IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 26, 2019, accepted April 26, 2019, date of publication May 30, 2019, date of current version June 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2919534

Survey on the Incorporation of NDN/CCN in IoT

AHED ABOODI[1], TAT-CHEE WAN[1,2], (Member, IEEE), AND GIAN-CHAND SODHY[1]
[1]School of Computer Sciences, Universiti Sains Malaysia (USM), Penang 11800, Malaysia
[2]National Advanced IPv6 Centre (NAV6), Universiti Sains Malaysia (USM), Penang 11800, Malaysia

Corresponding author: Tat-Chee Wan (tcwan@usm.my)

This work was supported by the USM RUI Grant No. 1001/PKOMP/811336.

ABSTRACT Named data networking (NDN) has emerged as a component of the Future Internet architecture to support scalable content distribution, mobility, security, and trust; as well as to provide access to information irrespective of its physical location. The NDN, and previously content-centric networking (CCN), has been considered as the enabler to address various Internet of Things (IoT) challenges, with the potential to outperform the current IP paradigm in many dimensions. The requirements and challenges of the IoT, imposed by constraints, such as limited memory and computational power, while requiring high energy efficiency in the face of unstable network connectivity, impose extra burdens on the IP paradigm. The named content, in-network caching and named-based routing approach in the CCN and NDN provide promising solutions to overcome these constraints and showcase alternative implementations. This paper aims to investigate and demonstrate the current incorporation of the NDN/CCN with the IoT in terms of in-network caching management, naming scheme for devices and data, access control and policies, forwarding strategies, device configuration, and discovery.

INDEX TERMS CCN, content-centric networking, ICN, information-centric networking, Internet of Things, named data networking, NDN.

I. INTRODUCTION

Content retrieval and distribution have evolved from centralized servers in data centres to disparate end-devices which dominate network usage in the form of Internet of Things (IoT). A thing can be a device, such as a sensor, or a system, such as learning thermostats, each playing the role of a host or a router in spontaneous or ad-hoc networks such as Wireless Sensor Networks (WSN). Most of the things are tiny devices with limited memory, CPU and battery capacities. Open IP-based standards and suite of protocols, e.g. 6LoWPAN [1], were proposed by the IETF working group to make WSN work coherently with the Internet and other type of compatible networks [2]. However, deploying IP-based IoT solutions on a large scale is still challenging to date, especially solutions with stringent requirements in terms of constraints such as energy efficiency, low capacity and scalability robustness intrinsic to IoT. Not to mention, the host-centric solutions which inherited the communication overhead issue of the current IP model caused by the packet header.

The associate editor coordinating the review of this manuscript and approving it for publication was Usama Mir.

A different communication approach, independent of the IP-based communication model, called Named Data Networking (NDN) [3], [4] has emerged recently which goes beyond end-to-end connections. NDN originated from an earlier project, Content-Centric Networking (CCN) [5], which Van Jacobson proposed at Xerox PARC. This information-based model retrieves content by name without directly mapping the content to host locations, and at the same time achieves better performance, scalability, and security. Both NDN and CCN are considered as Information-Centric Networking (ICN) [6]–[8] instances, similar to previous ICN oriented projects such as DONA [9], COMET [10], and PSIRP [11]. They natively support multicasting, mobility, and in-network caching. In contrast to the host-centric IP-based networking approach, content in CCN/NDN is an integral part of the network. According to Van Jacobson [5], every CCN node specifies *what* to search for but not *where* the information is located, unlike the IP-based paradigm. Any authorized consumer can acquire the data using a unique name (self-identifying), and the data can be authenticated with a signature (self-authenticating). The NDN project funded by the Future Internet Architecture program of the U.S. National Science Foundation (NSF), originally used CCN as the

baseline for its implementation (and later forked a clean-slate version). NDN became popular due to its simple communication model compared to that used by CCN. It offers lightweight configuration, easy management operations and scalable naming, which makes it a promising solution for addressing IoT challenges [2]. Both NDN and CCN architectures are fundamentally similar and follow the same communication principles. However, improvements incorporated into NDN reduce packet processing time and address issues such as packet looping [12] and fixed sized headers [13]. Other changes can be found in [13], [14]. This paper focuses on NDN due to the lower power and complexity requirements for IoT devices.

Various researchers have highlighted, with real-world experiments, the importance and benefits of incorporating NDN/CCN with IoT (henceforth referred to as NDN-based IoT) [15], [16] and have shown that NDN can fit the inherent requirements of the given IoT applications [17], [18]. In fact, the research community has been addressing the fundamental aspect for the deployment of ICN-based IoT starting with the Internet draft [19] and the work of Heidemann *et al.* [20] where they identified the beneficial effect of named data for WSN and described an architecture built around attribute-named data and in-network processing for data aggregation. Amadeo *et al.* [21] highlighted the major challenges and opportunities and discussed the inherent issues of integrating IoT with ICN to serve as a guideline for future implementations. Shang *et al.* [22] showed specifically how NDN addresses the IoT challenges and helps achieve the IoT vision. They also address, with reference to four example implementations, how to implement an IoT system using the NDN architecture, and at the same time achieving the IoT framework functionalities in terms of device and data naming, access control, initial configuration, discovery, schematizing trust, aggregation, Internet integration, and synchronization. Different research works have incorporated NDN/CCN into various IoT applications and tackled particular challenges of NDN-based IoT to address one or more IoT framework functionalities. Some works adopted the in-network caching of NDN/CCN to improve the network performance and increase the hit ratio in a constrained environment. Others use the NDN/CCN scalable naming feature to deploy many sensors and gather data for in a wide-scale application. Moreover, there are implementations which benefit from the small packet size of NDN/CCN and in-network caching to deploy energy-efficient IoT systems for their resource constrained-devices. Therefore, the NDN/CCN-based communication architecture is a promising solution for addressing the characteristics and requirements of the IoT ecosystem (e.g. interconnecting billions of heterogeneous objects) and for resolving the issues inherent in a host-centric IP-based IoT network.

While there is various research on using NDN/CCN as a communication architecture, to the best of our knowledge, only a few researchers have surveyed the incorporation of the content-centric approach for IoT systems. Amadeo *et al.* [23] discussed the design of the forwarding strategies in NDN-based wireless ad hoc networks and evaluated their performance. Meddeb *et al.* [24] detailed different approaches that address the producer mobility issue in the named data IoT networks. Arshad *et al.* [25] surveyed different ICN architectures over IoT and examined in-network caching, naming, security and mobility. However, they did not consider other important components of IoT framework functionality such as forwarding/routing, access control, data aggregation, device configuration and bootstrapping, service registration and discovery.

The objective of this paper is to present a survey on the incorporation of the NDN/CCN architecture in IoT systems, showing which modules were used in those incorporations to resolve IoT challenges, and how they achieve required IoT framework functionality including device and data naming, caching, forwarding, access control, data aggregation, configuration, bootstrapping and discovery. Since NDN is originally based on CCN, this paper will use NDN to describe and refer to both architectures. As an introduction, Fig. 1 shows an overview of the NDN/CCN-based IoT taxonomy.

Section II outlines briefly how the named data networking works. Section III lists the design of some NDN/CCN-based IoT applications and services. Section IV explains the designs of various naming schemes. Section V shows the strategies and policies for caching management. Section VI discusses the access control and policies used to manage the consumer and application authorization. Section VII reports and compares the forwarding strategies. Section VIII presents the data aggregation mechanisms. Section IX describes the device configuration and discovery.

II. HOW NAMED DATA NETWORKING WORKS

A. ARCHITECTURE

NDN is a data-oriented networking paradigm rooted in the idea of content-centric networking. Its design resolves many issues of the current IP connection-oriented paradigm including IP exhaustion, network load distribution, and network overhead. As shown in Fig. 2a, NDN's vision is to reshape the Internet Protocol stack by replacing the waist of the IP hourglass with *named data*, using various networking technologies below the waist for connectivity. Nevertheless, NDN can directly use some IP services such as inter-domain routing policies and Domain Name Service (DNS). IP routing protocols such as Border Gateway Protocol (BGP) and Open Shortest Path First (OSPF) can be slightly modified and adapted to NDN [26]. The strategy layer optimizes resource usage such as selecting a link in a multihomed device. It also allows specifying different transport and forwarding services, depending on the access network constraints and the application requirement. The security layer handles the security functionalities and applies them to the *name data*.

NDN has two packet types, Interest and Data, that carry the request and replies respectively, as shown in Fig. 2b. The sender's interest packet includes the names of the desired

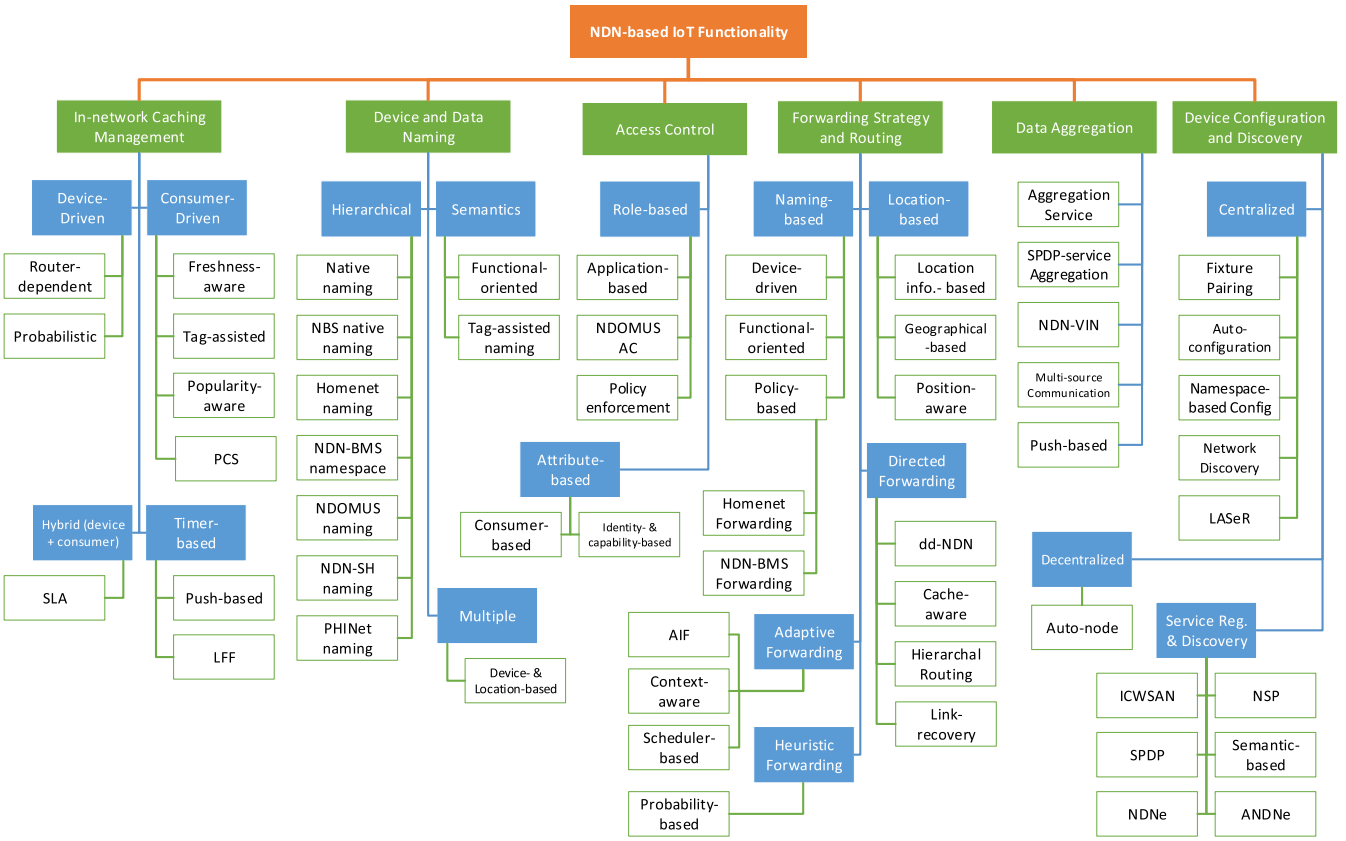

FIGURE 1. NDN/CCN -based IoT taxonomy.

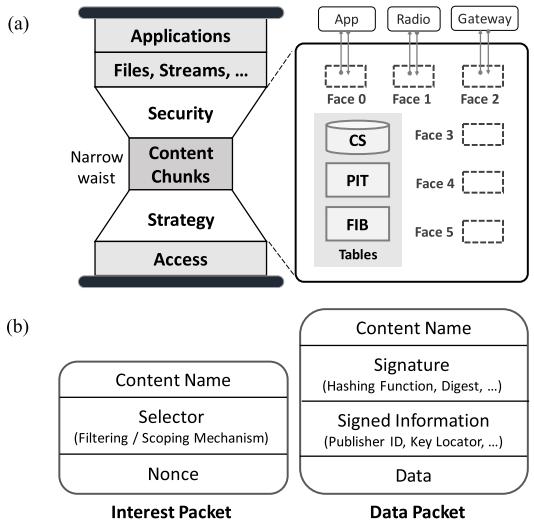

FIGURE 2. (a) NDN hourglass and node architecture, (b) NDN packets.

content and a cryptographic nonce to give a unique identifier. Data packets consist of a named payload to satisfy the requested Interests. The node in NDN maintains three data structures namely: Forwarding Information Base (FIB), which is a routing table used to guide the Interest towards Data; Pending Interest Table (PIT), which is used to keep track of the unsatisfied forwarded Interest along with its incoming and outgoing interfaces(s); and Content Store (CS), which caches the incoming Data packets temporarily but cannot be used for long-term storage.

B. NAMING

Names in NDN are hierarchal, opaque and application-specific. They are constructed and chosen by the application to reference its generated data and fit its need. Consumers send Interest packets that include names of the desired data, and other nodes in the network use the names to retrieve the requested data. For example, a home page produced by USM may have the name */usm.my/en/main.html*, similar to URLs but they do not have to be human readable. The components of the names can define the scope of the data (*/en*) and used to forward the requests. The hierarchical structure of the name helps track the information easily. Any requested name will consider a match if any piece of data has the same name prefix, i.e. at least one or more components starting from the left. For example, */usm.my/en/main.html* can be considered a match to the first segment of the third version of the USM home page (information object) under the name */usm.my/en/main.html/3/1*. The consumer could alternatively ask for the different version by requesting the name */usm.my/en/main.html/2*. The consumer can request segments by using the previous packet and naming convention

information. [27], [28] explains the sequence-based segmentation of standard NDN naming convention.

Local routing or local broadcasting may be necessary to find matching names for local networks, but not when names can be identified globally and uniquely. They may not be suitable for retrieving the data in large-scale and global networks. The names can follow the context of the application and be meaningful in different scopes. In other words, the modality of retrieving named data can be different to match the semantics of the IoT applications, e.g. a temperature sensor in a room or a country name in the world. Section III discusses various naming schemes for different IoT applications.

C. INTEREST AND DATA FORWARDING

The forwarding strategy plays an important part in NDN efficiency and flexibility. Using NDN's multipath capability, an adaptive forwarding strategy can forward Interest message along the best performing paths while maintaining load balancing and avoiding congestion. When the NDN node (e.g. router) receives an Interest message requesting for information, it first looks in its CS for a data that matches the name of the requested Interest. If a match is found, the Interest packet is discarded, and the requested information is returned, in the form of Data packet holding the same Interest name, to the incoming interface of the processed Interest. Otherwise, the PIT table is checked for a matching entry. If there is a match, it means a similar request has been already forwarded; the Interest interface is added to the found PIT entry and the Interest is then discarded. If not, the node performs a lookup match for the longest name-prefix in the FIB table; i.e. it performs a Longest Prefix Match (LPM) of the Interest name in the FIB. If the entry is found, then the Interest packet is forwarded, Multicast Forwarding (MF), through the interfaces assigned to the found entry and a new PIT entry is created with the incoming interface of the Interest. The Forwarding Strategy decides when to accept an Interest and where to forward it, after retrieving the longest-prefix matched entry from the FIB table, and decides when and where to forward the Interest.

When a Data packet is received by a node, it is first stored in the CS and then the longest-prefix-match is performed in the PIT for an entry that matches the Data packet name. If an entry is found, the node forwards a copy of the received Data packet to the interfaces found in the PIT entry, thus achieving multicast delivery; and deletes the found entry from the PIT. Otherwise, the received Data packet is discarded. The forwarding strategy, similarly, decides when to accept a Data packet and where to forward it.

D. CACHING

The in-network caching is natively supported in NDN and involves *cache decision/placement strategy* and *replacement policy (or cache replacement scheme)*. The first helps decide where and whether to cache an incoming Data packet while the second decides which existing Data element should be replaced in the CS when a new incoming Data packet needs to be cached and the CS is full. By default, the *cache placement strategy* in NDN uses Caching Everything Everywhere (CEE) or Leave Copy Everywhere (LCE) strategy where any incoming Data packet is cached. There are other caching strategies designed for ICN that can also be applied for NDN namely, Leave Copy Down (LCD) [29] which leaves one Data packet copy in the gateway close to the consumer, edge-caching strategy [30] which leaves one copy in the last gateway in the reverse path toward the consumer, and consumer-cache strategy [31] which keeps one copy of Data packet in gateways connected to the consumer directly. The default *replacement policy* in NDN is Priority-FIFO (First In First Out) where the oldest cache data is replaced by the newly arriving data [14]. Least Recently Used (LRU) policy is incorporated in NDN and has been adopted by many caching policies [32]. Its principle is to replace the cached data that is not being used for the longest time. The replacement task of LRU can be carried out in constant time ($O(1)$) together with the search task. Compared to LRU, FIFO has a vaguely simpler implementation as it repeats the same used policy to store content in the cache. Least Frequently Used (LFU) or Random Replacement (RR) can also be applied for NDN as a cache replacement policy. LFU replaces a least popular chosen cached data while RR replaces a randomly chosen cached data. The feasibility of applying the aforementioned *cache placement strategies* and *cache replacement policies* for NDN and CCN has been discussed and compared in [32] and [29]. Section III describes some caching strategies and policies developed for NDN-based IoT.

E. NDN FOR THE INTERNET OF THINGS

It is a great challenge for the IoT system, which comprises resource-constrained devices with highly heterogeneous traffic pattern, to have a network architecture that interconnects a huge ecosystem. It requires framework functionality and capability such as auto-configuration, service and peer discovery, bootstrapping, management, security, reliability, robustness, and scalability. NDN can directly manage several of these functionalities at the network layer level, due to its substantial features [33].

1) DEVICES AND DATA NAMING

NDN provides an easy, robust and scalable data retrieval, thanks to its application-specific hierarchical naming and forwarding strategy. Content provided by the IoT devices can be accessed by means of NDN name. Content retrieval upon request and scheduled content updates, using the named contents, are easily handled, even in a large network, with the help of in-network caching, hop-by-hop replication, and name-based routing, which eliminates the IP address assignment procedures. The naming convention can be customized for the application but should take account of common naming conventions and trust relationships. Hierarchal naming is not required but can be useful in forwarding strategy. The naming schemes of different NDN-based IoT applications are discussed in Section II.

2) DEVICE CONFIGURATION AND MANAGEMENT

An important part related to the usage of the IoT sensors and actuators over NDN infrastructure is the configuration and management of the IoT devices. Many implementations have used the communication mechanism available in NDN to enable operations related to the discovery, control, and configuration of IoT devices although NDN is based on named data and not on end-to-end communications. IoT systems must be able to adapt to topology changes in terms of device or service addition, removal or configuration update. Producer devices should be configured with at least named prefix, with self-signing and encryption options added for secure delivery. Autoconfiguration or bootstrapping of devices can be modeled with an Interest-Data packet exchange using a well-known, locally defined, namespace since any NDN device can request for data without having a name. Service and peer discovery can also use the same method assuming that nodes periodically express discovery Interest, which carries the namespace prefix that identifies either it is a service or an autoconfiguration protocol. Section IX discusses examples of implementations.

3) CACHING MANAGEMENT

As mentioned before, one of the characteristics of NDN is its in-network caching capability. It requires more resources to be available in the routers, which requires the need for resource management mechanism or proper cache placement strategy and replacement policy based on the device's available storage, power, and processing capabilities. The NDN's native caching strategy is designed for the Internet and may not be suitable for IoT systems, particularly wireless systems, due to rigorous information freshness and presence of resource-constrained devices and battery-powered producers [34]. Meddeb *et al.* [32] studied the impact of well-known caching policies, that were designed for computer architecture, databases, and ICN, and tested their performance in an NDN-based IoT environment. They found that the LRU outperforms other cache replacement policies, but the most convenient cache replacement policy is the combination of *consumer-cache placement strategy* [31] and the *random replacement policy*.

In fact, caching strategies and policies' performance depends on the context specificity in which they are used. Such context specificity is information freshness where preserving name to content (as a cache) that may change in time, in some cases, may not be consistent when the data is stored at various network points. To resolve this issue, a timestamp can be attached to the name content, but it is not optimal as it increases packet processing time at the routers when checking for matching content with the required timestamp. Version tag can replace the timestamp, but it requires the consumer to be aware of the proper content when requesting. Other context specificities can be the frequency of content generation, consumers' expectations, traffic volume and type, and the machine's computation capabilities. Section III discusses approaches designed to improve caching capability for different context specificities in NDN-based IoT systems.

4) DATA AGGREGATION

Sensor nodes of IoT observe the real-life environment and provoke information, while the sink nodes of IoT collect that information, which can generate a large amount of data continuously. Depending on the IoT application, it is necessary to pre-process and aggregate the raw data stream immediately after the data is captured in case it is infeasible, if not inefficient, to archive and analyze the raw data. After that, the aggregated data is processed (sometimes combined) or analyzed to satisfy the requester. The NDN has the ability to facilitate such mechanism, whereby data retrieval becomes efficient and fail-safe with its in-network caching, especially if the devices are deployed in a hierarchical topology. Some implementations use the naming convention to send continues Interest packets for a period of time to receive data from multiple producers (i.e. sensors), and the data are then added all together using a predefined format. Other implementations use the naming convention to define what type of information is needed and how to encode it, the location of measurement, and the time when it is taken. The sink node or the aggregation gateway constructs Interest packet with the defined naming convention and retrieves the result data periodically or with the help of a publish-subscribe mechanism. We further discuss these examples in Section VIII.

5) LIBRARY AND IMPLEMENTATION SUPPORT

The known protocols for CCN and NDN are CCNx by PARC [35] and NFD by NDN Project [36]. They are designed for PC type of hardware. However, there are other implementations specifically developed for IoT systems and consider its resource-constraint requirements, such as low processing power, low memory size, and energy efficiency of the devices' hardware. CCN-Lite [37] is a reduced lightweight implementation of CCNx and can be used with NDN protocol since it only implements the base forwarding daemon. Bacceli *et al.* [15] first ported CCN-Lite to RIOT [38], which is an open-source operating system (OS) for IoT sensor devices, as a third party package. Saadallah *et al.* [39] implemented and integrated the CCNx into Contiki [40], which is also an OS used for wireless sensor networks. Ren *et al.* [41] designed CCN-WSN, a lightweight alternative of CCNx protocol targeted for WSN. They modified the memory management and revised the computational constraints of sensor nodes and communication patterns. In contrast to [15], Shang *et al.* [42] integrated the core packet forwarding of NDN into RIOT OS as an optimized lightweight NDN protocol stack. Similar to RIOT and Contiki, the NDN implementation could use the other IoT operating systems referred to in [43] and [44]. Such operating systems include TinyOS [45], FreeRTOS [46] and OpenWSN [47].

III. NDN/CCN-BASED IOT APPLICATIONS AND SERVICES

A. CYBER-PHYSICAL SYSTEM

Hellbrück *et al.* [48] have implemented a facility management application, as an example of a service-centric architecture for the cyber-physical system (CPS). The architecture is developed to resolve issues such as service registration and discovery, raised on Service-Oriented Architecture (SOA) for large CPS like WSNs connected to clouds (backend). It is based on CCNx, as the communication protocol, and employs CCN-WSN [41] as the resource-efficient lightweight implementation to build the name-based service bus for CPS. The Named Service Bus (NSB) allows for registration of and transparent access to services (see Section IV), between CCN-WSN used in WSN and CCNx in the backend (cloud), in the entire CPS. A gateway is used to convert the message format between CCN-WSN and CCNx and serves as a service cache and proxy for services in WSN. CCN-daemon is a key component in NSB. It listens to service registration requests locally, with consumer queries locally and remotely, besides its basic routing and forwarding functionality.

B. ENVIRONMENTAL MONITORING SYSTEMS

Dinh and Kim [49] proposed a different type of service discovery that is based on the naming scheme. The authors explore new capabilities of ICN approach for Wireless Sensor and Actor Network (ICWSAN) which includes efficient coordination, interoperability, service discovery and prioritized routing, and establish a different foundation on performance improvement of challenges addressed in [50]–[52], in Wireless Sensor and Actor Networks (WSAN). This foundation, according to the authors, can be further extended for weather forecasting and environmental monitoring systems. The authors discuss why Internet-ICN is not directly applicable to WSAN due to device capabilities, reliability, and complicated communication.

C. HOMENET

Ravindran and Biswas *et al.* [53], [54] propose an ICN based Homenet highlighting how ICN virtues help overcome the issues of the IETF IP-based proposal for Homenet [55] such as security, mobility, and multicasting. They also propose dynamic attachment of nodes, called auto-node, and service discovery protocols enabling user interaction with devices inter-connected in infrastructure or ad hoc mode. The authors use the naming scheme, with the consideration of contextualization, service accessibility, extendibility and policy enforcement, to identify services, devices, and content offered by these devices.

The authors also introduce a prototype built over CCNx to achieve zero configuration neighbor and service discovery across router boundary and to enforce a policy in the forwarding plane. In addition, the prototype implements a name-based firewall at the gateway level of the experiment network. The prototype has two developed protocols, namely, Neighbor Discovery Protocol (NDP) and Service Publish and Discovery Protocol (SPDP). NDP is used to discover the nearby neighbor infrastructure nodes, such as CCN routers, while SPDP is used to advertise local services and dynamically handle the service request initiated by the consumer.

NDOMUS [56] is a framework built for smart home ecosystem as a typical service model that brings NDN in the home domain with constrained sensing and actuator devices. The framework is based on the previous work of the authors [33], but extended to specifically address the functionalities of smart home domain. Besides its service model feature, the framework defines a flexible naming scheme that can fit a variety of smart home applications and provides a different type of transmission modes/strategies for multiparty communication. It requires a Home Server (HS) that periodically communicates with a set of resource-constrained devices to maintain and list the active among them along with their parameters. The HS, which can be co-located in a gateway to provide global connectivity, controls and sets up the operations in the home network including device discovery, registration, key distribution, access control list and encryption mechanism.

Another example of home automation system is the work of Waltari and Kangasharju [57], which aims at a higher level of abstraction in accessing IoT using a presentation bridge, also called PB-CCNx. The authors address the actuator commands' issues where the physical location of the things must be unique and well known to the user and where in-network caching of acknowledgment message (content object of the actuator) could be harmful to the operation. The research uses There Corporation automation system, which supports multipurpose wireless sensors such as temperature, humidity, and energy, to implement the presentation bridge and to communicate with external networks through CCN. The research uses ThereGate as its gateway to many wireless technologies such as Z-Wave and ZigBee. The gateway is installed with CCN protocol in its transport layer to act as the presentation bridge of the network. The sensor readings are collected in another way using DBus. The proposed presentation bridge uses JSON to wrap the sensors readings in the content objects.

The implementation test bed was small and did not take into account the power consumption during transmission of the sensor readings as there is no CCN communication between the sensor and the gateway. This is considered a drawback for some applications in wireless sensor environment. Not to mention, CCN is tunneled in the gateway's transport layer and the device used has a fixed constant power. Moreover, the implementation uses CCNx that requires computational capabilities and cannot be applied for some less powerful devices such as transducers with IP connectivity because of the constraints regarding the processing power.

Ahmed and Kim [58] have introduced simpler architecture concept of smart home (NDN-SH) communication with a private cloud (PC) that stores the historical information produced by the sensors to enable users to retrieve the data when not in the proximity of the home server. The HS collects sensor data from different areas of a home while the PC stores the

cached data for distinctive usage and maximum availability. The designed architecture changes the stereotype of the HS, being the backbone of most of the smart-home designs that interact with all sensor devices and tend to send data to consumers. It tries to decouple the HS from sensor/actuator devices and allow consumers to retrieve data not only from the HS but also from specific sensors or area.

D. BUILDING AUTOMATION AND MANAGEMENT SYSTEMS

Burke *et al.* [59] constructed a secure NDN-based framework, for lighting control case (NDN-LC) as a Building Automation System (BAS) to satisfy low latency requirements of the lighting fixtures' communications. The constructed framework consists of a trusted model, to ensure security and ownership of fixtures contents, and a developed security protocol, to ensure authentication of the commands. It also provides authenticated access control to fixtures via authorization policies and uses NDN naming to reflect access restriction. The framework requires two important entities to function: a configuration manager (CM) for fixture initialization (namespace and identity/public key), and authorization manager (AM) for application policies on fixtures and signed access control list issuance. The authors further extended this work to a secure sensing framework (NDN-SSF) for BAS that provided secure sensor-bound communication over NDN [60]. The extended framework connects sensors with the application and supports three types of sensors, namely Pull Data Dissemination, Push Data Dissemination and Hybrid (Pull and Push) Data Dissemination. Corresponding communication protocols are also provided.

Shang *et al.* [61] proposed a data-centric design for Building Automation and Management Systems (BASs and BMSs) using NDN, called NDN-BMS. They changed the chilled water flow and electrical demand monitoring system that exists in the UCLA campus from using standard Modbus/TCP protocol and BACnet/IP protocol to publishing data using the designed NDN-BMS. They also employed a Python-based data publishing service and browser-based data visualization interface using their JavaScript NDN library, NDN.JS [62], to fetch and process the sensors' data. Their NDN-BMS uses the hierarchical namespace for data, encryption keys, and access control lists. The authors implement an encryption-based access control prototype for their deployed monitoring system testbed that collects sensing data from industry-standard components. The access control is based on the component's identity, as a shared symmetric key, to enforce trust relationships and uses encryption for unauthorized read protection. NDN-BMS employs public keys to uniquely identify users. Keys are then distributed as NDN data and encrypted or signed by a trusted authority using hash-based message authentication code (HMAC) [63].

Shang *et al.*, who proposed NDN-PS [64], developed a publish-subscribe communication framework on top of NDN-BMS to address the BMS data consumption. NDN-PS framework leverages distributed repositories as intermediaries to store and republish large-size data that can be employed by different applications. The publish-subscribe mechanism is incorporated to aggregate the published data streams from multiple producers or sensing devices so that consumers can subscribe to any interesting data sets of the BMS data streams. This is done by using a synchronization protocol known as PSync [65], which facilitates and provides adequate redundancy in multiple repositories and notifies consumers of new sensor readings.

E. VEHICULAR INFORMATION NETWORK

Yan *et al.* [66] designed a Vehicular Information Network (VIN) architecture, that can support traffic control, traffic scheduling, and emergency broadcast, based on NDN (NDN-VIN). They aim to improve content naming, addressing, data aggregation and mobility for Intravehicular Communication (IVC) in a VIN, also referred to as Intelligent Transportation System (ITS). Each NDN router in the architecture is deployed with location-based routing and aggregation/segregation services to ensure the scalability and efficiency of the VIN. The architecture uses a device-based naming scheme to address the sensors placed in the vehicle and location-based naming scheme for the geographical location. Two types of end/mobile devices are considered in the architecture: the first is a single mobile node such as a smartphone, and the second is the entire mobile subset (network mobility) including devices without Internet access capability. Both types of devices are placed in a vehicle. Thus, a mobile router or a gateway is needed to collect and distinguish the metadata from those devices.

F. SERVICE LEVEL AGREEMENT MANAGEMENT

Suarez *et al.* [67] proposed information freshness as service level parameter that can be negotiated with a gateway and later applied on and used by a device. They extended the previous investigation, in [68], on the management aspect in ICN solutions but focused on the use of IoT concepts as a relevant scenario of their works and on a specific use case, i.e. information freshness Service Level Agreement (SLA) application. The service level parameter is part of the information freshness management between the consumers and gateways to determine the suitability of the stored information. The gateway acts as an intermediate between consumers and IoT devices and is used to provide a supporting mechanism for the architecture operations in the form of discovery, registration, security, management, policy-enforcement, and aggregation of content from different physical devices. It is considered as a reference point that allows an authorized consumer to retrieve a set of permitted IoT devices along with its policy that indicates the list of commands and information items for which that consumer has authorized access in the IoT devices. This happens by the means of authentication and encryption mechanisms that are executed at the gateway.

G. HEALTH

PHINet [69] is an NDN-based testbed framework that integrates IoT-based Health system (Health-IoT) with cloud computing. The framework is designed to be used for development, testing and prototyping of Health-IoT applications using content-centric networking. It provides an ad-hoc platform for the application executed over it and seeks to perform data exchange over IP using NDN-compliant communication. The underlying framework of the PHINet's application uses UDP to facilitate the NDN-compliance, which can easily support the pull-based mechanism and multitasking that NDN uses. The authors implement PHINet using Node.js [70] with a PostgreSQL database for cloud server and Nod.js with Sqlite3 for the Android platform which provides native UDP support of the client-side applications. The framework includes a database handler to manipulate and store the PIT, FIB and CS entries in a database. It also includes both UDP sender and UDP listener modules to send and receive data to and from a UDP socket, respectively. Furthermore, forwarding and caching processes are handled by the parser of the framework. However, the devices communicate with each other over UDP, thus each device requires to be initialized by the server with five packet exchange process and later needs to periodically synchronize its up-to-date data (e.g. IP address) with the server to ensure long-term data storage.

H. SMART CITIES

Mick *et al.* [71] proposed a scalable NDN-based framework that provides lightweight authentication and hierarchical routing, called LASeR, for a large-scale deployment of IoT devices that reach as high as 40,000 devices/km^2, which makes it suitable for large-scale IoT applications such as smart cities. The framework supports three IoT functionalities namely network discovery, secure routing, and lightweight device authentication. It provides simultaneous on-board authentication, inspired by [72], and routing process/procedures to avoid additional overhead and overall latency imposed if those processes occurred serially. Those processes require only a few cryptographic operations and three round trips. The framework requires at least two unconstrained entities including a manager, called *Island Manager* (IM), to authenticate and register devices in the network and *anchor nodes/devices* (ANs) to form the network backbone. Constrained devices are defined by a flat identifier which is taken from its MAC address and then attached to one of the AN networks. Each AN has a forest network, which comprises hierarchal trees of constrained devices attached to the anchor device as the root of the forest. Multiple ANs may be linked and communicated via any appropriate routing protocol. A gateway can be placed as an edge router to connect all the ANs within a WAN while IM is a service placed in the cloud, within a node in the network or a shared and synchronized database between anchor devices.

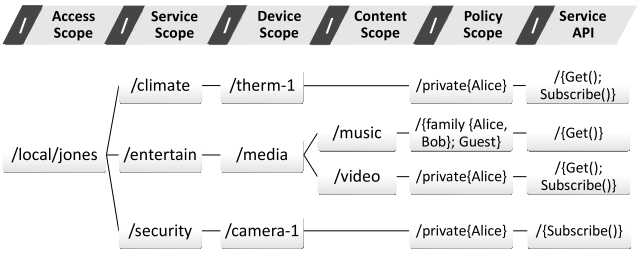

FIGURE 3. Homenet naming scheme.

IV. DEVICE AND DATA NAMING

A. HIERARCHAL NAMING SCHEME

Although the native naming scheme (see Section II) has a hierarchical structure, it assigns none of the semantics to names. It is not required but can be leveraged to support the forwarding process in the pull-based data retrieval mechanism, where an Interest carries the name of the requested data and forwarded one step at a time, based on the components of the Interest name, until it reaches the destination. The data is then carried back to the requester with the same name.

Many examples of NDN-based IoT has followed the native hierarchal naming structure to design their application or services. NSB [48] follows the NDN native naming and only uses the concept of CCNx c*ommand markers, %C1*, for its service naming convention to register and access the services in a CPS network. The marker followed by a namespace identifier specifies the service names followed by operation identifier (remote method) which is followed by the arguments delimited by tilde '∼', e.g. */C1.temprature.a∼4∼2*. While NSB uses the command markers to register and access its services in a cyber-physical system, it does not bind to specific naming scheme or topology. However, designing a naming scheme for a home/building automation system should be different as it requires consideration on contextualization, service accessibility, extendibility and policy enforcement, to identify services, devices, and content offered by these devices. Fig. 3 shows the *Homenet*naming scheme [53] as an example of naming in home automation. The first level in the naming hierarchy structure is the access scope, which identifies the reachability of the service. The second level is the service scope, such as climate-control, entertainment or security. Other levels are device scope, content scope, policies, and service API.

The naming scheme for building automation, NDN-BMS [61], uses the namespace to represent sensor data, devices, and users/consumers, as shown in Fig. 4. It has two parallel namespaces, */building* and */user*, under a root-prefix of their implementation testbed, unlike *Homenet* naming scheme which has cascaded scopes/namespace and did not consider the physical location of the devices. The */building* namespace follows the hierarchy of a building structure and can be partitioned into sub-namespaces of floor numbers and room numbers. Those sub-namespaces include devices grouped by their physical attachment and/or functionality such as power, voltage, and temperature. The gateway devices

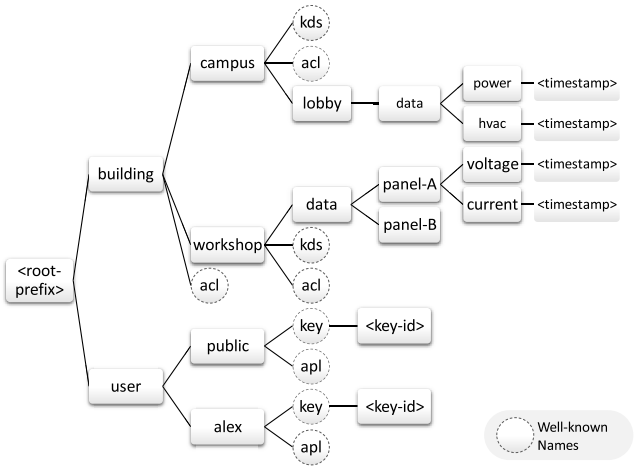

FIGURE 4. NDN-BMS namespace design.

are fitted with Key Distribution Services (KDS) which allow symmetrical keys to be generated for legitimate consumers. Each gateway performs this process regularly with the following name prefix structure: */<gateway-prefix>/kds/<user-public-key-id>/<timestamp>*, where *timestamp* denotes the key version. The */user* namespace is also used for identity management and privilege management of the users/consumers under identity-based access control scheme. Each consumer has its own name prefix in the namespace followed by its own security key prefix */key/<key-id>* and Access Privilege List (APL) prefix */apl*, which can be stored in or retrieved from a repository.

The naming scheme of NDOMUS [56] framework is designed to fit a variety of smart home applications in terms of sensing, actuating, configuration, and management operations. It has two main sub-namespaces: the first is the configuration and management namespace which uses the prefix */conf* for configuration update, bootstrap initialization, and management operations, and the second is the task namespace which uses the prefix */task* for placing control and monitoring operations. The task namespace has the prefix structure */task/type/subtype/location*, where the *type* component is used to differentiate between the task type of operations that a device is going to perform, whether they are action execution */task/action* or measurement reporting */task/sensing*. The *subtype* component specifies which type of action or sensing task must be executed, and the *location* component can follow the physical structure of the home building. The authors claim that the task namespace structure can be extended to include subcategories of its components. No mention is made regarding how configuration and management namespace */conf* is used for its device discovery process, device registration steps, active devices list, and other information related to security management, unlike NDN-BMS.

Similar to the task sub-namespace, */task*, NDN-SH [58] uses two main sub-namespaces in its naming scheme to identify tasks performed by actuators or to identify information type (i.e. sensor information of an application) required by the sensors. The second level of the naming scheme can follow the physical structure of the home building. For information retrieval, the namespace prefix */info* is used while for actuating task the namespace prefix */act* is used.

PHINet testbed [69] is an example of native hierarchical naming on an NDN-compliant system where its devices communicate with each other over UDP. The naming scheme used is more general and hierarchically defined as follows: domain, user identifier with the syntax *firstname+lastname+dateofbirth*, sensor identifier which can be one or more sensors separated with '-', time string and process identifier to distinguish the tasks.

B. SEMANTIC NAMING SCHEME

Functional-oriented naming: ICWSAN [49] provides a semantic naming structure by making the information identifiable and visible at the network layer, and argues it leads to efficient management of coordination and collaboration issues of sensors and actors. The proposed scheme is called functional-oriented naming scheme, where naming is addressed based on the type of information as well as the information producer's functions, thus reducing the naming space which makes the scheme different from the conventional CCN approach where naming is addressed by each piece of data. The naming of the devices (or entities as the authors refer to it) consists of two parts: *category prefix (CP)* which express a real-world category naming, and *InformationID* which is unique and can be the entity's ID, an event name, or type of sensing data, and denotes the detail information. Both of them are bound to make the information object name, *CP:InformationID,* as the structure of the used naming scheme. Hence, this binding allows for direct verification and helps utilize the semantic name. The proposed work includes four communication modes: broadcast, anycast, multicast and unicast which are specified by including two bits in the header of the information object name.

Tag-Assisted naming: The functional-oriented scheme categorizes naming, *CP*, based on the information type and the producer's (e.g. sensor) functions, and then uses these categorized prefixes instead of the native name prefix for its Interest forwarding and Data retrieval process, whereas Tag-assisted naming [73] uses tags to categorize the data. Furthermore, tag-assisted naming complies with native data naming for the data retrieval process, unlike functional-oriented naming which uses only the category prefix. The tags are carried by the data along with the hierarchal name and maintained in a tag list. Consumer or content generator can produce the tags with the prefix *'#'*. For example, the unique name for a lecture slides in the School of Computer Science can be stored as */usm.my/cs_school/ppt/courses /IoT/lecture3* with a set of tags *#IoT #courses #lecture2 #ppt*. Any consumer interested in this lecture, who does not know the hierarchical name, can send an Interest using any of the four tags in any sequence.

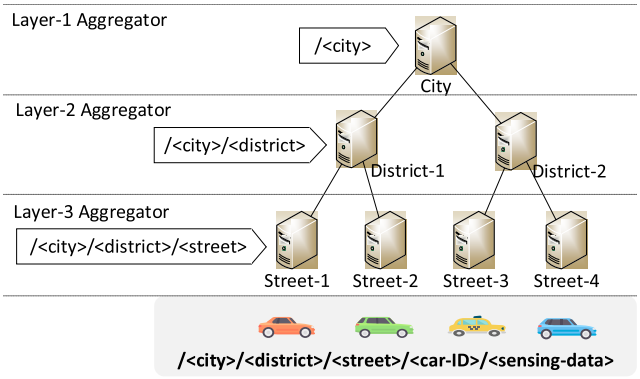

FIGURE 5. Location-based naming scheme and aggregators placement in NDN-VIN [66].

/<dL_1>: ... :<dL_n>:end/<$type_1$>: ... :<$type_t$>:end/
<sL>/<$type\_and\_data_1$>: ... :<$type\_and\_data_d$>:end/
<$next_1$>: ... :<$next_e$>:end/

FIGURE 6. Device-based naming scheme in NDN-VIN [66].

C. MULTIPLE NAMING SCHEME

While it is common in building automation system to name devices to reflect their physical location in an indoor environment, it is common for vehicular information network to reflect geographical location in an outdoor environment. The NDN-VIN architecture [66] proposes two naming schemes: device-based naming scheme for the sensors placed in the vehicle, and location-based naming scheme for the geographical location. In the location-based naming scheme, routers can be deployed along the hierarchy starting from a city to district until it reaches the streets. Hence, routers can also be extended to support aggregation processing, as in Fig. 5. In the device-based naming scheme, shown in Fig. 6, the structure consists of three fields: (1) multiple hierarchical structures of the destination location (*dL*) with a series of requested data types, (2) hierarchical structure of the source location (*sL*) with the data gathered and provided by this vehicle which instantiated the request, and (3) next location along the direction of the vehicle. A push-based data retrieval mechanism is used to piggyback the data gathered in *sL*, provided by the requesting vehicle, as labels in the Interest packet.

This section has discussed the naming schemes designed for NDN- and CCN-based IoT applications, which consist of three categories, namely hierarchal, semantic and multiple naming schemes. Our observation shows that the design of the naming scheme is the most important system design step for NDN-based IoT applications. The designed naming schemes reviewed in this section are application-specific and opaque to networks as they provide scalability and simplicity. All the hierarchal naming schemes follow the native NDN naming where names represent a piece of data provided by a producer. However, in semantic naming schemes, data names are defined by a function or a type and might have more than one producer. In multiple naming schemes, a producer follows more than one naming structure, i.e. the same data type can be provided by a producer when requested by any name prefix of a naming structure under the multiple naming schemes.

It is also important to mention that many of the discussed naming schemes are designed to support IoT application requirements and to help resolve one or more IoT challenges. As shown in Table 1, the naming scheme is different from one NDN-based IoT application to another. NSB used CCNx command markers for its communication and gives the freedom for the applications/services to design its own naming. Similarly, Tag-assisted and PHINet systems are not tied to a specific naming scheme and yet the former provides easier access to information if the consumer does not remember the named prefix of the required data. Some implementations such as PB-CCNx and SLA do not consider or mention the designed naming scheme in their applications or services. However, ICWSAN specifically designed the naming scheme for functional coordination and interoperability between WSAN devices. The designed naming scheme of Homenet [53] provides service accessibility, extendibility and policy enforcement over the devices of a building. NDN-BMS goes further and designed it to provide identity-, capability- and encryption-based access control and data-based security for a building. NDN-VIN used it for VIN traffic control and to represent the geographical areas starting from the city down to the district then to the street level, while at the same time used it to represent the devices' data on a moving vehicle.

V. IN-NETWORK CACHING MANAGEMENT

A. DEVICE-DRIVEN CACHING

Song *et al.* [74] have addressed that the CCN models are designed mainly oriented to super routers and not appropriate for IoT networks. Therefore, they propose an internetworking ***router-dependent*** scheme for resource-constrained devices based on task mapping. The scheme has two specific strategies for resource-constrained devices as producers and as consumers. The scheme must have a Super Router (SR) and all the resource-constraint devices must be connected to it, i.e. the SR being the main target and main resource for all producer and consumer devices in the network. The router will translate and map the Interest request and Data content (i.e. task or services such as storing or retrieving) from and to the IoT networks, resulting in all traffic passed and cached in the SR. Therefore, the in-network caching is also centralized and provided only by the SR and its neighbor routers, thus makes the scheme router-dependent. This is because all the consumer and producer interests must point at a service provided by the SR even if the Interest can be satisfied by any neighboring resources-constraint device that is under the service.

The work of Hail *et al.* [34] proposed a different device-driven caching strategy. Instead of the centralized caching and router dependency found in [74], the authors proposed

TABLE 1. The implementation differences of the applications and their naming schemes.

Application	Naming Scheme	Forwarding Strategy	In-network Caching	Data Aggregation	FIB Edit	PIT Edit	ICN Instance	Features
Cyber-physical System [48]	Native, *includes Extensible Command Markers*	Native	Native	No	No	No	CCNx; CCN-WSN	• Backend (cloud) service access • No global service profile
ICWSAN [49]	Functional-oriented Naming	Yes, based on the naming scheme	Native	Continuous Interest	-	-	CCNx	• Interest category-certifying • CI for event notification
Homenet [53], [54]	Homenet Naming	Policy-based Forwarding	Native	No	Yes	-	CCNx	• Service & access policies • Service discovery • Auto-configuration • Peer discovery
NDOMUS [56]	NDOMUS Naming	Native	Native	Yes, 1C:MC	No	No	NDN	• Multi-party communication • Access control • Data aggregation
PB-CCNx [57]	Native	Native	Native	No	-	-	CCNx	• There Corporation components
NDN-SH [58]	NDN-SH Naming	Native	Native	No	No	No	NDN: NS 2.34	• The concept of PC • Push-based support
NDN-LC [59]	NDN-LC Naming	Native	Native	No	No	-	CCNx 0.4	• Provide a trusted model • Application authorization • Symmetric authentication • Bootstrap • Encryption
NDN-SSF [60]	NDN-LC Naming	Native	Native	No	-	-	-	• NDN-SLCF features • Pull-based dissemination • Push-based dissemination • Hybrid dissemination
NDN-BMS [61]	NDN-BMS Namespace	Native	Native	No	-	-	NDN; NDN-JS	• Access control scheme • User privilege management • Key distribution service • Data encryption • Data-based security • Auto-configuration update
NDN-PS [64]	NDN-BMS Namespace	Native	Native	-	-	-	NDN; NDN-JS	• Support publish- subscribe communication. • Data synchronization for replication • Cryptographic signatures for packets
NDN-VIN [66]	Device-based Naming, Location-based Naming.	Location Information-based Routing	Native	Yes, 3 levels of aggregation	Yes	-	NDN	• Push-based data retrieval mechanism • Interest packet segregation • Data packet aggregation
SLA [67]	Native	Native	Yes, Freshness-based	No	-	-	NDN: ndnSIM	• Service level information freshness • Gateway-based device registration mechanism • Consumer-based policy enforcement
PHINet [69]	PHINet Naming	Native, and IP-overlaid	Native	No	Yes	Yes	NDN: UDP	• Plug-and-play testbed framework for Health-IoT • NDN overlay
LASeR [71]	Native	Hierarchal Routing	No	No	-	Yes	NDN: ndnSIM	• Network discovery and authentication • Device authentication • Key distribution

a ***probabilistic*** caching strategy, named pCASTING, specifically tailored to wireless multi-hop information-centric IoT systems which consider the data freshness and at the same time energy level and storage capability in heterogeneous resource-constrained devices. pCASTING is independent of network routing protocol, it requires no additional signaling or further information piggybacked in Interest/Data packets. Besides, the data packet residual freshness parameter used as an attribute for the probabilistic decision, battery energy level and node cache occupancy parameters are

also used as hardware attributes taken from the resource-constrained devices to decide whether or not to cache the received content. The caching decision in pCASTING is performed when a node receives content with a PIT matching. The caching probability is then calculated and defined with the *Caching Utility Function* F_u using the normalized parameters for energy, occupancy, and freshness. The function, shown in equation (1), can accept additional parameters.

$$F_u = \sum_{i=1}^{N_p} w_i g(x_i) \tag{1}$$

where N_p is the number of normalized parameters (pCASTING used three), the weight w_i assumes values where $0 \leq w_i \leq 1$ and $\sum_{i=1}^{N_p} w_i = 1$ to modify the effect of the normalized parameter x_i computing the utility value; and $g(x_i) = x_i^n$ with $n \geq 1$. This causes higher x_i value to have higher impact on the resulting utility. The F_u values are the node's cached probability and should be in the interval $[0 : 1]$.

B. CONSUMER-DRIVEN CACHING

Quevedo *et al.* [75] introduced a consumer-driven mechanism for information freshness (***freshness-aware***) in CCN network with IoT-enabled scenarios by analyzing the current caching mechanism of most IoT-based ICN approaches, considering the delay between the information generation and its consumption from the cache, to mitigate the negative effect of the information freshness requirement. The authors use the signed information to resolve some of the security threats (e.g. Denial of Service (DoS) attacks) resulting from the systematic requests of low freshness values in the proposed mechanism. The modification of this work is in the CS of the CCN protocol where a check step has been added to determine whether the stored copy meets the freshness requirement, which will be included as an optional field in the Interest packet. The mechanism should be applied for every node in the network. While the proposed mechanism shows significant improvement in the network performance when a requester has more strict freshness requirement, it also requires the requester to be more active to achieve that.

In [73], consumers are allowed to classify the caching content using tags. It proposed a ***Tag-assisted*** CCN (TCCN) scheme for IoT. Besides the traditional prefix-faces structure of the FIB table, prefix-based FIB (P-FIB), the scheme adds a Tag-based FIB (T-FIB) table that comprises the tags and the corresponding faces. The authors proposed a Tag Filter (TF), a tag list that consists of all the cached tags in the local CS, which is maintained by the CS and can be constructed based on Counting Bloom Filter (CBF) to quickly check whether the interest can be satisfied by local CS to reduce lookup time. This tag list is also updated when new contents are cached, or old contents are dropped. Furthermore, every node in the network checks all the incoming contents and then decide whether to cache it according to TF of CS and a counter threshold using a Tag-based Caching strategy (TCS). The caching strategy accepts only the popular tags which is decided by mapping the counters of the CBF to the incoming content tags and if the corresponding counters all exceed a threshold. By using this caching strategy, content with a similar set of tags (i.e. popularity) are cached intensively which will benefit the forwarding scheme of the network by improving the performance of interest-content matching since the size of the tag list of the CS will be reduced. However, the flexibility of tagging allows users to classify their collection of items in the ways that they find useful which will lead to the problems inherent in an uncontrolled vocabulary. The lack of synonym control can lead to different tags being used for the same concept, precluding collocation. The modification in this work involves all the tables of CCN (i.e. FIB, CS, PIT). FIB is attached with an additional table named tag-based FIB which records a set of tags in each face, its size equal to the number of faces. PIT and CS have a tag column which records the corresponding tags for each entry. Moreover, CS has a list of all tags of the cached contents to maintain and then used to construct the TF.

Another example of consumer-driven caching is in the work of Liu *et al.* [76], where content is cached based on its popularity with consumers (***popularity-aware***). The work targets the CCN caching decision policy for popular video streaming contents. The policy that decides whether the new arriving video content should be stored in a particular router or not is based on its popularity rank and the storage size of the router. The proposed policy can help solve the redundancy problem of the content by eliminating the redundant contents stored in the routers along the path from users to server, thus reducing the caching performance degradation. The Zipf [77] popularity distribution is adopted to calculate how popular a video is based on the number of requests. The frequency of video i with rank k_i for N total video elements is calculated as shown in equation (2).

$$f_{i,k_i,s,N} = \frac{1/k_i^s}{\sum_{n=1}^{n=N} 1/n^s} \tag{2}$$

where s denotes the skewness of popularity which characterizes the popularity distribution. Larger s value represents high video popularity diversity and lower s value represents less video popularity diversity. k_i is the popularity rank of the i^{th} video. The smaller it is, the higher the popularity. Thus, the popularity of a video i is defined as the ratio of the number of video i's requests to the total count of video requests.

However, the proposed design requires exchanging the information of the rank table of the popular videos so that the routers can decide whether or not they should cache the video. Although the simulation results on the static video samples show significant improvement on the average transmission hops and server hit ratio compared to the traditional CCN, there is no guarantee that this will improve the network performance when requesting limited time content (i.e. contents that are updated over time) such as temperature or humidity. Besides this, the scheme only tested on two known network topologies, binary tree and cascade network topologies.

Naeem *et al.* [78] goes a step further and proposed a periodic caching strategy (**PCS**) that addresses the cache-to-hits ratio, content retrieval latency, and cache distance between the source and consumer (stretch) to improve caching performance. Each node in PCS uses a distinctive table to store statistical information of the received Interest to find the most frequently requested content, i.e. popular content. The statistical information includes content name, frequency count, requested time, and threshold. The threshold is the maximum value that indicates whether or not a content is considered the most frequently requested content. If a content reaches the threshold, it is stored in the edge nodes of an autonomous system. When the edge node becomes full, PCE uses LRU as the replacement policy to release the cache of the incoming content. The evicted content follows the betweenness-centrality caching concept [32] in each autonomous system, hence, the evicted content cached at the nodes with the highest number of interfaces during content dissemination. Therefore, the content popularity will remain the same, which reduces the path length (stretch) to the desired content and improve the cache-to-hit ratio. Moreover, Naeem *et al.* implement PCE in an autonomous system with Abilene and tree topologies to prevent the replication of similar content within it, thus, reducing content retrieval latency and using memory efficiently.

C. HYBRID (DEVICE- AND CONSUMER-DRIVEN CACHING)

A CCN content object includes a freshness parameter which determines how long it should be kept in the content store before considered stale. Quevedo *et al.* [75] proposed a consumer-driven freshness approach as a different freshness mechanism, in which clients may specify their own freshness value per interest packet. However, Suarez *et al.* [67], added a further control on information freshness and proposed an information freshness as service level parameter that can be negotiated with a gateway and later applied on and used by the device. Both device (i.e. gateway) and consumer interfere with the freshness decision. The authors deploy two algorithms for information freshness management: Communication Signalling Algorithm (CSA) for the application side, and Freshness Control Algorithm (FCA) for the gateway. CSA is responsible to update and send freshness request specifying the desired freshness coefficient while FCA adds/updates a record of the freshness request, evaluate which coefficient to configure on the device, and sends the new freshness coefficient to the desired devices.

D. TIME-BASED CACHING

Zhang *et al.* [79] built a caching time model to characterize the caching process for an individual packet and to estimate (or ***forecast***) the packet's caching time on the next hop node. For each NDN router, the caching time model is used to predict how long the Data packet will be in the adjacent node (upstream node towards producer) when a data packet is cached. This caching model uses the LRU cache replacement policy [80] and the assumption technique proposed by

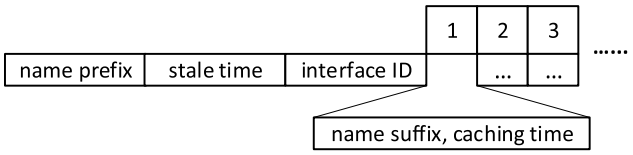

FIGURE 7. FIB structure in the caching time model [79].

Che *et al.* [81], along with the mean arrival rate (as constituent Poisson process) of Interest, to get the inter-arrival time, and from it, the cache-existing time is derived.

The FIB table is adjusted so that each face is attached with a list of all name suffixes belonging to the name prefix of the Data packets cached in the CS over time. Besides, the calculated cache-existing time is stored and attached to each name suffix in the FIB. Therefore, when a Data packet arrives, these two fields are actually processed and attached to the FIB outgoing face, where its corresponding forwarding name prefix is fully included in (or at least the first part of) the name prefix of the arrived Data packet that matches the incoming face of the arrived Data packet, as shown in Fig. 7. Moreover, the arrived Data packet must include status information, known as Status of the Next Upstream Interface (SNUI), of the next upstream node to calculate the caching time. The above considers a stochastic model that is built for the cache-based forwarding (see Section VII) to reduce the number of multicast forwarding packets sent by a router. While the model guarantees a valid hit ratio of content caching and decreases the Interest traffic, it does not provide a solution that minimizes the number of hops to retrieve the cached data.

The aforementioned caching managements are all pull-based caching, where a Data packet is cached after satisfying a pending Interest. Muralidharan, *et al.* [82], [83], however, introduced a ***Push-based*** caching model as a complement solution for categorizing the data traffic into three types; event-based traffic, query-based traffic, and time-based or periodic update traffic which is basically a push-based traffic. The Push-based caching is used for the time-based traffic where sensor devices send and update their information (added along with the Interest namespace) after a regular inter-arrival time using Interest in a pushed-based mechanism. This updates the caching routers within the network with fresh information (e.g. temperature) so that consumer will be able to retrieve fresh Data on demand. The CS in the caching routers acts as a temporary cache to satisfy the consumer Interests that requires Data packets from time-based, or push type, traffic. The sensor devices periodically generate and send update information in a fixed inter-arrival time to the caching routers. The data requested by the consumer is considered to be fresh if the inter-arrival time preceding the data arrival at the consumer is adjunct or overlaps with inter-arrival time period preceding the data generated by the producer. The authors also show that cache hit probability can be used to calculate the cache miss of a particular data packet since the inter-arrival time of the sensor devices is fixed, i.e. to ensure a reliable caching model.

Meddeb *et al.* [84] considered the cache freshness requirement and proposed the Least Fresh First (LFF) algorithm as a cache replacement policy for NDN. LFF relies on prediction calculations and estimates the residual life of the cached content. It is essentially designed for unpredicted or triggered type of traffic, e.g. presence sensor sends an update message on current motion. For that, LFF uses ARMA model [85], a time series analysis, as a forecasting tool to predict future sensors' events. The result will be seen as the remainder of the time for the future event or data. LFF is executed in a node when there is new Data to cache. If the node's cache is full, LFF performs a traverse over the cache contents to select and evict the one with the least remaining time plus the cache arrival time. The new Data is pushed into the CS stack along with the calculated residual lifetime of the Data and its cache time. The authors use their LFF replacement policy with consumer-cache caching strategy and argue that it can be used with any caching strategy.

Table 2 summaries the caching schemes of the aforementioned implementations for NDN- and CCN-based IoT applications. It lists the proposed caching policies and strategies and shows the modifications and differences along with the metrics used for evaluation. There are four categories: device-driven caching, consumer-driven caching, hybrid (both device and consumer) caching and time-based caching. From these categories, we observed that in-network caching impacts network bandwidth and energy efficiency for better IoT network performance. Although NDN and CCN have a native caching mechanism, it is not well suited for many wireless IoT network environments with stringent requirements of their resource-constrained devices and information freshness. The aforementioned works proposed alternative caching solutions to overcome this issue. Device-driven caching schemes use/compute the properties of the network devices to decide where to cache the data, though there are few proposals in this area. In router-dependent scheme, consumers and producers target the SR for their Interest and Data packet transmission, resulting in all traffic passed and cached centrally in the SR. pCASTING uses the device resources to decide whether or not to cache any received content. In contrast, consumer-driven caching schemes, which uses consumer's need as the target of where the data should be cached, have been addressed by the research community in terms of content freshness, information lookup, and content popularity. The freshness-aware strategy allows consumers to specify their information freshness level as an optional field added in the Interest packet, while TCS uses tags to allow consumers classify and store the cached contents using CBF. The popularity-aware caching policy caches content based on its popularity with consumers and includes a popularity rank to prioritize caching of contents. PCS also caches content based on its popularity with consumer, but additionally addresses the content retrieval latency and cache distance between the source and consumer. FCA strategy has considered both device-driven caching and consumer-driven caching. It proposed a consumer freshness mechanism similar to the freshness-aware strategy that also can be negotiated with and controlled by gateway device as a service level parameter. This hybrid mechanism needs further exploring by the research community on consumers and devices scalability. Time-based caching schemes involve a time parameter or counter to forecast the existence of a cache content. The *forecast* model estimates and includes the cache residual lifetime on the next hop to guarantee a valid hit ratio of content. LFF policy specifically estimates the cache residual lifetime for unpredicted or triggered type of traffic. Time-based traffic is also considered by the Push-based caching, which allows sensor devices to periodically send and update their contents by attaching the new content to the Interest namespace. However, the caching model was only tested on applications or services that generate small amounts of data, e.g. temperature. Although time-based caching scheme seems like a promising approach that increases the valid hit-ratio and decrease Interest traffic, only few researchers explored this approach. Further attention is required to check the feasibility of applying push-based caching in periodic monitoring services, as well as alarm propagation mechanisms that require large file transfers.

VI. ACCESS CONTROLS AND POLICIES

A. ROLE-BASED ACCESS CONTROL

In this section, we discuss the implementations which designed access control mechanism per-application or per-service. This mechanism is specifically designed to remove an application or service's discretion when accessing end devices. For example, a lighting application should not have permission (maybe a read policy only) to access temperature or humidity sensors over the network. This requires a predefined role (access policy) for each application to eliminates discretion over NDN end devices.

Application-based access control: The lighting control case [59] for building automation system provides authenticated access control to fixtures via authorization policies and uses NDN naming to reflect access restriction. The AM entity is in charge of issuing a signed access control list and uses it to assign application policies on fixtures. Naming, on the other hand, takes part of control policies using the key attributes in the syntax of the *attribute/value* pair. The trusted model of the framework proposes an application-based access control mechanism for multiple applications with a different namespace. Each fixture can have multiple namespaces to be accessed by the available applications in the system. For example, *fixture1* can have two namespaces, */ndn/app1/fixture1* and */ndn/app2/fixture1,* to be accessed by its owner applications app1 and app2 respectively. However, each application cannot become the owner of the fixture namespace that contains the other application's name without receiving additional namespace ownership from AM or the other application itself.

Similarly, NDOMUS [56] uses a configuration and authorization manager, co-located in an HS, that plays a similar role of AM and CM in lighting control case [59]. Besides its

TABLE 2. The implementation differences of the cache improvement schemes.

Scheme	Caching Policy	Caching Strategy	Forwarding Strategy	Modifications	Architecture	Evaluation Metrics
Router-dependent [74]	LRU	LCE	MF: based on LPM, mainly in SR	• Services mapped by the SR. Consumer & producer requests always target the SR (SR-dependent).	CCN	• Case Study: Camera; publish video & acquire computing service.
Probabilistic [34]	LRU	pCASTING	MF: based on LPM	• Energy level & cache occupancy are the hardware attributes used for the probabilistic decision besides the freshness attribute. • Data packet received, pCASTING is performed for caching decision.	NDN: ndnSIM Simulation tool	• Cache hit ratio. • Energy level using *Cumulative Distribution Function*. • Data retrieval delay. • Received data packets. • Storage capability.
Freshness-aware [75]	LRU	LCE based on adjusted freshness mechanism	MF: based on LPM	• Additional steps in CS to determine if the stored copy meets the freshness requirement or not.	NDN: ndnSIM	• Freshness for different CS size. • Cache hit ratio. • Average transmission hops.
TCCN [73], Tag-assisted	CBF	TCS	TCCN: based on LPM of T-FIB	• Additional tag-based table besides FIB (called T-FIB). • Additional tag column in CS & PIT. • Tag list in CS. • CCN lookup order change to tag list, CS, PIT and then T-FIB/FIB.	NDN: ndnSIM	• Matching and lookup time. • Number of name prefix.
Popularity-aware [76]	LRU & Popularity Rank using Zipf	Cache Path Redundancy -free	MF: based on LPM	• Cache decision policy replaced with the proposed scheme. • Zipf used to calculate the popularity rank of the data content (video).	CCN: Simulation	• Server hit ratio. • Average transmission hops for delay.
PCE [78]	LRU	Periodic caching strategy	Native	• Additional distinctive table for statistical information. • The table includes content name, frequency count, requested time, and a threshold. • Popular caches are stored at the nodes with highest number of interfaces. i.e. betweenness-centrality [32].	CCN: SocialCCNSim	• Cache-to-hit ratio. • Stretch decrement (path length between the cache-source and consumer). • Content retrieval latency.
SLA [67]	-	FCA *(freshness level adjusted by FAS)*	Native	• FCA: handles freshness decision; controlled by consumer and gateway (through internal notification). • FAS: handles freshness coefficient request.	NDN: ndnSIM	• Quality of information.
Forecast [79]	LRU	LCE based on Caching Time Model (Che)	Cache-aware Forwarding (CF): MF based on LPM and Caching Time Model	• Data packet received, Caching Time Model is applied. • CCN lookup order change to PIT, CS and then FIB. • SNUI cyclically executed. • FIB includes name suffixes of the cached packet	NDN: Simulation	• Hit ratio. • Average delay. • Forwarding number.
Push-based caching [82]	-	Push-based caching	Scheduler-based forwarding	• Cache content is included in the namespace of the Interest packet sent by the sensor devices. • Router updates the CS with the fresh information received from the Interest packet.	NDN: Simulation	• Network load. • Round-trip-time. • Link delay. • Cache Miss Ratio.
LFF [84]	LFF	Consumer-cache strategy	Native	• ARMA model is used to forecast and calculate the residual life of the cached content. • CS contains additional columns, i.e. residual life and cache time of each content.	CCN: ccnSim	• Server hit reduction ratio. • Hop reduction ratio. • Response latency.

purpose to control and set up the operations, such as device discovery, registration, key distribution in the home network, it also maintains access control list of the applications, including the owner's application, that requests access to the home network. The applications must first interact with the HS for configuration instructions and access control list

to be able to trigger an action or collect secure sensitive data.

Service-based policy enforcement: In Homenet [53], a naming scheme is developed to provide service accessibility and policy enforcement, which is then managed and controlled by the IR. The structure of the naming scheme, shown in Fig. 3, is composed of six levels: access scope, service scope, device scope, content scope, policy scope, and service-API scope. This means that services are driven by policy requirements, and the naming hierarchy helps accommodate expressions of policy at service, device or content level. The highest level, accessibility scope, can be used to restrict services from accessing only locally and/or globally, while the service API scope uses the attributes that express service actions and parameters for policy enforcement. The service API along with access policies are published as part of the service profile using the SPDP protocol. Network level policy is applied through policy-based routing. On the other hand, the Homenet gateway is equipped with a named-based firewall to impose service policies such as accessibility.

B. ATTRIBUTE-BASED ACCESS CONTROL

Different from the role-based access control systems, this section discusses implementations designed for access control mechanism per-consumer or per-user. The consumer here is granted access to specific end devices with specific right access through the use of policies regardless of the application or services used. For example, a consumer in a building network can have permission (maybe a read policy only) to access temperature sensors in room1 and humidity sensors in room2. This will not only allow the consumer to be granted access to those sensor devices, but it may also use their applications (temperature over room1 and humidity over room2).

Consumer-based policy enforcement: The architecture of secure IoT management, SLA [67], uses the gateway as the intermediate between the client/consumer and the IoT devices to provides a secure and authorized supporting mechanism for its operation. One of its functionalities is the policy-enforcement. The gateway also acts as the reference point that allows an authorized consumer to retrieve the consumer-specific policy before starting any operation on the IoT devices. Therefore, the consumer and gateways must first concertedly authenticate and set the parameters for the security mechanisms used in their future communication. Thus, consumers with verified *consumer-ID* (identity) may access the IoT devices, which means all authorized consumers must be pre-provisioned in the gateway using their *consumer-ID*s. Once the authorization procedure is finished and communication is secured, the consumer can request its access policy (consumer-specific policy) from the gateway. The policy includes a list of all the devices under the gateway authorized for the consumer. Each device in the policy list is attached with its CCN names, a set of information items (sensors data) and a set of commands that the consumer can access and use.

Identity- and capability-based: Unlike the previous per-application access control, NDN-BMS [61] has gone further and provides an access control scheme based on identity and capability, and supports user privilege management. It uses public keys, which is signed by a trusted authority, to identify consumers. The Access Control List (ACL) is the first configuration of each gateway to identify consumers to access its data. The consumers should have the APL, which is stored in a repository specifying the accessible data namespaces. NDN-BMS naming scheme has a *user* namespace, parallel to the *building* namespace, which employs the ACL structure to describe consumers with different privilege levels. Each prefix in the ACL records the identity of the consumers authorized to access the data under it, as shown in Fig. 4. NDN-BMS has BMS manager responsible to update the user privilege changes, through two-way Interest-Data exchanges for the affected gateways and maps ACL to APL (and vice versa), Notification Interest is sent by the BMS manager to the affected gateway. The gateway replays back with acknowledgment, following with an Interest request of the latest ACL. Consumers are granted access to the devices' data by the BMS manager using the identity certificate and the capability certificates, which are published and stored in a repository. However, this may create an ACL or user certificate management problem.

Although the sensors' data is encrypted by the gateway and only the legitimate consumer with access privileges can get it when requested, NDN-BMS avoids publishing an encrypted copy per user. It uses a shared symmetric key to encrypt the sensor data while using the asymmetric encryption scheme for the distribution of the shared symmetric key. The gateway uses Key Distribution Service (KDS) when generating a new shared symmetric key, similar to the TLS/SSL protocol used in today's Internet. However, this approach differs from TLS in that the encryption key (shared) is associated with the data itself rather than the communication channel. Therefore, there is no need to secure the communication perimeter.

As a summary, this section classifies the access controls and policies designed for NDN- and CCN-based IoT applications into two categories. Table 3 displays the access control mechanisms of the aforementioned works and their specific requirements. It also shows the granularity of the mechanisms along with whom they are enforced on. The first category addressed the proposed mechanisms of role-based access control, where such mechanisms eliminated the discretion of an application or service when providing access to end devices. Some of these mechanisms, as in the first category, provide access control to be enforced on multiple applications as in NDOMUS and NDN-LC, or multiple services as in Homenet. Their naming hierarchy helps accommodate the expressions, format, or syntax of the policy at the service, device or content level. The second category comprises a few proposed mechanisms of attribute-based access control that are enforced on consumers rather than applications or services, to implement role-based access control. The proposed mechanisms grant consumer access to specific end

TABLE 3. The implementation differences of the access control mechanisms.

Access Control Reference	Implicit in The Naming Scheme	Access Control Mechanism	Enforced on	Granularity (*Control Access to*)	Conditions
Application-based access control [59]	Yes	• Issues a signed access control list through AM. • Uses key attribute, syntax *attribute/value*, to grant ownership to app's namespace.	Application namespace	Nodes (*Fixtures*)	• Tied to naming scheme. • Fixtures have their own namespace for each app. • Public key installation, per namespace, is required for fixtures.
NDOMUS [56] *(Application-based access control)*	Yes	• Co-located in HS, as AM. • Has similar mechanism to [59]. • Also applied for action-based applications.	Application namespace (*action or sensing tasks*)	Nodes (*End devices*)	• Tied to naming scheme. • Devices have their own namespace for each app. • Public key installation, per namespace, is required for end devices.
Service-based policy enforcement [53]	Yes	• Through IR, translates *action-flags* to apply appropriate actions. • Uses API scope to indicate service actions and parameters. • Applies network level policy.	Producer	Nodes (*Devices*)	• Tied to naming scheme. • Well-known policy enforcement rule must be encoded to action flags. • SPDP needed to publish and access the service. • Policy-based routing required.
Consumer-based policy enforcement [67]	No	Uses the gateway: • for policy enforcement. • to deliver *consumer-specific* policy to authorized consumer, including a list of devices *(i.e. names, info items and commands*).	Consumer	Nodes (*IoT devices*)	• Consumer must be pre-provisioned in the gateway. • Consumer must be authorized by the gateway.
Identity- and capability-based access control [61]	Yes	Uses: • gateways to manage ACL. • naming (*/user*) to form consumer's APL. • BMS manger to update consumer privileged changes for the affected gateways. • identity capability certificates to grant access to consumers.	Consumer	Hierarchal (*Gateways, which applies them to devices*)	• Naming structure (*/user*) needed to form consumer's APL. • Authentication needed before granting both certificates. • Repository needed to store ACL and APL. • Gateway needs to obtain its own configuration when first installed.

devices with specific rights access through defined policies, regardless of the applications or services that use the end devices. SLA uses a reference node to authenticate, authorize, and grant consumers a consumer-specific policy before any operation, and enforces policies on the consumer. NDN-BMS used a similar approach, and further provides capability-based access control using APL attached for each user and supports user privilege management using two-way Interest-Data exchange with the BMS manager. However, NDN-BMS is tied to its naming schemes to identify and assign ACLs and APLs. In fact, most of the access control mechanisms are implicit to the designed naming scheme of their implementations. Only SLA does not implicitly use a hierarchal naming scheme in its implementation, therefore, there is no support for hierarchal granularity, unlike NDN-BMS.

VII. FORWARDING STRATEGIES OR ROUTING

A. NAMING-BASED FORWARDING

Policy-based Forwarding: Homenet [53] has introduced policy-based routing that imposes action-flags on consumer requests which helps published services to implement policy enforcement. It presumes that the services must associate their well-known policy enforcement rules when first published in the FIB, which is then translated into appropriate actions and applied at the gateways or internal routers. The FIB is modified to include action-flags, encoded and mapped with the service prefixes and its list of next hop faces. Similarly, gateways in NDN-BMS [61] use consumer identities and capabilities for the forwarding decisions. The gateways grant access to consumers and forward consumer requests after checking the ACLs. Their ACLs are configured on bootstrap by the BMS manager, which is also responsible for any ACL updates.

Device-driven forwarding: Song *et al.* [74], in its proposed internetworking scheme, groups the network devices into edge networks, which contain producers (resource-constrained devices) and consumers, as well as core networks, which contains SRs. They did not modify the CCN structure for the proposed scheme. Instead, they interposed the SRs between the consumers and the producers. The internetworking scheme lets the edge network devices depend on the SRs in the core network. It is based on task mapping where each SR must respond to all the service requests from consumers, and then communicate with the devices (e.g. sensors) required to perform those services. In other words, the consumer is no longer required to know and

include the resource-constrained device name in its Interest request to retrieve the required data (e.g. snapshot from a camera). The consumer only needs to include the service name provided by the SR (e.g. retrieval service), in its Interest request. However, this process requires task mapping and translating of at least two Interest request packets (from the consumer in the edge network to the SR in the core network, and then from the SR to the sensor device in the edge network) and two corresponding Data packets (from sensor devices to the SR and from the SR to the consumer).

Function-oriented forwarding: ICWSAN [49] alters the concept of CCN hierarchical naming to become function-oriented naming, which addresses the producer's function instead of the piece of data. The ICSWAN naming scheme contains the information object name, *CP:InformationID*. Therefore, the forwarding mechanism is altered to suit the new naming scheme. It includes four modes: (1) anycast mode, which is the default mode that forwards an Interest to a nearest producer holding the corresponding information using the longest prefix-matching approach at the CP level, (2) multicast mode (usually used by sink node), which forwards Interest to multiple producers in the area with the CP prefix, (3) unicast mode, which forwards an Interest to the exact producer using the longest prefix-matching, and (4) broadcast mode, which targets all nodes. The modes are specified by including two bits in the information object name header.

Moreover, ICWSAN rarely follows the request and response operation of the CCN convention. The authors proposed a new forwarding mechanism, called CI, to support event/emergency notification and to save devices' energy when there is no need to generate the same Interest packet for the same type of information. The Interest lifetime in CI is longer compared to native CCN and must not be deleted even if a corresponding Data packet is received. It remains valid until an explicit cancel message is received or the lifetime expires. Whenever changes of sensing data or events occur, a Data packet is issued and forwarded back to the requester continuously following the Interest.

B. LOCATION-BASED FORWARDING

Location information-based routing: The NDN-VIN [66] forwarding strategy is based on location-based naming, which is designed to consider locations, and changed to suit its architecture. The authors extended the FIB of the NDN router to include the entry of the location information-based routing (i.e. city, district, street). The vehicle interested in the next location information sends its Interest piggybacked with three fields as the name structure (see Section III). The fields include *dLs* with request information type (e.g. traffic), *sL* with gathered information from the vehicle itself (e.g. vehicle speed), and the next locations on the path of *sL* to *dLs*. The router (or aggregator) that receives the Interest decides on the forwarding path based on the location information of the vehicle and its next locations. It decides if the received Interest packet should be segregated over their related outgoing faces, based on the dL labels, and sent to every router (lower level aggregator) in charge of the split *dLs*. A timer, which differs from the basic PIT timer, is needed at the NDN router to wait for multiple Data packets. The timer duration must be inherited from the upper-level aggregator and decremented by the lower level aggregator. The corresponding Data packets containing requested information are then aggregated and forwarded back to the requester. The above process enables Data packet aggregation and Interest packet segregation in NDN. However, it requires an extension to the PIT steps when an Interest packet is received and stored by the router. The router needs to decide whether to directly forward the Interest based on the matched entry in the PIT or to split the original Interest into multiple new Interest packets, and then send them to the corresponding outgoing faces toward the lower level aggregators.

Geographical-based forwarding: Navigo [86], another vehicular network system, dubbed V-NDN [87], introduced geographical-based forwarding that uses geographical areas in its implementation. Different from [66] which uses a predefined naming scheme to identify the geographical locations, Navigo follows Military Grid Reference System (MGRS) to map data names to geo-areas in a learning process. The geographical-based forwarding strategy works as follows: (1) explore the surrounding areas of the node for producer or intermediate nodes, and then, when the producer responds with a Data packet, (2) calculate the shortest path towards the producer, and use it for forwarding future Interests under the same name prefix that the exploring Interest used.

The first phase is an exploration phase (broadcast) performed using a Link Adaptation Layer (LAL) [87], which is conceptually Layer 2.5, designed to overcome collision detection/recovery for broadcast transmissions in IEEE 802.11. The LAL maintains of a Face-to-Area (F2A) table, which stores and maps FIB faces to geo-areas (MGRS). The records of the F2A table are updated by extracting the MGRS information from any received Data packets sent to it by the provider. By broadcasting the Interest, all the surrounding nodes will learn the geo-area of the sender and the farthest nodes from it. The second parameter used is obtained by calculating the shortest path using the Dijkstra algorithm (using streets as edges and intersections as nodes) with a probabilistic approach that prioritized edges (streets) having larger roads. The FIB table may have multiple entries with the same name prefix bound to different faces (having different geo-areas for the same data name). Therefore, as part of the shortest path calculation, the forwarding strategy selects one face with the highest probability of success in a round-robin manner. If only one name prefix is available with very low probability, the exploration process will be repeated.

Position-aware forwarding: Rehman and Kim [88] proposes a forwarding mechanism based on the current physical position of the various nodes: consumer, producer and intermediate nodes. Using the position coordinates, the physical distance between the consumer and the producer is calculated

and the shortest physical distance is chosen for the forwarding transmission. When processing an Interest request, each intermediate node will calculate its distance to the producer using the Euclidean distance formula, and forward the Interest (with its current position) only if it is nearer to the producer when compared with the consumer's distance. Meanwhile, the intermediate node also checks its remaining energy and will not forward the Interest if it is less than a threshold. These processes are repeated for each node receiving the Interest packet until it reaches the producer. The Data packet follows the same process when sent back towards the consumer. While the proposed forwarding sustains the same NDN table structure, it assumes all nodes know the current positions of each other including intermediate nodes and producers. Moreover, the Data packet is forwarded towards the consumer using the same forwarding mechanism as the Interest, which adds extra computation that might be avoided if the native NDN mechanism is used for Data packet forwarding.

C. ADAPTIVE FORWARDING

Hail *et al.* [89] presented an *Adaptive Interest Forwarding* (AIF) strategy, or energy-aware forwarding, that implements controlled flooding with a timer-based packet suppression technique, which exploits Interest overhearing where the node defers the Interest forwarding with a specific delay and drops it if the same Interest were overheard on the radio channel. AIF uses capability-based calculation for the deferment time and prioritizes forwarding from higher-capability (i.e. higher remaining battery level) nodes. AIF limits the number of Interest transmissions by calculating longer deferment times for nodes with low battery levels, causing nodes to only forward the Interest packet if they do not overhear an identical Interest transmission, consuming minimal additional energy for the overhearing operation. However, this means that such a node will not be involved in decisions regarding the caching of the corresponding Data packet.

Context-aware adaptive forwarding: Saxena and Raychoudhury [90] proposed an adaptive forwarding strategy, called Cdf, for forwarding emergency events traffic of a NDN-based smart health IoT system. The Cdf strategy involves the integration and calculation of three context parameters: packet type, signal quality and prefix forwarding status, in the forwarding process. The packet type represents the type of traffic, whether emergency event packet or normal packet. Signal quality (low or high) represents the resource availability which can be detected from the Signal-to-Noise Ratio (SNR). The prefix forwarding status is included for each FIB entry of each interface and has three status values (i.e. average, good, or bad) for representing link performance, based on the round-trip-time (RTT) for the given FIB entry. These parameters values are used as condition attributes for the Cdf forwarding decision, and for rules to be executed accordingly. The authors defined specific conditions based on the attribute values to forward emergency event traffic: e.g., forwarding emergency traffic to all interfaces with good RTT values regardless of its signal quality.

Scheduler-based forwarding: Muralidharan *et al.* [82], [91] proposed a Markov Decision Process (MDP)-based scheduler for the forwarding strategy, which considers the type of requested message/Interest. The *MDP-based scheduler* satisfies the latency prerequisites by choosing the best interfaces with low RTT values, based on the received traffic statistics, to fetch Data. The forwarding strategy in [92] is also an MDP-based forwarding strategy using a similar approach that direct Interests to the best interfaces, to improve the QoS in NDN. However, it did not consider the delay-tolerance needed in most IoT applications. Muralidharan *et al.* addressed that issue by categorizing traffic Interests into three types: event-based traffic with low latency requirement, query-based traffic with medium priority, and periodic traffic with no latency prerequisites. The Interest type is included in the namespace. The scheduler changes the interface ranking, i.e. applies the forwarding decision, for every message received and calculates the state transition probabilities by relying on the IoT traffic type, channel condition, latency prerequisite factor, and RTT.

In fact, categorization of traffic types has been addressed previously by the PPT model [83] for efficient data exchange in IoT-NDN environment. However, PPT uses a naive forwarding strategy and does not distinguish between the traffic types unlike the strategy adopted in [82], where the interface ranking can be changed based on the RTT and other additional parameters.

D. DIRECTED FORWARDING

Amadeo *et al.* [93] has extended the forwarding strategy of the basic NDN using *Direct Diffusion* [94] principles, called *dd-NDN*, and came up with an enhanced NDN forwarding strategy for WSNs that establishes a path from the sink node to a producer. The forwarding strategy includes a *direction state* that is set during a discovery (broadcast) phase to establish paths from Data producers to the sink. These paths are used by the sink for future data retrieval from the producers and would reduce collisions and traffic overheads, as claimed by the authors.

In the discovery phase, the sink broadcasts an Interest packet targeting a producer. The intermediate nodes rebroadcast it randomly within a short time interval to control the flooding. The producer then issues a Data packet and forwards it back to the sink through the intermediate nodes. Each intermediate node capable of forwarding the Data packets stores a named entry in the Next Hop Table (NHT), binding the Interest packet from the discovery phase with the identifier for the previous node where the Data packet came from. This entry is known as the *direction state*. In other words, the NHT table is formed during the discovery phase to establish a path among the nodes from the sink to the producer. Therefore, every subsequent Interest requests by the sink will use the same path, by including the identifier of the sink's immediate next hop, taken from NHT, allowing only that node to receive the Interest. The immediate next hop node, in turn, includes its next hop identifier, taken from

its NHT, to forward the Interest. This process is repeated for every node along the path until the Interest reaches the producer.

Cache-aware forwarding: Zhang *et al.* [79] proposed a cache-aware forwarding scheme based on a caching time model (described in section IV). This approach changes the NDN forwarding behaviour by reducing the number of multicast forwarding packets (Interests) sent by the router. It forecasts the existence of the needed Data packet on the next hop nodes by calculating/estimating its caching time, and based on it, chooses the next hop node.

The forwarding scheme starts when an Interest packet arrives at the NDN router and there is no data satisfying the request. The router will look for a match in the FIB. If found, the router will predict the cache existence time of the next upstream node, and forward the Interest to one of the randomly selected upstream nodes with equal probability, instead of multicasting the Interest to matched interfaces as in the case for native NDN. The order of the routing steps in this scheme is modified where the router first processes the PIT table instead of the CS to reduce the lookup overhead and duplicated interests, and to obtain a more reasonable cache-access distribution. The FIB lookup comes after the PIT and CS lookups. The scheme reduces the number of forwarded multicast packets and guarantees a valid hit ratio for cached content. However, the caching time calculation increased the forwarding delay and became noticeable when data size is small. In addition, it is not known how this model will work in a large-sized network and how it will impact memory usage.

Hierarchal routing: LASeR [71] proposes a hierarchal routing scheme with a custom forwarding strategy based on device advertisement similar to the RPL [95] approach. It constructs a forest of trees, each rooted to an anchor device. As mentioned previously, LASeR has three phases: the first and second phases are designed for network discovery, network authentication, device authentication, and key delivery. The third phase is designed for routing construction or path advertisement but depends on the success of the first two phases (see Section IX for more details on these phases - discovery and authentication). When a device connects to the forest network, the first two phases are performed, not only to gather the security information from the IM but to ensure mutual trust between the device and the network before attaching the device to the forest network of the Anchor Node (AN). Once the device is on-board and attached to a path (route) toward AN, the third phase starts with the device using *SetNext* message to advertise its route. The message is then propagated hop-by-hop upstream towards the AN informing each device in the path about the advertised route, i.e. to update their routing states. After that, the AN sends a *SetPrefix* message notifying the IM about the device registration.

In fact, each device, including the AN, maintains its routing state using a table called Downstream Forwarding Base (DFB), which maps the MAC address of the next-hop device to the device ID. The forwarding strategy uses both the DFB and FIB for making forwarding decisions to destination devices within the same AN network. However, if a device wants to send a message to another device in the same network, it must climb the tree toward the anchor device until either reaches a common ancestor or reach the anchor itself. Then, it is forwarded downstream through another branch of the tree to the destination. Furthermore, LASeR routing within two or more anchors requires prefix resolution through the IM, which might raise the question on the applicability of the NDN concept, i.e. data-centric approach.

Link-recovery-based forwarding: Meddeb *et al.* [96] targets producer mobility and proposed an adaptive forwarding-based link recovery (AFIRM) strategy. The AFIRM forwarding strategy has two basic phases namely, the FIB construction phase and the link recovery phase for the producer mobility. The FIB construction phase adopts the concept of leaving Data traces used in COBRA [97], but without replacing the classical structure of the node's FIB. It is basically a flooding phase that sends out an Interest for the required information, and used in two situations: at the beginning when all the FIBs are empty, and during Interest forwarding when there is no forwarding information in the FIBs. When a node receives a Data packet in response to the flooding process, the entire name and its prefixes are stored in the FIB along with the incoming and outgoing (according to the PIT) interfaces of the Data packet as FIB entries. Moreover, the number of hops from the producer/cache-node to the receiving node is also stored with each entry. Therefore, any future request will be forwarded to the entry interface with the least number of hops. The link recovery phase is used to avoid transmission failure due to producer mobility and to update the old and new paths. This phase uses a keep-alive mechanism, handled by the gateway, to detect the producer's movement. If there is a failure or mobility change for a producer, the gateway sends a *recovery* packet to delete the FIB entries that lead to the failed producer, via the corresponding outgoing interfaces stored in the FIB. If the producer is attached to another/same gateway, it sends a *positive recovery* packet to add the new forwarding information, based on the outgoing interfaces and the LPM.

E. HEURISTIC FORWARDING

Probability-based multipath forwarding: Lei *et al.* [98] designed a probability-based multipath forwarding strategy for efficient distribution of increasingly large data volumes in a large-scale IoT application (e.g. video streaming in 5G). They adapted the reinforcement learning approach to routing [99], specifically in the distributed implementation for its forwarding strategy [100], and combined it with a network coding technique to utilize idle paths in the network to disseminate data contents to as many routers as possible (for in-network caching), and to balance the traffic load among the network paths. Each node in the forwarding strategy learns the delivery times of a file from one or more Data packets coming from different interfaces. Thus, each node learns the delivery time of each interface involved in transmitting

the file. The content forwarding of a large file is represented in two phases: the exploration phase and the exploitation phase. The exploration phase forwards Interest to two interfaces; the first interface is randomly selected (to explore new paths) with a probability less than a given threshold, and the second interface is selected with a probability calculated from the previously received data belonging to the requested file. This phase requires the forwarding of a few Interests in order to predict the estimated delivery time for the file. In the exploitation phase, only one interface will be selected for forwarding. A probability calculation is used to select the best interface based on the estimated delivery time. This phase will last for a certain amount of Interests or until the network status changes and substantially affects the delivery time.

Table 4 summarizes the aforementioned forwarding strategies, including the forwarding mechanisms and their specific conditions. The table also shows the modification on both FIB and PIT tables for the NDN architecture and the necessity of using specific naming schemes (implicit in-naming scheme) required for the forwarding mechanisms. The forwarding strategies are classified into five categories including naming-based, location-based, adaptive, directed, and heuristic forwarding. The naming-based forwarding mechanisms require a specific naming scheme to function. It is usually devised to complement the main purpose of the specified application. The *policy-based* forwarding includes service policies in the naming scheme to provide efficient and secure access control; the consumer requests in device-driven forwarding include service names and are directed to the SR which performs a task mapping based on the provided service names; ICWSAN forwards Interest to producers based on its functional name; each forwarding mechanism has therefore been developed to conform to the design of the naming scheme.

The second category, location-based forwarding, showed how forwarding strategies use the physical location in their mechanism to achieve efficient forwarding. NDN-VIN decides the forwarding path based on the location information of the vehicle and its next locations gathered by the vehicle. Navigo calculates the shortest-path towards the producer using MGRS mapping for data-names and a flooding phase. The position-aware strategy forwards the information to the nearest, in physical distance, nodes. However, the location information in NDN-VIN and in position-aware strategies depends on the quality of the estimated vehicle/node location/position, while in Navigo, it depends on the flooding phase and an additional forwarding table. Those systems might not work efficiently in some environments and raised the question of how to handle location estimation and network delays caused by the flooding phase.

The above works nevertheless demonstrates the importance of NDN, especially for VIN systems, in terms of delivery efficiency and scalability but more attention from the research community is necessary.

In the third category, adaptive forwarding, the forwarding strategies use multiple parameters to select the best forwarding path. These parameters may be energy, RTT, signal quality or type of traffic, and can be used for the calculation of probability to help in forwarding decisions. AIF uses the node's energy level and limits the flooding phase with a timer-based suppression technique. Cdf does not use a flooding phase but used condition attributes comprising the traffic type, signal quality, and RTT for the forwarding decision. The scheduler-based forwarding approach goes a step further and uses the traffic type, RTT, and additional parameters for interface ranking calculations based on MDP. Although there are only a few researchers working on adaptive forwarding strategies, it seems to be a promising area for applications with latency constraints, since Muralidharan *et al.* [82] managed to reduce the RTT by 20-30% compared to other traditional strategies based on best route selection and multicasting.

In the fourth category, directed forwarding, the forwarding strategies use different forwarding guidelines alongside the FIB. dd-NDN uses a NHT table to store the direction state as the forwarding guidelines; cache-aware forwarding uses SNUI for calculating the caching time; LASeR uses a DFB that maps the MAC address of the next-hop device to the device ID; AFIRM extends the FIB and includes the number of hops and output interfaces for each Data name and its prefixes. These guidelines might be constructed or derived from a calculation model or an initial phase (discovery/flooding phase). The implementations using directed forwarding proves to be more suitable in ad-hoc networks since they are able to adapt to topology changes, but they are of limited use in NDN-based IoT applications which have stringent power requirements. This may be due to the need for additional computation power as its calculations are complex. Moreover, in a highly dynamic environment, the discovery phase may cause an expensive delay every time a network change occurs, in order to estimate the new forwarding direction. The authors of AFIRM addressed that in AFIRM with a link-recovery phase for producer mobility that limits the overhead of the flooding phase. In heuristic forwarding, the proposed forwarding strategy used a reinforcement learning approach. More research in this area is needed, in order to reduce the exploration phase time, since it needs to select at most two random interfaces using probability calculation each time an Interest is received.

VIII. DATA AGGREGATION

In many IoT systems, the amount of raw data generated for archiving and analysis can be large, but nonetheless, highly correlated for a given application [101]. Therefore, IoT systems can pre-process and combine the raw data while forwarding to a sink or storage for future analysis and retrieval by interested consumers. The IoT gateways can execute the analytical operations that the consumers are interested in for a given data stream if the system distributes those operations to the gateways in advance. The NDN transmission mechanism is data-oriented, and can therefore easily adopt

TABLE 4. The implementation differences of the forwarding strategies.

Forwarding Strategy	Implicit in The Naming Scheme	Forwarding Mechanism	FIB Edit	PIT Edit	Conditions
Policy-based forwarding [53]	Yes	• Follows attached action-flag and naming hierarchy.	Includes encoded *action-flag*	-	• Tied to naming scheme. • Policy rules should be applied in the gateway.
Device-driven [74]	No	• Uses task mapping. • At least two pairs of Interest-Data involved.	No	No	• Consumer should know target services. • Services need to be configured.
Functional-oriented [49]	Yes	• Targets producer's function. • Uses 4 transmission modes; anycast, unicast, multicast and broadcast. • Uses Continuous Interest for event.	-	-	• Tied to naming scheme. • Longer lifetime period for CI mode. • The modes are identified using two bits set on the name prefix (info. object).
Geographical-based forwarding [86]	No	• Requires exploration phase (Interest flooding by LAL). • Calculates the shortest-path using probability of roads (the larger the better). • Requires implicit ACK concept for next hop.	FIB bound to F2A table *(i.e. geo-area to face)*	-	• Geo-area dimension must be carefully selected. It may affect the size of the FIB. • Timer that removes unused faces and any reference to them from FIB & F2A tables. • Nodes equipped with GPS.
Location information-based routing [66]	Yes	• Based on vehicle's next location. • Includes push-based data delivery using Interest. • Uses Interest packet segregation. • Uses Data packet aggregation.	Includes location info (*City, District and Street)*	-	• Tied to naming scheme. • PIT timer value in lower level aggregator must be less than upper level one. • Aggregators and segregators follow the naming hierarchy.
Position-aware forwarding [88]	No	• Chooses the shortest physical distance between consumer and producer for both delivery of Interest and Data packets. • Considers node's remaining energy.	No	No	• Assumes all nodes know the current position of each other.
Adaptive interest forwarding [89]	No	• A controlled flood mechanism. • Calculates defer time based on energy level of the nodes.	No	No	• Requires timer-based packet suppression to control the Interest broadcast. • Low energy nodes will not be involved in caching.
Context-aware adaptive forwarding [90]	No	• Integrates three context parameters namely, packet type, signal quality, and prefix forwarding status. • Uses parameter values as condition attributes for forwarding decision.	Includes prefix forwarding status	-	• FIB needs to periodically check the status of the interfaces through interface probing. • Packet traffic should be categorized into emergency or normal type. • Requires SNR and RTT values for each interface.
Scheduler-based forwarding [82], [91]	No	• Based on MDP. • Calculates state transition probabilities to rank interfaces with the best performance. • Chooses the best fitting interfaces based on the Strategy Choice Table (SCT) which contains interface ranking.	FIB bound to SCT table (i.e. interface ranking)	Includes Congestion Control	• Traffic type is categorized into event-based, query-based, and time-based. Then, it is included in the name field of the Interest packet. • Requires calculation for RTT, channel condition, and latency requisite. These calculations are included in the SCT table for the interface ranking.
Direct diffusion-based forwarding [93]	-	• Requires a discovery phase to establish a path from sink to producer. • Stores the direction state in NHT table. • Subsequent Interest is attached with next hop ID, taken from NHT.	No	No	• Follows Direct Diffusion principles. • Timer-based packet suppression for discovery phase.
Cache-aware forwarding [79]	No	• Based on caching time model, predicts cache existing time of upstream nodes. • Forwards to those nodes with equal probability.	No	No	• SNUI status is recorded (in a cyclic manner) for Poisson process. • Interest processing order changed to PIT, CS and then FIB.
Hierarchal routing [71]	No	• Chooses the shortest path towards the anchor device. • Advertises its route hop-by-hop upstream towards AN. • Stores next-hop state in DFB table.	-	Yes	• Requires network discovery and authentication phase, as well as device authentication phase. • Interest must pass through a common anchor device or ancestor device of the sender and receiver. • Routing between two anchors requires prefix resolution through IM.

TABLE 4. Continued.

Link-recovery-based forwarding [96]	No	• Requires FIB construction phase using leaving trace concept used in COBRA. • Forwards Interest to interface with least number of hops. • Uses link recovery phase to handle producer's mobility.	Includes outgoing interface (from PIT) and number of hops	No	• PIT entries (i.e. outgoing interface) must be stored in the FIB table, before they are deleted, along with the number of hops from the producers. • Requires a keep-alive mechanism to handle producer's movement.
Probability-based multipath forwarding [98]	No	• Adapts the reinforcement learning approach and network coding technique. • Learns and estimates the delivery times of a large file from arrived packet(s) of that file. Then, select the best path (interface) based on that. • Discovers new path during packet transmissions of the large file.	Only in query processing operation	Only in query processing operation	• To discover new path, the selection of new interface is based on the probability calculated from the previous received data. • Requires exploration phase to discover new paths and predict the delivery time. • Requires exploitation phase to select the best interface.

data aggregation processes. An application can encode the metadata of the sensor within the NDN naming convention; for example, data type, measurement location, and time of acquisition. Therefore, Interests can be constructed by the aggregation gateways using that naming convention to retrieve the data from the network. For example, multiple sensor readings can be fused and sent together for the same physical event using a multicast Interest for those sensors. The aggregation process can help reduce the network traffic to be sent over expensive wireless communication links, and thus, cause a significant reduction in energy consumption [102]. This section addresses the various ways and techniques used to aggregate and forward the data toward the gateways.

Teubler *et al.* [103] designed a *forwarding service* (i.e.*aggregation service*) on top of the native CCN forwarding that allows the applications to aggregate the data. The service works only when data aggregation is demanded by an application, i.e. through Interest registration request, which includes a unique application name as an identifier. Each time a Data packet is received for an application, the forwarding service will hand it to the application for the aggregation process. The aggregated data from the application is then transferred by the forwarding service to the next node. Ravindran *et al.* [53] uses similar application-driven concept but includes data aggregation process within the service publish and discovery protocol (SPDP) (see Section IX). A SPDP instance is attached locally for every node in the network to provide distributed service discovery without centralized service control. Therefore, when SPDP discovers the available services in the network, it sends a discovery request (Interest) to get multi-source data from the aggregated service list. The Interest is forwarded to the local SPDP instance, which locally stores a state corresponding to the received request, as the Interest is processed and forwarded hop-by-hop. When a SPDP instance receives a Data packet (containing an *aggregated service list*), it combines the received information with its locally stored service information to generate a new aggregated service list to be forwarded back to the original requester.

The NDN-VIN [66] architecture uses the hierarchy of the location-based naming scheme to organize the aggregators for various areas into three levels: */city* as the top-level aggregator, */city/district*as a middle-level aggregator, and */city/district/street* as the bottom-level aggregator, as shown in Fig. 5. As discussed in section VII (forwarding strategies), this architecture uses an Interest packet segregation mechanism to disseminate new Interest packets towards the lower level aggregators to request the data from multiple producers (e.g. traffic information for street-1 and 2). As a result, the Data packets are generated by producers (e.g. cars), and then sent back along the reverse path from the lower level aggregators to the upper-level aggregators via the data aggregation mechanism. This requires managing the PIT entries' lifetimes so that the PITs in upper-level aggregators provide more time for the lower-level aggregators to process and respond to the Interest requests before their corresponding PIT entries are deleted.

Amadeo *et al.* [56] includes the data aggregation mechanism in one of its strategies for multi-party communications, i.e. Multi-source communication (1C:MS). The designed namespace allows the producers to have a common name prefix that defines a similar task. Nevertheless, the producers can also be distinguished by producer-specific name components. For example, when the HS receives a single Interest with a name prefix *myHouse/task/sensing/temperature*, the HS broadcasts it (most likely over a wireless link), while the producers (e.g. thermostat sensors) reply individually with the Data packet carrying the name prefixes in both the Interest, i.e. *myHouse/task/sensing/temperature/*, and the producer-specific components, e.g. */bedroom/nest*. Therefore, when the HS receives all the Data packets from the thermostat sensors, it aggregates all the gathered data and sends a single aggregated Data packet to the consumer. In fact, the above method follows the data retrieval principle from their previous work [104].

Aside from the traditional pull-based data aggregation, Saxena *et al.* [105] proposed a push-based data aggregation (PDA) to periodically update the location of RFID-based tags of the items/devices. The proposed mechanism is designed to regularly transmit the item location in order to overcome the limited RFID signal range, in case data could not reach the central server. The central server monitors for emergency situations and stores the location of instantly needed items in order to deal with future queries by consumers. It assumes the existence of mediators/nodes that are placed in specific locations to collect the location information of in-range items, which periodically transmit their tag information via push-based communications. These tags are assigned with IDs to help identify their distance from the central server, where more distant nodes have larger IDs. Each node collects the location information and combine them with other received location information, then passes them to other mediators with lower IDs. The process is repeated and information from the combined locations is transferred to another lower ID mediator until they reach the central server with ID 0.

The aforementioned works show that data aggregation is necessary for some IoT scenarios in which consumers require data from different and/or large number of producers. They also show that the mechanism for aggregating the data can be applied without extending the completion time or violating basic NDN functionality. Consequently, the congestion and heavy load on devices and routers are decreased. Teubler *et al.* [103] designed the data aggregation mechanism as a service/application attached to the router that requires an application identifier and a registration step. SPDP uses a similar application-driven concept to get an aggregated service list. It includes the aggregation process within the distributed service publish and discovery protocol. NDN-VIN proposed a mechanism for data aggregation which does not require a registration step. It is designed to comply with the Interest packet segregation mechanism for the VIN networks. However, it is only limited to three-level aggregators/routers and requires managing the PIT-entries' lifetimes among the in-path aggregators. 1C:MS integrates data aggregation into the multi-source communication protocol. The consumer uses a common name prefix for all the producers. The producers must reply with Data packets which include their own name prefix and the common name prefix so that the HS will know when to aggregate them. In contrast, PDA uses a push-based data transmission, to attach the aggregated location information of RFID-based tags that send location information regularly. This work assigns IDs for the mediator devices, responsible for the aggregation process, to help identify their distances from the central server. This mechanism helps reduce energy consumption and maintain up-to-date data on the server, consisting of small-sized messages containing the location information of the tags. Nevertheless, push-based data aggregation is expected to improve efficiency further for large data sizes while minimizing energy consumption and network traffic.

IX. DEVICE CONFIGURATION AND MANAGEMENT

A. CENTRALIZED CONFIGURATION AND MANAGEMENT

Fixtures pairing: Burke *et al.* [59] pairs the new fixture with the CM for the bootstrap process, so that applications will execute commands on the fixture. A symmetric key is used to initialize the new fixture. In this system, the symmetric key is obtained from the fixture's enclosure, via a factory-installed barcode. The CM assigns a NDN name to the fixture, synchronizes the fixture clock to its own clock and installs a trusted public key per application domain that belongs to the AM. The fixture then generates a paired key and sends its public key to the CM for future communications. Further distribution mechanisms are then established; for example, ACL names for application access and a long-term secret master key for authentication. Finally, the fixture sends a respond to the CM with the current time and a hash of the previously exchanged information.

Auto-configuration: In [61], the designed system uses the BMS manager daemon to distribute the configuration to any newly installed gateway which connect to the network for the first time. The gateway obtains from the BMS manager its own identity name, access control list, the name prefix of the application and installation that will be used for publishing data, and the public key of the BMS manager. The above process is started when the gateway establishes an NDN Interest/Data exchange. The gateway sends an Interest with a well-known name prefix to the BMS manager, which its daemon should already be listening for, and waiting to service any gateway request. The gateway's serial number is used to derive a shared symmetric key to bootstrap the trust relationship, similar to the device initialization step in [59]. The key can be manually installed on the manager. It is used to sign or encrypt the communication process between the gateway and the manager daemon for further security during the bootstrap process of the gateway.

Namespace-based configuration: NDOMUS [56] sets up all the configuration and operations under one namespace */homeID/conf,* including support for end-devices discovery. It uses a similar configuration process to that used in [59] to enable the fixture pairing configuration and to assign application namespaces under which they can operate, along with the other security management information configuration. The configuration manager maintains a list of active devices using keep-alive broadcast messages that are sent periodically. Devices which fail to react to the broadcast message after a certain interval will be deleted from the active device list.

Network discovery: The LASeR [71] protocol proposes a secure on-boarding mechanism for devices to join a NDN-based IoT network. Inspired by OnboardICNg [72], which introduced the first secure on-boarding protocol that provides authentication and authorization for ICN-based devices, LASeR adapts the onboarding protocol of OnboardICNg to work in conjunction with its proposed hierarchical routing protocol. Both protocols involve at least one trusted neighbor, a device which has already joined the network and has been authenticated by the authentication manager, for the

authentication process of new devices. Since both LASeR and OnboardICNg have similar authentication steps for onboard new devices, this paper will just discuss the on-boarding mechanism for the LASeR protocol.

In LASeR, new devices first discover the neighbors which are already authenticated, registered and part of the hierarchal topology (already on-board) in the LASeR network. The neighbor devices then ask the IM device for the required information to authenticate the network to the new device. In fact, the idea of neighbor discovery in LASeR is more related to discovering a path in the hierarchical network of LASeR and verifying the legitimacy of the network it is connecting to, rather than finding a proper place/order in the discovered path after performing authentication. To do that, the new device sends a *Discovery Request*, which is a broadcast-forwarded Interest with a */discover/* prefix, constituting a join request with a long PIT lifetime and includes its ID, a nonce and its current hop-distance (initially large number) from the AN. Neighbor devices receiving the message will then relay the message, as a new *Onboarding Request*, to its AN along with the authentication parameter. The *Onboarding Request* is signed by the neighbor device, indicating assent for the new device to receive a route towards the AN. The IM completes the authentication process and derives the security keys required for future transmission, and then replies with a network authentication message (including the new device's next-hop neighbor and hop-distance, and AN) to the neighbor devices, which then forward the reply to the new device after reverse-mapping it to satisfy the original *Discovery Request*. However, the processes of *Discovery Request* and *Onboarding Request* may be repeated multiple times to locate a shorter path to the AN. Furthermore, it might trigger subsequent discovery processes for other devices along the downstream path that the new device was added to, in order to adjust their hop-order in the hierarchal network.

B. DECENTRALIZED CONFIGURATION AND MANAGEMENT

Auto-node: The Homenet [53] architecture includes a protocol for network discovery, named *neighbor discovery protocol (NDP)*, that can discover the identifiers of the neighboring nodes in a network. The NDP in a node aims to discover the available neighboring nodes in the network with a simple exchange of information, even in an ad-hoc network. The well-known namespace */ndp* is used for the discovery process. The NDP starts by adding a temporary FIB entry */ndp/pseudo_x* for each active physical interface to enable discovery over *face x*. NDP adds the FIB entry */ndp:s,* to map the NDP service locally to *face s* so that it serves any arriving discovery request. After that, the NDP periodically sends discovery Interests along with their identifiers over the physical faces. When a discovery request (Interest) is received in a neighbor node at *face 1*, it is forwarded to the local *face s* of its NDP service, which learns about the sender's identifier. Finally, the neighbor node includes the sender's (discovered neighbor) identifier (ID_{S1}) when sending subsequent Interests to establish bi-directional adjacency, and then replaces */ndp/pseudo_1* with */ndp/<IDs_1>*. In fact, the protocol requires manual entry of the device ID and does not mandate the installation of a centralized configuration manager. NDP enables network devices to be discovered in a plug-and-play manner, but it does not include any security measures. However, it can be extended to include more information, other than the identifier, to support additional services. An example extension can be security credentials for negotiating neighbor relationships.

C. SERVICE REGISTRATION AND DISCOVERY

In [49], the functional-oriented naming scheme supports service discovery for WSAN networks. The naming is based on the type and functions of the information making it identifiable and visible at the network layer. The structure of this naming, discussed in Section III, comprises two parts, namely the *CP,* and the *InformationID* of producers which facilitate the coordination and collaboration between the producers, based on the service function defined in the CP. For example, any sensor which needs to communicate with other nearby sensors with the name */sensor:xxx* can establish an event for a service called */sensorCategory:Eventxxx*. The semantic name of the interest message is enough to coordinate with one or more actors that have the corresponding name in anycast, unicast or multicast mode. Any sensor that receives the event knows that it may collaborate with other sensors having the same *CP* name */sensor:xxx*. However, a category prefix needs to be configured and set in all the devices along with the service event prefix.

The proposed work for a cyber-physical system [48] introduced a named service bus for service registration and discovery. NSB acts as a mediator between a service and its service consumer (e.g. application). Nodes that provide services must register in the NSB at its local daemon, specifically in the FIB of the NSB's gateway with a common interface and wait for service requests coming from the NSB. On the other hand, a consumer can submit encoded request for the desired service to the local NSB daemon. The local NSB daemon searches for a matching entry in the FIB and then forwards the Interest to another daemon in the network that provides the service, or to a local face that provides the service directly.

In fact, this method does not require a service profile. Services register directly to the NSB using their local daemon via the name prefix of the service, and consumers can request for the services using the naming conventions *%C1*. However, consumers need to know the service's commands and call syntax. There is no consistent naming system within the CPS. Furthermore, the messages exchanged among the three parties might raise the issue of fault tolerance, where it is not clear if the services were unavailable or if the messages were just delayed, especially in scenarios with high system load.

SPDP [53] is developed for service registration and discovery and uses the */spdp* namespace for interest and data exchange. Unlike NSB, SPDP publishes a service profile to register the service. Using the SPDP API, service producers register their services locally following the proposed naming

structure (see Section III). The registered profile of the service includes service ID, access policies, reachability scope, TTL and API to access the service. Any application can discover all the services, or else a specific service with matching criteria, by forwarding the request to the local SPDP, which in turn sends the Interest, including the original node identifier to all active neighbors. SPDP instances receive the request, process it and then forward it hop-by-hop to another instance. The data containing the aggregated service list is combined with the original request, while the service policies are applied and enforced when accessing the FIB. While SPDP can be applied to a tree-based topology, it requires a name-based mesh topology routing protocol to determine the appropriate next hop for request forwarding.

Quevedo *et al.* [106] has extended the core concept of semantic features in [107] and proposed a *semantic-based* service discovery mechanism developed on top of the NDN application layer. The authors developed a novel service-query matchmaking interface for its broker-based service discovery mechanism. Fig. 8 shows the solution entities and interfaces. The *Discovery Broker* holds the information about the available services and interacts with the *Semantic Matching Engine (SME)* to match the incoming queries against the available services. It also listens for service (de)registration requests and consumers' discovery queries, then forwards them to the *SME* over an available transport protocol such as UDP, TCP or ICN. The *SME*is considered an external entity with respect to the *Discovery Broker*. It performs actual queries, runs a different matching algorithm, and keeps track of the registered service for the matching process. Each *Service Provider* must be registered in the *Service Discovery Broker (SDB)* before it can interact with the consumer. On the other hand, any consumer interested in a specific piece of information must interact first with the *Discovery Broker* to get the available service providers that can satisfy its needs. The system has the feature to send service (de)registration requests individually, or else aggregated in one packet, which reduces the network overhead. However, it is only utilized for the face that connects the *Discovery Broker* with the *Semantic Matching Engine*.

Amadeo *et al.* [108] proposed *NDNe* to extend the NDN semantics, packets, and primitives to support service provider discovery and service provisioning phases. They designed the naming scheme to identify the different types of services (e.g. storage or video processing) with a well-known name prefix. In NDNe, the Interest and Data packets are equipped with context-rich attributes to describe consumer demands and provider candidates (i.e. eInterest), and to select capable candidate providers for task accomplishment (i.e. eData). The attributes used are consumer position, direction, speed, content size, and other attributes related to delay and energy needed for task completion. NDNe has a two-step process, namely a *provider discovery* phase and a *service provisioning* phase. In provider discovery, consumers broadcast eInterest requests to their neighborhood using an expanding ring search technique. On the other hand, each provider receiving the

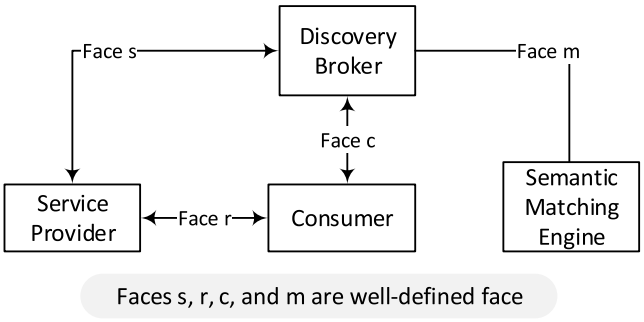

FIGURE 8. Semantic-based service discovery [106].

eInterest estimates the expected time interval needed to connect to the consumer and estimates the energy needed to complete the task. This is to verify whether or not it is a proper candidate for the task, and, therefore, include those estimations in the eData. Each candidate provider calculates random deferral times using a smart deferral scheme that is based on the estimated time interval and energy cost, in order to avoid collisions, and most importantly, to let the best candidate provider for the task be the first respondent. In the service provisioning phase, the consumer chooses the candidate provider who replies first with eData and ignores other late replies. Then, it performs an initial handshake with the selected provider using legacy NDN Interest-Data packets. After that, a unicast content transfer session is established between the consumer and provider, leveraging the layer-2 acknowledged transmission mode for Dynamic Unicast [109].

Amadeo *et al.* [110] extended NDNe into ANDNe to provide a mutual authentication mechanism (trust) between the consumer and the provider, to secure service provisioning at the mobile edge. They enhanced the procedures in the two phases with the means to verify the legitimacy of a content request or service and the means to provide data confidentiality. Both phases are based on authenticated packet exchanges and are cryptographically confidential. The access server is an entity added to the system that enables consumers to verify authorized candidate providers for performing a service, and for providers to verify the consumers asked for the service. Furthermore, packets in ANDNe are enhanced with specific fields carrying security-related information to establish trust between consumers and providers and ensure consumer privacy during data exchange. These security measures are designed to protect against possible attacks, such as eavesdropping, user profiling, and man-in-the-middle attacks. They are also designed to protect service providers from unauthorized consumers that attempt to obtain their services.

Table 5 compares the differences among various implementations providing solutions for device configuration and bootstrapping, along with service registration and discovery. Device configuration and management features are classified into three categories namely centralized configuration, decentralized configuration, and service registration and discovery.

TABLE 5. The implementation differences of discovery services and bootstrapping.

System	Service Registration	Service Discovery	Peer Discovery	Configuration Assigned	Bootstrap (Granularity)	Bootstrap Requirement	Features	ICN Instance
NDN-LC [59]	-	-	No	• Synchronized clocks • AM's public key • Key-pair for communication and identity • ACL • Long-term master key	Yes (fixtures)	• Symmetric key (factory-installed barcode) installation • AM in the network • Selecting fixture's name	-	NDN
NDN-BMS [61]	-	Through user's APL	No	Gateway *Configuration File:* • Identity name • Application-specific prefixes • ACL • BMS manager's public key • Local routable name	Yes (gateways)	For Gateways: • Symmetric key (serial number) installation • BMS manager • Well known-prefix of the BMS manager	• Configuration update	CCN
NDOMUS [56]	-	-	No	• Similar to [59]	Yes (end-devices)	• Similar to [59] • HS advertise the root-name prefix • Well-known name */<root-name>/conf*	• Maintain a list of active devices	CCN
LASeR [71]	-	-	Yes	• Routing Authentication Key • The next-hop ID toward AN • The AN's ID	Yes (devices)	• AN, IM & Trusted neighbor • Long-life PIT for discovery request • Well-known name */discovery* • ID & PSK to derive the Long-Lived keys, Transient Authentication & Encryption Keys	• Network Authentication • Device Authentication • Network Discovery • Secure On-boarding	NDN
Homenet [53]	SPDP	SPDP	NDP	• SPDP: Service profile (service ID, access policies, TTL, scope & API) • NDP: FIB entry (ID) for each neighbor device discovered in the network • Aggregated service list	Yes (devices)	• Well-known name */spdp* • Well-known name */ndp* • Assigning Identifier for each device & service	• Distributed service and node discovery • Network Discovery • Service list is aggregated on discovery	CCN
ICWSAN [49]	No	Yes	Yes	• Through event requests • Based on the functional, name-prefix definition of the designed functional-oriented naming	-	-	• No functional discovery required • Service info. at network level	CCN
NSB [48]	Yes	No	No	• Service producer register its name-prefix in NSB's FIB	No	-	• No service profile (global profile)	CCN
Semantic-based SD [106]	SDB	SME	No	• Service record includes unique ID, name, metadata, semantic description. It is stored in the service table.	-	-	• Semantic service discovery and (un)registration	NDN
NDNe [108]	No	Provider discovery phase	Yes	• Phase-1: Consumer discovers service providers based on context-rich (position, direction, speed, size, delay, and energy), attached to Interest request. • Phase-2: Consumer selects the service provider candidate who replays first.	-	-	• Decentralized service discovery • No service registration • Service provider selection • Consumer to service-provider handshake	NDN
ANDNe [110]	No	Provider discovery phase	Yes	• Extended NDNe [108] to secure its phase-2 (service provisioning) • Security-related information is attached to the packets • An access server is used to verify authorized provider candidates	-	-	• Similar to NDNe but requires access server entity • Mutual (consumer-provider) authentication	NDN

TABLE 6. Incorporation of NDN in IoT with supported functionalities, features, and additions.

Functionality/ Feature/ Addition / NDN-IoT Incorporation	Naming Scheme	Caching Policy	Caching Strategy	Forwarding Strategy	Exploration Phase Required	Role-based Access Control	Attribute-based Access Control	Access Control Granularity	Service Registration	Service Discovery	Peer Discovery	Bootstrap	Centralized Configuration/ Management	Data Aggregation	Publish and Subscribe Support	Push-based Support	FIB Table Modification	PIT Table Modification	Requires Additional Table
NSB [48]	√	×	×	×	×	×	×	×	√	×	×	×	√	×	×	×	×	×	×
ICWSAN [49]	√	×	×	√	×	×	×	×	×	√	√	-	×	×	√	×	-	-	×
Homenet [53], [54]	√	×	×	√	×	√	×	×	√	√	√	√	√	×	√	×	√	-	×
NDOMUS [56]	√	×	×	×	×	√	×	×	-	-	×	√	√	√	×	√	-	-	×
PB-CCNx [57]	×	×	×	×	×	×	×	×	×	×	×	×	√	×	×	√	-	-	×
NDN-SH [58]	√	×	×	×	×	-	-	-	-	-	×	-	√	×	×	√	×	×	×
NDN-LC [59]	√	×	×	×	×	√	×	×	×	×	×	×	√	×	×	×	×	-	×
NDN-SSF [60]	√	×	×	×	×	×	×	×	×	×	×	×	-	×	×	√	-	-	-
NDN-BMS [61]	√	×	×	×	×	×	√	√	-	√	×	√	√	×	×	×	-	-	×
NDN-PS [64]	√	×	×	×	×	×	√	√	-	√	×	√	√	-	√	×	-	-	-
NDN-VIN [66]	√	×	×	√	×	×	×	×	×	×	-	-	×	√	×	√	√	√	×
SLA [67]	×	-	√	×	×	×	√	×	×	×	×	×	√	×	×	×	-	-	×
PHINet [69]	√	×	×	×	×	×	×	×	×	×	×	√	√	×	×	×	√	√	-
LASeR [71]	×	×	×	√	√	×	×	×	-	-	√	√	√	×	×	×	-	√	√
OnboardICNg [72],	×	×	×	√	√	×	×	×	-	-	√	√	√	×	×	×	-	√	√
Router-dependent [74]	×	×	×	√	×	×	×	×	×	×	×	×	√	×	×	×	×	×	-
pCASTING [34]	×	×	√	×	×	×	×	×	×	×	×	×	√	×	×	×	×	×	×
Freshness-aware [75]	×	×	√	×	×	×	×	×	×	×	×	×	-	×	×	×	-	-	×
TCCN [73], Tag-assisted	×	√	√	×	×	×	×	×	×	×	×	×	×	×	×	×	√	×	√
Popularity-aware [76]	×	√	√	×	×	×	×	×	×	×	×	×	×	×	×	×	-	-	√
PCE [78]	×	×	√	×	×	×	×	×	×	×	×	×	×	×	×	×	-	-	√
Forecast [79]	×	×	√	√	×	×	×	×	×	×	×	×	×	×	×	×	√	-	√
Push-based caching [82]	×	-	√	√	×	×	×	×	×	×	×	×	×	×	×	√	√	√	√
LFF [84]	×	√	√	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Navigo [86]	×	×	×	√	√	×	×	×	×	×	×	×	×	×	×	×	√	-	√
Rehman *et al.* [88]	×	×	×	√	×	×	×	×	×	×	×	×	×	×	×	×	×	×	-

TABLE 6. Continued.

Hail *et al.* [89]	×	×	×	√	√	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Saxena *et al.* [90]	×	×	×	√	×	×	×	×	×	×	×	×	×	×	×	√	√	-	×
Amadeo *et al.* [93]	-	×	×	√	√	×	×	×	×	×	×	×	×	×	×	×	×	×	√
Meddeb *et al.* [96]	×	×	×	√	√	×	×	×	×	×	×	×	×	×	×	-	√	×	×
Lei *et al.* [98]	×	×	×	√	√	×	×	×	×	×	×	×	×	×	×	×	√	√	×
Teubler *et al.* [103]	-	×	×	×	×	×	×	×	×	×	×	×	-	√	-	×	×	×	×
Saxena *et al.* [105]	-	×	×	×	×	×	×	×	×	×	×	×	-	√	√	√	×	×	×
Quevedo *et al.* [106]	×	×	×	×	×	×	×	×	√	√	×	-	√	×	-	×	-	-	√
Amadeo *et al.* [108]	×	×	×	×	×	×	×	×	×	√	√	-	×	×	-	-	×	×	×
Amadeo *et al.* [110]	×	×	×	×	×	×	×	×	×	√	√	-	×	×	-	-	×	×	×

The first category requires a centralized entity that the network devices rely on for configuration. *Fixtures-pairing* uses the CM to assign name prefix, install public key for authentication and authorization purposes, and distribute ACL to devices. *Namespace-based configuration* implements a similar configuration process to *Fixtures-pairing* and further includes a keep-alive broadcast message to maintain a list of active devices. *Auto- configuration* uses the BMS manager to distribute and update the gateways' configuration, including information on security management. LASeR provides authentication and authorization for device configuration, similar to *Fixtures-pairing*, but must involve another party, i.e. a trusted neighbor which is already authenticated by the network, before the device can communicate with the IM (configuration manager). This type of configuration approach is not applicable for mobile networks because authentication and authorization of new devices take a great deal of time. Any change in the network topology or device hop-order will involve the same parties repeating the configuration and security processes.

The second category does not require a centralized entity for configuration as it assumes every node is already equipped with the necessary information, e.g. an identifier, for communicating with the other nodes. *Auto-node* is a protocol that can discover neighboring nodes in a network and register their identifier in the local FIB of the discovering-node. However, it is an application-level discovery process with no security features, and no restraint on the discovery range. Further research is needed in this area to address those issues since it will benefit decentralized ad-hoc networks. As a point of reference, trust and authentication issues in ad-hoc networks were addressed in [111], [112].

In the third category, service registration and discovery is included as an integral part of the configuration process. This is a complex and dynamic issue in IoT systems as it requires interoperability among different applications and providers. ICWSAN [49] includes service discovery as part of its solution. The service identifies and broadcasts itself with the category namespace included in the Interest packet. The NSB registers the services in the FIB table of the NSB gateway, which then waits for any incoming service requests. SPDP [53] publishes a service profile instead of a FIB to register the services provided by producers. The profile includes ID, access policies, reachability, scope, TTL and service API. Semantic-based service discovery also publishes a service profile, i.e. discovery broker, similar to SPDP but features a semantic concept to support efficient consumer service queries. It includes a semantic matching engine to interact with consumers' queries to discover available services.

The NSB, SPDP, and *Semantic-based* solutions require a centralized entity, gateway, or broker to handle the service registration and discovery requests, unlike ICWSAN which rely on broadcasting the service requests over the network to reach available service providers. In contrast, NDNe and ANDNe do not use centralized entities but use an expanding ring search technique (controlled broadcast) as part of the provider discovery phase. NDNe uses Interests with attached context-rich attributes to describe consumer demands and to determine provider capability for task completion. A provisioning phase is also taken into account so that the consumer chooses the best service provider(s) for the task. ANDNe improves upon this and provides mutual authentication between the consumer and the provider to protect them against possible attacks. The decentralized service discovery approaches are useful for many IoT systems where centralized service profiles are not applicable. Nevertheless, the service registration and discovery mechanisms need to be further enhanced for better and efficient performance over a wide range of possible use-cases such as different mobility patterns, communication technologies, and service workloads or types.

Table 6 shows an overall summary of all the NDN-IoT implementations with their supported functionalities, features, and additions.

X. CONCLUSION

This paper discusses the state of the art related to the incorporation of named data networking for Internet of Things applications. The various architectures and mechanisms of NDN and how they addressed and resolved IoT challenges in a resource-constrained environment to implement IoT framework functionality were explored in detail. This paper provides insights into various NDN-enabled IoT applications and services, discussed modules developed to address IoT-specific challenges and examined the feasibility of their proposed features. Furthermore, the paper classifies the incorporation of NDN and IoT paradigms into six categories, in terms of framework functionality: device and data naming scheme, caching strategies and policies, access control and policies, forwarding strategies and routing, data aggregation, as well as device configuration, bootstrapping and service discovery. The proposed mechanisms and approaches for each category are also compared and analyzed to help the reader understand the strengths and weaknesses of the surveyed works.

REFERENCES

[1] G. Mulligan, "The 6LoWPAN architecture," in *Proc. 4th Workshop Embedded Networked Sensors (EmNets)*, Cork, Ireland, 2007, pp. 78–82.
[2] Z. Sheng, S. Yang, Y. Yu, A. Vasilakos, J. A. Mccann, and K. K. Leung, "A survey on the IETF protocol suite for the Internet of Things: Standards, challenges, and opportunities," *IEEE Wireless Commun.*, vol. 20, no. 6, pp. 91–98, Dec. 2013.
[3] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, and C. Papadopoulos, Named Data Networking Project, Tech. Rep. NDN-0001, Oct. 2010. [Online]. Available: http://named-data.net
[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.
[5] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. 5th Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, Rome, Italy, 2009, pp. 1–12.
[6] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking (draft)," in *Proc. Dagstuhl Seminar 10492 Inf.-Centric Netw.*, Wadern, Germany, 2011, Art. no. 10492.
[7] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, Jul. 2012.
[8] 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," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 1024–1049, May 2014.
[9] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, p. 181, Oct. 2007.
[10] *COMET Project Overview*. Accessed: Sep. 25, 2018. [Online]. Available: http://www.comet-project.org/
[11] M. Ain *et al.* "D2.3—Architecture definition, component descriptions, and requirements," Deliverable D2.3, Publish-Subscribe Internet Routing Paradigm Project, Helsinki Univ. Technol., Feb. 2009. [Online]. Available: http://www.psirp.org
[12] M. F. Majeed, S. H. Ahmed, and M. N. Dailey, "Enabling push-based critical data forwarding in vehicular named data networks," *IEEE Commun. Lett.*, vol. 21, no. 4, pp. 873–876, Apr. 2017.
[13] S. Guizani, "Internet-of-Things (IoT) feasibility applications in information centric networking system," in *Proc. 13th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Valencia, Spain, Jun. 2017, pp. 2192–2197.
[14] A. Afanasyev *et al.*, "NFD developer's guide," NDN Tech. Rep. NDN-0021-Revision 10, Jul. 2018. [Online]. Available: http://named-data.net
[15] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Wählisch, "Information centric networking in the IoT: Experiments with NDN in the wild," in *Proc. 1st Int. Conf. Inf.-Centric Netw. (INC)*, Paris, France, 2014, pp. 77–86.
[16] A. Abane, P. Muhlethaler, M. Daoui, and H. Afifi, "A down-to-earth integration of named data networking in the real-world IoT," in *Proc. 6th Int. Conf. Future Internet Things Cloud Workshops (FiCloudW)*, Barcelona, Spain, Aug. 2018, pp. 243–249.
[17] S. K. Datta and C. Bonnet, "Integrating named data networking in Internet of Things architecture," in *Proc. IEEE Int. Conf. Consum. Electron.-Taiwan (ICCE-TW)*, Nantou, Taiwan, May 2016, pp. 1–2.
[18] M. Amadeo, C. Campolo, and A. Molinaro, "Internet of Things via named data networking: The support of push traffic," in *Proc. Int. Conf. Workshop Netw. Future (NOF)*, Paris, France, Dec. 2014, pp. 1–5.
[19] K. Pentikousis *et al.*, "ICN baseline scenarios," IETF, Tech. Rep., Mar. 2013. [Online]. Available: https://tools.ietf.org/html/draft-irtf-icnrg-scenarios-00
[20] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building efficient wireless sensor networks with low-level naming," *ACM SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 146–159, Dec. 2001.
[21] M. Amadeo, C. Campolo, J. Quevedo, D. Corujo, A. Molinaro, A. Iera, R. L. Aguiar, and A. V. Vasilakos, "Information-centric networking for the Internet of Things: Challenges and opportunities," *IEEE Netw.*, vol. 30, no. 2, pp. 92–100, Mar. 2016.
[22] W. Shang, A. Bannis, T. Liang, Z. Wang, Y. Yu, A. Afanasyev, J. Thompson, J. Burke, B. Zhang, and L. Zhang, "Named data networking of things (invited paper)," in *Proc. IEEE 1st Int. Conf. Internet-Things Design Implement. (IoTDI)*, Berlin, Germany, Apr. 2016, pp. 117–128.
[23] M. Amadeo, C. Campolo, and A. Molinaro, "Forwarding strategies in named data wireless ad hoc networks: Design and evaluation," *J. Netw. Comput. Appl.*, vol. 50, pp. 148–158, Apr. 2015.
[24] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, and K. Drira, "Producer mobility support in named data Internet of Things network," *Procedia Comput. Sci.*, vol. 109, pp. 1067–1073, Dec. 2017.
[25] S. Arshad, M. A. Azam, M. H. Rehmani, and J. Loo, "Recent advances in information-centric networking-based Internet of Things (ICN-IoT)," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2128–2158, Oct. 2017. [Online]. Available: https://arxiv.org/abs/1710.03473v3
[26] D. Saxena, V. Raychoudhury, N. Suri, C. Becker, and J. Cao, "Named data networking: A survey," *Comput. Sci. Rev.*, vol. 19, pp. 15–55, 2016.
[27] I. Moiseenko, "Fetching content in Named Data Networking with embedded manifests," NDN, Tech. Rep. NDN-0025, Revision 1, Sep. 2014. [Online]. Available: http://named-data.net
[28] NDN Project Team, "NDN technical MEMO: Naming conventions," Tech. Rep. NDN-0022, Revision 1, Jul. 2014. [Online]. Available: http://named-data.net
[29] L. Saino, I. Psaras, and G. Pavlou, "Icarus: A caching simulator for information centric networking (ICN)," in *Proc. 7th Int. Conf. Simul. Tools Techn.*, Lisbon, Portugal, 2014, pp. 66–75.
[30] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. C. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable ICN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 147–158, Aug. 2013.
[31] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, and K. Drira, "Cache coherence in machine-to-machine information centric networks," in *Proc. IEEE 40th Conf. Local Comput. Netw. (LCN)*, Clearwater Beach, FL, USA, Oct. 2015, pp. 430–433.
[32] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, and K. Drira, "How to cache in ICN-based IoT environments?" in *Proc. IEEE/ACS 14th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Hammamet, Tunisia, Oct./Nov. 2017, pp. 1117–1124.
[33] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro, "Named data networking for IoT: An architectural perspective," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Bologna, Italy, Jun. 2014, pp. 1–5.
[34] M. A. Hail, M. Amadeo, A. Molinaro, and S. Fischer, "Caching in named data networking for the wireless Internet of Things," in *Proc. Int. Conf. Recent Adv. Internet Things (RIoT)*, Singapore, Apr. 2015, pp. 1–6.
[35] CCNx Project, (2012). *PARC, CCNx*. [Online]. Available: http://ccnx.org
[36] NDN Team, (2017). *NDN Forwarding Process*. [Online]. Available: http://named-data.net/doc/NFD/current/
[37] University of Basel. *CCN-Lite*. [Online]. Available: http://www.ccn-lite.net/

[38] E. Baccelli, O. Hahm, M. Gunes, M. Wahlisch, and T. C. Schmidt, "RIOT OS: Towards an OS for the Internet of Things," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Turin, Italy, Apr. 2013, pp. 79–80.

[39] B. Saadallah, A. Lahmadi, and O. Festor, "CCNx for contiki: Implementation details," INRIA, Tech. Rep. RT-0432, 2012.

[40] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki—A lightweight and flexible operating system for tiny networked sensors," in *Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw.*, Tampa, FL, USA, Nov. 2004, pp. 455–462.

[41] Z. Ren, M. A. Hail, and H. Hellbrück, "CCN-WSN—A lightweight, flexible content-centric networking protocol for wireless sensor networks," in *Proc. IEEE 8th Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process.*, Melbourne, VIC, Australia, Apr. 2013, pp. 123–128.

[42] W. Shang, A. Afanasyev, and L. Zhang, "The design and implementation of the NDN protocol stack for RIOT-OS," in *Proc. IEEE Globecom Workshops, GC Wkshps*, Dec. 2016, pp. 1–6.

[43] O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes, "Operating systems for low-end devices in the Internet of Things: A survey," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 720–734, Oct. 2016.

[44] Y. B. Zikria, H. Yu, M. K. Afzal, M. H. Rehmani, and O. Hahm, "Internet of Things (IoT): Operating system, applications and protocols design, and validation techniques," *Future Gener. Comput. Syst.*, vol. 88, pp. 699–706, Nov. 2018.

[45] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An operating system for sensor networks," in *Ambient Intelligence*. Berlin, Germany: Springer-Verlag, 2007, pp. 115–148.

[46] *FreeRTOS, a FREE Open Source RTOS for Small Embedded Real Time System*. Accessed: Oct. 21, 2018. [Online]. Available: https://www.freertos.org

[47] *Berkeley's OpenWSN Project*. Accessed: Oct. 21, 2018. [Online]. Available: https://openwsn.atlassian.net/wiki

[48] H. Hellbrück, T. Teubler, and S. Fischer, "Name-centric service architecture for cyber-physical systems (short paper)," in *Proc. IEEE 6th Int. Conf. Service-Oriented Comput. Appl.*, Koloa, HI, USA, Dec. 2013, pp. 77–82.

[49] N.-T. Dinh and Y. Kim, "Potential of information-centric wireless sensor and actor networking," in *Proc. Int. Conf. Comput., Manage. Telecommun. (ComManTel)*, Ho Chi Minh City, Vietnam, Jan. 2013, pp. 163–168.

[50] P. De, N. Riou, and W. Vermeylen, *Building Automation Routing Requirements in Low-Power and Lossy Networks*, document RFC 5867, Jun. 2010.

[51] N.-T. Dinh, "RESTful architecture of wireless sensor network for building management system," *KSII Trans. Internet Inf. Syst.*, vol. 6, no. 1, p. 18, 2012.

[52] Y. Ma, X. He, and Z. Cao, "Draft-ma-core-dhcp-pd-01–dhcp option for CoAP proxy discovery," IETF, Tech. Rep., 2012. [Online]. Available: https://tools.ietf.org/html/draft-ma-core-dhcp-pd-01

[53] R. Ravindran, T. Biswas, X. Zhang, A. Chakraborti, and G. Wang, "Information-centric networking based homenet," in *Proc. IFIP/IEEE IM Workshop, 5th Int. Workshop Manage. Future Internet (ManFI)*, Ghent, Belgium, May 2013, pp. 1102–1108.

[54] T. Biswas, A. Chakraborti, R. Ravindran, X. Zhang, and G. Wang, "Contextualized information-centric home network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 461–462, Aug. 2013.

[55] T. Chown, J. Arkko, A. Brandt, O. Troan, and J. Weil, "IPv6 home networking architecture principles," draft-ietf-homenet-arch-07, IETF, Tech. Rep., Feb. 2013. [Online]. Available: https://tools.ietf.org/html/rfc7368

[56] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro, "Information centric networking in IoT scenarios: The case of a smart home," in *Proc. IEEE Int. Conf. Commun. (ICC)*, London, U.K., Jun. 2015, pp. 648–653.

[57] O. Waltari and J. Kangasharju, "Content-centric networking in the Internet of Things," in *Proc. 13th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, Jan. 2016, pp. 73–78.

[58] S. H. Ahmed and D. Kim, "Named data networking-based smart home," *ICT Express*, vol. 2, no. 3, pp. 130–134, Sep. 2016.

[59] J. Burke, P. Gasti, N. Nathan, and G. Tsudik, "Securing instrumented environments over content-centric networking: The case of lighting control and NDN," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Turin, Italy, Apr. 2013, pp. 394–398.

[60] J. Burke, P. Gasti, N. Nathan, and G. Tsudik, "Secure sensing over named data networking," in *Proc. IEEE 13th Int. Symp. Netw. Comput. Appl.*, Cambridge, MA, USA, Aug. 2014, pp. 175–180.

[61] W. Shang, Q. Ding, A. Marianantoni, J. Burke, and L. Zhang, "Securing building management systems using named data networking," *IEEE Netw.*, vol. 28, no. 3, pp. 50–56, May/Jun. 2014.

[62] W. Shang, J. Thompson, M. Cherkaoui, J. Burkey, and L. Zhang, "NDN.JS: A javascript client library for named data networking," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Turin, Italy, Apr. 2013, pp. 399–404.

[63] J. Burke, A. Horn, and A. Marianantoni, "Authenticated lighting control using named data networking," NDN Tech. Rep. NDN-0011, Oct. 2012. [Online]. Available: http://named-data.net

[64] W. Shang, A. Gawande, M. Zhang, and A. Afanasyev, "Publish-subscribe communication in building management systems over named data networking," NDN Tech. Rep. NDN-0066, Oct. 2018. [Online]. Available: http://named-data.net

[65] M. Zhang, V. Lehman, and L. Wang, "Scalable name-based data synchronization for named data networking," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Atlanta, GA, USA, May 2017, pp. 1–9.

[66] Z. Yan, S. Zeadally, and Y.-J. Park, "A novel vehicular information network architecture based on named data networking (NDN)," *IEEE Internet Things J.*, vol. 1, no. 6, pp. 525–532, Dec. 2014.

[67] J. Suarez, J. Quevedo, I. Vidal, D. Corujo, J. Garcia-Reinoso, and R. L. Aguiar, "A secure IoT management architecture based on information-centric networking," *J. Netw. Comput. Appl.*, vol. 63, pp. 190–204, Mar. 2016.

[68] D. Corujo, R. L. Aguiar, I. Vidal, and J. Garcia-Reinoso, "A named data networking flexible framework for management communications," *IEEE Commun. Mag.*, vol. 50, no. 12, pp. 36–43, Dec. 2012.

[69] C. Seales, T. Do, E. Belyi, and S. Kumar, "PHINet: A plug-n-play content-centric testbed framework for health-Internet of Things," in *Proc. IEEE Int. Conf. Mobile Services*, New York, NY, USA, Jun./Jul. 2015, pp. 368–375.

[70] Node.js Foundation. (2017). *Node.js*. Accessed: Nov. 14, 2018. [Online]. Available: https://nodejs.org

[71] T. Mick, R. Tourani, and S. Misra, "LASeR: Lightweight authentication and secured routing for NDN IoT in smart cities," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 755–764, Apr. 2018.

[72] A. Compagno, M. Conti, and R. Droms, "OnboardICNg: A secure protocol for on-boarding IoT devices in ICN," in *Proc. 3rd ACM Conf. Inf.-Centric Netw.*, Kyoto, Japan, 2016, pp. 166–175.

[73] Y. Song, H. Ma, and L. Liu, "TCCN: Tag-assisted content centric networking for Internet of Things," in *Proc. IEEE 16th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Boston, MA, USA, Jun. 2015, pp. 1–9.

[74] Y. Song, H. Ma, and L. Liu, "Content-centric internetworking for resource-constrained devices in the Internet of Things," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Budapest, Hungary, Jun. 2013, pp. 1742–1747.

[75] J. Quevedo, D. Corujo, and R. Aguiar, "Consumer driven information freshness approach for content centric networking," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, Apr./May 2014, pp. 482–487.

[76] Z. Liu, M. Dong, B. Gu, C. Zhang, Y. Ji, and Y. Tanaka, "Inter-domain popularity-aware video caching in future Internet architectures," in *Proc. 11th EAI Int. Conf. Heterogeneous Netw. Quality, Rel., Secur. Robustness*, Taipei, Taiwan, Aug. 2015, pp. 404–409.

[77] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, YouTube, everybody tubes: Analyzing the world's largest user generated content video system," in *Proc. 7th ACM SIGCOMM Conf. Internet Meas. (IMC)*, San Diego, CA, USA, 2007, p. 1.

[78] M. A. Naeem, R. Ali, B.-S. Kim, S. A. Nor, and S. Hassan, "A periodic caching strategy solution for the smart city in information-centric Internet of Things," *Sustainability*, vol. 10, no. 7, p. 2576, Jul. 2018.

[79] Z. Zhang, H. Ma, and L. Liu, "Cache-aware named-data forwarding in Internet of Things," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, USA, Dec. 2015, pp. 1–6.

[80] A. Dan and D. Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 18, no. 1, pp. 143–152, Apr. 1990.

[81] H. Che, Y. Tung, and Z. Wang, "Hierarchical Web caching systems: Modeling, design and experimental results," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1305–1314, Sep. 2002.

[82] S. Muralidharan, A. Roy, and N. Saxena, "MDP-IoT: MDP based interest forwarding for heterogeneous traffic in IoT-NDN environment," *Future Gener. Comput. Syst.*, vol. 79, pp. 892–908, Feb. 2018.

[83] S. Muralidharan, B. J. R. Sahu, N. Saxena, and A. Roy, "PPT: A push pull traffic algorithm to improve QoS provisioning in IoT-NDN environment," *IEEE Commun. Lett.*, vol. 21, no. 6, pp. 1417–1420, Jun. 2017.

[84] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, and H. Mathkour, "Least fresh first cache replacement policy for NDN-based IoT networks," *Pervas. Mobile Comput.*, vol. 52, pp. 60–70, Jan. 2019.
[85] S. Makridakis and M. Hibon, "ARMA models and the box-Jenkins methodology," *J. Forecasting*, vol. 16, no. 3, pp. 147–163, May 1997.
[86] G. Grassi, D. Pesavento, G. Pau, L. Zhang, and S. Fdida, "Navigo: Interest forwarding by geolocations in vehicular named data networking," in *Proc. IEEE 16th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Boston, MA, USA, Jun. 2015, pp. 1–10.
[87] G. Grassi, D. Pesavento, G. Pau, R. Vuyyuru, R. Wakikawa, and L. Zhang, "VANET via named data networking," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, Apr./May 2014, pp. 410–415.
[88] R. A. Rehman and B.-S. Kim, "Location-aware forwarding and caching in CCN-based mobile ad hoc networks," *IEICE Trans.*, vol. E99-D, no. 5, pp. 1388–1391, May 2016.
[89] M. A. M. Hail, M. Amadeo, A. Molinaro, and S. Fischer, "On the performance of caching and forwarding in information-centric networking for the IoT," in *Wired/Wireless Internet Communications* (Lecture Notes in Computer Science), vol. 9071. Cham, Switzerland: Springer, 2015, pp. 313–326. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-22572-2_23
[90] D. Saxena and V. Raychoudhury, "Design and verification of an NDN-based safety-critical application: A case study with smart healthcare," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 5, pp. 991–1005, May 2019.
[91] S. Muralidharan, A. Roy, and N. Saxena, "MDP-based model for interest scheduling in IoT-NDN environment," *IEEE Commun. Lett.*, vol. 22, no. 2, pp. 232–235, Feb. 2018.
[92] J. Su, X. Tan, Z. Zhao, and P. Yan, "MDP-based forwarding in named data networking," in *Proc. 35th Chin. Control Conf. (CCC)*, Chengdu, China, Jul. 2016, pp. 2459–2464.
[93] M. Amadeo, C. Campolo, A. Molinaro, and N. Mitton, "Named data networking: A natural design for data collection in wireless sensor networks," in *Proc. IFIP Wireless Days (WD)*, Valencia, Spain, Nov. 2013, pp. 1–6.
[94] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 2–16, Feb. 2003.
[95] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, and R. Alexander, *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, document RFC Editor 6550, Mar. 2012.
[96] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, and S. Gannouni, "AFIRM: Adaptive forwarding based link recovery for mobility support in NDN/IoT networks," *Future Gener. Comput. Syst.*, vol. 87, pp. 351–363, Oct. 2018.
[97] M. Tortelli, L. A. Grieco, G. Boggia, and K. Pentikousisy, "COBRA: Lean intra-domain routing in NDN," in *Proc. IEEE 11th Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, Jan. 2014, pp. 839–844.
[98] K. Lei, S. Zhong, F. Zhu, K. Xu, and H. Zhang, "An NDN IoT content distribution model with network coding enhanced forwarding strategy for 5G," *IEEE Trans. Ind. Informat.*, vol. 14, no. 6, pp. 2725–2735, Jun. 2018.
[99] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Proc. 6th Int. Conf. Neural Inf. Process. Syst.*, San Francisco, CA, USA, 1993, pp. 671–678.
[100] R. Chiocchetti, D. Perino, G. Carofiglio, D. Rossi, and G. Rossini, "INFORM: A dynamic interest forwarding mechanism for information centric networking," in *Proc. 3rd ACM SIGCOMM Workshop Inf.-Centric Netw. (ICN)*, Hong Kong, 2013, pp. 9–14.
[101] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: A survey," *IEEE Trans. Wireless Commun.*, vol. 14, no. 2, pp. 70–87, Apr. 2007.
[102] S. Harada, Z. Yan, Y.-J. Park, K. Nisar, and A. A. A. Ibrahim, "Data aggregation in named data networking," in *Proc. IEEE Region 10 Conf. (TENCON)*, Penang, Malaysia, Nov. 2017, pp. 1839–1842.
[103] T. Teubler, M. A. M. Hail, and H. Hellbrück, "Efficient data aggregation with CCNx in wireless sensor networks," in *Advances in Communication Networking* (Lecture Notes in Computer Science), vol. 8115. Berlin, Germany: Springer, 2013, pp. 209–220. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-40552-5_19
[104] M. Amadeo, C. Campolo, and A. Molinaro, "Multi-source data retrieval in IoT via named data networking," in *Proc. 1st Int. Conf. Inf.-Centric Netw. (INC)*, Paris, France, 2014, pp. 67–76.
[105] D. Saxena, V. Raychoudhury, and C. Becker, "An NDNoT based efficient object searching scheme for smart home using RFIDs," in *Proc. 18th Int. Conf. Distrib. Comput. Netw. (ICDCN)*, Hyderabad, India, 2017, pp. 1–6.
[106] J. Quevedo, M. Antunes, D. Corujo, D. Gomes, and R. L. Aguiar, "On the application of contextual IoT service discovery in Information Centric Networks," *Comput. Commun.*, vols. 89–90, pp. 117–127, Sep. 2016.
[107] M. Antunes, D. Gomes, and R. Aguiar, "Semantic features for context organization," in *Proc. 3rd Int. Conf. Future Internet Things Cloud*, Rome, Italy, Aug. 2015, pp. 87–92.
[108] M. Amadeo, C. Campolo, and A. Molinaro, "NDNe: Enhancing named data networking to support cloudification at the edge," *IEEE Commun. Lett.*, vol. 20, no. 11, pp. 2264–2267, Nov. 2016.
[109] C. Anastasiades, J. Weber, and T. Braun, "Dynamic Unicast: Information-centric multi-hop routing for mobile ad-hoc networks," *Comput. Netw.*, vol. 107, no. 2, pp. 208–219, Oct. 2016.
[110] M. Amadeo, C. Campolo, A. Molinaro, C. Rottondi, and G. Verticale, "Securing the mobile edge through named data networking," in *Proc. IEEE 4th World Forum Internet Things (WF-IoT)*, Singapore, Feb. 2018, pp. 80–85.
[111] S. Guleng, C. Wu, X. Chen, X. Wang, T. Yoshinaga, and Y. Ji, "Decentralized trust evaluation in vehicular Internet of Things," *IEEE Access*, vol. 7, pp. 15980–15988, 2019.
[112] H. Zhao, X. Yang, and X. Li, "cTrust: Trust management in cyclic mobile ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 62, no. 6, pp. 2792–2806, Jul. 2013.

AHED ABOODI received the B.Sc. degree in information technology and engineering from the University of Aden, Yemen, in 2007, and the M.Sc. degree in computer science from Universiti Sains Malaysia, Penang, Malaysia, in 2012, where he is currently pursuing the Ph.D. degree in computer science with the School of Computer Sciences. He has been with Aden Community College, Yemen, since 2012. His current research interests include wireless sensor networks, namely, data networking, the Internet of Things, and mobile applications.

TAT-CHEE WAN received the B.S.E.E. and M.S.E.E./C.E. degrees from the University of Miami, Coral Gables, FL, USA, and the Ph.D. degree in computer science from Universiti Sains Malaysia (USM), Penang, Malaysia, where he is currently an Associate Professor with the School of Computer Sciences, and the Deputy Director of the National Advanced IPv6 Centre (NAV6), with 20 years of teaching and research experience. He was with Motorola Malaysia Sdn. Bhd. as a Senior R&D Engineer in software development for two-way radios. This research is also conducted in affiliation with the National Advanced IPv6 Centre (NAV6), USM. His current research interests include efficient protocols for the Internet of Things, wireless sensor networks, satellite communications, quality of service, and real-time systems.

GIAN-CHAND SODHY received the B.Comp.Sc. degree (Hons.) and the M.Sc. degree in computer science from Universiti Sains Malaysia, Penang, Malaysia. He has been a Lecturer with the School of Computer Sciences, Universiti Sains Malaysia, since 1999. His current research interests include data replication and data grid.