling efficient dissemination of popular data and facilitating local recovery. Different classes of devices can adjust the cache size and management policy based on available storage, power, and processing capabilities. In addition to opportunistic caches, NDN networks can include persistent data repositories (repos) that provide long-term managed storage for data [28].

IoT applications may leverage each type of in-network storage at the same time. For example, sensors with limited storage deployed in an agricultural field can transfer monitoring data immediately after its acquisition to a nearby repository. A remote controller can later retrieve this data from the repository, more effectively using available bandwidth and consuming less energy. When measurement data needs to be stored in multiple repositories, in-network opportunistic caches and NDN Interest packet aggregation will assist to effectively multicast data to the repositories. In wireless mesh networks, in-network caching can significantly improve efficiency of data dissemination: each intermediate mesh forwarder can cache recent Data packets to serve retransmitted requests in the future. In typically disconnected environments, "data mules" can carry Data packets in their in-network storage, enabling data to be diffused even when consumers and producers never have a directly connected channel between them.

V. Related work

NDN provides a single network protocol that "changes the game" for deploying the Internet of Things, by providing name-based, request-response semantics at packet granularity. In this section, we briefly discuss a few representative IoT frameworks that achieve similar high-level functionalities to what NDN provides, but take fundamentally different approaches in the lower-layer details. Their relationship is illustrated conceptually in Figure 1.

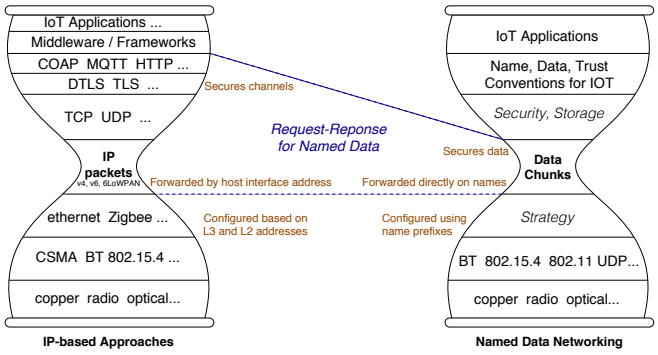

Fig. 1. NDN changes the "thin waist" of IoT networking from communication between source and destination hosts to dissemination of named, signed data.

An open architecture that is gaining popularity among many IoT vendors is specified by a collection of IETF standards, including 6LoWPAN [29], RPL [30], DTLS [31] and CoAP [32], which can be used together to provide *request-response* exchange of data, with session-based security semantics [33]. 6LoWPAN defines the link adaptation layer for transporting IPv6 packets over IEEE 802.15.4 [34] networks. RPL is the routing protocol for constrained networks to facilitate IPv6 forwarding. CoAP is an application-layer protocol that can run on top of UDP and 6LoWPAN to provide an HTTP-style communication interface and optionally use DTLS to provide channel security. Collectively, these standards provide a full-stack architecture that supports IoT applications in constrained environments. While they achieve the functionality of naming resources, request-response communication, and caching at the application layer, the underlying layers must deal with interconnecting participating devices using a host-to-host model with secure channels [19], resulting in the complex mapping problem described earlier. The mismatch between application and network layers results in inefficiency and brittleness to configuration changes that impact host to name mapping or security perimeters. Dealing with such contingencies yields bulky system implementations.

AllJoyn [35] is an application-layer framework that provides a common software interface across various network technologies (TCP/IP, Bluetooth, etc.) and heterogeneous systems. It uses the abstraction of a message bus that spans across multiple devices to interconnect different applications, hiding the details of the underlying connectivity. AllJoyn applications advertise and discover each other's services via unique bus names that are similar to hierarchical NDN names. One major difference between AllJoyn and NDN is that AllJoyn still preserves the notion of a connected *session* between AllJoyn applications, which is provided by the networking technology under the bus abstraction. Network security is implemented by protecting the sessions with session keys that are established by a key exchange process. In contrast, NDN provides a pure data-centric communication semantics directly at the network layer with name-based routing and forwarding, which eliminates the concept of sessions among the network nodes.

The ZigBee Alliance standardized the ZigBee protocol stack [36], which also runs on top of IEEE 802.15.4 networks like 6LoWPAN. ZigBee by default does not use IP, but defines its own network layer that extends 802.15.4 MAC layer operations to support device addressing, network formation, routing, and forwarding. On top of the network layer, ZigBee provides an application framework that allows developers to specify common application behaviors using service and device *profiles*, which is essential for cross-vendor compatibility. ZigBee also provides a service discovery framework for ZigBee nodes to explore applications and neighboring devices by sending queries that contain profile IDs. This profile-based service discovery has been incorporated by other industrial IoT architectures as well, including Bluetooth and Bluetooth LE [37]. Similar to the frameworks discussed above, ZigBee focuses primarily on interconnecting individual devices at the network layer and implements application-layer services, such as profile-based discovery, on top of the device-oriented communication model. This is fundamentally different from NDN's data-centric semantics, which support naming with application semantics directly at the network layer.

VI. Achieving IoT Framework Functionality

In this section, we describe how to employ NDN's core network-layer protocol to achieve IoT framework functionality.

A. Naming Things, Devices, and their Data

NDN uses application-meaningful names at the network layer, which makes proper naming design a high priority when creating an NDN application. Although NDN applications are free to choose any naming model, following common naming conventions and trust relationships, and data payload formats can enable interoperability at the network and application layers. Examples of NDN naming schemes exist for applications including building management [26], lighting control [24], person tracking [38], video conferencing [39], scientific data [40], network routing [41], and others.

NDN names are very flexible. In this paper, we use hierarchical, human-readable English-language names for clarity, but names can also include machine-readable and encrypted components. A forwarder simply treats each name component as a string of bits. Name length is also flexible, so that applications can utilize short names to reduce overhead. Hierarchy is not required, but can be leveraged to support forwarding as well as security, aggregation, and other critical features. NDN also allows the use of flat names, which are simply a special case of hierarchical names that have only one component.

This paper's examples follow emerging conventions in our application research for naming *things* and their data, as well as devices themselves. In these examples, typical names include a root prefix that describes the scope and can be used for forwarding,[4] a middle set of components that name a thing or device, and final components that name the specific instance of data. For example, "`/AliceFarms/field/21SUJ22850705/soil/pH/201412021339`" could name an NDN data object corresponding to a soil pH measurement in a given U.S. National Grid (USNG) grid square for land owned by Alice Farms (the *thing*), measured on December 2, 2014 at 1:39PM. Such a packet can be signed by a device-specific key (e.g., "`/AliceFarms/devices/sensors/field/soil/685b359aec5b/key/27`" for the Alice's Farms sensing device with serial number `685b...`), providing provenance for the data. For brevity, such examples omit longer prefixes or the use of forwarding hints [42] that may be needed to provide global access to the named data.

In constrained cases, long, human-readable names may lead to undesirable overhead. A number of techniques can be employed to let applications work with efficient names while preserving NDN's benefits. One may use a combination of efficient name component encoding schemes, application-side lookup tables, which can be easily distributed as Data packets, and encapsulation of packets by more powerful nodes on the network using techniques such as those discussed in [42].

The remainder of this section explores how in NDN devices can get their names and keys, and how data consumers can authenticate the validity of a Data packet and, if necessary, decrypt it. We will also illustrate how to implement data aggregation and publish-subscribe communication using NDN primitives, and conclude with techniques for efficient multi-party communication and bridging local and global networks.

B. Bootstrapping and Discovery

IoT networks must handle ongoing addition, removal, and configuration of devices and services. In IP networks, this involves establishing and securing the many device-, network-, and application-level mappings. In NDN, devices that produce data must be configured with the naming prefixes to use and appropriate signing and encryption keys. Devices that consume data need to obtain proper trust anchors and decryption keys.

NDN enables nodes to request data without having a name or address of their own. Bootstrapping in NDN can thus be achieved through well-known naming and trust schemes for initial configuration data. For example, a new device can express a special Interest with a well-known name prefix and information identifying the device and its function (e.g., "`/local/discovery/lighting/serial=123456`"). When received by a bootstrap controller, this Interest can initiative the device's initial configuration, secured through a pre-shared secret, such as an out-of-band PIN, a pre-scanned barcode, etc. During the initial configuration, the device can be given its own identity "`/LivingRoom/_fixtures/123456`" that can be used for further configuration, associated with a *thing* ("`/LivingRoom/Lighting/TableLamp`", given a set of functions ("`../ON`", "`../OFF`"), and configured with a proper set of trust anchors and trust model(s).

Once devices are named and their root(s) of trust established, capability and service discovery can be implemented by the machine-to-machine (M2M) exchange of metadata associated by convention with each prefix, e.g., "`../TableLamp/`

[4] Often called the "routable prefix", early components in names are used by NDN forwarders to direct Interests towards possible data locations and/or restrict propagation of Interests and Data within the defined area.

`_capabilities`". For example, a controller may need to obtain device information such as the model and the manufacturer, while the neighboring devices may want to know what kind of services the new device can support. In both cases, metadata with well-known names can be used to describe device information like "`.../TableLamp/_manufacturer`" or implement typical IoT *profile* mechanisms. In larger networks, name synchronization schemes can be used to efficiently discover and manage new devices; this is discussed in more detail in Section VI-G. In Section VII-B, we describe NDN-IoT, a software package that implements bootstrapping and discovery frameworks for generic smart home applications.

C. Schematizing Trust

Managing trust within IoT networks is still an open challenge within the broader area of IoT security [43]. In NDN, trust decisions can leverage the structure of names to schematize decision-making on a packet-by-packet basis that does not require channel- or session-based semantics.[5]

As each Data packet has its own name and also carries the signer's name in the *KeyLocator* field, NDN enables applications to express trust relationships through rules that regulate allowable relationships between the Data packet name and signing key name. Depending on the specifics of an application, the desired trust model can be quite fine-grained. For example, consider a new wearable wellness device that Alice just bought. To bootstrap, Alice would use a mobile health app on her phone to performs two functions:

- configure the wearable device to publish activity data under the prefix "`/Alice/health/activity`", and
- assign the device a key ("`/Alice/_devices/fitbit,id=2211213/key`"), signing it with Alice's mobile health root key.

Later, during physical activity, the wearable device would periodically—or, as a response to Interests from the mobile health application—generate step count data under the configured prefix, signing it with the assigned key. For example, a packet for January 10, 2016 3:34pm would be named "`/Alice/health/activity/step_count/20160110-153400`" and signed by "`/Alice/_devices/fitbit,id=2211213/key`").

In this example, besides simply verifying signature validity, the analytics application would want to restrict Alice's health data to be signed by keys within the "`/Alice/_devices`" namespace only. When an analytics application on the phone retrieves the step count data, it can also retrieve the key that signed the data, and, recursively, the keys that signed the keys. The chain from data to key (to key...) is considered verified when these recursive operations terminate at an already trusted key, such as Alice's mobile health root key. In this way, the relationship between data and key's hierarchical names gives the context for data authentication.

We have proposed a policy language to express various trust models in terms of relationships between data and key names [44]. With this formalization, it is possible to automate authentication and simplify key bootstrapping. Note that in resource-constrained contexts such as IoT, generation, assignment and evaluation of keys can be done on more powerful devices and the resulting trusted keys for publishing and consuming data can be stored on the constrained nodes.

[5]NDN can support channel- and session-based solutions as well, but these schemes inherit the limitations of perimeter-based security in highly heterogeneous IoT networks, as well as brittleness with intermittent connectivity.

D. Name-Based Access Control

Just as NDN enables trust to be evaluated independently of how data is communicated, it can also provide channel-independent, data-centric confidentiality through per-packet encryption. Building on schematized trust, names can further be used to organize fine-grained access control, an example of which is found in recent work on Name-based Access Control [45]. NAC enables a data owner, like Alice in the preceding example, to enforce access control policies based on data names. It aims to enable the principle of least privilege security to be applied to NDN data access.

Utilizing NDN's power of fetching named data, NAC makes use of an additional access control namespace, which is parallel to the actual data namespace, to facilitate the distribution of encryption and decryption keys. For example, a mobile health application can designate the namespace "`/Alice/activity/NAC/read`" for the access control for Alice's activity data by publishing in this namespace:

- the public (encryption) key "`/Alice/activity/NAC/ekey`" to be used by wearable devices to encrypt produced data under "`/Alice/activity`" prefix;
- the private (decryption) key, encrypted for each authorized device, application, or group of applications and devices: "`/Alice/activity/NAC/dkey/FOR/Alice-Family/health/...`", "`/Alice/activity/NAC/dkey/FOR/UCLA-Health/physicians/...`", etc.

To encrypt the data, the wearable device would simply retrieve the encryption key. To access the data, the data owner (Alice) will need to give explicit permission to each legit device or user by encrypting the data decryption key with their public key, then simply publish these encrypted decryption keys.

The above illustrates the high-level idea of NAC; the proposed protocol includes several additional elements to improve performance and security [45]. The building management application in Section VII-A includes an experimental implementation of the NAC protocol.

E. Data Aggregation

After the IoT devices establish network connectivity and trust relationships, a common function in IoT networks is to aggregate data in a subsystem. Just as in the previous sections, names can be leveraged in this case as well.

IoT systems can generate a large amount of data on an ongoing basis. As it is often inefficient and, in some cases, infeasible to archive and analyze the raw data, it is common for IoT systems to pre-process and aggregate the raw data stream immediately after the data is captured. The processed and aggregated data is then transferred to some permanent

storage for future retrieval and analysis by other IoT applications. If the system can determine in advance what kind of analysis the high-level IoT applications are interested in, those analytic operations can be distributed down to the intermediate gateways who can execute them as data becomes available.

NDN's data-oriented semantics facilitate such data aggregation. NDN IoT applications first define the naming convention of how to encode sensor data information, such as the data type, the location of measurement and the time when it is taken. The aggregation gateways then construct Interest packets following that naming convention, and retrieve the data over the NDN network. The gateways can either pull the sensor data periodically or use a publish-subscribe framework, as described in Section VI-F, to subscribe to certain sensor data. In-network caching makes data retrieval efficient and fail-safe, especially if the device deployment features a hierarchical topology, where the data is pulled from the lower level of the hierarchy to the higher level and cached along the way. The building management application discussed in Section VII-A implements this approach as proof-of-concept.

F. Application-Level Publish-Subscribe

Publish-subscribe (pub-sub) is a common communication paradigm for asynchronous messaging applications. For example, in simple IoT applications employing a pub-sub model, sensors "publish" their data as it is generated, while aggregators, analytics engines, and actuators "subscribe" to such data sources of interest to receive notifications of new data. It is a common misconception to confuse NDN's basic Interest-Data exchange model with the pub-sub pattern. The core NDN protocol implements a pull-based request-response paradigm. To ensure flow balance at the network layer, it does not directly provide persistent subscriptions with publisher-initiated communication of new data. However, it is straightforward to build application support for pub-sub communication using NDN. Hierarchical NDN names can be used to define the categorization of various Data packets. The Interest-Data exchange mechanism enables asynchronous fetching of existing data. Data publishers announce the name prefix, through the NDN routing system, under which they will publish new data. Subscribers issue Interest packets with those names. To implement *push*-style notification for new data at the application layer, a consumer needs only to ensure that this soft state is refreshed at the minimum acceptable notification time, which can be supported by the library. PITs in each node collapse duplicate Interests and provide efficient multicast distribution when new data is available. The building management application in Section VII-A implements this mechanism to obtain the sensor data it uses in multi-level aggregation as described in the previous section.

G. Sync: Efficient Multi-party Communication

Pub-sub, while useful, is challenging to employ when there are many producers in the same namespace, a likely scenario in IoT. For example, consider all of the data that might be published, by all sorts of devices, as part of the Smith family smart home, "`/smith/family/house`".

Building on Interest-Data exchange, NDN can provide new types of high-level data dissemination functionality that are useful in such circumstances. Distributed synchronization of shared data sets is one such example. In NDN, synchronization (sync) refers to a multi-party communication paradigm that aims to efficiently reconcile collections of named data. Specific example protocols are given in [46], [47], which allow the participants in the sync group to exchange their knowledge about data published under a namespace. When new data is generated, nodes advertise their updated knowledge about the collection, using tools such as digest trees to represent them efficiently, and synchronize with other nodes.[6]

NDN sync is suitable for high-level device, thing, and service discovery as well as for implementing lightweight data sharing across multiple IoT devices and repositories in a local environment. Names that are synchronized can correspond to prefixes that identify *things*, enabling efficient discovery on shared media, or reachable by multicast, as an extension to the basic bootstrapping process described in VI-B. Or, the names of Data objects themselves, such as sensor readings, can be synchronized, enabling multi-publisher scenarios.

NDN's approach to synchronization is session-less and based on representing the knowledge of each participant. This makes it particularly useful to assist information dissemination in disruptive environments where the network exhibits intermittent connectivity, dynamic topology, or can communicate over multiple media. For example, in agricultural IoT applications, solar-powered sensors may need to periodically shut down the wireless network interface to conserve energy, leaving only a small time window for communication. In those environments it is often impractical to achieve long-lasting, session-based data transfer. Synchronization, on the other hand, allows a group of IoT nodes to quickly discover the missing (or new) data over short-lived ad-hoc links, which can be used to implement efficient message forwarding across wireless mesh networks with lots of sleeping nodes.

H. Integrating IoT with the Global Internet

Finally, the IoT vision is of an *Internet* connecting *things* at a global scale. While the discussion above has focused on local communication, NDN is being developed as a future Internet architecture suitable for a wide variety of applications deployed globally. For example, scalability of NDN forwarding is discussed in [48], and some of the wide variety of research on NDN's applications and opportunities is covered in [49].

Local and global NDN networks can be bridged by leveraging the approaches introduced in previous sections: data-centric security protects authenticity and confidentiality of data without relying on secure channels, name-based forwarding and signature verification can be used to limit traffic

[6]Unlike more familiar cloud-based synchronization solutions (e.g., Dropbox, Google Drive), the sync protocols in NDN are decentralized and serverless, and further benefit from the in-network caches in the NDN forwarders.

that traverses local networks, while caching, persistent in-network storage, and Interest aggregation in forwarders make it straightforward to handle many consumers of data from resource-constrained devices. Forwarding hints and encapsulation, discussed further in [42], can be used to bridge namespaces.

In practice, an application-level example that has proved useful in our work is IoT integration with existing Web technologies. NDN allows easy integration of Web components with IoT applications in a protocol-independent fashion, since the network layer and all application protocols share the same data unit: the NDN Data packet. This universal data unit can be conveniently transported across different network environments, storage, and platforms via basic Interest-Data exchange. For example, as a proof-of-concept, the NDN team developed a JavaScript library that implements NDN communication support directly in web browsers, which is used to provide user interfaces that directly access data from IoT nodes in many of the examples in the next section [50].

VII. Implementations

In this section, we describe a few ongoing research projects that apply NDN to various IoT scenarios.

A. NDN-BMS

NDN-BMS [26] is an application-driven project that designs and implements an NDN-based building management system to be used by facility management personnel. The prototype system deployed on the UCLA campus captures, archives, and visualizes time-series data generated by industry standard sensors located in campus buildings. In NDN-BMS, the sensor data namespace is based on naming the *things* being measured, such as electrical current and chilled water flow, according to the physical hierarchy of the building structure. For example, the prefix "/bms/building1/floor1/room1 covers the data generated in Room1 on Building1s first floor, including such child Data objects as ".../current/201602101210" for the current draw measured for the room at 12:10 p.m. on February 10, 2016. This namespace facilitates routing and caching: each node could register the physical location name it represents with its upstream node, expect to receive Interest for data generated by itself or its downstream nodes, and cache data from downstream for later access. The system employs a basic encryption-based access control scheme that limits data access within a group of authorized users.

Mini-BMS [51] is an extension of this work that adopts a data namespace design similar to NDN-BMS, while incorporating more recent work on schematized trust (Section VI-C), name-based access control (Section VI-D), and data aggregation. It uses Mini-NDN, a Mininet-based NDN network emulation tool [52], to emulate nodes in a larger BMS network driven by real data from the UCLA campus. Following the basic design in Section VI-E, each node implements application-level pub-sub semantics by keeping outstanding Interests for the data produced by its child nodes to gather the data for aggregation in a fixed time window. The system uses a hierarchical trust schema in which the certificate of a child node is signed by its parent, and a predetermined root of trust is installed on each node. Its name-based access control system allows managers of the system to configure data access privileges based on the *thing*'s physical location and data type.

B. NDN-IoT

NDN-IoT [53] is a development toolkit for setting up simple smart home networks. It provides an experimental platform running on Raspberry Pi devices which can be outfitted with a number of simple sensors via GPIO pins. NDN-IoT contains templates for two types of nodes—controllers and devices—that implement the basic bootstrapping and discovery mechanism described in Section VI-B. The controller node maintains a directory of available services, represented by NDN names and an internal mapping from service names to the supporting devices on the network. It controls the addition and removal of devices by requiring a pairing code provided by a device to be entered by the user during initial bootstrap. When a new device is added, it also issues credentials, e.g., identity certificates, that allow for authenticated interactions between devices.

The other type of node is a device node, which manages sensors and/or actuators (the *things*) connected to it. When devices are added, they provide a description of their capabilities to the controller who will add this information to the service directory so that other devices can search for the services they need. For example, the NDN-IoT toolkit includes a sample application that incorporates infrared proximity sensors and an HDMI connection to a television that can process Consumer Electronics Control (CEC) commands. The sensing process consults the service directory to discover the television control service provided by a different device and uses that service to switch the television on and off depending on room occupancy.

C. NDN over Arduino

Arduino single-board microcontrollers represent a class of constrained devices with a low-power, slow-speed CPU and a few kilobytes of RAM and Flash. When deploying IoT applications in wide-area infrastructure-less environments such as agricultural fields, it is common to employ sensors and actuators running on such constrained hardware platforms. To bring NDN applications to such platforms, we developed a special version of the NDN client library called NDN-CPP Lite [54].

The Lite API, combined with HMAC signature support, can be used to fit an NDN producer application in a few tens of kilobytes. Given the highly constrained memory and CPU, NDN Arduino applications may be "hard-wired" to sending and receiving Interests under a single namespace. In these cases, the memory and processing footprints can be further reduced by eliminating the dynamic data structures of PIT, CS, and FIB. For example, the "NDN over Bluetooth Low-energy" [55] project uses NDN-CPP Lite to implement a demonstration producer application on an RFduino device

using Bluetooth Low Energy interface. This project demonstrates the possibility of adapting an NDN application to run on a Arduino-class device, with open engineering challenges further discussed in Section VIII.

D. NDN-ACE

When running IoT applications on constrained devices, a common challenge is that the device themselves may not have enough storage or computational power to support complex security mechanisms, such as maintaining per-device security materials for each peer or executing expensive public key cryptography. NDN-ACE [56] is an access control framework for securing actuation operations using constrained IoT devices, which is designed to meet those challenges. It adopts a protocol architecture where the constrained actuators offload the authorization and key management tasks to a trusted third-party called an *authorization server* (AS) that runs on more powerful platforms. It assumes that a basic bootstrapping mechanism described in Section VI-B is available for establishing a trust relationship between the devices and the AS.

In NDN-ACE, the actuator generates a root symmetric key and shares the key with the AS. The AS authorizes the access request from the client devices through identity verification via schematized trust as described in Section VI-C. It then computes per-client, per-service access keys, derived from the root key via HMAC chaining. Upon receiving the command Interests, the actuator recomputes the access key using the local root key and the client information carried in the Interest packets, and then verifies the command signature using the derived key. Compared with the simple CoAP+DTLS approach, NDN-ACE avoids the overhead of maintaining secured sessions and key materials per client, which makes it suitable for constrained devices and a *thing*-based naming approach.

VIII. Open Problems

Several open problems exist for realizing IoT over NDN. This section discusses some of the most significant.

A. Naming with Multiple Hierarchies

Through our application research, it has become clear that IoT applications often desire to publish the same data under different namespaces, in order to simplify data discovery and facilitate access to data. For example, in NDN-BMS (see Section VII-A), applications might wish to organize data by both *data type* (e.g., voltage) and *location* (e.g., which building). This can simplify data retrieval and access control. For example, using the NAC framework described in Section VI-D, a consumer who is authorized to access all the voltage data only needs to register its read-access under the "`/Voltage`" prefix, rather than having multiple registration under the "`.../Voltage`" sub-namespace of every location prefix. Given that an NDN network supports data retrieval by one-dimensional names, the challenge of supporting multi-dimensional naming is two-fold. First, the number of possible combinations grows exponentially with the number of components in a name. Second, multiple combinations of name components lead to names not present in router FIBs. For example, if routers propagate reachability to data based on locations, they know how to forward interests with prefix of "`/location/voltage/`", but not "`/voltage/location/`". At the time of this writing, several solutions are being explored to ensure that the Interests designated to different namespaces can be satisfied, including approaches at both the network and application layer, as well as hybrids of the two.

B. Routing over Infrastructure-less Environments

Network routing protocols provide the essential support for scalable and efficient packet forwarding in an established network infrastructure, without having to flood packets through the network. However, many IoT systems are deployed in infrastructure-less environments, e.g., distributing the sensors and actuators in agricultural fields or embedding them inside the structure of a large building, where running a routing protocol is infeasible. The key to solving this challenge in NDN is to use the expressiveness of NDN names to encode information that can assist Interest forwarding, and to employ the stateful forwarding layer to make intelligent forwarding decisions. For example, in an agricultural monitoring system like the example of Section VI-A, data names that contain geographic grid coordinates can be used by infrastructure to guide the Interest toward the location where nodes are capturing and storing the data. Additionally, the NDN forwarding layer maintains soft-state information about the reachability of the names that have been requested over the network, which allows each forwarder to independently adapt to the network environment, whether static or dynamic. Per-prefix forwarder strategies complement traditional routing announcements by enabling powerful and application-specific forwarding logic in the NDN data plane. It remains an active research area for how to combine all the new features provided by NDN to support real applications at scale.

C. Implementation for Highly Constrained Devices

Many IoT systems are deployed on constrained devices that typically operate on battery power and have stringent requirements on energy efficiency. As demonstrated by the NDN over Arduino project mentioned above, it is feasible to deploy NDN on microcontroller-class devices which typically have tens of kilobytes of RAM and flash, and low-power processors clocked at tens of megahertz. However, limitations in memory, processing, and power do create engineering challenges in the implementation of the core NDN protocol and higher level frameworks.

Unlike stateless IP forwarding, NDN's stateful forwarding mechanism requires every node to maintain PIT, CS and FIB tables. Due to memory limitations, constrained nodes cannot keep track of a large number of pending Interests or cache a lot of data. The number of FIB entries will also be limited, which can be a problem for mesh networks where no default route is available for the intermediate forwarders. NDN's requirement of per-data packet signatures also requires careful selection and optimization of authentication schemes that are appropriate

for constrained platforms. Finally, energy limitations require higher-level protocols and frameworks, such as NDN sync, to minimize network transmission operations and always consider sleeping nodes in their protocol design.

D. Push-Style Data Collection

As a specific way to reduce network transmission and accommodate the sleep time of power-constrained nodes, some have proposed to change NDN's Interest-Data exchange model to allow a producer, such as a constrained sensor node, to push out data as soon as it is produced without having to stay online and wait for an incoming request. It is feasible to engineer such optimized solutions for a local environment. There are at least two ways to implement data push at the network layer in NDN. For data that is small in size, an energy-constrained device may include the data itself in an Interest packet to send to a less constrained collector. Another engineering optimization would be to establish a stable PIT entry on a forwarder that accepts unsolicited Data packets from a nearby sensor, eliminating the need for Interest packets. This second approach is distinct from the pub-sub framework described in Section VI-F, where subscribers assume a relatively stable network forwarding plane and keep refreshing the outstanding Interests for the subscribed data.

We would like to emphasize that such engineering approaches trade off functionality to obtain effective local optimization. As such, they should not be considered as general solutions for large-scale environments. For example, it is important to keep NDN's Interest-Data exchange communication model intact for flow balance and to prevent abuses of these optimizations, including data flooding through such hypothetical stable PIT entries.

IX. Conclusion

This paper aimed to show that the semantics of NDN naturally fit the inherent requirements of IoT applications, consequently the Internet of Things can be enabled at scale by NDN's data-centric model for networking.

NDN enables applications to name *things* and their data, and have the network forward packets directly based on those names, addressing core challenges of the IoT vision by closing the gap between application and network semantics. Instead of building up new layers to achieve request-response communication of named data, as today's frameworks do, NDN implements this functionality at the network layer as the common "thin waist". Rather than struggling to define and manage security in terms of subnetworks, channels, and sessions that are largely orthogonal to application security requirements, NDN's data-centric security solutions provide a robust alternative to building up granular, packet-level authentication and access control. It does so for realistic IoT scenarios, where devices use a variety of means to communicate in networks supporting heterogeneous applications. The architecture also naturally and effectively supports local machine-to-machine communication, while providing mechanisms for secure integration with global networks. The paper described the current design and implementation work in each of these areas.

Here, we introduced the basic NDN protocol and illustrated how to build framework-level solutions on top of it without any use of IP. Because NDN can operate over any medium that can carry bits, it can also be deployed over existing IP architecture. In fact, this is how the NDN testbed currently operates.[7] This suggests an evolutionary path for IoT deployment, where NDN can be used natively in IoT subnetworks, while gateways between subnetworks can use NDN over IP transport for interconnection.

We are continuing to validate the applicability of NDN's core protocol and the framework concepts given above by developing prototype applications and software packages that target various IoT use cases. There are still challenging open problems, such as those described in naming, routing, and power-efficient communications. We invite researchers from different backgrounds to join the NDN community and explore this exciting way to realize the vision of a Named Data Internet of *Things*.

Acknowledgment

This work has been supported by the National Science Foundation under award CNS-1345318, CNS-1345142, CNS-1455794, and CNS-1455850, as well as by Huawei and Qualcomm Research.

References

[1] R. Want, B. N. Schilit, and S. Jenson, "Enabling the Internet of Things," *Computer*, no. 1, pp. 28–35, 2015.

[2] C. C. Aggarwal, N. Ashish, and A. P. Sheth, "The Internet of Things: A survey from the data-centric perspective," 2013.

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '09. New York, NY, USA: ACM, 2009, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1658939.1658941

[4] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2656877.2656887

[5] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 2, pp. 1024–1049, 2014.

[6] Nest Labs, "Nest Thermostat," https://nest.com/thermostat/meet-nest-thermostat/, accessed on Jan. 29, 2016.

[7] Belkin International, Inc., "WEMO Light Switch," http://www.belkin.com/us/p/P-F7C030/, accessed on Jan. 29, 2016.

[8] KIWI.KI GmbH, "KIWI Homepage," https://kiwi.ki/en/, accessed on Jan. 29, 2016.

[9] Fitbit Inc., "Fitbit Official Site," https://www.fitbit.com/, accessed on Jan. 29, 2016.

[10] Misfit Inc., "Misfit Homepage," http://misfit.com/, accessed on Jan. 29, 2016.

[11] Jawbone, "Jawbone Homepage," https://jawbone.com/, accessed on Jan. 29, 2016.

[12] C. Arkan, "How IoT enables smart agriculture," https://www.microsoft.com/enterprise/industry/caglayan-arkan-blog/articles/how-iot-enables-smart-agriculture.aspx, accessed on Jan. 29, 2016.

[13] SCR Dairy, "HealthyCow24 Solution," http://www.scrdairy.com/cow-intelligence/hc24-solution.html, accessed on Jan. 29, 2016.

[7] See http://named-data.net/ndn-testbed/, which uses UDP tunnels between NFD [57] forwarders.

[14] L. Heikell, "Connected cows help farms keep up with the herd," http://news.microsoft.com/features/connected-cows-help-farms-keep-up-with-the-herd/, aug 2015.

[15] S. Dawson-Haggerty, A. Krioukov, J. Taneja, S. Karandikar, G. Fierro, N. Kitaev, and D. Culler, "Boss: building operating system services," in *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, 2013, pp. 443–457.

[16] D. Estrin and I. Sim, "Open mhealth architecture: an engine for health care innovation," *Science*, vol. 330, no. 6005, pp. 759–760, 2010.

[17] K. Taylor, C. Griffith, L. Lefort, R. Gaire, M. Compton, T. Wark, D. Lamb, G. Falzon, and M. Trotter, "Farming the web of things," *Intelligent Systems, IEEE*, vol. 28, no. 6, pp. 12–19, 2013.

[18] K. Aberer, M. Hauswirth, and A. Salehi, "The global sensor networks middleware for efficient and flexible deployment and interconnection of sensor networks," Tech. Rep., 2006.

[19] W. Shang, Y. Yu, R. Droms, and L. Zhang, "Challenges in IoT Networking via TCP/IP Architecture," NDN Project, Tech. Rep. NDN-0038, February 2016.

[20] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[21] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, "Lithe: Lightweight secure CoAP for the Internet of Things," *Sensors Journal, IEEE*, vol. 13, no. 10, pp. 3711–3720, 2013.

[22] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security Challenges in the IP-based Internet of Things," *Wireless Personal Communications*, vol. 61, no. 3, pp. 527–542, 2011.

[23] R. Roman, P. Najera, and J. Lopez, "Securing the Internet of Things," *Computer*, vol. 44, no. 9, pp. 51–58, 2011.

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

[25] NDN Project Team, "NDN Packet Format Specification," http://named-data.net/doc/ndn-tlv/, accessed on Feb. 11, 2016.

[26] W. Shang, Q. Ding, A. Marianantoni, J. Burke, and L. Zhang, "Securing building management systems using Named Data Networking," *Network, IEEE*, vol. 28, no. 3, pp. 50–56, May 2014.

[27] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A Case for Stateful Forwarding Plane," *Comput. Commun.*, vol. 36, no. 7, pp. 779–791, Apr. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2013.01.005

[28] NDN Project Team, "Repo-ng," Available at https://github.com/named-data/repo-ng.

[29] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944 (Proposed Standard), September 2007.

[30] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550 (Proposed Standard), Mar. 2012.

[31] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," RFC 6347 (Proposed Standard), Jan. 2012.

[32] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252 (Proposed Standard), Jun. 2014.

[33] Z. Sheng, S. Yang, Y. Yu, A. Vasilakos, J. Mccann, and K. Leung, "A survey on the IETF protocol suite for the Internet of Things: Standards, challenges, and opportunities," *Wireless Communications, IEEE*, vol. 20, no. 6, pp. 91–98, 2013.

[34] IEEE, "IEEE Standard for Local and metropolitan area networks–Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)," *IEEE Std 802.15.4-2006*, June 2006.

[35] AllSeen Alliance, "AllJoyn Framework," https://allseenalliance.org/framework, accessed on Jan. 29, 2016.

[36] ZigBee Alliance, "ZigBee Specification," ZigBee Document 053474r20, September 2012, available at http://www.zigbee.org/zigbee-for-developers/network-specifications/zigbeepro/.

[37] Bluetooth Special Interest Group (SIG), "Bluetooth Specification Version 4.2," dec 2014, available at https://www.bluetooth.com/specifications/adopted-specifications.

[38] P. Gusev, Z. Wang, J. Burke, L. Zhang, T. Yoneda, R. Ohnishi, and E. Muramoto, "Real-time streaming data delivery over named data networking," *IEICE Transactions*, 2016.

[39] P. Gusev and J. Burke, "NDN-RTC: Real-time videoconferencing over Named Data Networking," in *Proceedings of the 2nd International Conference on Information-Centric Networking*. ACM, 2015, pp. 117–126.

[40] C. Fan, S. Shannigrahi, S. DiBenedetto, C. Olschanowsky, C. Papadopoulos, and H. Newman, "Managing scientific data with named data networking," in *Proceedings of the Fifth International Workshop on Network-Aware Data Management*. ACM, 2015, p. 1.

[41] A. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, "NLSR: Named-data link state routing protocol," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, 2013, pp. 15–20.

[42] A. Afanasyev, C. Yi, L. Wang, B. Zhang, and L. Zhang, "SNAMP: Secure namespace mapping to scale NDN forwarding," in *Proceedings of 18th IEEE Global Internet Symposium (GI 2015)*, April 2015.

[43] L. Belli, S. Cirani, L. Davoli, A. Gorrieri, M. Mancin, M. Picone, and G. Ferrari, "Design and Deployment of an IoT Application-Oriented Testbed," *Computer*, no. 9, pp. 32–40, 2015.

[44] Y. Yu, A. Afanasyev, D. Clark, k. claffy, V. Jacobson, and L. Zhang, "Schematizing trust in named data networking," in *Proceedings of the 2Nd International Conference on Information-Centric Networking*, ser. ICN '15. New York, NY, USA: ACM, 2015, pp. 177–186. [Online]. Available: http://doi.acm.org/10.1145/2810156.2810170

[45] Y. Yu, A. Afanasyev, and L. Zhang, "Name-Based Access Control," NDN Project, Tech. Rep. NDN-0034, Revision 2, jan 2016.

[46] Z. Zhu and A. Afanasyev, "Let's ChronoSync: Decentralized dataset state synchronization in Named Data Networking," in *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, Oct 2013, pp. 1–10.

[47] W. Fu, H. Ben Abraham, and P. Crowley, "Synchronizing namespaces with invertible bloom filters," in *Architectures for Networking and Communications Systems (ANCS), 2015 ACM/IEEE Symposium on*, May 2015, pp. 123–134.

[48] T. Song, H. Yuan, P. Crowley, and B. Zhang, "Scalable name-based packet forwarding: From millions to billions," in *Proceedings of the 2nd International Conference on Information-Centric Networking*. ACM, 2015, pp. 19–28.

[49] A. Afanasyev, Y. Yu, L. Zhang, J. Burke, J. Polterock *et al.*, "The Second Named Data Networking Community Meeting (NDNcomm 2015)," *ACM SIGCOMM Computer Communication Review*, vol. 46, no. 1, pp. 58–63, 2016.

[50] W. Shang, J. Thompson, M. Cherkaoui, J. Burkey, and L. Zhang, "NDN.JS: A JavaScript client library for Named Data Networking," in *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, April 2013, pp. 399–404.

[51] Z. Wang and J. Meng, "NDN EBAMS node running in Mini-NDN," Available at https://github.com/zhehaowang/bms-node.

[52] NDN Project Team, "Mini-NDN," Available at https://github.com/named-data/mini-ndn.

[53] A. Bannis, "Named Data Network Internet of Things Toolkit (NDN-IoTT)," Available at https://github.com/remap/ndn-pi.

[54] NDN Project Team, "NDN Client Library for C++ and C," Available at https://github.com/named-data/ndn-cpp.

[55] ——, "NDNcomm Hackathon: Demonstrate NDN over Bluetooth LE on the Arduino," Available at https://github.com/ndncomm/ndn-btle/tree/arduino.

[56] W. Shang, Y. Yu, T. Liang, B. Zhang, and L. Zhang, "NDN-ACE: Access Control for Constrained Environments over Named Data Networking," NDN Project, Tech. Rep. NDN-0036, Revision 1, dec 2015.

[57] A. Afanasyev, J. Shi, B. Zhang, L. Zhang, I. Moi-seenko, Y. Yu, W. Shang, Y. Huang, J. P. Abraham, S. DiBenedetto, C. Fan, C. Papadopoulos, D. Pesavento, G. Grassi, G. Pau, H. Zhang, T. Song, H. Yuan, H. B. Abraham, P. Crowley, S. O. Amin, V. Lehman, , and L. Wang, "NFD Developers Guide," NDN Project, Tech. Rep. NDN-0021, Revision 5, oct 2015.