Chapter 3
Content-Centric Networks (CCN)

Syed Hassan Ahmed, Safdar Hussain Bouk and Dongkyun Kim

Abstract Several initiatives have been taken in the past decade to improve the architecture and performance of information-centric networks (ICN). In this context, Van Jacobson presented a new architecture called the "content-centric network" (CCN), which was funded by the PARC research company. The main goal of CCN was to change host-centric communication into content-centric communication. In CCN, the requester is known as a "consumer" who sends an "interest" to the network, and any node with the requested data can send back the "content" to the consumer by way of the same path. This simple overview seems superficial without explanation. Therefore, in this chapter, we provide readers with the history of CCN followed by its basic operations. Moreover, we describe the different components of CCN in detail such as the following: (1) What constitutes "content"? (2) What is the structure of content and an interest message? (3) How can the interest can be forwarded and, in response, how can data retrieval be efficient compared with the current Internet architecture? We believe that this chapter will enable our respective readers have a solid background about the CCN and, in later stages, that they can become active researchers in the given field.

Keywords CCN · CCNx · Content · Data · Interest · Content retrieval · Applications

3.1 Introduction

The Content Centric Networking (CCN) [1] Project was started and managed by the PARC (Palo Alto Research Center) with the aims to develop a flexible, simple, and universal next-generation communication architecture. It is also targeted that CCN will alleviate communication complexities, will require less configuration, must be efficient and scalable, and will have the application design patterns of the current communication technologies. The implementation of CCN is called "CCNx," and the project has gone through different stages, such as the preliminary implementation of CCNx and currently different issues are being researched. The next phase of this project is the formalization of different standardization proposals that plan to

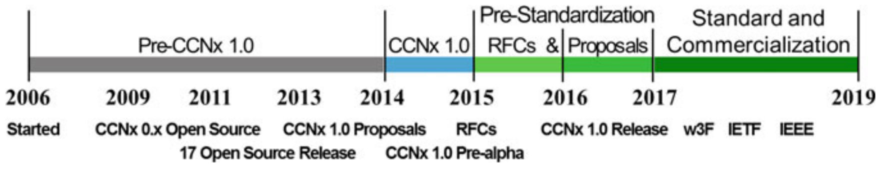


Fig. 3.1 CCN timeline

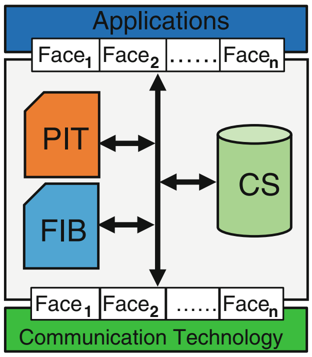
standardize CCN in 2017 to 2018. The detailed timeline of the CNN Project is shown in Fig. 3.1.

It is stated by the Jacobson [2], who coined the concept of CCN, "*that any architecture designed to run over anything is necessarily an overlay. The things that matter are the capabilities.*" The argument was based on the historical overlay technologies, such as IP technology, which was initiated as an overlay on the phone system; in turn, currently the phone system is used as an overlay on IP technology. IP in its entirety is independent of any layer in the current Internet technology. Likewise, CCN demonstrates and envisions the same characteristics as an overlay that can run over any technology, including IP, and the converse is also possible, i.e., anything can run over CCN including IP. A similar project was started and funded by the NSF (National Science Foundation) in mid-2010, under the Future Internet Architecture program, called "named-data networking" (NDN) [3]. NDN follows the basic CCN functionality with some modifications in the forwarding daemon implementation.

Basically, CCN is one of the future Internet architectures to provide location-independent communication of data, contents, information elements, streaming contents (video/audio), or content segments using content *names* [note that the terms "data," "contents," "content objects," or "information elements" are interchangeable in the context of this book; similarly, a complete content or content object can be a combination of multiple fragments]. It is considered that the content name must uniquely identify the complete content or any fragment(s) of the content and that it is independent of the location information. The content communication is also performed using these names. Thus, it is the center of the CCN architecture. The implementation of CCN [1, 4], i.e., CCNx, provides multiple services including end-to-end, loop-free, multi-hop, multi-path communication, flow control, palpable and impulsive multicast service, security intrinsic to content rather than the connection, content integrity irrespective of the delivery path, distributed caching in the network, etc. The terms "content" or "data" are referred as "ContentObjects" in CCNx. CCN relinquishes the execution of application notwithstanding the type and nature of the lower-layer technology.

End-to-end communication is provided by CCN applications using CCN as an innate part of the application to avoid a more layered complexity. A simplified pull-based or receiver-initiated content communication is adapted in basic CCN architecture. However, a few proposals have adopted push-based CCN architecture [5] suitable for specific applications, e.g., Internet of Things (IoT) applications [6],

Fig. 3.2 CCN daemon

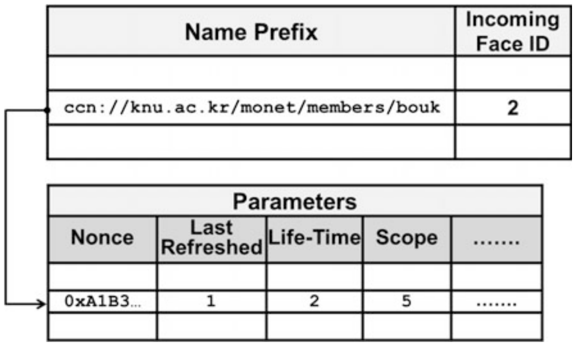

Applications
Face$_1$ Face$_2$ ...... Face$_n$
PIT
FIB
CS
Face$_1$ Face$_2$ ...... Face$_n$
Communication Technology

wireless-sensor network (WSN) applications [7], etc. In a pull-based communication, the receiver node or device sends a content request message (called an "interest message" in CCN terminology), and the node with requested data replies with the data message. In CCN, the requesting node is called the "consumer node," and the request satisfying or data bearer node is called the "provider node." All CCN-enabled nodes implement the forwarding and content-caching capabilities (Fig. 3.2).

CCN uses two elementary messages to achieve the simplified pull-based communication, i.e., "*interests*" and "*data*" messages [8]. Along with these two messages, some data structures are maintained at each node to properly forward interest data messages in the network including pending interest table (PIT), forwarding information base (FIB), and the content store (CS). The consumer node sends the interest message if it requires any content, and the provider node replies with the data message with the content or a fragment of the content. The interest message contains the content name and other information to properly *identify* (not *locate*) the content. All of the intermediate nodes that receive the interest message first search in their CS, and if they do not find the specified content, they make an entry in the PIT. Every PIT entry contains content name, incoming face, and other information of every received interest. Then, the intermediate node searches the content name within the FIB to find the outgoing face. The outgoing face linked to the entry with the longest prefix match is used to forward the interest message. When an intermediate node receives the data message, it may hold the copy of the content message depending on whether a caching policy is implemented. Following is the brief description of the data structures and concepts partly mentioned previously that are used by the CCN node to achieve loop-free forwarding and caching in the network.

- **Face**: The term "face" is taken from the word "interface." The face in CCN is nothing but a simplified concept of communication or application interface. This means that the information is exchanged through face(s) between the CCN core (CCN forwarding daemon as well as the data structures used by the CCN) and the software or application(s) running on the node and/or the communication interface(s) installed on the CCN-enabled node. In the latter case, the CCN node configures the communication interface(s) in such a way so that the node can receive and send messages to and from the other nodes in the network. Communication in CCN is mostly either broadcast or multicast in nature; however, there are the cases where point-to-point communication is performed using these faces. The application faces are used by the CCN to communicate with the application instance(s) or process(es) running on the node.

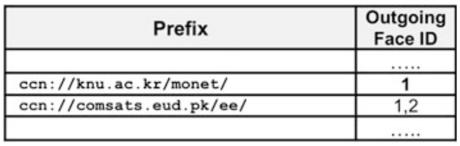
- **Pending interest table (PIT)**: This is a data structure that stores information about unsatisfied interests including content name, incoming face from where the interest message was received, and timer(s). The term "unsatisfied" means that the node has forwarded the interest message received from the downstream and forwarded it to the upstream; however, the interest forwarder is awaiting the response. The timers are used to limit the duration of PIT entry and, ultimately, the size of the PIT. The entries in PIT are organized in such a way that insert, search, delete, and modify functions can be performed easily. An entry is created in the PIT when a node receives an interest message that it cannot satisfy with the requested content. This means that the content is not available in the CS. The pending interest recorded is purged from the PIT in two cases: (1) the node received the content or data in response to that interest message or (2) the message time has expired. Before the creation and deletion of the PIT entry, a lookup operation is performed to make sure that the entry is still present or not. This lookup mechanism uses the longest prefix matching algorithm on *names* to find the entry with the best-matching name in the PIT. Therefore, entries in the PIT should be managed in such a way that it should expedite the lookup process because there may be a huge number of requests in the network. In technical documents, it is also suggested that the PIT should also keep track of the interest's scope (the maximum hop limit, i.e., a two-byte field in the interest message). If an interest with a shorter hop limit is stored in the PIT and a similar interest with a larger hop limit arrives, then the forwarder node must forward the shorter-hop interest and update the PIT entry accordingly. A large random number, called "NONCE," is used in the interest message to uniquely identify it. The NONCE value is stored in the PIT to detect the interest loop as well. If a node receives an interest with similar-value NONCE, it is dropped, and no further action is required. A detailed illustration of a PIT is shown in Fig. 3.3.
- **Forwarding information base (FIB)**: The FIB is analogous to the IP table maintained on Internet routers. When an interest is unsatisfied by the node (i.e., there is no matching content in the CS and no entry in the PIT), then it is forwarded upstream toward potential providers using FIB. Every entry in the FIB is the tuple of name prefix and outgoing face(s). A single prefix entry may have more than one outgoing face associated with it. The interested content name is searched within the FIB using longest-prefix matching. The interest is forwarded to the face associated with resultant entry of the LPM search. A simplified FIB is shown in Fig. 3.4.
- **Content store (CS)**: This is a buffer memory or cache used to store full or partial contents or data packets; however, it is not a persistent storage. The



Name Prefix	Incoming Face ID
ccn://knu.ac.kr/monet/members/bouk	2

Parameters				
Nonce	Last Refreshed	Life-Time	Scope	....
0xA1B3...	1	2	5	....

Fig. 3.3 Details of pending interest table



Prefix	Outgoing Face ID
	.....
ccn://knu.ac.kr/monet/	1
ccn://comsats.eud.pk/ee/	1,2
	.....

Fig. 3.4 FIB structure

contents are stored in the CS per cache policies to satisfy future interest messages. Two types of policies for the CS have been discussed: replacement policies and *caching-decision* policies. These policies are important to maximize caching and content-satisfaction efficiency. Cache-replacement policies effectively use the CS capacity. It indicates that when a new content is received, the CS should replace it with a certain content: the content that is least recently used (LRU) or least frequently used (LFU), random content, content with the maximum time in the CS (i.e., FIFO [first in first out]), etc. In contrast, caching-decision policies decide whether or not the received content should be cached. Caching-decision policies may include leave copy everywhere (LCE), leave copy down (LCD), fixed probability p caching, most popular caching (MPC), leave copy on the edge, etc. [9–14]. The content store also uses a 1-bit "stale" flag with each content object. A high stale flag indicates that content will not be sent in response to the interest message and that the interest message should be forwarded to upstream for other potential providers. This bit is set high when the content's freshness time has expired, and the content will be given priority to be eliminated first when the cache reaches to its capacity.

The CS is organized in such a way as to efficiently retrieve contents based on the prefix lookup using content names and the selectors. That is the reason the content table is built on a **name_tree** structure that is keyed by the flat-name representation of the content name.

The same structure can also be used as a basic structure that requires or depends on prefix-matching operations [15], i.e., FIB entries, PIT entries, statistics used by the strategy layer, name enumeration, and creation/deletion notifications. Most of the CCN data structure requires a longest-prefix lookup, and the name_tree structure is inherent in several data structures. Readers interested in a

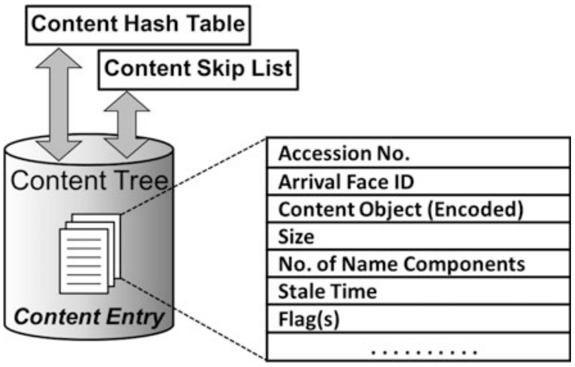


Fig. 3.5 CS structure

detailed description of name_tree implementation are advised to refer the name_tree structure in the CCNx. The structure of the content store is shown in Fig. 3.5.

A “hash table” is maintained to store content information, and a separate “skip list” is used to orderly maintain the content objects and is used over the stored content objects. The content hash table stores all the content entries. Its “key” type is a portion of the data or the content message other than the actual content object.

“Content entry” represents a single entry in the content store and contains the following:

- The accession number (accession) is a unique identifier assigned to each unique content stored in the content store. Accession numbers are assigned in the order of arrival, and the highest number is assigned to the content that has been received most recently. The most recent number is saved in the CCNx data structure to keep track.
- Face no. or ID from which the content object has arrived initially.
- Size of the content object.
- Index and number of the name components (comps, ncomps)
- Time in seconds before the content object becomes stale.
- Flags, e.g., whether or not the content object is stale.

Name Skip List

The skip list uses the content name to index content entries in an increasing order. The maximum depth of this skip list is up to 30 [readers are advised to refer to the information regarding skip list from the plethora of online and offline resources because skip lists are out of the scope of this book]. The traversal of the skip links using pointers to the next entries will walk through the names in increasing order.

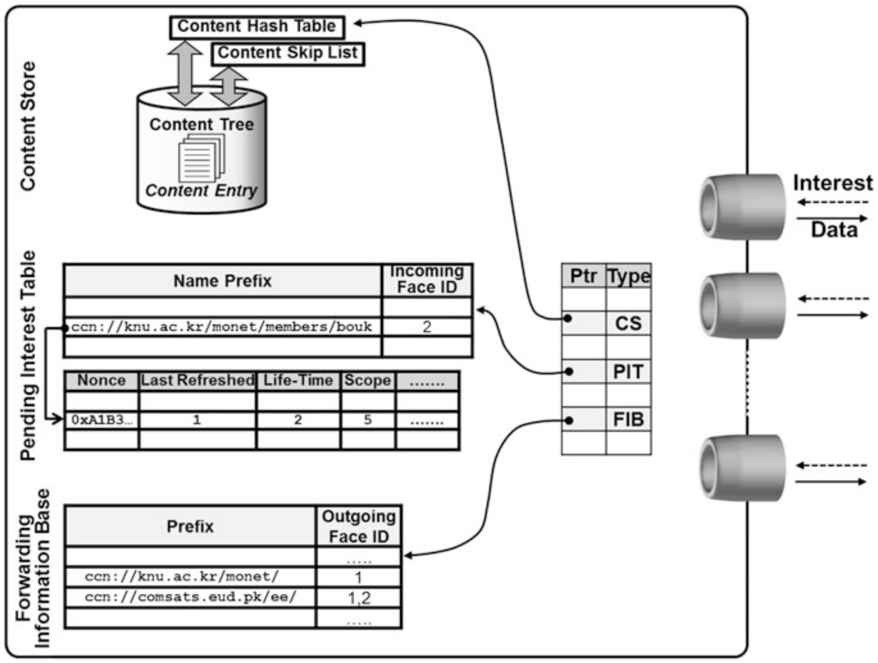


Fig. 3.6 A complete CCN architecture

Accession Skip List
The accession skips list indexes content entries using the accession number information. The content entries may use an accession number, such as (Accession Base + Accession window), to access the content entry.

There may be a case when a content or data message is received by a node that has no previous PIT information or the node did not receive the interest message requesting that content message. This type of content object is called “unsolicited data,” which may also be stored in a separate list. Accession numbers of the unsolicited content objects are separately recorded in the data structure.

In CCNx, all three structures, i.e., FIB, PIT, and CS, are connected through a single index, which may result in alleviation of the lookup cost of each message received, processed, and forwarded by the CCNx. This front index is optimized and ordered in such a way as to efficiently perform the operations specific to the received message type as shown in Fig. 3.6.

3.2 Basics of CCN

In CCN, a consumer node sends an interest message if it requires a content. The consumer node uses the content name and its selectors (the content attributes, resolutions, filters, etc.) in the interest message to request the content object. When any node receives the message, it first searches the desired content in the CS using longest prefix matching based on the content name. If there more than one content matches the content name, the selectors are used to precisely identify the content object.

3.2.1 Content

The terms "data" and "content" are used analogously in the context of this book. Any content that is desirable for the application, we call a "content object" or "data." Therefore, we will not go into more detail about the differences between data, content, and information. There are so many resources available online that differentiate between these terms [16] as follows:

- Data are the facts and figures known to be or assumed as facts that do no convey any meaning on their own.
- Content is a way to contextualize data. Date can become a content when they are presented in a usable form envisioned for one or more resolutions.
- Information is processed data that conveys the proper meaning or, simply put, information is data with meaning.

As mentioned previously, the terms "data" or "content," in the context of this writing, refers to the files or streams published by a producer or publisher. Therefore, the data or content entity can be a text document, video clip, audio file, or stream of text, audio, or video information. The size of the file can range from a few bytes to multiple gigabytes. These files are also known as "binary large objects" (BLOBs). In [17], it is stated that a large file or document is divided into multiple content objects and that the size of a content object can be up to 64 kB. Similarly, the maximum transmission unit (MTU) size of the Ethernet is 1500 bytes, which is the maximum allowable size of information that can be sent in a single transmission unit or frame. Therefore, it is evident that the data file or document can be larger than the content object as well as the MTU. As a result, the data or content or content object is fragmented in such a way to fit the permissible size of the MTU. After successful reception of these MTUs, the receiver node reassembles the content objects to form the whole. Content fragmentation and reassembly is discussed in detail later in this chapter.

Because fragmentation is unavoidable when we communicate large contents, a different approach, called "manifest," is used to efficiently manage the fragmentation and reassembly of contents. Simply stated, the manifest is a structure to index

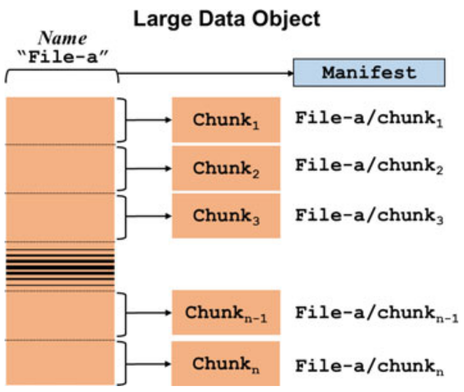


Fig. 3.7 Large data object with manifest

large contents and content objects. They represent the information as a metadata about the whole content object. The manifest by itself is considered as a content object and is sent through the data message with the payload of type "manifest," not as a "data." The following figure shows that a large content with the name "file-a" is divided into "n" chunks [18], and each chunk is named as "file-a/chunk$_i$." The manifest is maintained in this file, and it records the information regarding the type of content, access rights, publisher information, number of chunks, the size of the chunks, names, chunk-level hash-based names of the data objects, etc. [19]; refer to Fig. 3.7.

A manifest may also be used to represent the information about the stream data (either video, audio, text, etc.).

The manifests can also be considered the collection of linked content objects with corresponding hash digests. Large data objects are chunked, and each chunk shares the same data object name along with additional chunk information. Therefore, a manifest structure can be represented in the hierarchical structure, and the root of the manifest represents the content name. The root of the hierarchical manifest requires the cryptographic signature using the single public key for a whole data object and the subsequent chunks. In result, the large contents can be restricted to the size of the MTU, and thus fragmentation can be avoided. Due to this reason, a manifest can easily represent the represent the stream data (video, audio, text, etc.) as shown in Fig. 3.8.

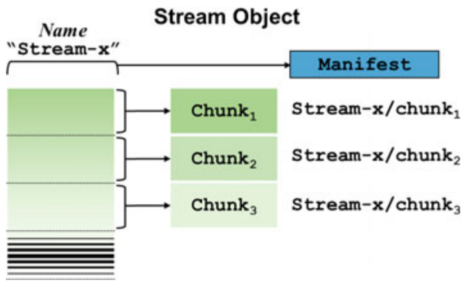


Fig. 3.8 Stream object with manifest

As discussed previously, the content object is divided into chunks, and CCN names every chunk. To secure a content object, either the whole content object is signed, or every chunk is signed individually. It is easy to sign a whole content object by signing the manifest of the content. Readers are suggested to read [19] for detailed information.

The content chunks are numbered and in a data packet, and each chunk is identified by its number. Along with that, the last chunk number field is also sent in the data packet to notify the maximum number of chunks into which the content is divided. These concepts are further explained in the following text.

3.2.2 Naming

The future communication architecture CCN uses content names instead of the host ID (IP address) to identify the content, and the name is also used to forward contents on the network. This means that every CCN-enabled node operates on the content names [20]. Here we focus on the content naming used in the CCN and its variant implementation in NDN. In CCN, the names are also termed "HSVLIs" (hierarchically structured variable length identifiers).

CCN and NDN use a human-readable naming scheme that uses the URL (uniform resource locator) like a hierarchical structure to name to the content. For example, the image content "safdar.jpg" of a Mobile and Network Laboratory (MoNeT) member in Kyungpook National University (KNU), Korea, can be represented as follows:

URL: **monet.knu.ac.kr/images/people/safdar.jpg**
Name: **/knu/monet/images/people/safdar/jpg/date/20151131/res/500/800**

Multiple segments in the URL provided above are used by the current communication protocols. The same content represented by the URL is also presented with the corresponding name used by the CCN. Each segment has been divided into multiple components, and the adjacent name segment components are separated by the delimiter "/". The content name is divided into different parts that are set and provided by the content provider. Unlike IP addresses, the length of the name components, as well as the number of components in the name, is arbitrary. The CCN protocol only uses the content name and its hierarchical structure to forward messages in the network.

The content name combines a routable prefix with an arbitrary suffix assigned by the publisher to a content or a piece of content. The routable prefixes and the suffixes do not have a fixed length. The routable prefix, e.g., "**/knu/Monet/**," is assigned the outgoing face and will be used to forward the interest message containing the content names with similar prefixes. Figure 3.9 shows the example content name divided into different segments that include the following:

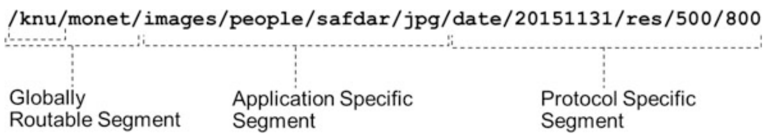

/knu/monet/images/people/safdar/jpg/date/20151131/res/500/800
Globally Routable Segment
Application Specific Segment
Protocol Specific Segment

Fig. 3.9 Content name used in CCN

(a) The globally routable segment, which is the network-wide coordinated name space, i.e., domain name. This name segment is used to route messages in the CCN.

(b) The content name assigned by the application (the applications can assign their contents a name in their own way). This application content-naming convention can vary and is application dependent so that each application can use any scheme.

(c) The last part of the content-name segment provides the components or attributes related to the contents used by the protocol. These name components include content-version information, chunk number, date and time stamp, and other content features.

The CCN protocol does not infer the meaning of the name but only it matches the name within the routing and related data structures. Therefore, the name components are set, assigned, and used by the application, and only the application can infer the purpose of each name component as in the previous example. The name components represent the application, organization, and global resolution, and they are reflected in forwarding rules. In CCN, the required content names can be provided by users (simply by copy-and-paste method or typing the name), a document or Web page in the form of links, generated by search engines, provided by the near field communication (NFC), or scanned from QR (quick response) code, or it can be provided in other formats.

Each content or every chunk of the content, collection of contents, or collection of chunks of contents is represented by the name. The name part that commonly represents the collection of the contents or chunks of contents is called the "name prefix." Therefore, the content names are also called "name prefixes" or simply "prefixes." The name prefixes are used to forward messages as well as find the contents by matching these name prefixes.

The advantages and disadvantages of hierarchical naming are briefly discussed in [21]. Hierarchical content naming has many advantages including, but not limited to, the following:

- The similarity of the content name with the current URL structure makes is easy to use it with most of the URL-based applications.
- The hierarchical structure of the content name prefix can easily aggregate different content names with similar prefixes under a single name prefix. For example, consider the following content names:
  /knu/monet/images/people/**safdar/jpg/**date/20151131/dim/500/700

/knu/monet/images/people/**hassan/jpg**/dim/430/620/color/RGB
/knu/monet/images/people/**azfar/jpg**/dim/300/403/res/180dpi
can all be aggregated under the name prefix:
/knu/monet/images/people/.
Instead of saving all the content names in the forwarding table, only the aggregated name prefixes are saved, which minimizes the size of the name-prefix table.

- Due to the small size of the forwarding table, the lookup operation can be accelerated.
- The lookup operation used for searching a name prefix from the name tables is similar to the one used in current routing devices. One of the most popular methods used for the lookup is the longest prefix matching (LPM) method.

Along with many advantages, the hierarchical naming scheme comes with a few shortcomings [21]. The most prominent problem with hierarchical naming is that it has variable size because the name is composed of a series of components with variable number and length. This will result in large forwarding tables and slow lookup process [22]. CCN-enabled applications may be interested in the content without knowing the full name of the content. Searching a name prefix from the table solely based on the content name is quite difficult and currently not supported by the CCN. Along with that, the forwarding tables store only the prefixes and have no information about the aggregated, which may cause the "suffix hole" problem in CCN forwarding. For example, the prefix "/knu/monet/images/" has only a few image contents that are available at the node or outgoing interface associated with that prefix. However, based on the prefix information, all requests are forwarded to that face even if the content is not available at that face. The CCN must use additional information, i.e., content attributes, to search the specific prefix and the appropriate content. For more information regarding security and other related information about CCN-naming schemes, readers are advised to refer [20, 21].

3.2.3 Interest

It was clearly mentioned previously that CCN uses two types of messages: interest messages and data messages (referred as a ContentObject message in CCNx documentation.). Here we will briefly discuss the *interest* message and its format as well as how it is forwarded, i.e., what steps are taken by each node to forward the message in CCN.

An interest message is generated by a node that requires any content, i.e., a content-requesting node, called the "consumer node." The interest message uses the content name to request and specifically identify the content or its chunks. Along with the name prefixes, the interest message may also provide some specifiers or selectors related to the content object that help to ascertain the explicit data in the repository.

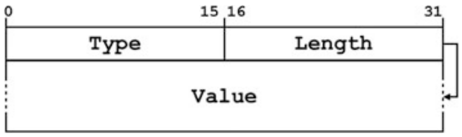


Fig. 3.10 Type–length–value format

CCN defines the detailed description of the interest message in the CCN message semantics [23] and CCN message type length value (TLV) [24] documents as shown in Fig. 3.10. Here we summarize the interest message in the TLV encoding and TLV fields in detail.

Basically, CCN uses **type** and **length** fields, both 16 bits in size, to encode the TLV packet format. The 16-bit size of a **type** field gives 2^{16} possible types, and the range of types from 0 × 1000 to 0 × 1FFF is reserved for experimental use, but it provides ample space for the protocol types. Similarly, the **length** field defines the size of the **value** field (in octets) according to the **type** and **length** fields, and it can be up to 2^{16} bytes (64 KiB). The length value does not include the 4 bytes of the **type** and **length** fields. It is stated in [24] that a 0 value of the **length** field is permissible. The TLV is hierarchical in nature, and the **value** field may also contain other TLV structures (called "member" or "subordinate" TLV structures); that TLV is called the "container TLV."

The complete CCN packet format is shown in Fig. 3.11. Every CCN packet consists of a fixed 8-byte static header, which is common to all CCN packets and does not follow the TLV format. This common header is followed by the TLV(s) that represent the optional hop-by-hop header and packet payload (which is a CCN message) encoded in TLV format. There is an optional CCN message-validation TLV at the end of the packet.

The **packet payload TLV** is the CCN message that is an interest, data, or any other type of message used by CCN. The **version** field represents the packet version, and **version** = 1 at the time of writing this book. The **PacketType** field identifies that the packet is either 0 = interest, 1 = data, or other packet type (i.e., 2 = interest return, etc.). The **PayloadLength/PacketLength** is the complete packet length (in octets) starting from the **Version** field to the end of the packet. The **HeaderLength** represents the total length (in octets) of all the headers (including the fixed header) before the Packet Payload TLV or CCN message TLV. The **PacketType-dependent field** is divided into multiple other fields based on the packet type.

The fixed header is followed by the optional header or optional hop-by-hop header TLVs. The hop-by-hop headers are used in a CCN packet that requires per-hop (interest or data message) information propagation and/or operations.

Interest LifeTime TLV: As evident from the name, this header may be included in the interest message. The header represents the duration that an unsatisfied interest should be kept in the PIT. If this value is not used, i.e., not propagated with the interest message, then the default period that an interest must be kept in the PIT is

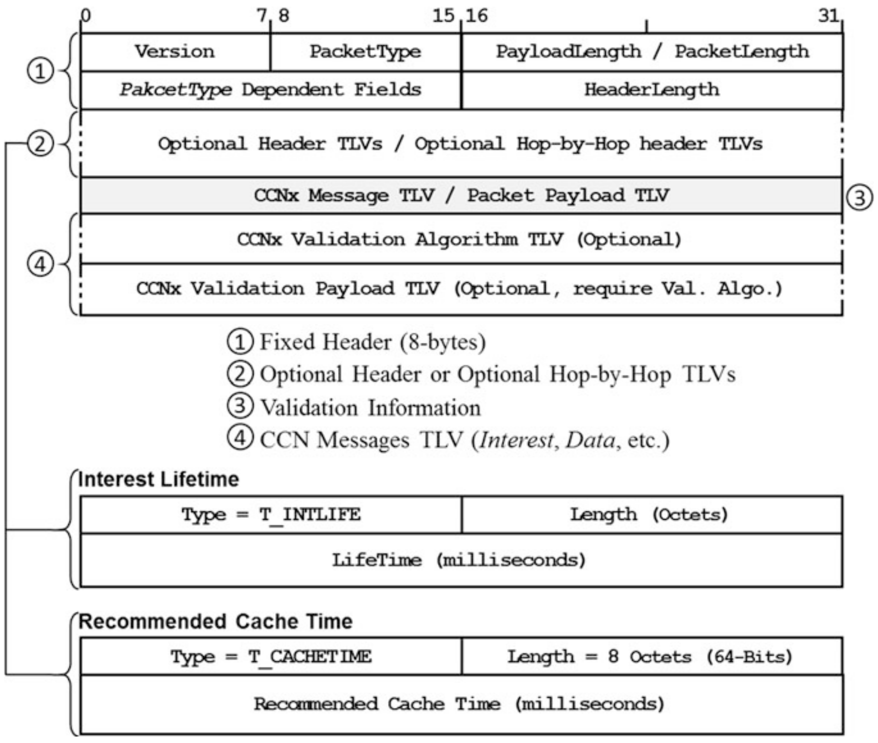

0
7 8
15 16
31
Version
PacketType
PayloadLength / PacketLength
PakcetType Dependent Fields
HeaderLength
Optional Header TLVs / Optional Hop-by-Hop header TLVs
CCNx Message TLV / Packet Payload TLV
CCNx Validation Algorithm TLV (Optional)
CCNx Validation Payload TLV (Optional, require Val. Algo.)
① Fixed Header (8-bytes)
② Optional Header or Optional Hop-by-Hop TLVs
③ Validation Information
④ CCN Messages TLV (Interest, Data, etc.)
Interest Lifetime
Type = T_INTLIFE
Length (Octets)
LifeTime (milliseconds)
Recommended Cache Time
Type = T_CACHETIME
Length = 8 Octets (64-Bits)
Recommended Cache Time (milliseconds)

Fig. 3.11 CCN complete packet format

4 s. The **LifeTime** value is measured in milliseconds. It is also stated that the interest **LifeTime** header with **LifeTime** = 0 ms (**%×00**) should be forwarded; however, no data message reply is expected for that kind of interest message.

Recommended Cache Time TLV: This header is included in the data message to represent the suitable **LifeTime** of the content or data in the cache or CS. This value is recommended by the content producer or by any of the upstream data-forwarder nodes. The recommended cache time value is represented by the millisecond timestamp. Because it is an optional TLV, the nodes may ignore the suggested period.

(a) Interest Message Format

The complete CCNx interest message is shown in Fig. 3.12. The CCNx interest message TLV area, as well as the fixed header fields related to the interest message, are shown in the shaded area. **HopLimit** in the fixed header sets the maximum number of hops that the interest message can be forwarded in the CCN. The HopLimit value is set by the consumer node and is decremented at each hop. The interest message is forwarded until the **HopLimit** does not reach zero after the decrement. The maximum value of the HopLimit is 2^8 = 255. Some of the

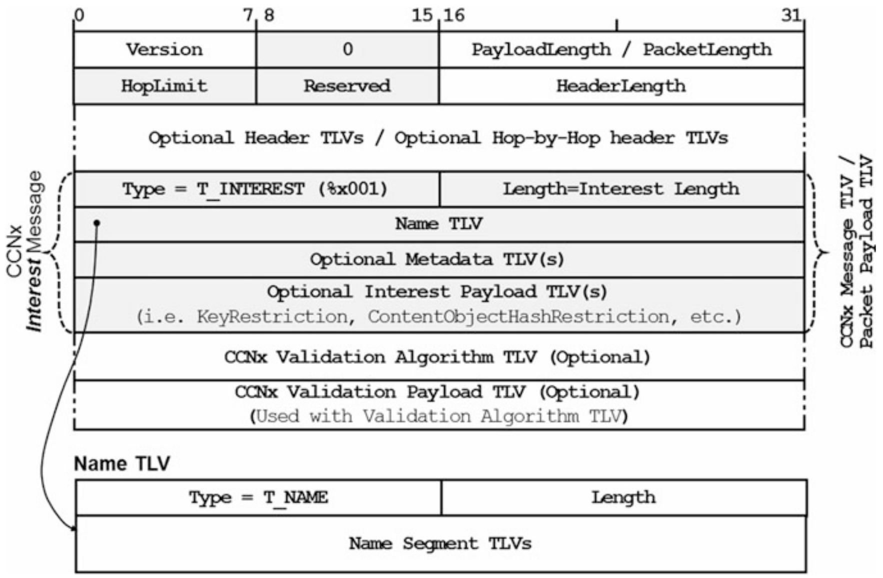

0
7
8
15
16
31
Version
0
PayloadLength / PacketLength
HopLimit
Reserved
HeaderLength
Optional Header TLVs / Optional Hop-by-Hop header TLVs
CCNx *Interest* Message
Type = T_INTEREST (%x001)
Length=Interest Length
Name TLV
Optional Metadata TLV(s)
Optional Interest Payload TLV(s)
(i.e. KeyRestriction, ContentObjectHashRestriction, etc.)
CCNx Message TLV / Packet Payload TLV
CCNx Validation Algorithm TLV (Optional)
CCNx Validation Payload TLV (Optional)
(Used with Validation Algorithm TLV)
Name TLV
Type = T_NAME
Length
Name Segment TLVs

Fig. 3.12 Interest message TLV

Reserved field bits are used for the interest flags; however, there is no description of the flags used in the Internet draft [24].

The interest message container TLV starts with **Type = T_INTEREST** or is encoded as **%×001** (interest message type). The interest message TLV must contain the **Name TLV,** and a similar TLV is mandatory for the data message as well (the data message is discussed later in the chapter). There are other optional TLVs that can be included in the interest message.

(a) **Name TLV**: The name TLV container encodes the content name segment, and each segment of the naming convention is represented with a separate TLV and is assigned different **Type** values Refer [25] for more information on the Labeled Content Information document plus Fig. 3.13.

A generic name segment, which includes the arbitrary octets (hierarchical naming convention) is represented by the name segment TLV. The interest payload ID is used in the name segment to properly multiplex the interests based on the name with different payloads. The payload ID is created by the consumer node to identify the payload in the interest message. The interest payload ID is commonly represented by the hash of payload, which requires the hash algorithm information as well in the interest message. Another type of interest payload ID, called **NONCE**, is the name segment that uniquely identifies the interest message.

The application-component TLV shows the application-oriented payload in the name segment. It is suggested that the application name should be identified in the name segment and that the semantics of these components are application

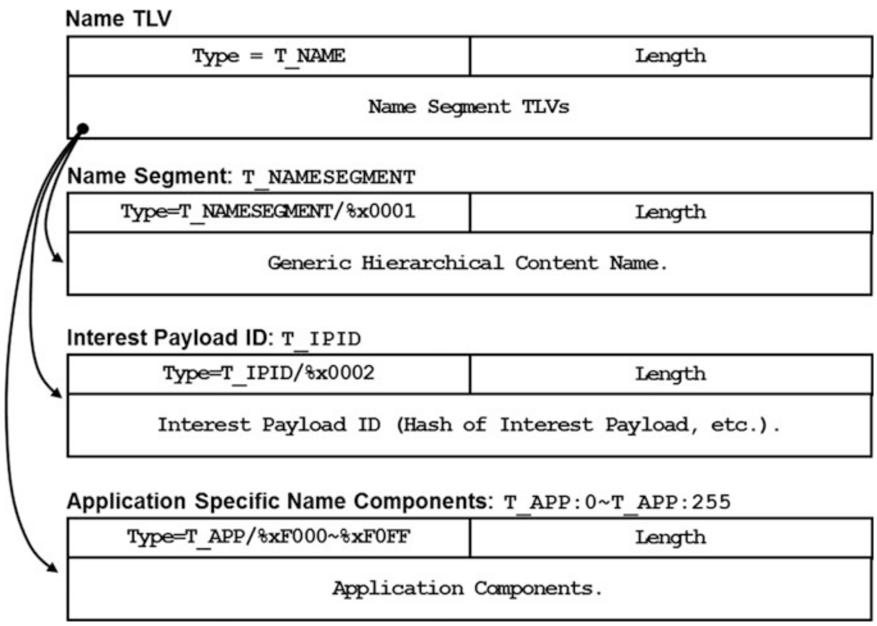

Name TLV
Type = T_NAME
Length
Name Segment TLVs
Name Segment: T_NAMESEGMENT
Type=T_NAMESEGMENT/%x0001
Length
Generic Hierarchical Content Name.
Interest Payload ID: T_IPID
Type=T_IPID/%x0002
Length
Interest Payload ID (Hash of Interest Payload, etc.).
Application Specific Name Components: T_APP:0~T_APP:255
Type=T_APP/%xF000~%xF0FF
Length
Application Components.

Fig. 3.13 Various formats of name TLVs

dependent. The name TLV is common in the interest and data messages; therefore, the same name TLV is used in the data message as well.

(b) **Interest-Specific TLVs**

These TLVs are optional and specific to the message type, i.e., interest message or data message. The metadata TLV(s) and interest payload TLVs are optional and are used with both interest messages and data messages. Here we will discuss the interest-specific optional TLV(s); the data message related TLV(s) are discussed later in the chapter.

At the time of writing this book, only two optional metadata TLVs related to the interest message are used to assist in identifying the closest matching content required by the consumer node. The interest-specific optional TLVs are the **KeyIdRestriction** and **ContentObjectHashRestriction** (refer to Fig. 3.14).

The key identifier is the unique string that offers methods to identify certificates that contains the particular key. The identifier can be a 160-bit SHA-256 algorithm hash of the key that is used to satisfy the interest or any unique number-generation mechanism. The **KeyIdRestriction** TLV sends a byte string that identifies the publisher signing key that will satisfy the interest carrying this TLV. The **ContentObjectHashRestriction** is the hash of the content object computed using an SHA-256 algorithm. This content object is the one that satisfies the interest message.

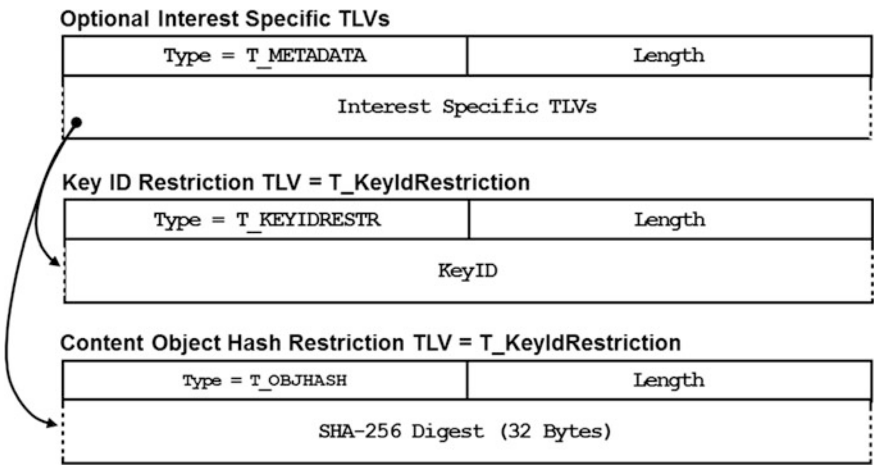

Optional Interest Specific TLVs
Type = T_METADATA
Length
Interest Specific TLVs
Key ID Restriction TLV = T_KeyIdRestriction
Type = T_KEYIDRESTR
Length
KeyID
Content Object Hash Restriction TLV = T_KeyIdRestriction
Type = T_OBJHASH
Length
SHA-256 Digest (32 Bytes)

Fig. 3.14 Interest-specific TLVs

The validation algorithm TLV container comprises different TLVs that contain information about the specific algorithm used to validate the interest message and can be a cyclic redundancy check (CRC), message signature, SHA algorithm, or any cryptographic algorithm. The validation information, i.e., digest, CRC, signature, etc., of the interest is also sent in the nested TLVs of the interest message. It is important to note that the message validation is optional as depicted in Fig. 3.12. The next section briefly discusses the steps that are followed to forward an interest message.

3.2.4 Interest Forwarding (Interest Message Forwarding)

CCN uses pull-based or receiver-driven communication. The consumer node requires content to generate an interest message and transmits it in the network. The interest message is forwarded in the network until the required content is found or until the **LifeTime** of the interest message expires. The interest message is forwarded in the upstream direction, that is, from the consumer node toward the content-provider node.

Any node that has the requested or interested content that matches the specifications mentioned in the interest message replies with the content or ContentObject in the data message (it sends either whole content, if the content fits the permissible size of the data message, or a chunk of the content of the data message size in case of large content.). An interest message is satisfied with the content if the name in the interest message precisely matches the content name. Further, if the security is enabled, then the **KeyID** in the optional **Validation Algorithm** TLV of the data message should match with the **KeyID** in the **KeyIdRestriction** TLV of interest message. Moreover, the optional **ContentObjectHash** must be similar to the Content Object hash restriction within interest message.

Before discussing the interest message processing and forwarding steps, it is necessary to discuss the concept of local and remote next hops. In [17], the authors define that an application running on the node is the local next hop, and the remote next hop is the node that is not local to the current system. The concept can be easily understood by considering the **HopLimit** parameter in the CCN messages. If an interest with **HopLimit** = 0 is sent by the application running on the consumer node, then it can only be sent to the other applications running locally on the same node. However, if an interest with **HopLimit** = *m* is sent by the local application, then the message will be transferred and processed by the nodes that are *m*-hops away from the sender node. Similarly, when a node receives an interest with **HopLimit = 1**, it decrements it first (**HopLimit = 0**) and then forwards it to the local applications.

The stepwise processing and forwarding of the received interest message is discussed below:

1. Interest received from local applications (**HopLimit** = 0): Forwarded to the other applications running on the same node.
2. Interest received from the remote or downstream node with **HopLimit** ≥ 1:

   (a) Decrement the **HopLimit**.
   (b) If the resulting **HopLimit** = 0, then forward to local applications.
   (c) If the resulting **HopLimit** ≥ 1:

      (i) Search the requested content name (and other optional search constrains if provided in the interest message) in the content store. If any match is found in the CS, then it generates a data message that carries the content or chunk of the content. The interest message is then discarded.
      (ii) If no matching content is available in the CS, then search the PIT. In the case that the node has already received the interest message requesting the same content, but with a different receiving face or with a different NONCE or interest payload ID, then the additional face or NONCE information is added in the previous PIT entry. However, discarding or re-forwarding the received interest message depends on the **LifeTime** value in the interest message and PIT record. The CCN suggests that the interest is forwarded only if the **LifeTime** of the previous entry in the PIT is less than the **LifeTime** in the received interest message. In this case, the interest message is forwarded when the current PIT entry **LifeTime** expires and the remaining **LifeTime** of the interest is set in the PIT entry. In contrast, the interest message is discarded and no further processing is required when the **LifeTime** of the received interest is less than or similar to the PIT record.
      (iii) In the case of no record found in the PIT for the received interest, a new PIT entry is created with a **LifeTime** similar to the one that is set in the interest message. Move to the next step (step iv).

(iv) Search name prefix in the FIB:

- If the FIB search returns more than one next hops, then the interest is forwarded based on the forwarding strategy [17], i.e., send the interest to all, best path, or alternatively to the paths based on their ranks, etc.
- If no FIB entry is found, then CCN holds the entry for a short time before discarding the interest message because the new FIB entry creation entry may provide a way to satisfy the interest. This is the crude functionality of the CCN, which requires more research.

(v) Send interest toward the upstream.

The reliability of the interest message is not presumed by the CCN, however, the reliability can be supported by retransmission of the interest message (either by the consumer node or the intermediate forwarder node) once the **LifeTime** of the message expires [26]. The total number of retransmissions and **LifeTimes** of the interest message should be recorded with each pending entry in PIT to support constrained retransmission of the interest message. In CCN, the default value of **LifeTime** is 4 s, and there is no specific value for the number of interest message retries. Detailed illustration of interest-message processing is depicted in Fig. 3.15.

3.2.5 Data Retrieval

The interest message pulls the desired content from the network using the content name (and the optional additional qualifiers to specifically identify the content or chunk of the content.) The content, either whole or chunk, is transferred by CCN protocol in data message(s). Here we will discuss the data-message format as well as the steps to successfully forward the data message toward the consumer node in the *downstream* direction.

The data message carries content as the payload, the name that identifies it, and additional information to verify and validate the content and the data message. This additional information includes the signature (cryptographic), the identification information of the publisher or signer, etc. Simply, the data message binds the *content* (may be chunk), its name, and the publisher, all together (refer to Fig. 3.16). It is required that all communicated data in CCN is verified through the signature and that every data message must contain the valid signature. An unverified data message should be discarded by the receiving node. Data verification requires signature verification from the signing authority using the public key; therefore, the public key should either be in the interest or included in the data message. However, in CCN there is no specific protocol for key distribution, and the keys are considered and distributed as the contents are.

The data message TLV consists of similar TLVs (e.g., **Version**, **PayloadLength**, **Name** TLV, etc.) similar to the interest message (refer to Sect. 3.2.3). The only difference is the **Packet Type** field, which identifies that the packet is *data = 1* packet, and there are separate data message specific TLVs. At the

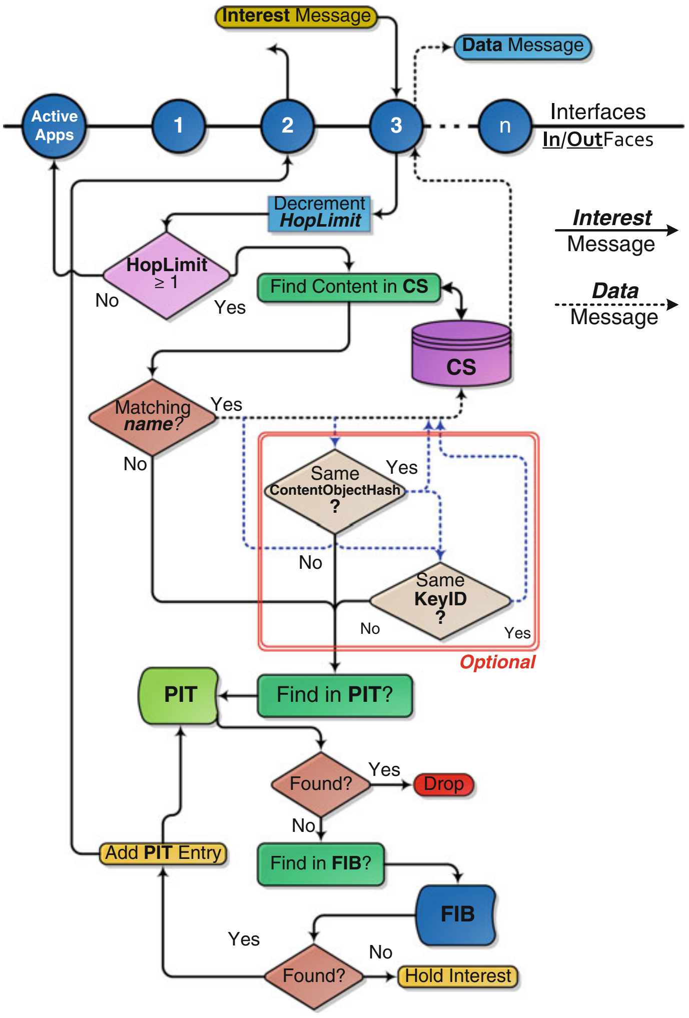


Fig. 3.15 Interest message flow diagram

time of writing this book, there are only two optional data message specific TLVs: called **PayloadType** (**%×0005**) and **ExpiryTime** (**%×0006**).

The **PayloadType** TLV defines the general type of the contents in optional **Data Payload** TLV, which follows this TLV. The broad types of the content types include the following:

- **%×00**: Data/***Content Object***: This is the default payload type when the **PayloadType** TLV is not included in the packet. Its contents represent the content object [complete or chunk of content] bytes used by the application.
- **%×01**: ***Key*** contents: Payload contains the encoded public key.

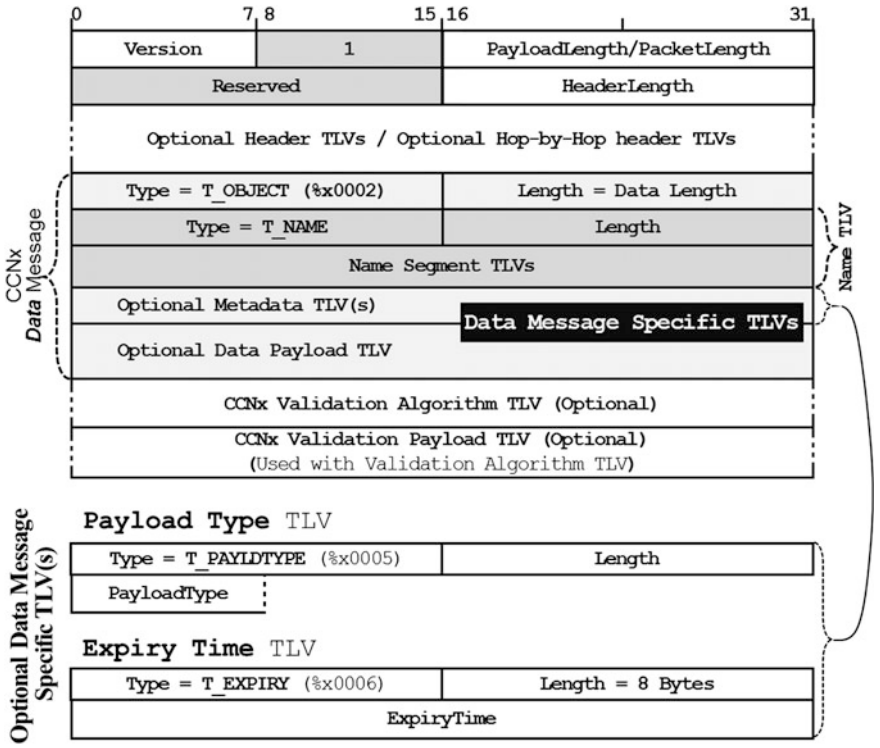

0
7
8
15
16
31
Version
1
PayloadLength/PacketLength
Reserved
HeaderLength
Optional Header TLVs / Optional Hop-by-Hop header TLVs
Type = T_OBJECT (%x0002)
Length = Data Length
Type = T_NAME
Length
Name Segment TLVs
Optional Metadata TLV(s)
Data Message Specific TLVs
Optional Data Payload TLV
CCNx Validation Algorithm TLV (Optional)
CCNx Validation Payload TLV (Optional)
(Used with Validation Algorithm TLV)
CCNx Data Message
Name TLV
Payload Type TLV
Type = T_PAYLDTYPE (%x0005)
Length
PayloadType
Expiry Time TLV
Type = T_EXPIRY (%x0006)
Length = 8 Bytes
ExpiryTime
Optional Data Message Specific TLV(s)

Fig. 3.16 Complete data message TLV

- ***%×02***: ***Link*** information: This indicates that payload contains the link that is the tuple: {CCNx Name, KeyId, ContentObjectHash}.
- ***%×03***: ***Manifest*** contents: The payload contains the *manifest* of the content object.

ExpiryTime TLV indicates an unsigned 64-bit integer that shows the time instance (encoded in the UNIX timestamp containing the number of milliseconds) when the payload will expire. The content provider should not satisfy the interest with a content object that has passed its expiry time instance. Content in the data message with no **ExpiryTime** TLV can be kept by the consumer or caching node as long as needed. The data message forwarder should not check the **ExpiryTime** field. The **Data payload** TLV contains the contents of the data packets.

Data Forwarding

When a data message is received, the following steps occur:

1. The node first searches all the related entries in the PIT by matching the information from the data message, i.e., content name, content hash, etc.

(a) If PIT search returns one or more entries and there is no **ContentObjectHash** restriction requirement, then the data message is sent to the recorded incoming faces. In the case of a **ContentObjectHash** restriction requirement, the **ContentObjectHash** in the data message is matched with the content hash in the PIT. *The* data message is transferred to the incoming faces recorded in the PIT with matching entries.

(b) When a **ContentObjectHash** restriction is requested in the interest message, the content hash in the data message should be checked in the PIT. In case of multiple PIT entries returned by the search with matching **ContentObjectHash**, the data message is forwarded to those faces that are associated with the PIT entries. Otherwise, the data message is discarded.

2. If PIT search does not return any entries, then the data message is dropped.
3. After transferring data message downstream toward the next hop nodes connected with the faces associated with PIT, the PIT entries are deleted. PIT entries with expired **LifeTime** are also purged.
4. Content received in the data message may be stored in the CS based on the caching policy [27], i.e., all caching, no caching, popularity-based caching, etc.

Detailed processing steps of a data message are depicted in Fig. 3.17.

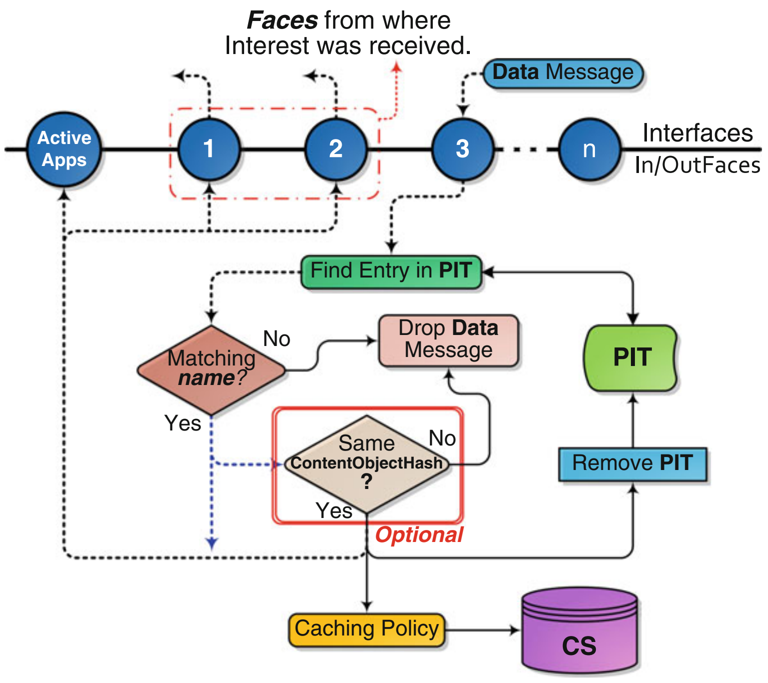


Fig. 3.17 Data-message flow diagram

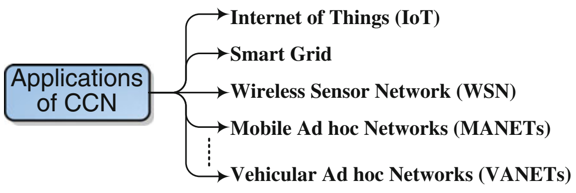

Applications of CCN
Internet of Things (IoT)
Smart Grid
Wireless Sensor Network (WSN)
Mobile Ad hoc Networks (MANETs)
Vehicular Ad hoc Networks (VANETs)

Fig. 3.18 A few examples of CCN-application areas

3.3 Applications for CCN

CCN is one of the promising future Internet architectures, and this is the reason that CCN has been actively investigated in many applications. Following is the list of applications where CCN has been investigated including, but not limited to, Internet of Things (IoT), Smart Grid, wireless-sensor networks (WSN), vehicular ad hoc networks (VANETs), etc. Figure 3.18 show a few of the CCN applications that are covered in this book.

3.3.1 CCN in IoT

IoT is the collection of small battery-operated devices having sensing, computation, and communication capabilities attached to the objects (anything) that connect them to the Internet. The attachment of the devices makes the objects "smart" to exchange and consume information to/from other devices autonomously or with little human intervention. Instead of communication in a point-to-point scenario, this huge network with a plethora of devices mainly focuses on data and information. Hence, the smart applications for IoT require contextual information that is generated either proactively or reactively by these devices [28]. There are several issues with these types of networks including, but not limited to, the following [29]:

- Dynamically addressing and naming each device
- Energy efficiency algorithms and protocols
- Network management and self-organization
- Network scalability and interoperability standards
- Network and service discovery
- Cloud connectivity and computation
- Real-time communication with small latencies
- Dynamic network partitioning and merging
- Mass data mining, filtering, and processing
- Scalable security solutions
- Mostly push-based communication
- Security, privacy, and trust technologies, etc.

IoT applications have been envisioned and realized in many applications as follows [30]:

- Smart healthcare system: anytime, anywhere person monitoring; remote healthcare; vital sign sensing, monitoring, and communication; emergency rescue and response; smart implants; telemedicine, etc.
- Intelligent transportation: connectivity and communication between vehicles, ships, trains, road signs, intelligent roads, tolls, planes, traffic lights, etc.
- Building and infrastructure interconnectivity and monitoring.
- Intelligent home.
- Connected smart city.
- Smart grid and energy sources.
- Connected manufacturing and intelligent products.
- Cooperative sensing.
- Smart logistics and retail of goods, etc.

IoT has been actively researched due to its huge range of applicability. Therefore, the future communication architecture, CCN, is also investigated in the context of IoT, and the solutions are discussed here. In [31], the author proposed and implemented ICN architecture for IoT. IoT contents are addressed using the names and its benefits of implementing the concept in home-automation systems are highlighted. Its implementation uses push-based communication with the home-automation system.

In [6], the authors used a push-based CCN-communication mechanism for IoT traffic. Initially they described a subscription scenario where the subscription is performed by sending an interest message that uses the hierarchical name for the desired content without creating a PIT entry for each interest forwarded. The reason for not creating the PIT entry is that no data are immediately expected for this interest. However, the authors mention that PIT-entry creation will avoid the interest looping. The authors use a timestamp in each content name (the time stamp is stored in the FIB as a separate field, called "last_seen") to avoid the routing loops. Each time a content is forwarded, its timestamp is matched with this field in FIB, and only the most recent data are forward to the subscribed consumers. To avoid redundant forwarding of the regularly sampled data, an optimal forwarding strategy is also proposed. In the optimal forwarding strategy, the provider pushes data only for the requested time period that is demanded by the consumer (the consumer subscribes for the data sampled in the specific period.). The intermediate nodes keep track of this dedicated period using the counter. Data messages with similar samples and their overlapping counters or sampling periods are combined and forwarded as a single data message instead of separate messages. In the end, a smart-pushing strategy is proposed to limit the number of forwarded messages. Identical-interval subscriptions uses one counter, and the remaining subscription intervals consider separate interval counters.

Another proposal that analyzes the application of CCN in the context of IoT is presented in [32]. It suggests that the complete ICN mechanism cannot be implemented on IoT nodes because of their power, sensing, processing, and memory constraints. Therefore, some functionalities, i.e., security and caching options, are delegated to the third-party trusted nodes. Optimal caching can be achieved by only storing the latest sensed information obtained using the counter or sequence numbers. Because data from different network segments can be combined, the use of a sequence number in data naming is not a valid solution. The authors tackle this issue by using the lifetime parameter, called "FreshenessSeconds," in data packets, which indicates the period until which data can be held in the CS. The authors simulated the IoT environment to check energy and consumption, as well as bandwidth use for varying CS sizes, and compared it with IP-based communication.

The authors in [33] performed experiments with an NDN implementation on IoT deployment in multiple office buildings. The bare-bones open-source Linux version of the CCN, called CCN-Lite [34], is ported over the RIOT [35], which is an operating system for resource-constrained IoT devices. The implementation uses the hierarchical naming scheme, which is better suited for routing aggregation. In the experiment, a simplest interest-flooding mechanism, named "vanilla interest flooding" (VIF), is used where each node forwards the interest in the network when it receives the interest for the first time. There is only one producer and many consumers in the network. The number of radio transmissions of interest messages are reduced, and the dynamic FIB is managed by the reactive optimistic name-based routing (RONR). RONR assumes that the FIB is empty at the network initialization; then first interest is flooded in the network, and the name prefix of that interest is stored in the FIB. The subsequent interests for the chunks of the same content are auto-configured in the FIB and forwarded in a unicast manner based on the name prefix in the FIB. It is assumed that the whole content is available at a single provider, which may not be the general case. The FIB entries are timed out to minimize its size. The authors also analyzed the impact of caching versus no caching with single or multiple consumers.

Another CCN proposal for IoT was proposed in [36] that uses NDN for an IoT scenario with multiple data sources located at single hop, single interest, multiple data (SIMD). They proposed a collision-avoidance mechanism by introducing the old concept of the contention window. Multiple data sources simply select random number $\mathrm{CW_r}$ between $[0, \mathrm{CW_{max}} - 1]$. The data source defers its transmission until the collision-avoidance random period $\tau_{\mathrm{ca}} = \mathrm{V_{slot}} * \mathrm{CW_r}$. A new interest packet is introduced that requests data from multiple sources, multisource interest (msINT). The consumer advertises $\mathrm{V_{slot}}$ and $\mathrm{CW_{max}}$ in msINT. The $\mathrm{CW_{max}}$ is dynamically adjusted based on the number of the data sources. The simulations are performed to measure the number of interest, interest overhead, and collection time, and compares it with the single interest, single data (SISD) scenario.

Most of the IoT proposals just focused on the simple scenario where an interest is used to subscribe for data, and the data are sent for that subscription period. Generally, there is periodic sensing in the IoT application, but the data may also be generated in response to the event detected by the sensors: this is called an "unsolicited emergency notification." The application requires this notification to be pushed into the network in either a unicast or multicast manner. However, there should be CCN proposals that implement the unsolicited emergency-message communication support along with the solicited data communication in IoT. There are several other issues that must be addressed when adopting CCN in IoT, and the readers are suggested to explicitly refer the related work.

3.3.2 CCN in Smart Grid

It is quite hard to define the term "smart grid"; however, we define it as follows: *Smart grid integrates advanced communication, control and automation, computer-based technology and systems that manages, regulates, and brings the responsive and resilient utility electricity network.* The grid connects the power-generation sources and manages the electricity demand in a reliable, sustainable, and economic manner. Balanced demand-based supply of electricity is one of the main objectives of the smart grid because most of the electricity generation relies on fossil fuels, which increases the amount of harmful gases in the environment.

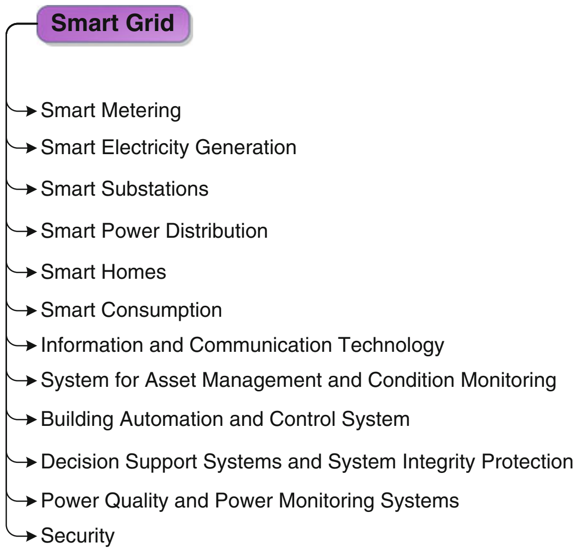
The smart grid is a system of systems consisting many components including, but not limited to, the following:

- **Smart meters**: Utility meters are enabled with communication technology to connect energy consumers and providers to automate billing, regulate demand, and detect faults for speedy recovery.
- **Smart electricity generation**: Several power-generation sources ranging from renewable to ones that consume fossil fuels. A smart electricity-generation system optimally generates electricity to meet users' demand with minimum cost and carbon emission.
- **Smart power distribution**: A power-distribution system connects the power-generation sources with the consumers through distribution lines and smart substations. It has self-optimizing, self-healing, and self-balancing capabilities to automatically predict and detect power failures in real time.
- **Smart substations**: These control and monitor critical and noncritical operational data, i.e., battery status, transformer status, breaker information, power-factor performance, security, etc.
- **Information and communication technologies**: These provide means for all the components to interact with each other to conserve energy by efficiently using, distributing, and generating electricity.

The rest of the components are shown in Fig. 3.19, but further details on the topic are out of the scope of this book.

Recently ICN architectures and their effectiveness has been investigated in smart-grid systems to investigate its feasibility and effectiveness. These proposals are summarized below.

The authors in [37] promote the use of ICN in smart-grid applications and suggest the use of publish/subscribe, such as communication, to ease the smart-grid control with simple and secure data sharing. The smart grid also uses a


Smart Grid
Smart Metering
Smart Electricity Generation
Smart Substations
Smart Power Distribution
Smart Homes
Smart Consumption
Information and Communication Technology
System for Asset Management and Condition Monitoring
Building Automation and Control System
Decision Support Systems and System Integrity Protection
Power Quality and Power Monitoring Systems
Security
Fig. 3.19 Components of a smart grid

many-to-many data-communication approach between devices and applications; therefore, ICN is envisioned to be a proper communication architecture for smart grids in the future. Currently, the ICN framework is used as an overlay on smart-grid communication to enable seamless and robust communication.

CCN-based advanced metering infrastructure (CCN-AMI) for the smart grid was proposed in [38]. The main objective of the proposal was aimed to efficiently control communication congestion, support mobility, and ensure security. The authors explicitly discussed the naming scheme for the home network, which is an integral part of AMI. The naming scheme consists of several components—including access scope (service, device, content, policy, etc.), transmission mode, and message type—along with version and segmentation information. In addition to the naming scheme, the authors briefly discussed the components encompassed by the CCN-AMI architecture: smart meter, customer energy-management system, smart-home appliances, data connector, meter data management system, demand response management system, load-management system, etc. The authors evaluated the performance of CCN-AMI with the current IP-based smart-metering infrastructure. The results show that CCN-AMI can significantly reduce network congestion traffic. The same authors proposed a key-management scheme for the CCN-AMI in [39].

In [40], the authors claim they implemented the ICN-based communication infrastructure, called "C-DAX," to support data communication in a smart grid. They proposed the C-DAX architecture and its components and planned to have a fully functional laboratory demonstration as well as port the implementation of the actual smart-grid system in the Netherlands.

3.3.3 CCN in WSN

Wireless-sensor networks (WSN), an integral part of IoT, are the collection of a large number of small battery-operated devices that have sensing and communication capabilities. The WSN consists of inexpensive and large number of devices spread or installed in the sensing area that monitor environmental or physical parameters, i.e., humidity, temperature, pressure, vital signs, water salinity, soil moisture, etc. A typical tiny sensor node consists of the following components connected as a single component:

- Communication
- Computing
- Sensing
- Power source
- Actuation

The sensed parameters are communicated wirelessly in an ad hoc mode toward the collection point(s), called "sink node," either periodically or based on an event for further decision making. The sink node(s) can be static or mobile to collect data from the WSN. The main objective of WSN is to monitor the field for a very long duration, and that is only possible by conserving the battery power [41]. The operations that consume battery power are wireless communication, sensing, processing and listening. Therefore, a node must efficiently schedule its operations. Along with the longer lifetime, the WSN must self-configure and self-organize due to dynamic network architecture caused by node failures [42]. Several routing solutions and proposals have been presented in the past to achieve energy efficiency, self-configuration, and self-organization. Researchers in the area of WSN may further refer [43–45].

Recently CCN communication architecture was investigated in WSN [46–51]. In [46], Bilel Saadallah et al. implemented the CCN named-data communication stack in the Contiki [contiki] operating system. *Contiki* is one of the operating systems for the wireless-sensor networks and embedded systems that contain resource-constrained devices. The implementation considers hierarchical naming (similar to CCN) consisting of the name prefix followed by the content attributes as follows:

Prefix	**/Temperature/halinria**
Content Attributes	/Temperature/halinria**/Bld1/Floor2/Office1**
	/Temperature/halinria**/Bld1/Floor1/Office12**

The implementation uses interest and data messages of 102 bytes to match the IEEE 802.15.4 frame, which is 127-bytes long (127 to 25 (-bytes MAC header) = 102 bytes). The processing steps of these messages are modified to suit the processing capabilities of the WNS nodes. CCN PIT, FIB, and CS are also implemented accordingly. CCN implementation in Contiki is evaluated through simulations and real deployment using a synthetic monitoring application for varying network sizes.

The authors in [47] implemented the CCN-communication architecture for WSN in Wieslib [52], which is a library of algorithms that form heterogeneous sensor networks. CCN implementation for WSN is termed “CCN-WSN” and implements hierarchical naming, interest message, data message, PIT, CS, and FIB. Evaluation of this flexible implementation of CCN-WSN demonstrates the suitability of CCN in WSN. The CCN architecture for WSN was proposed in [48]. The architecture is divided into two tiers: The first tier manages the heterogeneity devices that comprise the WSN (sensor node, sink, remote server). CCN is enhanced with some changes to the forwarding strategies to improve data collection. The second tier is the modified, lightweight, and shortened CCN forwarding strategy encompassing forwarding data structures (FIB, CS, and PIT), message (interest and data) transmission, and techniques for message retransmission.

Amadeo et al. [49] explored the application of NDN ICN architecture in WSN. To increase the reliability of interest and data message communication, the authors proposed the defer-window concept for both types of messages where the defer window of the interest message is larger than the data message. Rebroadcast of data and interest is deferred for the duration of D_{DW} and I_{CW} duration, respectively, as follows:

$$D_{\mathrm{DW}} = \mathrm{rand}[0, \mathrm{DW}] * \mathrm{DeferSlotTime}$$

$$I_{\mathrm{DW}} = (\mathrm{DW} + \mathrm{rand}[0, \mathrm{DW}]) * \mathrm{DeferSlotTime}$$

The DeferSlotTime is a short time and fix interval. The reason for smaller data DW than the interest message is to provide it the higher priority. The results of the proposed defer window based NDN implementation are compared with the basic NDN implementation in WSN. The results show that the proposed defer window based NDN architecture is more energy efficient than the basic NDN implementation in WSN. The CCN implementation of Contiki [53] has been simulated and implemented on TelosB nodes, and the results are discussed in [50]. Both the simulations and the implementation show that the proposed solution has less message exchange overhead with acceptable data-retrieval delays. The authors in [51] proposed the fine-grained NDN-based trust model, which allows both consumers and providers to authenticate sensed data and access control mechanisms based on data encryption and key attributes.

The previously discussed solutions comprise the preliminary CCN implementation and their validations in WSN. However, more serious attention from researchers is required to propose more energy efficient CCN-based solutions for WSNs.

3.3.4 CCN in MANETs

Mobile ad hoc network (MANET) is an autonomous, infrastructureless, self-healing, self-organizing, dynamic topology, and multi-hop network of mostly battery-operated nodes. These characteristics of the network make data delivery between source and destination node a more challenging. Due to the multi-hop nature and dynamic network topology, data dissemination with less control overhead conserves energy and is more reliable. There has been much research to resolve these issues; however, the issues still remain, and research is still ongoing. Much literature and many books are available on the topic, and readers can consult the following very recent articles to quickly obtain information about the topic [54–58].

After the invention of a new future emerging network paradigm, CCN, which substitutes the traditional IP-based, location-dependent, and host-centric networks with the name-based, location-independent, and data-centric network, researchers investigated CCN in MANETs. Following is the brief summary of CCN-based schemes for MANETs.

In [59], the authors proposed the NDN-based new data-forwarding scheme for MANETs termed “listen first, broadcast later” (LFBL). LFBL uses three-way message exchange (announce the name prefixes, forward the interest, and return the data), which is supported by NDN. Initially a node disseminates the network-wise request carrying the names of the data requested by the application. Any node with the requested named data sends a response packet backward to the requesting node. Next, the data-receiving node sends the acknowledgement (positive or negative) packet. LFBL saves the prefixes that are flooded with the requests. When an intermediate node receives the packet, it defers its transmission for a short time and listens to that channel for any potential forwarder or responder. The results show that LFBL has better data-delivery efficiency than the famous reactive routing protocol called “ad hoc on-demand distance vector” (AODV) running over 802.11MAC and CSMA MAC.

The authors in [60] implemented the CCN architecture on laptops running Linux OS to create on-demand MANET. The CCN-based protocol suite was implemented by the authors. The CCN data structures implemented includes CS, metadata registry, and an interest table. The large MANET for emergency and tactical scenarios with hierarchical architecture, group mobility, and operation are considered in the paper. The hierarchical storage and search mechanisms for the topic-based and spatiotemporal contents are considered in the implementation. Initially the publisher disseminated the same meta-content information that is disseminated through gateway nodes at the upper layer of the hierarchy in each group and recorded at each node’s registry in each group. The node sends the interest to the gateway node, and the gateway replies with the matching content if it has the copy. Otherwise, the interest is sent to other gateways to find the content, which makes these gateways responsible to publish and deliver data.

In [61], the authors pointed out basic approaches to identify fundamental points in the design of content-centric MANET (CCM). The performance and efficiency of the identified design are evaluated through modeling. Announce content availability, send query to find the content, and fetch the content are three major operations considered in the modeling space. Three schemes are identified in the paper to retrieve content in the MANET. Reactive flooding describes the requesting node flooding the requested message to find the content, service, or location of the content or service-provider node. In this case, there is no announcement; a node just floods the query message, and the content provider replies with data or service in a unicast manner to the requesting node. The second proposed design is proactive flooding where each node periodically floods the resource or content available in MANET. The requesting node just listens to the periodic announcements, and then the request and content delivery are performed in a unicast manner. The third and the last CCN design for MANET uses geographic hash tables (GHT). This design assigns a key to each resource in the network and, when operated through this key, it may provide a pair of two-dimensional (x, y) coordinates. The node first announces the pairs (resource, host) to the nodes closest to the resource key. The requester first computes the hash of the resource in which it is interested, which provides the location of node(s) holding the pair of (resource, host). The query is forwarded in a unicast manner to these nodes through a GPSR protocol. The content fetches operation is performed once the resource host information is retrieved. Fetch and content forwarding are unicast operations. Content availability, latency, and overhead cost are modeled by the authors.

In [62], content centric fashion MANET (CHANET) has been proposed. The CHANET architecture provides detailed processing of the broadcast-nature interest and data messages, and the many features for 802.11-based MANET include the following:

- Hierarchical naming
- Content segmentation and reassembly (content divided into chunks)
- Content advertisement (periodically by fixed providers, e.g., access point)
- Content discovery (interest message) and delivery (data message)
- Retransmission request (int-ack message)

CHANET brings simplicity and robustness by using the broadcast nature of CCN messages. Nodes overhear the broadcasted messages and defer the time to reduce the number of collisions. The interest and retransmission request message forwarding decision is made by the node itself that receives the messages. It indirectly provides the retransmission request (embedding acknowledgments in the interest packets) and sequence control mechanism. The consumer provider mobility are inherently implemented and supported by the scheme.

The open-source CCN emulation for MANETs in [63] is called CCN-Joker "CCN-Java Open Source Kit EmulatoR." The emulation considers basic aspects of CCN including interest, data, cache-replacement policy, etc. The authors did not consider the CCNx (the implementation of CCN) because they wanted to have full control over the replacement policy and a lightweight application program that can easily be executed by resource-constrained devices. The main objective of selecting Java was due to its networking library (e.g., java.net) and multithreading capability. CCN-Joker implements FIB, PIT, and CS along with two additional structures called "repository" (permanent storage for content seed copies) and "my requests" (pending requests that are generated by the node itself). The implementation distinguishes between local and external interests. The interest and data messages are transferred to their respective handling modules called “request manager” and “content manager.” The assignment of interest and data messages to their respective handling modules after they are received is performed by the Joker Server module. The operations of these modules are similar to CCN core implementation.

Adem et al. [64] proposed a CCN scheme for MANETs that avoids the message loss in MANETs. The scheme uses the neighborhood information and defer time to achieve this goal. Detection of neighbors is performed by periodic overhearing of the communication activities of the neighbors. If a node does not hear any periodic advertisement or any communication activity, the link to that neighbor is considered broken. The interest and data messages may be broadcasted or unicasted. In case of broadcast messages, the time to live (TTL) value is used; however, TTL is not considered for the unicast messages. If a node has any content, it is advertised by that node. For intermediate nodes that receive this content, advertisements may record all the paths toward the content source in the FIB. Each path entry has hop-count and may record any other path metrics. The FIB entries are sorted based on the hop-count. When an interest packet arrives, most of the operations are similar to CCN other than the FIB search and next-hop forwarding. The FIB entry with minimum hop-count is used to forward the interest message. The interest message is broadcasted when no alternative path is available. In the case of node mobility, the path loss is frequent. When a node has no entry in the FIB, then it calculates the defer time and starts overhearing the neighborhood communication. At the expiration of the deferred window period, the node checks the FIB for the possible next hop entry; otherwise, it broadcasts the interest message. It is the same case with the data message. To summarize the discussion, the proposed scheme ensures the next hop entry before using it and discovers the alternative path or means to forward the messages.

An interest flooding control scheme, called “neighborhood aware interest forwarding” (NAIF), for NDN enabled MANETs has been proposed in [65]. NAIF initially prunes the ineligible forwarders because it selects the potential forwarders from the forwarder set based on the data-retrieval rate and its distance to the content provider. Afterward, the eligible potential forwarder may probably drop the received interest based on the updated smoothed forwarding rate of the related name prefix. After every update interval, t, the forwarding rate (F_t) is smoothed as $F_t = s_{\mathrm{int}}/r_{\mathrm{int}}$ where s_{int} is the number of interests sent, and r_{int} is number of non-cached interests received. This information is used to adjust the interest-forwarding rate.

The authors in [66] proposed an enhanced version of CHANET [62] called “E-CHANET” (enhanced content-centric multi-hop wireless network). It is claimed that E-CHANET solves issues such as channel unreliability, dynamic network topology, broadcast storm, data message reliability, and interest rate control as well as achieves energy efficiency in wireless multi-hop ad hoc networks. The interest and data messages are extended with additional information, and a new data structure—along with FIB, PIT, and CS—called "CPT" (content provider table) is used to achieve the above-mentioned goals. The interest message piggybacks hop-count (*iHC*), provider node ID (*sProvID*), and expected hop distance to the selected provider (D_p). Similarly, the data message carries additional information, for instance, sustainable interest rate information, hop-count same as the interest message, provider node ID, and distance to the consumer. If a consumer has no previous information of the potential provider, then *sProvID* and D_p are set to null. When a node receives an interest with the matching content provider, it initiates random defer time before data transmission. If the requesting content is not found, then the node checks the PIT and adds an entry. Before forwarding the interest message, the node must check the CPT. The interest message may include the D_p and provider ID if the potential provider is found in the CPT. This node uses a counter-based suppression technique to schedule the interest forwarding. The provider node only replies to the first interest received by it and discards copies of the same interest received later. Each intermediate data message receiving node uses hop-count information to forward the data packet toward the consumer node. Each interest message is assigned a timer (retransmission time-out) to schedule the next retransmission of the interest message. If a consumer node fails to receive data after the specified number of tries, then the consumer gives up trying to obtain the data and may try later.

The authors in [67] implemented the content-centric design for MANETs one handheld device called "SCALE." The authors use nodeID to reflect its geographic coordinates, content name, and several other parameters to achieve content communication. The demonstration supports three functions: publish, subscribe, and query the content. The data in the cache is indexed based on the actual content instead of the content name.

A topology-aware CCN protocol (TOP-CCN) was proposed in [68] employing multi-point relays (MPRs), publisher MPR (PMPR), and a hop count based flooding-control mechanism. The PMPR announces multiple contents in a single announcement to reduce announcement overhead and reduce flooding range using the hop-count limit. The MPRs are used to discover and announce the contents, and every node periodically elects MPRs. These MPRs reduce the number of announcement and content-discovery broadcast messages. The topology information is stored in tables that include one-hop and two-hop neighbor node IDs and lifetimes. The content announcement message includes the content prefix, MAC address of the node, content prefix list of neighbors, neighbor IDs, type of neighbors, the TTL of announcement message, cache hit ratio, and flooding type (one-hop/multi-hop). This information is used to maintain a neighbor list as well as make MPR selection.

A topic-based publish/subscribe content-centric network (TPS-CCN) system for MANET was proposed in [69]. The proposed TPS-CCN system uses a hierarchical naming scheme, CCN functionality, in-network caching, and multicast functionality to minimize delay and maximize delivery efficiency. Through dynamic configuration of CCN-forwarding tables, the mobile device caches data and can function as a data “mule” to transport contents toward the disconnected parts of the network. This realizes the delay tolerance in CCN networks.

TPS-CCN generally performs four functionalities: publish contents, publisher discovery and sync, pull-based data communication, and delay-tolerant data communication support. A data publish procedure executing on a node makes information items available to the related TPS-CCN subscribers. Published and unsatisfied information is stored in the information item repository (IIR) and a pending interest repository (PIR), respectively. The interest messages are processed and forwarded based on these tables. The data-pull functionality is executed by the TPS-CCN subscriber to pull content from the publishers. In case of multiple publishers of the same topic, a separate window is maintained for each provider to speed up the transfer procedure in the presence of long round-trip times. To pull information content from the publisher, the subscriber requires the topic name, publisher ID, and the sequence number of the last produced content. The subscriber periodically executes the publisher discovery and the sync procedure to retrieve updated contents. When a subscriber becomes aware that it is disconnected from the publisher, the delay-tolerant mode is activated. Publisher disconnectivity can be detected by the publisher-discovery procedure. The DTN mode is well explained in [69]. The performance evaluation of the CCN-MANET routing engine was developed as a plug-in of the OLSR Linux daemon and executed under the emulated Linux virtual-machine environment.

The authors in [70] analyzed two NDN-forwarding strategies: the blind-forwarding or provider-blind forwarding strategy and the provider-aware forwarding strategy. The blind-forwarding strategy is a counter-based broadcasting scheme, which defers data and interest message transmissions to limit collision probability and controls interest redundancy by overhearing the messages. This scheme achieves the above-mentioned performance in a scenario where NDN messages are broadcasted without any neighborhood knowledge and identity of the content sources. In contrast, the provider-aware forwarding strategy uses the soft-state information of the content sources, i.e., the distance to content source, content source ID, etc., to facilitate content retrieval. This soft-state information about the content sources is maintained in table(s) at the nodes, such as a distance table, and this information is piggybacked in interest and data messages. The above discussed schemes are summarized in Table 3.1.

Because most of the MANET nodes are battery operated, mobile, and communicate without infrastructure support, the CCN solution must efficient in content discovery, message-forwarding reliability, and support security. The proposed solutions focus solely on the application and testing of CCN architecture, and more solutions are required to support the reliability and efficiency of CCN messages as well as alleviate bandwidth congestion resulted from the broadcast nature of the network.

Table 3.1 CCN-/NDN-based schemes for MAMETs

Name	Scheme	Architecture	Simulator	Comparison	MANET
LFBL [59]	Reliable forwarding	NDN	Qualnet	AODV over 802.11MAC and CSMA-MAC	General with RWP mobility
MANET CCN [60]	Metadata and content forwarding in hierarchical MANETs through gateways that manage the group	CCN	Implementation	OLSRD	Hierarchical architecture, group mobility
CCM [61]	Announce, query, and fetch operations; reactive and proactive flooding; hash table based design	CCN	Modeling latency, delay and availability of content.	–	General
CHANET [62]	Content-centric fashion MANET with retransmission support	CCN	NS-2	FTP over TCP/IP	802.11 with AP*
CCN-Joker [63]	Java-based fully customizable and open-source emulation of CCN-like architecture	CCN	Java based implementation	–	General
Adem et al. [64]	CCN message loss avoidance in dynamic MANETs using the defer-time scheme	CCN	OPNET	Typical CCN	General
NAIF [65]	Neighborhood-aware interest forwarding based on interest-forwarding rate	NDN	Qualnet	LFBL and NDN forwarding daemon	General
E-CHANET [66]	Enhanced CHANET with retransmission and congestion-control support	CCN	NS-2	CCN, CHANET, and TCP/IP	802.11 with AP*
SCALE [67]	Geographic location of device, content, name, and other parameters are used to implement content communication	ICN	Implementation on Android devices	–	Generic
TOP-CCN [68]	MPR-based content announcement-and-discovery protocol	CCN	NS-3	CCN and E-CHANET	Grid wireless network
TPS-CCN [69]	TPS-CCN publishes content, publisher discovery, and sync pull-based data communication and delay-tolerant data communication support	CCN	CCN MANET routing engine implemented as a plug-in of OLSR Linux daemon	UDP-Ack, no DTN, no cache, and reliable IP multicast	Sparse MANET
Amadeo et al. [70]	Blind forwarding and provider-aware forwarding	NDN	NS-3	–	Wireless network with AP* and MANET

*Access point

3.3.5 CCN in VANETs

Vehicular ad hoc networks (VANETs) are a reality of the near future and play vital role in everyday life by providing services, e.g., safety and comfort of passengers while driving, infotainment, etc. The network comprises vehicles with communication capabilities. One of the most prominent characteristic of the VANET is its highly dynamic topology, which makes it challenging to propose any efficient and promising communication solution. Along with that, it has also been proven by many researchers that the TCP/IP communication protocol stack is inefficient for mobile networks. This is the reason that a separate protocol stack for VANETs, called "wireless access in vehicular environments" (WAVE) [71], has been proposed. WAVE supports data exchange without the TCP/IP overhead through the WAVE short message protocol (WSMP), which was designed for safety critical and control messages.

Recently, there has been much research with the intention to reliably communicate emergency information, traffic status, vehicle sensory data, and infotainment information in a non host centric manner by using an information-centric communication mechanism. Content-centric approaches are briefly discussed and summarized in [72]. Here, we will discuss and summarize the very recent CCN-based schemes for VANETs.

The very first proposal in [73] employs a named data communication mechanism to collect vehicles' sensory data from the mobile vehicular network. This information may be used by the manufacturing or related companies to provide vehicle management, safety, and alert information to the drivers or owners of the vehicles or companies. A hierarchical naming scheme, with company, type of information, country, state, etc., is used to request this sensory information.

A CCN-based multi-interface enabled vehicular network has been proposed in [74] where each vehicle may be equipped with more than one network interface to communicate with other vehicles or infrastructure (e.g., 802.16 (WiMAX), IEEE 802.11p, WiMAX, and UMTS). It supports a general pull-based vehicular network data communication and push-based (unsolicited messages) safety-messages dissemination as well. The push-based packet is called the "event packet." It is possible that any vehicle can generate a denial of service (DoS) attach by consuming the network resource by sending a large number of event packets. This is controlled by limiting the number of such packets from the neighboring nodes. The format or structure of the event packet is similar to the CCN's ContentObject or data message.

CCN and NDN use a hierarchical naming scheme with general components to identify the contents. The naming scheme for vehicular networks to represent spatial and temporal vehicular information has been proposed in [vccn03]. This naming scheme is used by vehicular applications to communicate contents between vehicles and the infrastructure. The vehicular network information is identified by the following naming scheme: **/traffic/geographic_scope/temporal_scope/data_type/NONCE.**

The CCN architecture was implemented in 802.11p-enabled vehicles and RSUs in [75]. It uses three types of messages: **B-Int**, **A-Int**, and **C-Obj**. The content name is used in all messages with the potentiality to discover, identify, and retrieve content or a chunk of the content. **The B-Int message** is broadcasted to discover the content provider including the content name, sequence number (to avoid duplicate), hop count, and additional options. On receiving the **B-Int**, a node searches its content store, and if it finds the matching content it sends the C-Obj (s) after time-out of the random defer time. In case of multiple copies of the **A-Int** message, only the first one is answered. Intermediate nodes that maintain **C-Obj** maintain entries about the content provider in CPT: provider ID (MAC Address), hop-count distance, and the provided content ID. The **C-Obj** message consists of the content name, provided content chunk information, segment information, hop-count, provider ID, and the content itself. After receiving the first content chunk, successive chunks are requested by sending the **A-Int**, which carries additional information compared with the **B-Int**: preferred provider ID, expected hop distance to provider, and acknowledgement of previously received chunks in a bitmap format. The simulations show that CRoWn has a better content-retrieval rate compared with the legacy TCP/IP-based communication protocol.

Authors in [76] proposed a content centric communication scheme for autonomous vehicles, named “CarSpeak.” CarSpeak enabled autonomous vehicles communicate sensory information from neighboring cars as well as the sensor installed on the static infrastructures, e.g., RSU, road side buildings, etc. using the content-centric interest-data mechanism.

A content-centric vehicular network (CCVN) transport strategy has been proposed in [78] that is almost similar to the one proposed in [77]. A preliminary analysis of CCVN and its effectiveness was performed in [79]. The main difference compared with [77] is the consumer- and provider-driven handovers proposed in [78]. In the consumer-driven handover, the consumer selects the nearest provider based on the providers’ information from the CPT. In the case of no provider information, the consumer sends a new **B-Int** message. If a node that is not enlisted in A-Int as a potential provider receives the **A-Int** message and also has the matching content in CS, it can reply with the **C-Obj**. This is called the provider-driven handover. A node only selects itself as a new potential provider if it has less hop distance than the provider in **A-Int** message.

Due to the broadcast nature of wireless faces, collision of interest and data messages is inevitable. The authors in [80] introduced different timers to avoid the NDN-message collisions in an NDN-enabled vehicle-to-vehicle (V2V) multi-hop highway communication network. The timers include a collision-avoidance timer, pushing timer, NDN-layer retransmission, and application retransmission timer. With the collision-avoidance timer, a node randomly delays the interest in data transmission between 0 and 2 ms. Push type messages are scheduled based on the pushing timer, which is computed based on the transmitter-receiver distance, maximum transmission range, and minimum next-hop delay. The NDN-layer retransmission (50 ms) and application retransmission timers are used to schedule the retransmission of NDN messages over the lossy wireless network.

Hierarchical Bloom-filter routing (HBFR) has been proposed for CCN-enabled VANET in [81]. HBFR identifies each chunk of the content object with a hierarchal name e.g., /category/service-name/additional-info/. The category shows the type of content based on its size, type, popularity, etc. The service-name identifies the data services provided by the node(s) based on time-sensitivity of the content. The additional-info component contains any additional information about the content. The proposed HBFR routing framework adaptively performs reactive and proactive content discovery based on content characteristics. It uses Bloom filters to announce the popular prefixes, and the announcement is restricted within a geographical region. The network is divided into hierarchical geographic regions to restrict the distribution of messages in order to reduce overhead. Vehicles in a single region form a cluster, and every vehicle in that region is a member of that cluster. Contents advertisement by a vehicle includes its region information, and it adds the content prefixes using Bloom filters in the content advertisements. These Bloom filter based advertisements are shared between regions to obtain a full view of the contents as well as their respective regions to easily discover and communicate the desired content.

The authors in [82] proposed the content-segmentation reassembly and reliable content delivery scheme by scheduling the retransmission of interests for lost-data messages. To quickly recover lost-data messages, the interests are retransmitted depending on the dynamic round-trip time (RTT) of the interest data exchange. The average RTT over a multi-hop path is estimated as a weighted moving average of RTT samples, and based on this information the retransmission time out is scheduled.

Last-encounter content routing (LER) in [83] is an opportunistic geo-inspired routing scheme for CCN-enabled VANET. Each vehicle that runs LER maintains two tables called the “content list” and the “last encounter list” (LEL). The content list contains the list of all contents that the vehicle holds and is shared within one-hop neighbors. The LEL enlists the content and information received from neighboring vehicles and enlists the provider ID, encounter position, and time. A content provided by multiple vehicles has multiple entries in the LEL. The interests are forwarded toward the location provided in the message.

A content-centric networking scheme for vehicular networks that uses provider information has been proposed in [84]. The scheme is proposed by the same authors and is an extension of the one proposed in [78]. This is the reason the scheme uses the same messages as in [78]: ***A-Int***, ***B-Int***, and ***C-Obj***. The scheme uses provider selection mechanisms called “most responsive provider” (CCVN-MRP) and “nearest most responsive provider” (CCVN-NMRP). CCVN-MRP maintains the provider ID performed by the data message. The content-provider table is maintained at each node and stores content name, provider ID, and response counter.

The response counter is the number of times the provider replied to a data message for the said content. A consumer selects the provider with the highest response value, and that information of the selected provider is sent in the interest message. In contrast, the CCVN-NMRP scheme uses response count and hop-distance between the provider and consumer vehicles to select the potential provider. In CCVN-MRP, a consumer vehicle selects the nearest and most responsive provider.

The results of the vehicular NDN implementation in vehicles have been presented in [85]. The implementation uses various wireless faces, i.e., Wi-Fi, WiMAX, IEEE802.11p, etc., and the messages are transmitted over all the available faces to avoid disruptions caused by intermittent connectivity. The experiments have been performed in the vehicular network with no mobility, vehicles moving in a platoon, and vehicles moving around the University of California, Los Angeles (UCLA) campus.

In [86], the authors proposed a forwarding scheme that fetches data from a plethora of providers using digital map information. Navigo binds the NDN data names to the producers' geographic area(s). It uses a shortest-path algorithm to forward interests to the geographic area of the potential provider. The authors also claim the application of an adaptive best data provider discovery and selection scheme from multiple geographic areas.

The traffic violation ticketing (TVT) application for CCN-enabled vehicular networks has been proposed in [87]. It discusses the use of CCN interest and data messages used by police officers to issue violation tickets to drivers who commit violations. Additional data structures are maintained to achieve this application perspective. The same authors proposed and evaluated a hierarchal and hash-based content-naming scheme for vehicular networks [88]. The naming scheme encompasses the provider's identity, different components that represent the content attributes, and the spatio-temporal resolution of contents. A small hash component is also part of the name, which helps to precisely identify the content. Along with that, a compact tri-based name-management scheme has been adapted to manage the content name to perform speedy search, delete, and add name in the name prefix tables. Analysis shows that the proposed name-management scheme is more suitable for variable length name prefix management in content-centric networks.

A robust interest forwarder selection (RUFS) scheme for vehicular ad hoc networks has been proposed in [89]. RUFS mitigates the interest broadcast storm by selecting the suitable next-hop forwarder vehicle. In RUFS, each vehicle shares its satisfied-interests statistics with neighboring nodes. This information is managed in the neighbors-satisfied list (NSL). The authors also summarized the problems, challenges, and future perspectives of CCN and NDN in VANETs in [90]. ICN schemes for vehicular networks discussed previously are summarized in Table 3.2.

Table 3.2 CCN-/NDN-based schemes for vehicular networks

Name	ICN architecture	Scheme	Comparison	Network type	Simulator
NMND [72]	NDN	Collect vehicles' sensory information from the infrastructure (RSU)-supported vehicular network	MobileIP	Infrastructure-supported vehicular network	Qualnet
Hybrid VANET [74]	CCN	Implemented push-based event packet for CCN-based VANETs	–	VANETs	NS-3
Vehicular naming [75]	CCN/NDN	Hierarchical naming scheme for vehicular network representing temporal and spatial scope information	–	VANETs	–
CarSpeak [76]	MAC layer multiresolution naming	Collects sensory information from neighboring vehicles and nearby infrastructure	802.11 and 802.11 + naming	Golf Car and 10 iRobots connected with Xbox 360 Kinect sensors	Implemented on robots and Golf Car
CRoWn [77]	CCN	Forwarding strategy for V2V and V2I; it uses advertise, discover, and transfer content (***A-Int***, ***B-Int***, and ***C-Obj***) messages	Legacy TCP/IP-based architecture using AODV	Infrastructure-supported vehicular network	NS-2
CCVN [78]	CCN	Forwarding strategy with consumer- and provider-driven handover schemes; it uses the same messages as [vccn5]	Legacy TCP/IP-based architecture using AODV	Infrastructure-supported vehicular network	NS-2
CCVN [79]	CCN	Evaluates effectiveness of CCN in VANETs; it shows preliminary simulation results	Legacy TCP/IP-based architecture	Infrastructure-supported vehicular network	NS-2
V2V NDN [80]	NDN	Set of timers is used to avoid collision of NDN messages	–	V2V highway VANET scenario	NS-3 (ndnSIM)
HBFR [81]	CCN	Hierarchical Bloom-filter routing uses Bloom filters to advertise and communicated the contents in region-based clusters	CCN	VANET	Qualnet

(continued)

Table 3.2 (continued)

Name	ICN architecture	Scheme	Comparison	Network type	Simulator
CCN retransmission [82]	CCN	Content segmentation/reassembly and reliable content delivery (interest-retransmission scheme for lost-data messages)	–	Infrastructure-supported vehicular network	NS-2
LER [83]	CCN	Last-encounter content Routing is opportunistic geo-inspired content-based routing	Flooding	VANET	NS-3
CCVN-(MRP and NMRP) [84]	CCN	The provider is selected based on the large number of responses, number of responses, and the smallest hop distance	CCN	Infrastructure-supported vehicular network	NS-2
V-NDN [85]	NDN	NDN implementation over vehicular network	–	V2I, I2V, and Infrastructure-supported V2V	UCLA test bed
NAVIGO [86]	NDN	Geographic area based forwarding scheme for NDN-enabled VANETs	GPSR	Infrastructure-supported vehicular network	NS-3 (ndnSIM)
TVT [87]	CCN	Application of CCN in traffic violation ticketing use case	–	VANET	–
Hierarchical and hash-based Naming [88]	CCN	Hierarchical and hash-based naming scheme using compact Trie management scheme for vehicular CCN	Simple Trie and Bloom filters	VANET	C/C++
RUFS [89]	CCN	RobUst interest-forwarder selection scheme to mitigate interest-forwarder broadcast storm	CCN, DR-based, and NAIF	VANET	

References

1. CCNx—Content Centric Networking. url https://www.ccnx.org/
2. Jacobson V, Smetters DK, Thornton JD, Plass MF, Briggs NH, Braynard RL (2009) Networking named content. In: Proceedings of the 5th international conference on Emerging networking experiments and technologies (CoNEXT'09). ACM, New York, NY, USA, pp 1–12
3. Named Data Networking (NDN)—A future internet architecture. url http://named-data.net/, NSF's Future Internet Architecture Program
4. Content-Centric Networking CCNx Reference Implementation. url https://github.com/ProjectCCNx/ccnx
5. Kim K, Choi S, Kim S, Roh B-h (2013) A push-enabling scheme for live streaming system in content-centric networking. In: Proceedings of the 2013 workshop on Student workshop (CoNEXT Student Workshop'13). ACM, New York, NY, USA, pp 49–52
6. Francois J, Cholez T, Engel T (2013) CCN traffic optimization for IoT. In: 2013 Fourth international conference on the network of the future (NOF), pp 1–5, 23–25 Oct 2013
7. Teubler T, Hail MAM, Hellbrück H (2013) Efficient data aggregation with CCNx in wireless sensor networks. In: Advances in communication networking, vol 8115. Lecture notes in computer science, pp 209–220
8. Mosko M, Solis I, Uzun E, Wood C (2015) CCNx 1.0 protocol architecture. Technical report, Aug 2015. url http://www.ccnx.org/pubs/CCNxProtocolArchitecture.pdf
9. Zhang B, Afanasyev A, Burke J, Jacobson V, Crowley P, Papadopoulos C, Wang L, Zhang B (2010) Named data networking
10. Ren J et al (2014) MAGIC: a distributed MAx-Gain In-network caching strategy in information-centric networks. IEEE INFOCOM NOM workshop
11. Bernardini C, Silverston T, Festor O (2014) Socially-aware caching strategy for content centric networking. IFIP networking
12. Bernardini C, Silverston T, Festor O (2013) MPC: popularity-based caching strategy for content centric networks. IEEE ICC
13. Psaras I, Chai WK, Pavlou G (2012) Probabilistic in-network caching for information-centric networks. In: Proceedings of the second edition of the ICN workshop on Information-centric networking. ACM, New York
14. Chai WK et al (2012) Cache less for more in information-centric networks. In: NETWORKING 2012. Springer, Berlin, pp 27–40
15. Private definitions for ccnd—the CCNx daemon "ccnd_private.h". url https://github.com/ProjectCCNx/ccnx/blob/master/csrc/ccnd/ccnd_private.h
16. Rowley Jennifer (2007) The wisdom hierarchy: representations of the DIKW hierarchy. J Inf Sci 33(2):163–180. doi:10.1177/0165551506070706
17. Mosko M, Solis I, Uzun E (2015) CCN 1.0 protocol architecture. Palo Alto Research Center. url http://ccnx.org/pubs/ICN_CCN_Protocols.pdf
18. Mosko M (2015) CCNx content object chunking. ICNRG, Internet-Draft, draft-mosko-icnrg-ccnxchunking-01, 1 July 2015
19. Mosko M, Scott G, Solis I, Wood C (2015) CCNx manifest specification. ICNRG, Internet-Draft, draft-wood-icnrg-ccnxmanifests-00, 24 June 2015. url http://www.ccnx.org/pubs/draft-wood-icnrg-ccnxmanifests-00.html
20. Mosko M, Solis I, Mahadevan P, Uzun E (2014) CCNx 1.0 naming: transforming network addresses to application value. Technical report, PARC, 16 March 2014
21. Zhang H, Quan W, Guan J, Xu C, Song F (2016) Uniform information with a hybrid naming (hn) scheme. (https://tools.ietf.org/html/draft-zhang-icnrg-hn-01) draft-zhang-icnrg-hn-01.txt, Expires: 7 April 2016
22. Ding S, Chen Z, Liu Z (2012) Parallelizing FIB lookup in content centric networking. 2012 Third international conference on networking and distributed computing (ICNDC). IEEE, pp 6–10

23. Mosko M (2015) CCNx semantics. IETF Internet-Draft, ccnx-mosko-semantics-01, 21 Jan 2015
24. Mosko M (2015) TLV packet format. IETF Internet-Draft, ccnx-mosko-tlvpackets-01, 21 Jan 2015
25. Mosko M (2015) Labeled content information. ICNRG, Internet-Draft, draft-mosko-icnrg-ccnxlabeledcontent-00, 13 July 2015
26. Abu AJ, Bensaou B, Wang JM (2014) Interest packets retransmission in lossy CCN networks and its impact on network performance. ICN'14, 24–26 Sept 2014, Paris, France, pp 167–176
27. Bernardini C, Silverston T, Festor O (2013) Cache management strategy for CCN based on content popularity. In: Emerging management mechanisms for the future internet, vol 7943. Lecture notes in computer science. Springer, Berlin, pp 92–95
28. Miorandi D, Sicari S, De Pellegrini F, Chlamtac I (2012) Internet of things: vision, applications and research challenges. Ad hoc Networks 10(7):1497–1516, Sept 2012. ISSN 1570-8705, http://dx.doi.org/10.1016/j.adhoc.2012.02.016
29. IERC-European Research Cluster on the Internet of Things (2011) Internet of things—Pan European Research and Innovation Vision, 2011. url http://www.internet-of-things-research.eu/pdf/IERC_IoT-Pan%20European%20Research%20and%20Innovation%20Vision_2011_web.pdf
30. Wilson S (2013) Rising tide—Exploring pathways to growth in the mobile semiconductor industry, 6 Nov 2013. url http://dupress.com/articles/rising-tide-exploring-pathways-to-growth-in-the-mobile-semiconductor-industry/
31. Waltari OK (2013) Content-centric networking in the internet of things. MSc thesis, Department of Computer Science, University of Helsinki, 25 Nov 2013. url http://hdl.handle.net/10138/42303
32. Quevedo J, Corujo D, Aguiar R (2014) A case for ICN usage in IoT environments. In: Global Communications Conference (GLOBECOM), 2014. IEEE, 8–12 Dec 2014, pp 2770–2775
33. Baccelli E, Mehlis C, Hahm O, Schmidt TC, Wählisch M (2014) Information centric networking in the IoT: experiments with NDN in the wild. In: Proceedings of the 1st international conference on Information-centric networking (ICN'14). ACM, New York, NY, USA, pp 77–86
34. CCN Lite: lightweight implementation of the content centric networking protocol, 2014. url http://ccn-lite.net
35. RIOT: the friendly operating system for the internet of things. url http://www.riot-os.org/
36. Amadeo M, Campolo C, Molinaro A (2014) Multi-source data retrieval in IoT via named data networking. In: Proceedings of the 1st international conference on Information-centric networking (ICN'14). ACM, New York, NY, USA, pp 67–76
37. Katsaros K, Chai W, Wang N, Pavlou G, Bontius H, Paolone M (2014) Information-centric networking for machine-to-machine data delivery: a case study in smart grid applications. In: Network, vol 28, no 3. IEEE, May–June 2014, pp 58–64
38. Yu K, Zhu L, Wen Z, Mohammad A, Zhou Z, Sato T (2014) CCN-AMI: performance evaluation of content-centric networking approach for advanced metering infrastructure in smart grid. In: 2014 IEEE international workshop on applied measurements for power systems proceedings (AMPS), 24–26 Sept 2014, pp 1–6
39. Yu K, Arifuzzaman M, Wen Z, Zhang D, Sato T (2015) A key management scheme for secure communications of information centric advanced metering infrastructure in smart grid. In: IEEE transactions on instrumentation and measurement, vol 64, no 8, Aug 2015, pp 2072–2085
40. Chai WK, Katsaros KV, Strobbe M, Romano P, Ge C, Develder C, Pavlou G, Wang N (2015) Enabling smart grid applications with ICN. 2nd ACM conference on information-centric networking (ICN 2015), 30 Sept–2 Oct 2015, pp 207–208
41. Rault T, Bouabdallah A, Challal Y (2014) Energy efficiency in wireless sensor networks: a top-down survey. Comput Netw 67(4):104–122

42. Kafi MA, Djenouri D, Ben-Othman J, Badache N (2014) Congestion control protocols in wireless sensor networks: a survey. In: Communications surveys & tutorials, vol 16, no 3. IEEE, pp 1369–1390, Third Quarter 2014
43. Rawat P, Singh KD, Chaouchi H, Bonnin JM (2014) Wireless sensor networks: a survey on recent developments and potential synergies. J Supercomput 68(1):1–48, 09 Oct 2013
44. Singh SP, Sharma SC (2015) A survey on cluster based routing protocols in wireless sensor networks. In: Procedia computer science, vol 45, pp 687–695
45. Butun I, Morgera SD, Sankar R (2014) A survey of intrusion detection systems in wireless sensor networks. In: Communications surveys & tutorials, vol 16, no 1. IEEE, pp 266–282, First Quarter 2014
46. Saadallah B, Lahmadi A, Festor O (2012) CCNx for Contiki: implementation details. Technical report RT-0432, INRIA, p 52
47. Ren Z, Hail MA, Hellbruck H (2013) CCN-WSN—A lightweight, flexible Content-Centric Networking protocol for wireless sensor networks. In: 2013 IEEE eighth international conference on intelligent sensors, sensor networks and information processing, 2–5 April 2013, pp 123–128
48. Meijers JP, Amadeo M, Campolo C, Molinaro A, Paratore SY, Ruggeri G, Booysen MJ (2013) A two-tier content-centric architecture for wireless sensor networks. In: 2013 21st IEEE international conference on network protocols (ICNP), 7–10 Oct 2013, pp 1–2
49. Amadeo M, Campolo C, Molinaro A, Mitton N (2013) Named data networking: a natural design for data collection in wireless sensor networks. In: Wireless Days (WD), 2013 IFIP, 13–15 Nov 2013, pp 1–6
50. Abidy Y, Saadallahy B, Lahmadi A, Festor O (2014) Named data aggregation in wireless sensor networks. In: Network operations and management symposium (NOMS), 2014 IEEE, 5–9 May 2014, pp 1–8
51. Burke J, Gasti P, Nathan N, Tsudik G (2014) Secure sensing over named data networking. In: Proceedings of the 13th IEEE international symposium on network computing and applications (NCA)
52. Baumgartner T, Chatzigiannakis I, Fekete S, Koninis C, Kröller A, Pyrgelis A (2010) Wiselib: a generic algorithm library for heterogeneous sensor networks. In: Sá Silva J, Krishnamachari B, Boavida F (eds) Proceedings of the 7th European conference on wireless sensor networks (EWSN'10). Springer, Berlin, pp 162–177
53. Contiki: The open source OS for the internet of things. url http://www.contiki-os.org/
54. Dorronsoro B, Ruiz P, Danoy G, Pigne Y, Bouvry P (2014) Evolutionary algorithms for mobile ad hoc networks. Wiley, New York
55. Mahmood BA, Manivannan D (2015) Position based and hybrid routing protocols for mobile ad hoc networks: a survey. Wireless Personal Commun 83(2):1009–1033, 21 Feb 2015
56. Reina DG, Askalani M, Toral SL, Barrero F, Asimakopoulou E, Bessis N (2015) A survey on Multihop ad hoc networks for disaster response scenarios. Int J Distrib Sensor Netw 2015:16 p
57. Attia R, Rizk R, Ali HA (2015) Internet connectivity for mobile ad hoc network: a survey based study. In: Wireless networks, vol 21, No 7, 1 Oct 2015, pp 2369–2394
58. Ruiz P, Bouvry P (2015) Survey on broadcast algorithms for mobile ad hoc networks. ACM Comput Surv 48(1):35 p, Article 8
59. Meisel M, Pappas V, Zhang L (2010) Ad hoc networking via named data. In: Proceedings of the fifth ACM international workshop on mobility in the evolving internet architecture (MobiArch'10). ACM, New York, NY, USA, pp 3–8
60. Oh SY, Lau D, Gerla M (2010) Content centric networking in tactical and emergency MANETs. In: IFIP wireless days (WD), 20–22 Oct 2010, pp 1–5
61. Varvello M, Rimac I, Lee U, Greenwald L, Hilt V (2011) On the design of content-centric MANETs. In: 2011 Eighth international conference on wireless on-demand network systems and services (WONS), 26–28 Jan 2011, pp 1–8
62. Amadeo M, Molinaro A (2011) CHANET: a content-centric architecture for IEEE 802.11 MANETs. In: 2011 International Conference on the network of the future (NOF), 28–30 Nov 2011, pp 122–127

63. Cianci I, Grieco LA, Boggia G (2012) CCN—Java opensource kit EmulatoR for wireless ad hoc networks. In: Proceedings of the 7th international conference on future internet technologies (CFI'12). ACM, New York, NY, USA, pp 7–12
64. Adem O, Kang S-j, Ko Y-B (2013) Packet loss avoidance in content centric mobile adhoc networks. In: Proceedings of the 15th international conference on advanced communication technology (ICACT), 2013, 27–30 Jan 2013, pp 245–250
65. Yu Y-T, Dilmaghani RB, Calo S, Sanadidi MY, Gerla M (2013) Interest propagation in named data manets. In: International conference on computing, networking and communications (ICNC), 2013, 28–31 Jan 2013, pp 1118–1122
66. Amadeo M, Molinaro A, Ruggeri G (2013) E-CHANET: routing, forwarding and transport in information-centric multihop wireless networks. In: Computer communications, vol 36, No 7, pp 792–803, April 2013
67. Varvello M, Schurgot M, Esteban J, Greenwald L, Guo Y, Smith M, Stott D, Wang L (2013) SCALE: a content-centric MANET. In: 2013 IEEE conference on computer communications workshops (INFOCOM WKSHPS), 14–19 April 2013, pp 29–30
68. Kim J, Shin D, Ko Y-B (2013) TOP-CCN: topology aware content centric networking for mobile ad hoc networks. In: 2013 19th IEEE international conference on networks (ICON), 11–13 Dec 2013, pp 1–6
69. Detti A, Tassetto D, Melazzi NB, Fedi F (2015) Exploiting content centric networking to develop topic-based, publish–subscribe MANET systems. In: Ad hoc networks, vol 24, Part B, Jan 2015, pp 115–133
70. Amadeo M, Campolo C, Molinaro A (2015) Forwarding strategies in named data wireless ad hoc networks: design and evaluation. J Network Comput Appl 50:148–158
71. IEEE Standard for Wireless Access in Vehicular Environments (WAVE)—Networking services—Redline. In: IEEE Std 1609.3-2010 (Revision of IEEE Std 1609.3-2007)—Redline, 30 Dec 2010, pp 1–212
72. TalebiFard P, Leung VCM, Amadeo M, Campolo C, Molinaro A (2015) Information-centric networking for VANETs, Chap 17. In: Campolo C, Molinaro A, Scopigno R (eds) Vehicular ad hoc networks. Springer, Berlin, pp 503–524
73. Wang J, Wakikawa R, Zhang, L (2010) DMND: collecting data from mobiles using named data. In: IEEE vehicular networking conference (VNC), 2010, 13–15 Dec 2010, pp 49–56
74. Arnould G, Khadraoui D, Habbas Z (2011) A self-organizing content centric network model for hybrid vehicular ad-hoc networks. In: Proceedings of the first ACM international symposium on design and analysis of intelligent vehicular networks and applications (DIVANet'11). ACM, New York, NY, USA, pp 15–22
75. Wang L, Wakikawa R, Kuntz R, Vuyyuru R, Zhang L (2012) Data naming in vehicle-to-vehicle communications. In: IEEE conference on computer communications workshops (INFOCOM WKSHPS), 25–30 March 2012, pp 328–333
76. Kumar S, Shi L, Ahmed N, Gil S, Katabi D, Rus D (2012) CarSpeak: a content-centric network for autonomous driving. SIGCOMM Comput Commun Rev 42(4):259–270
77. Amadeo M, Campolo C, Molinaro A (2012) CRoWN: content-centric networking in vehicular ad hoc networks. IEEE Commun Lett 16(9):1380–1383
78. Amadeo M, Campolo C, Molinaro A (2012) Content-centric vehicular networking: an evaluation study. In: Third international conference on the network of the future (NOF), 21–23 Nov 2012, pp 1–5
79. Amadeo M, Campolo C, Molinaro A (2012) Content-centric networking: is that a solution for upcoming vehicular networks? In: Proceedings of the ninth ACM international workshop on vehicular inter-networking, systems, and applications (VANET'12). ACM, New York, NY, USA, pp 99–102
80. Wang L, Afanasyev A, Kuntz R, Vuyyuru R, Wakikawa R, Zhang L (2012) Rapid traffic information dissemination using named data. In: Proceedings of the 1st ACM workshop on emerging name-oriented mobile networking design—Architecture, algorithms, and applications (NoM'12). ACM, New York, NY, USA, pp 7–12

81. Yu Y-T, Li X, Gerla M, Sanadidi MY (2013) Scalable VANET content routing using hierarchical bloom filters. In: 9th international wireless communications and mobile computing conference (IWCMC), 1–5 July 2013, pp 1629–1634
82. Amadeo M, Campolo C, Molinaro A (2013) Design and analysis of a transport-level solution for content-centric VANETs. In: Proceedings of the IEEE international conference on communications workshops (ICC), 9–13 June 2013, pp 532–537
83. Yu Y-T, Li Y, Ma X, Shang W, Sanadidi MY, Gerla M (2013) Scalable opportunistic VANET content routing with encounter information. In: 2013 21st IEEE international conference on network protocols (ICNP), 7–10 Oct 2013, pp 1–6
84. Amadeo M, Campolo C, Molinaro A (2013) Enhancing content-centric networking for vehicular environments. Comput Networks 57(16):3222–3234, 13 Nov 2013
85. Grassi G, Pesavento D, Pau G, Vuyyuru R, Wakikawa R, Zhang L (2014) VANET via named data networking. In: IEEE conference on computer communications workshops (INFOCOM WKSHPS), 27 April–2 May 2014, pp 410–415
86. Grassi G, Pesavento D, Pau G, Zhang L, Fdida S (2015) Navigo: interest forwarding by geolocations in vehicular named data networking. IEEE 16th international symposium on "a world of wireless, mobile and multimedia networks" (WoWMoM), June 2015, pp 1–10
87. Ahmed SH, Yaqub MA, Bouk SH, Kim D (2015) Towards content-centric traffic ticketing in VANETs: an application perspective. In: 2015 Seventh international conference on ubiquitous and future networks (ICUFN), 7–10 July 2015, pp 237–239
88. Bouk SH, Ahmed SH, Kim D (2015) Hierarchical and hash based naming with Compact Trie name management scheme for vehicular content centric networks. Comput Commun. Available online, 3 Oct 2015
89. Ahmed SH, Bouk SH, Kim Dongkyun (2015) RUFS: RobUst forwarder selection in vehicular content-centric networks. IEEE Commun Lett 19(9):1616–1619
90. Bouk SH, Ahmed SH, Kim D (2015) Vehicular content centric network (VCCN): a survey and research challenges. In: Proceedings of the 30th annual ACM symposium on applied computing (SAC'15). ACM, New York, NY, USA, pp 695–700