ICN Research Group
Internet-Draft
Intended status: Experimental
Expires: March 25, 2021

H. Asaeda
A. Ooka
NICT
X. Shao
Kitami Institute of Technology

September 21, 2020

CCNinfo: Discovering Content and Network Information in Content-Centric Networks

draft-irtf-icnrg-ccninfo-05

Abstract

This document describes a mechanism named "CCNinfo" that discovers information about the network topology and in-network cache in Content-Centric Networks (CCN). CCNinfo investigates: 1) the CCN routing path information per name prefix, 2) the Round-Trip Time (RTT) between the content forwarder and consumer, and 3) the states of in-network cache per name prefix.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In Content-Centric Networks (CCN), publishers provide the content through the network, and receivers retrieve it by name. In this network architecture, routers forward content requests through their Forwarding Information Bases (FIBs), which are populated by name-based routing protocols. CCN also enables receivers to retrieve content from an in-network cache.

In CCN, while consumers do not generally need to know the content forwarder that is transmitting the content to them, the operators and developers may want to identify the content forwarder and observe the routing path information per name prefix for troubleshooting or investigating the network conditions.

Traceroute [5] is a useful tool for discovering the routing conditions in IP networks because it provides intermediate router addresses along the path between the source and destination and the Round-Trip Time (RTT) for the path. However, this IP-based network tool cannot trace the name prefix paths used in CCN. Moreover, such IP-based network tools do not obtain the states of the in-network cache to be discovered.

Contrace [6] enables end users (i.e., consumers) to investigate path and in-network cache conditions in CCN. Contrace is implemented as an external daemon process running over TCP/IP that can interact with a previous CCNx forwarding daemon (CCNx-0.8.2) to retrieve the caching information on the forwarding daemon. This solution is flexible, but it requires TCP/IP networks and defining the common APIs for the global deployment.

This document describes the specifications of "CCNinfo", an active networking tool for discovering the path and content caching information in CCN. CCNinfo defines the protocol messages to investigate path and in-network cache conditions in CCN. It is embedded in the CCNx forwarding process and facilitates with non-IP networks as with the basic CCN concept.

CCNinfo is intended as a comprehensive network management tool for CCN networks. It provides a wealth of information from CCN forwarders, including on-path in-network cache conditions as well as forwarding path instrumentation of multiple paths toward content forwarders. ICN ping [7] and traceroute [8], in contrast, are lightweight operational tools that enable a user to explore the path(s) that reach a publisher or a cache storing the named content. As a management capability that exposes detailed information about the forwarders deployed by a network operator, CCNinfo employs more granular authorization policies than those required of ICN ping or ICN traceroute.

Usually, the CCNinfo user (e.g., consumer) invokes the CCNinfo user program (such as "ccninfo" command described in Appendix A) with the name prefix of the content. The user program initiates the "Request" message (described in Section 3.1) to obtain routing path and cache information. When an appropriate adjacent neighbor router receives the Request message, it retrieves own cache information. If the router is not the content forwarder for the specified name prefix, it inserts its "Report" block (described in Section 3.1.2) into the Request message and forwards it to its upstream neighbor router(s) decided by its FIB.

The Request message is forwarded by routers toward the content publisher and the Report record is inserted by each intermediate router. When the Request message reaches the content forwarder (i.e., a router that can forward the specified content), the content forwarder forms the "Reply" message (described in Section 3.2) and sends it to the downstream neighbor router. The Reply message is forwarded back toward the user in a hop-by-hop manner (along the PIT entries).

These two message types, Request and Reply messages, are encoded in the CCNx TLV format [1]. The request-reply message flow, walking up the tree from a consumer toward a publisher, is similar to the behavior of the IP multicast traceroute facility [9].

CCNinfo facilitates the tracing of a routing path and provides: 1) the RTT between the content forwarder (i.e., the caching or first-hop router) and consumer, 2) the states of the in-network cache per name prefix, and 3) the routing path information per name prefix.

In addition, CCNinfo identifies the states of the cache, such as the following metrics for Content Store (CS) in the content forwarder: 1) size of cached content objects, 2) number of cached content objects, 3) number of accesses (i.e., received Interests) per content, and 4) elapsed cache time and remaining cache lifetime of content.

```
           1. Request    2. Request    3. Request
              (+U)          (U+A)         (U+A+B)
             +----+        +----+        +----+
             |    |        |    |        |    |
             |    v        |    v        |    v
    +--------+    +--------+    +--------+    +--------+    +---------+
    | CCNinfo|----| Router |----| Router |----| Router |----|Publisher|
    |  user  |    |   A    |    |   B    |    |   C    |    |         |
    +--------+    +--------+    +--------+    +--------+    +---------+
                                         \
                                          \          +-------+
                                3. Request \         | Cache |
                                   (U+A+B)  \ +---------+     |
                                             v| Caching |----+
                                              |  router |
                                              +---------+
```

Figure 1: Request messages forwarded by consumer and routers. CCNinfo user and routers (i.e., Router A, B, C) insert their own Report blocks into the Request message and forward the message toward the content forwarder (i.e., caching router and publisher).

```
           3. Reply(P)   2. Reply(P)   1. Reply(P)
             +----+        +----+        +----+
             |    |        |    |        |    |
             v    |        v    |        v    |
    +--------+    +--------+    +--------+    +--------+    +---------+
    | CCNinfo|----| Router |----| Router |----| Router |----|Publisher|
    |  user  |    |   A    |    |   B    |    |   C    |    |         |
    +--------+    +--------+    +--------+    +--------+    +---------+
                                         ^
                                          \          +-------+
                               1. Reply(C) \         | Cache |
                                            \ +---------+   |
                                             \| Caching |----+
                                              |  router |
                                              +---------+
```

Figure 2: Default behavior. Reply messages forwarded by routers. Each router forwards the Reply message along its PIT entry and finally, the CCNinfo user receives a Reply message from Router C, which is the first-hop router for the Publisher. Another Reply message from the Caching router (i.e., Reply(C)) is discarded at Router B when the other Reply message (i.e., Reply(P)) was already forwarded by Router B.

CCNinfo supports multipath forwarding. The Request messages can be forwarded to multiple neighbor routers. When the Request messages are forwarded to multiple routers, the different Reply messages are forwarded from different routers or publishers.

Within a network with multipath condition, there is a case (Figure 3) wherein a single CCNinfo Request is split into multiple Requests (e.g., at Router A), which are injected into a single router (Router D). In this case, multiple Replies with the same Request ID and Node Identifier including different Report blocks are received by the router (Router D). To recognize different CCNinfo Reply messages, the routers MUST distinguish the PIT entries by the Request ID and exploiting path labels, which could be a hash value of the concatenation information of the cumulate Node Identifiers in the hop-by-hop header and the specified content name. For example, when Router D in Figure 3 receives a CCNinfo Request from Router B, its PIT includes the Request ID and value such as H((Router_A|Router_B)|content_name), where "H" indicates some hash function and "|" indicates concatenation. When Router D receives a CCNinfo Request from Router C, its PIT includes the same Request ID and value of H((Router_A|Router_C)|content_name). Two different Replies are later received on Router D and each Reply is appropriately forwarded to Router B and Router C, respectively. Note that two Reply messages coming from Router B and Router C are reached at Router A, but the CCNinfo user can only receive the first Reply message either from Router B or Router C as Router A removes the corresponding PIT entry after it forwards the first Reply.

```
                          +--------+
                          | Router |
                          |   B    |
                          +--------+
                         /          \
                        /            \
  +--------+    +--------+            +--------+     +---------+
  | CCNinfo|----| Router |            | Router | ... |Publisher|
  |  user  |    |   A    |            |   D    |     |         |
  +--------+    +--------+            +--------+     +---------+
                        \            /
                         \          /
                          +--------+
                          | Router |
                          |   C    |
                          +--------+
```

Figure 3

To avoid routing loop, when a router seeks the cumulate Node Identifiers of the Report blocks in the hop-by-hop header, it MUST examine whether its own Node Identifier is not previously inserted. If a router detects its own Node Identifier in the hop-by-hop header, the router inserts its Report block and terminates the Request as will be described in Section 6.8.

Furthermore, CCNinfo implements policy-based information provisioning that enables administrators to "hide" secure or private information but does not disrupt message forwarding. This policy-based information provisioning reduces the deployment barrier faced by operators in installing and running CCNinfo on their routers.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [3]; they indicate the requirement levels for the compliant CCNinfo implementations.

2.1. Definitions

This document follows the basic terminologies and definitions described in [1]. Although CCNinfo requests flow in the opposite direction to the data flow, we refer to "upstream" and "downstream" with respect to data, unless explicitly specified.

Router
: It is a router that facilitates CCN-based content retrieval in the path between the consumer and publisher.

Scheme name
: It indicates a URI and protocol. This document only considers "ccnx:/" as the scheme name.

Prefix name
: A prefix name, which is defined in [2], is a name that does not uniquely identify a single content object, but rather a namespace or prefix of an existing content object name.

Exact name
: An exact name, which is defined in [2], is one that uniquely identifies the name of a content object.

Node
: It is a router, publisher, or consumer.

Content forwarder
: It is either a caching router or a first-hop router that forwards content objects to consumers.

CCNinfo user
: It is a node that initiates the CCNinfo Request, which is usually invoked by the user program (described in Appendix A) or other similar commands.

Incoming face
: The face on which data are expected to arrive from the specified name prefix.

Outgoing face
: The face to which data from the publisher or router are expected to transmit for the specified name prefix. It is also the face on which the Request messages are received.

Upstream router
: The router that connects to an Incoming face of a router.

Downstream router
The router that connects to an Outgoing face of a router.

First-hop router (FHR)
The router that matches a FIB entry with an Outgoing face referring to a local application or a publisher.

Last-hop router (LHR)
The router that is directly connected to a consumer.

3. CCNinfo Message Formats

CCNinfo uses two message types: Request and Reply. Both messages are encoded in the CCNx TLV format ([1], Figure 4). The Request message consists of a fixed header, Request block TLV (Figure 8), and Report block TLV(s) (Figure 13). The Reply message consists of a fixed header, Request block TLV, Report block TLV(s), and Reply block/sub-block TLV(s) (Figure 15).

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|    Version    |  PacketType   |         PacketLength          |
+---------------+---------------+---------------+---------------+
|           PacketType specific fields          | HeaderLength  |
+---------------+---------------+---------------+---------------+
/ Optional Hop-by-hop header TLVs                               /
+---------------+---------------+---------------+---------------+
/ PacketPayload TLVs                                            /
+---------------+---------------+---------------+---------------+
/ Optional CCNx ValidationAlgorithm TLV                         /
+---------------+---------------+---------------+---------------+
/ Optional CCNx ValidationPayload TLV (ValidationAlg required)  /
+---------------+---------------+---------------+---------------+
```

Figure 4: Packet format [1]

The Request and Reply Type values in the fixed header are PT_CCNINFO_REQUEST and PT_CCNINFO_REPLY, respectively (Figure 5). These messages are forwarded in a hop-by-hop manner. When the Request message reaches the content forwarder, the content forwarder turns it into a Reply message by changing the Type field value in the fixed header from PT_CCNINFO_REQUEST to PT_CCNINFO_REPLY and forwards it back toward the node that initiated the Request message.

Code	Type name
%x00	PT_INTEREST [1]
%x01	PT_CONTENT [1]
%x02	PT_RETURN [1]
%x03	PT_CCNINFO_REQUEST
%x04	PT_CCNINFO_REPLY

Figure 5: Packet Type Namespace

The CCNinfo Request and Reply messages MUST begin with a fixed header with either a Request or Reply type value to specify whether it is a Request message or Reply message. Following a fixed header, there can be a sequence of optional hop-by-hop header TLV(s) for a Request message. In the case of a Request message, it is followed by a sequence of Report blocks, each from a router on the path toward the publisher or caching router.

At the beginning of PacketPayload TLVs, a top-level TLV type, T_DISCOVERY (Figure 6), exists at the outermost level of a CCNx protocol message. This TLV indicates that the Name segment TLV(s) and Reply block TLV(s) would follow in the Request or Reply message.

Code	Type name
%x0000	Reserved [1]
%x0001	T_INTEREST [1]
%x0002	T_OBJECT [1]
%x0003	T_VALIDATION_ALG [1]
%x0004	T_VALIDATION_PAYLOAD [1]
%x0005	T_DISCOVERY

Figure 6: Top-Level Type Namespace

3.1. Request Message

When a CCNinfo user initiates a discovery request (e.g., via the ccninfo command described in Appendix A), a CCNinfo Request message is created and forwarded to its upstream router through the Incoming face(s) determined by its FIB.

The Request message format is shown in Figure 7. It consists of a fixed header, Request header block TLV (Figure 8), Report block TLV(s) (Figure 13), Name TLV, and Request block TLV. The Type value of the Top-Level type namespace is T_DISCOVERY (Figure 6). The Type value for the Report message is PT_CCNINFO_REQUEST.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------------------------------+
|    Version    |  PacketType   |         PacketLength          |
+---------------+---------------+---------------+---------------+
|    HopLimit   |   ReturnCode  | Reserved(MBZ) | HeaderLength  |
+===============+===============+===============+===============+
/                    Request header block TLV                   /
+---------------+---------------+---------------+---------------+
/                      Report block TLV 1                       /
+---------------+---------------+---------------+---------------+
/                      Report block TLV 2                       /
+---------------+---------------+---------------+---------------+
/                               .                               /
/                               .                               /
+---------------+---------------+---------------+---------------+
/                      Report block TLV n                       /
+===============+===============+===============+===============+
|      Type (=T_DISCOVERY)      |         MessageLength         |
+---------------+---------------+---------------+---------------+
|             T_NAME            |             Length            |
+---------------+---------------+---------------+---------------+
/   Name segment TLVs (name prefix specified by CCNinfo user)   /
+---------------+---------------+---------------+---------------+
/                       Request block TLV                       /
+---------------+---------------+---------------+---------------+
/ Optional CCNx ValidationAlgorithm TLV                         /
+---------------+---------------+---------------+---------------+
/ Optional CCNx ValidationPayload TLV (ValidationAlg required)  /
+---------------+---------------+---------------+---------------+
```

Figure 7: Request message consists of a fixed header, Request block TLV, Report block TLV(s), and Name TLV

HopLimit: 8 bits

HopLimit is a counter that is decremented with each hop whenever a Request packet is forwarded. It is specified by the CCNinfo user program. It limits the distance that a Request may travel on the network. Only the specified number of hops from the CCNinfo user traces the Request. Each router inserts its own Report block and forwards the Request message to the upstream router(s). The last router stops the trace and sends the Reply message back to the CCNinfo user.

ReturnCode: 8 bits

ReturnCode is used for the Reply message. This value is replaced by the content forwarder when the Request message is returned as the Reply message (see Section 3.2). Until then, this field MUST be transmitted as zeros and ignored on receipt.

Value	Name	Description
%x00	NO_ERROR	No error
%x01	WRONG_IF	CCNinfo Request arrived on an interface to which this router would not forward for the specified name/function toward the publisher.
%x02	INVALID_REQUEST	Invalid CCNinfo Request is received.
%x03	NO_ROUTE	This router has no route for the name prefix and no way to determine a route.
%x04	NO_INFO	This router has no cache information for the specified name prefix.
%x05	NO_SPACE	There was not enough room to insert another Report block in the packet.
%x06	INFO_HIDDEN	Information is hidden from this discovery owing to some policy.
%x0E	ADMIN_PROHIB	CCNinfo Request is administratively prohibited.
%x0F	UNKNOWN_REQUEST	This router does not support/recognize the Request message.
%x80	FATAL_ERROR	In a fatal error, the router may know the upstream router but cannot forward the message to it.

Reserved (MBZ): 8 bits

The reserved fields in the Value field MUST be transmitted as zeros and ignored on receipt.

3.1.1. Request Header Block and Request Block

When a CCNinfo user transmits the Request message, s/he MUST insert her/his Request header block TLV (Figure 8) into the hop-by-hop header and Request block TLV (Figure 8) into the message before sending it through the Incoming face(s).

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|     Type (=T_DISC_REQHDR)     |             Length            |
+---------------+---------------+-------+-------+-------+-+-+-+-+
|           Request ID          |SkipHop|      Flags    |V|F|O|C|
+---------------+---------------+-------+-------+-------+-+-+-+-+
```

Figure 8: Request header block TLV (hop-by-hop header)

Code	Type name
%x0000	Reserved [1]
%x0001	T_INTLIFE [1]
%x0002	T_CACHETIME [1]
%x0003	T_MSGHASH [1]
%x0004-%x0007	Reserved [1]
%x0008	T_DISC_REQHDR
%x0009	T_DISC_REPORT
%x0FFE	T_PAD [1]
%x0FFF	T_ORG [1]
%x1000-%x1FFF	Reserved [1]

Figure 9: Hop-by-Hop Type Namespace

Type: 16 bits

Format of the Value field. For the type value of the Request block TLV MUST be T_DISC_REQHDR.

Length: 16 bits

Length of Value field in octets.

Request ID: 16 bits

This field is used as a unique identifier for the CCNinfo Request so that the duplicate or delayed Reply messages can be detected.

SkipHop (Skip Hop Count): 4 bits

Number of skipped routers for a Request. It is specified by the CCNinfo user program. Routers corresponding to the value specified in this field are skipped and the CCNinfo Request messages are forwarded to the next router without the addition of Report blocks; the next upstream router then starts the trace. The maximum value of this parameter is 15. This value MUST be lower than that of HopLimit at the fixed header.

Flags: 12 bits

The Flags field is used to indicate the types of the content or path discoveries. Currently, as shown in Figure 10, four bits, "C", "O", "F", and "V" are assigned, and the other 8 bits are reserved (MBZ) for the future use. Each flag can be mutually specified with other flags. These flags are set by the CCNinfo user program when they initiate Requests (see Appendix A), and the routers that receive the Requests deal with the flags and change the behaviors (see Section 5 for details). The Flag values defined in this Flags field correspond to the Reply sub-blocks.

Flag	Value	Description
C	0	Path discovery (i.e., no cache information retrieved) (default)
C	1	Cache information retrieval
O	0	Request to any content forwarder (default)
O	1	Publisher discovery (i.e., only FHR can reply)
F	0	Request based on FIB's strategy (default)
F	1	Full discovery request. Request to possible multiple upstream routers specified in FIB simultaneously
V	0	No reply validation (default)
V	1	Reply sender validates Reply message

Figure 10: Codes and types specified in Flags field

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|       Type (=T_DISC_REQ)      |             Length            |
+---------------+---------------+---------------+---------------+
|                      Request Arrival Time                     |
+---------------+---------------+---------------+---------------+
/                         Node Identifier                       /
+---------------+---------------+---------------+---------------+
```

Figure 11: Request block TLV (packet payload)

Code	Type name
%x0000	T_NAME [1]
%x0001	T_PAYLOAD [1]
%x0002	T_KEYIDRESTR [1]
%x0003	T_OBJHASHRESTR [1]
%x0005	T_PAYLDTYPE [1]
%x0006	T_EXPIRY [1]
%x0007	T_DISC_REQ
%x0008	T_DISC_REPLY
%x0009-%x0012	Reserved [1]
%x0FFE	T_PAD [1]
%x0FFF	T_ORG [1]
%x1000-%x1FFF	Reserved [1]

Figure 12: CCNx Message Type Namespace

Type: 16 bits

Format of the Value field. For the Request block TLV, the type value(s) MUST be T_DISC_REQ (see Figure 12) in the current specification.

Length: 16 bits

Length of Value field in octets.

Request Arrival Time: 32 bits

The Request Arrival Time is a 32-bit NTP timestamp specifying the arrival time of the CCNinfo Request packet at a specific router. The 32-bit form of an NTP timestamp consists of the middle 32 bits of the full 64-bit form; that is, the low 16 bits of the integer part and the high 16 bits of the fractional part.

The following formula converts from a UNIX timeval to a 32-bit NTP timestamp:

```
request_arrival_time
= ((tv.tv_sec + 32384) << 16) + ((tv.tv_nsec << 7) / 1953125)
```

The constant 32384 is the number of seconds from Jan 1, 1900 to Jan 1, 1970 truncated to 16 bits. ((tv.tv_nsec << 7) / 1953125) is a reduction of ((tv.tv_nsec / 1000000000) << 16).

Note that it is RECOMMENDED for all the routers on the path to have synchronized clocks to measure one-way latency per hop;

however, even if they do not have synchronized clocks, CCNinfo measures the RTT between the content forwarder and consumer.

Node Identifier: variable length

This field specifies the node identifier (e.g., node name or hash-based self-certifying name [10]) or all-zeros if unknown. This document assumes that the Name TLV defined in the CCNx TLV format [1] can be used for this field and the node identifier is specified in it.

3.1.2. Report Block TLV

A CCNinfo user and each upstream router along the path would insert their own Report block TLV without changing the Type field of the fixed header of the Request message until one of these routers is ready to send a Reply. In the Report block TLV (Figure 13), the Request Arrival Time and Node Identifier MUST be inserted.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|     Type (=T_DISC_REPORT)     |             Length            |
+---------------+---------------+---------------+---------------+
|                      Request Arrival Time                     |
+---------------+---------------+---------------+---------------+
/                         Node Identifier                       /
+---------------+---------------+---------------+---------------+
```

Figure 13: Report block TLV (hop-by-hop header)

Type: 16 bits

Format of the Value field. For the Report block TLV, the type value(s) MUST be T_DISC_REPORT in the current specification. For all the available types for hop-by-hop type namespace, please see Figure 9.

Length: 16 bits

Length of Value field in octets.

Request Arrival Time: 32 bits

Same definition as given in Section 3.1.1.

Node Identifier: variable length

Same definition as given in Section 3.1.1.

3.1.3. Content Name Specification

Specifications of the Name TLV (whose type value is T_NAME) and the Name Segment TLVs are described in [1], which are followed by CCNinfo. CCNinfo enables to specification of the content name either with a prefix name without chunk number (such as "ccnx:/news/today") or an exact name (such as "ccnx:/news/today/Chunk=10"). When a CCNinfo user specifies a prefix name, s/he will obtain the summary information of the matched content objects in the content forwarder. In contrast, when a CCNinfo user specifies an exact name, s/he will obtain only about the specified content object in the content forwarder. A CCNinfo Request message MUST NOT be sent only with a scheme name, ccnx:/. It will be rejected and discarded by routers.

3.2. Reply Message

When a content forwarder receives a CCNinfo Request message from an appropriate adjacent neighbor router, it inserts its own Reply block TLV and Reply sub-block TLV(s) to the Request message and turns the Request into the Reply by changing the Type field of the fixed header of the Request message from PT_CCNINFO_REQUEST to PT_CCNINFO_REPLY. The Reply message (see Figure 14) is then forwarded back toward the CCNinfo user in a hop-by-hop manner.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +---------------+---------------+---------------+---------------+
   |    Version    |  PacketType   |          PacketLength         |
   +---------------+---------------+-------------+-+---------------+
   |    HopLimit   |  ReturnCode   |Reserved(MBZ)| HeaderLength    |
   +===============+===============+=============+=+===============+
   /                  Request header block TLV                     /
   +---------------+---------------+---------------+---------------+
   /                               .                               /
   /                               .                               /
   /                      n Report block TLVs                      /
   /                               .                               /
   /                               .                               /
   +===============+===============+===============+===============+
   |      Type (=T_DISCOVERY)      |         MessageLength         |
   +---------------+---------------+---------------+---------------+
   |            T_NAME             |             Length            |
   +---------------+---------------+---------------+---------------+
   /   Name segment TLVs (name prefix specified by CCNinfo user)   /
   +---------------+---------------+---------------+---------------+
   /                       Request block TLV                       /
   +---------------+---------------+---------------+---------------+
   /                        Reply block TLV                        /
   +---------------+---------------+---------------+---------------+
   /                     Reply sub-block TLV 1                     /
   +---------------+---------------+---------------+---------------+
   /                               .                               /
   /                               .                               /
   +---------------+---------------+---------------+---------------+
   /                     Reply sub-block TLV k                     /
   +---------------+---------------+---------------+---------------+
   / Optional CCNx ValidationAlgorithm TLV                         /
   +---------------+---------------+---------------+---------------+
   / Optional CCNx ValidationPayload TLV (ValidationAlg required)  /
   +---------------+---------------+---------------+---------------+
```

Figure 14: Reply message consists of a fixed header, Request block TLV, Report block TLV(s), Name TLV, and Reply block/sub-block TLV(s)

3.2.1. Reply Block TLV

The Reply block TLV is an envelope for the Reply sub-block TLV(s) (explained from the next section).

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|      Type (=T_DISC_REPLY)     |             Length            |
+---------------+---------------+---------------+---------------+
|                     Request Arrival Time                      |
+---------------+---------------+---------------+---------------+
/                        Node Identifier                        /
+---------------+---------------+---------------+---------------+
```

Figure 15: Reply block TLV (packet payload)

Type: 16 bits

Format of the Value field. For the Reply block TLV, the type value MUST be T_DISC_REPLY shown in Figure 12 in the current specification.

Length: 16 bits

Length of the Value field in octets. This length is the total length of Reply sub-block(s).

Request Arrival Time: 32 bits

Same definition as given in Section 3.1.1.

Node Identifier: variable length

Same definition as given in Section 3.1.1.

3.2.1.1. Reply Sub-Block TLV

The router on the traced path will add one or multiple Reply sub-blocks followed by the Reply block TLV before sending the Reply to its neighbor router. This section describes the Reply sub-block TLV for informing various cache states and conditions as shown in Figure 16. (Other Reply sub-block TLVs will be discussed in separate document(s).)

Note that some routers may not be capable of reporting the following values, such as Object Size, Object Count, # Received Interest, First Seqnum, Last Seqnum, Elapsed Cache Time, and Remain Cache Lifetime, shown in Figure 16, or do not report these values due to their policy. These values therefore MAY be returned with null. Also, the value of each field will be all-one when the value is equal to or bigger than the maximum size expressed by the 32-bit field.

If the cache is refreshed after reboot, the value in each field MUST be refreshed (i.e., MUST be set to 0). If the cache remains after reboot, the value MUST NOT be refreshed (i.e., MUST be reflected as it is).

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|             Type              |             Length            |
+---------------+---------------+---------------+---------------+
|                          Object Size                          |
+---------------+---------------+---------------+---------------+
|                          Object Count                         |
+---------------+---------------+---------------+---------------+
|                       # Received Interest                     |
+---------------+---------------+---------------+---------------+
|                          First Seqnum                         |
+---------------+---------------+---------------+---------------+
|                          Last Seqnum                          |
+---------------+---------------+---------------+---------------+
|                       Elapsed Cache Time                      |
+---------------+---------------+---------------+---------------+
|                     Remain Cache Lifetime                     |
+---------------+---------------+---------------+---------------+
|            T_NAME             |             Length            |
+---------------+---------------+---------------+---------------+
/                       Name Segment TLVs                       /
+---------------+---------------+---------------+---------------+
```

Figure 16: Reply sub-block TLV for T_DISC_CONTENT and T_DISC_CONTENT_PUBLISHER (packet payload)

Code	Type name
%x0000	T_DISC_CONTENT
%x0001	T_DISC_CONTENT_PUBLISHER
%x0FFF	T_ORG
%x1000-%x1FFF	Reserved (Experimental Use)

Figure 17: CCNinfo Reply Type Namespace

Type: 16 bits

Format of the Value field. For the Reply sub-block TLV, the type value MUST be either T_DISC_CONTENT or T_DISC_CONTENT_PUBLISHER defined in the CCNinfo Reply Type Namespace (Figure 17). T_DISC_CONTENT is specified when the cache information is replied from a caching router. T_DISC_CONTENT_PUBLISHER is specified when the content information is replied from a FHR attached to a publisher.

Length: 16 bits

Length of the Value field in octets.

Object Size: 32 bits

The total size (KB) of the unexpired content objects. Note that the maximum size expressed by the 32-bit field is approximately 4.29 TB.

Object Count: 32 bits

The number of the unexpired content objects. Note that the maximum count expressed by the 32-bit field is approximately 4.29 billion.

Received Interest: 32 bits

The total number of the received Interest messages to retrieve the cached content objects.

First Seqnum: 32 bits

The first sequential number of the unexpired content objects.

Last Seqnum: 32 bits

The last sequential number of the unexpired content objects. The First Seqnum and Last Seqnum do not guarantee the consecutiveness of the cached content objects; however, knowing these values may help in the analysis of consecutive or discontinuous chunks such as [11].

Elapsed Cache Time: 32 bits

The elapsed time (seconds) after the oldest content object of the content is cached.

Remain Cache Lifetime: 32 bits

The lifetime (seconds) of a content object, which is lastly cached.

4. CCNinfo User Behavior

4.1. Sending CCNinfo Request

A CCNinfo user invokes a CCNinfo user program (e.g., ccninfo command) that initiates a CCNinfo Request message and sends it to the user's adjacent neighbor router(s) of interest. The user later obtains both the routing path information and in-network cache information in the single Reply.

When the CCNinfo user program initiates a Request message, it MUST insert the necessary values, i.e., the "Request ID" and the "Node Identifier", in the Request block. The Request ID MUST be unique for the CCNinfo user until s/he receives the corresponding Reply message(s) or the Request is timed out.

Owing to some policies, a router may want to validate the CCNinfo Requests using the CCNx ValidationPayload TLV (whether it accepts the Request or not) especially when the router receives the "full discovery request" (see Section 5.3.2). Accordingly, the CCNinfo user program MAY require validating the Request message and appending the user's signature into the CCNx ValidationPayload TLV. The router then forwards the Request message. If the router does not approve the Request, it rejects the Request message as described in Section 6.11.

After the CCNinfo user program sends the Request message, until the Reply is timed out or the expected numbers of Replies or a Reply message with a non-zero ReturnCode in the fixed header is received, the CCNinfo user program MUST keep the following information: HopLimit, specified in the fixed header, Request ID, Flags, Node Identifier, and Request Arrival Time, specified in the Request block.

4.1.1. Routing Path Information

A CCNinfo user can send a CCNinfo Request for investigating the routing path information for the specified named content. Using the Request, a legitimate user can obtain 1) the node identifiers of the intermediate routers, 2) node identifier of the content forwarder, 3) number of hops between the content forwarder and consumer, and 4) RTT between the content forwarder and consumer, per name prefix. This CCNinfo Request is terminated when it reaches the content forwarder.

4.1.2. In-Network Cache Information

A CCNinfo user can send a CCNinfo Request for investigating in-network cache information. Using the Request, a legitimate user can obtain 1) the size of cached content objects, 2) number of cached content objects, 3) number of accesses (i.e., received Interests) per content, and 4) lifetime and expiration time of the cached content objects, for Content Store (CS) in the content forwarder, unless the content forwarder is capable of reporting them (see Section 3.2.1.1). This CCNinfo Request is terminated when it reaches the content forwarder.

4.2. Receiving CCNinfo Reply

A CCNinfo user program will receive one or multiple CCNinfo Reply messages from the adjacent neighbor router(s). When the program receives the Reply, it MUST compare the kept Request ID and Node Identifier to identify the Request and Reply pair. If they do not match, the Reply message MUST be silently discarded.

If the number of Report blocks in the received Reply is more than the initial HopLimit value (which was inserted in the original Request), the Reply MUST be silently ignored.

After the CCNinfo user has determined that s/he has traced the whole path or the maximum path that s/he can be expected to, s/he might collect statistics by waiting for a timeout. Useful statistics provided by CCNinfo are stated in Section 8.

5. Router Behavior

5.1. User and Neighbor Verification

Upon receiving a CCNinfo Request message, a router MAY examine whether a valid CCNinfo user has sent the message. If the router recognizes that the Request sender's signature specified in the Request is invalid, it SHOULD terminate the Request, as defined in Section 6.4.

Upon receiving a CCNinfo Request/Reply message, a router MAY examine whether the message comes from a valid adjacent neighbor node. If the router recognizes that the Request/Reply sender is invalid, it SHOULD silently ignore the Request/Reply message, as specified in Section 10.9.

5.2. Receiving CCNinfo Request

After a router accepts the CCNinfo Request message, it performs the following steps.

1. The value of "HopLimit" in the fixed header and that of "SkipHop (Skip Hop Count)" in the Request block are counters that are decremented with each hop. If the HopLimit value is zero, the

router terminates the Request, as defined in Section 6.5. If the SkipHop value is equal to or more than the HopLimit value, the router terminates the Request, as defined in Section 6.4. Otherwise, until the SkipHop value becomes zero, the router forwards the Request message to the upstream router(s) without adding its own Report block and without replying to the Request. If the router does not know the upstream router(s) regarding the specified name prefix, it terminates the Request, as defined in Section 6.5. It should be noted that the Request messages are terminated at the FHR; therefore, although the maximum value for the HopLimit is 255 and that for SkipHop is 15, if the Request messages reach the FHR before the HopLimit or SkipHop value becomes 0, the FHR silently discards the Request message and the Request is timed out.

2. The router examines the Flags field (specified in Figure 10) in the Request block of the received CCNinfo Request. If the "C" flag is not set, it is categorized as the "routing path information discovery". If the "C" flag is set, it is the "cache information discovery". If the "O" flag is set, it is the "publisher discovery".

3. If the Request is either "cache information discovery" or "routing path information discovery", the router examines its FIB and CS. If the router caches the specified content, it sends the Reply message with its own Reply block and sub-block(s). If the router cannot insert its own Reply block or sub-block(s) because of no space, it it terminates the Request, as specified in Section 6.7. If the router does not cache the specified content but knows the upstream neighbor router(s) for the specified name prefix, it creates the PIT entry, and inserts its own Report block in the hop-by-hop header and forwards the Request to the upstream neighbor(s). If the router cannot insert its own Report block because of no space, or if the router does not cache the specified content and does not know the upstream neighbor router(s) for the specified name prefix, it terminates the Request, as defined in Section 6.5.

4. If the Request is the "publisher discovery", the router examines whether it is the FHR for the requested content. If the router is the FHR, it sends the Reply message with its own Report block and sub-blocks (in the case of cache information discovery) or the Reply message with its own Report block without adding any Reply sub-blocks (in the case of routing path information discovery). If the router is not the FHR but knows the upstream neighbor router(s) for the specified name prefix, it creates the PIT entry, and inserts its own Report block and forwards the Request to the upstream neighbor(s). If the router cannot insert

its own Report block in the hop-by-hop header because of no space, it it terminates the Request, as specified in Section 6.7. If the router is not the FHR and does not know the upstream neighbor router(s) for the specified name prefix, it terminates the Request, as defined in Section 6.5. Note that in Cefore [13], there is an API by which a publisher informs the application prefix to the FHR and the FHR registers it into the FIB. The prefix entry then can be statically configured on other routers or announced by a routing protocol.

5.3. Forwarding CCNinfo Request

5.3.1. Regular Request

When a router decides to forward a Request message with its Report block to its upstream router(s), it specifies the Request Arrival Time and Node Identifier in the Report block of the Request message. The router then forwards the Request message upstream toward the publisher or caching router based on the FIB entry like the ordinary Interest-Data exchanges in CCN.

When the router forwards the Request message, it MUST record the F flag and Request ID in the Request block of the Request message and exploiting path labels (specified in Section 1) at the corresponding PIT entry. The router can later check the PIT entry to correctly forward the Reply message(s) back.

CCNinfo supports multipath forwarding. The Request messages can be forwarded to multiple neighbor routers. Some routers may have a strategy for multipath forwarding; when a router sends Interest messages to multiple neighbor routers, it may delay or prioritize to send the message to the upstream routers. The CCNinfo Request, as the default, complies with such strategies; a CCNinfo user could trace the actual forwarding path based on the forwarding strategy and will receive a single Reply message such as a content object.

5.3.2. Full Discovery Request

There may be a case wherein a CCNinfo user wants to discover all possible forwarding paths and content forwarders based on the routers' FIBs. The "full discovery request" enables this functionality. If a CCNinfo user sets the F flag in the Request block of the Request message (as seen in Figure 10) to request the full discovery, the upstream routers simultaneously forward the Requests to all multiple upstream routers based on the FIBs. Then, the CCNinfo user can trace all possible forwarding paths.

```
           3. Reply(C)   2. Reply(C)
           3. Reply(P)   2. Reply(P)   1. Reply(P)
             +----+        +----+        +----+
             |    |        |    |        |    |
             v    |        v    |        v    |
    +--------+    +--------+    +--------+    +--------+    +---------+
    | CCNinfo|----| Router |----| Router |----| Router |----|Publisher|
    |  user  |    |   A    |    |   B    |    |   C    |    |         |
    +--------+    +--------+    +--------+    +--------+    +---------+
                                         ^
                                          \          +-------+
                               1. Reply(C) \         | Cache |
                                            \ +---------+    |
                                             \| Caching |----+
                                              |  router |
                                              +---------+
```

Figure 18: Full discovery request. Reply messages forwarded by publisher and routers. Each router forwards the Reply message along its PIT entry and finally, the CCNinfo user receives two Reply messages: one from the FHR (Router C) and the other from the Caching router.

To receive different Reply messages forwarded from different routers, the PIT entries initiated by CCNinfo remain until the configured CCNinfo Reply Timeout (Section 7.1) is expired. In other words, unlike the ordinary Interest-Data exchanges in CCN, if routers that accept the full discovery request receive the full discovery request, the routers SHOULD NOT remove the PIT entry created by the full discovery request until the CCNinfo Reply Timeout value expires.

Note that the full discovery request is an OPTIONAL implementation of CCNinfo; it MAY NOT be implemented on routers. Even if it is implemented on a router, it MAY NOT accept the full discovery request from non-validated CCNinfo users or routers or because of its policy. If a router does not accept the full discovery request, it rejects the full discovery request as described in Section 6.11. Routers that enable the full discovery request MAY rate-limit Replies, as described in Section 10.8 as well.

5.4. Sending CCNinfo Reply

If there is a caching router or FHR for the specified content within the specified hop count along the path, the caching router or FHR sends back the Reply message toward the CCNinfo user and terminates the Request.

When a router decides to send a Reply message to its downstream neighbor router or the CCNinfo user with NO_ERROR return code, it inserts a Report block with the Request Arrival Time and Node Identifier to the Request message. Then, the router inserts the corresponding Reply sub-block(s) (Figure 16) to the payload. The router finally changes the Type field in the fixed header from PT_CCNINFO_REQUEST to PT_CCNINFO_REPLY and forwards the message back as the Reply toward the CCNinfo user in a hop-by-hop manner.

If a router cannot continue the Request, the router MUST put an appropriate ReturnCode in the Request message, change the Type field value in the fixed header from PT_CCNINFO_REQUEST to PT_CCNINFO_REPLY, and forward the Reply message back toward the CCNinfo user to terminate the Request (see Section 6).

5.5. Forwarding CCNinfo Reply

When a router receives a CCNinfo Reply whose Request ID and Node Identifier match those in the PIT entry, sent from a valid adjacent neighbor router, it forwards the CCNinfo Reply back toward the CCNinfo user. If the router does not receive the corresponding Reply within the [CCNinfo Reply Timeout] period, then it removes the corresponding PIT entry and terminates the trace.

The Flags field in the Request block TLV is used to indicate whether the router keeps the PIT entry during the CCNinfo Reply Timeout even after one or more corresponding Reply messages are forwarded. When the CCNinfo user does not set the F flag (i.e., "0"), the intermediate routers immediately remove the PIT entry whenever they forward the corresponding Reply message. When the CCNinfo user sets the F flag (i.e., "1"), which means the CCNinfo user chooses the "full discovery request" (see Section 5.3.2), the intermediate routers keep the PIT entry within the [CCNinfo Reply Timeout] period. After this timeout, the PIT entry is removed.

CCNinfo Replies MUST NOT be cached in routers upon the transmission of Reply messages.

6. CCNinfo Termination

When performing a hop-by-hop trace, it is necessary to determine when to stop the trace. There are several cases when an intermediate router might return a Reply before a Request reaches the caching router or the FHR.

6.1. Arriving at First-hop Router

A CCNinfo Request can be determined to have arrived at the FHR. To ensure that a router recognizes that it is the FHR for the specified content, it needs to have a FIB entry (or attach) to the corresponding publisher or the content.

6.2. Arriving at Router Having Cache

A CCNinfo Request can be determined to have arrived at the router having the specified content cache within the specified HopLimit.

6.3. Arriving at Last Router

A CCNinfo Request can be determined to have arrived at the last router of the specified HopLimit. If the last router does not have the corresponding cache, it MUST insert its Report block and send the Reply message with NO_INFO return code without appending any Reply (sub-)block TLVs.

6.4. Invalid Request

If the router does not validate the Request or the Reply, the router MUST note a ReturnCode of INVALID_REQUEST in the fixed header of the message, insert its Report block, and forward the message as the Reply back to the CCNinfo user. The router MAY, however, randomly ignore the received invalid messages. (See Section 10.7.)

6.5. No Route

If the router cannot determine the routing paths or neighbor routers for the specified name prefix within the specified HopLimit, it MUST note a ReturnCode of NO_ROUTE in the fixed header of the message, insert its Report block, and forward the message as the Reply back to the CCNinfo user.

6.6. No Information

If the router does not have any information about the specified name prefix within the specified HopLimit, it MUST note a ReturnCode of NO_INFO in the fixed header of the message, insert its Report block, and forward the message as the Reply back to the CCNinfo user.

6.7. No Space

If appending the Report block or the Reply (sub-)block would make the hop-by-hop header longer than 247 bytes or the Request packet longer than the MTU of the Incoming face, the router MUST note a ReturnCode of NO_SPACE in the fixed header of the message and forward the message as the Reply back to the CCNinfo user.

6.8. Fatal Error

If a CCNinfo Request has encountered a fatal error, the router MUST note a ReturnCode of FATAL_ERROR in the fixed header of the message and forward the message as the Reply back to the CCNinfo user. This may happen, for example, when the router detects some routing loop in the Request blocks (see Section 1). The fatal error can be encoded with another error: if a router detects routing loop but cannot insert its Report block, it MUST note NO_SPACE and FATAL_ERROR ReturnCodes (i.e., %x85) in the fixed header and forward the message back to the CCNinfo user.

6.9. CCNinfo Reply Timeout

If a router receives the Request or Reply message that expires its own [CCNinfo Reply Timeout] value (Section 7.1), the router will silently discard the Request or Reply message.

6.10. Non-Supported Node

Cases will arise in which a router or a FHR along the path does not support CCNinfo. In such cases, a CCNinfo user and routers that forward the CCNinfo Request will time out the CCNinfo request.

6.11. Administratively Prohibited

If CCNinfo is administratively prohibited, the router rejects the Request message and MUST send the CCNinfo Reply with the ReturnCode of ADMIN_PROHIB. The router MAY, however, randomly ignore the Request messages to be rejected (see Section 10.7).

7. Configurations

7.1. CCNinfo Reply Timeout

The [CCNinfo Reply Timeout] value is used to time out a CCNinfo Reply. The value for a router can be statically configured by the router's administrators/operators. The default value is 3 (seconds). The [CCNinfo Reply Timeout] value SHOULD NOT be larger than 4 (seconds) and SHOULD NOT be lower than 2 (seconds).

7.2. HopLimit in Fixed Header

If a CCNinfo user does not specify the HopLimit value in the fixed header for a Request message as the HopLimit, the HopLimit is set to 32. Note that 0 HopLimit is an invalid Request; hence, the router in this case follows the way defined in Section 6.4.

7.3. Access Control

A router MAY configure the valid or invalid networks to enable an access control. The access control MAY be defined per name prefix, such as "who can retrieve which name prefix" (see Section 10.2).

8. Diagnosis and Analysis

8.1. Number of Hops and RTT

A CCNinfo Request message is forwarded in a hop-by-hop manner and each forwarding router appends its own Report block. We can then verify the number of hops to reach the content forwarder or publisher and the RTT between the content forwarder or publisher.

8.2. Caching Router Identification

While some routers may hide their node identifiers with all-zeros in the Report blocks (as seen in Section 10.1), the routers in the path from the CCNinfo user to the content forwarder can be identified.

8.3. TTL or Hop Limit

By taking the HopLimit from the content forwarder and forwarding the TTL threshold over all hops, it is possible to discover the TTL or hop limit required for the content forwarder to reach the CCNinfo user.

8.4. Time Delay

If the routers have synchronized clocks, it is possible to estimate the propagation and queuing delays from the differences between the timestamps at the successive hops. However, this delay includes the control processing overhead; therefore, it is not necessarily indicative of the delay that would be experienced by the data traffic.

8.5. Path Stretch

By obtaining the path stretch "d / P", where "d" is the hop count of the data and "P" is the hop count from the consumer to the publisher, we can measure the improvements in path stretch in various cases, such as in different caching and routing algorithms. We can then facilitate the investigation of the performance of the protocol.

8.6. Cache Hit Probability

CCNinfo can show the number of received interests per cache or chunk on a router. Accordingly, CCNinfo measures the content popularity (i.e., the number of accesses for each content/cache), thereby enabling the investigation of the routing/caching strategy in networks.

9. IANA Considerations

New assignments can only be made via a Standards Action as specified in [4]. This document does not intend to be the standard document. However, the new assignments such as the ReturnCode and various type values will be considered when this specification becomes the RFC.

10. Security Considerations

This section addresses some of the security considerations.

10.1. Policy-Based Information Provisioning for Request

Although CCNinfo gives excellent troubleshooting cues, some network administrators or operators may not want to disclose everything about their network to the public or may wish to securely transmit private information to specific members of their networks. CCNinfo provides policy-based information provisioning, thereby allowing network administrators to specify their response policy for each router.

The access policy regarding "who is allowed to retrieve" and/or "what kind of cache information" can be defined for each router. For the former type of access policy, routers with the specified content MAY examine the signature enclosed in the Request message and decide whether they should notify the content information in the Reply. If the routers decide to not notify the content information, they MUST send the CCNinfo Reply with the ReturnCode of ADMIN_PROHIB without appending any Reply (sub-)block TLVs. For the latter type of policy, the permission, whether (1) All (all cache information is disclosed), (2) Partial (cache information with a particular name prefix can (or cannot) be disclosed), or (3) Deny (no cache information is disclosed), is defined at the routers.

In contrast, we entail that each router does not disrupt the forwarding of CCNinfo Request and Reply messages. When a Request message is received, the router SHOULD insert the Report block if the ReturnCode is NO_ERROR. Here, according to the policy configuration, the Node Identifier field in the Report block MAY be null (i.e., all-zeros), but the Request Arrival Time field SHOULD NOT be null. Finally, the router SHOULD forward the Request message to the upstream router toward the content forwarder if the ReturnCode is kept with NO_ERROR.

10.2. Filtering CCNinfo Users Located in Invalid Networks

A router MAY support an access control mechanism to filter out Requests from invalid CCNinfo users. To accomplish this, invalid networks (or domains) could, for example, be configured via a list of allowed/disallowed networks (as observed in Section 7.3). If a Request is received from a disallowed network (according to the Node Identifier in the Request block), the Request MUST NOT be processed and the Reply with the ReturnCode of INFO_HIDDEN SHOULD be used to note that. The router MAY, however, perform rate-limited logging of such events.

10.3. Topology Discovery

CCNinfo can be used to discover actively used topologies. If a network topology is not disclosed, CCNinfo Requests SHOULD be restricted at the border of the domain using the ADMIN_PROHIB return code.

10.4. Characteristics of Content

CCNinfo can be used to discover the type of content being sent by publishers. If this information is a secret, CCNinfo Requests SHOULD be restricted at the border of the domain, using the ADMIN_PROHIB return code.

10.5. Computational Attacks

CCNinfo may impose heavy tasks at content forwarders. The current CCNinfo specification allows to return null values for several fields, such as First/Last Seqnum or Elapsed Cache Time fields in the Reply sub-block. As mentioned in Section 3.2.1.1, these values MAY be null. This means that the content forwarder can not only hide these values owing to privacy/security policies, but also skip the implementations of the complex functions to report these values.

10.6. Longer or Shorter CCNinfo Reply Timeout

Routers can configure CCNinfo Reply Timeout (Section 7.1), which is the allowable timeout value to keep the PIT entry. If routers configure a longer timeout value, there may be an attractive attack vector against the PIT memory. Moreover, especially when the full discovery request option (Section 5.3) is specified for the CCNinfo Request, several Reply messages may be returned and cause a response storm. (See Section 10.8 for rate limiting to avoid the storm). To avoid DoS attacks, routers MAY configure the timeout value, which is shorter than the user-configured CCNinfo timeout value. However, if it is too short, the Request may be timed out and the CCNinfo user does not receive all Replies; s/he only retrieves the partial path information (i.e., information about a part of the tree).

There may be a way to enable incremental exploration (i.e., to explore the part of the tree that was not explored by the previous operation); however, discussing such mechanisms is out of scope of this document.

10.7. Limiting Request Rates

A router may rate-limit CCNinfo Requests by ignoring some of the consecutive messages. The router MAY randomly ignore the received messages to minimize the processing overhead, i.e., to keep fairness in processing requests, or prevent traffic amplification. In such a case, no error message is returned. The rate limit function is left to the router’s implementation.

10.8. Limiting Reply Rates

CCNinfo supporting multipath forwarding may result in one Request returning multiple Reply messages. To prevent abuse, the routers in the traced path MAY need to rate-limit the Replies. In such a case, no error message is returned. The rate limit function is left to the router’s implementation.

10.9. Adjacency Verification

It is assumed that the CCNinfo Request and Reply messages are forwarded by adjacent neighbor nodes or routers. The CCNx message format or semantics do not define a secure way to verify the node/ router adjacency, while HopAuth [10] provides a possible method for an adjacency verification and defines the corresponding message format for adjacency verification as well as the router behaviors. CCNinfo MAY use a similar method for node adjacency verification.

11. Acknowledgements

The authors would like to thank Spyridon Mastorakis, Paulo Mendes, Ilya Moiseenko, David Oran, and Thierry Turletti for their valuable comments and suggestions on this document.

12. References

12.1. Normative References

[1] Mosko, M., Solis, I., and C. Wood, "CCNx Messages in TLV Format", RFC 8609, July 2019.

[2] Mosko, M., Solis, I., and C. Wood, "CCNx Semantics", RFC 8569, July 2019.

[3] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[4] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

12.2. Informative References

[5] Malkin, G., "Traceroute Using an IP Option", RFC 1393, January 1993.

[6] Asaeda, H., Matsuzono, K., and T. Turletti, "Contrace: A Tool for Measuring and Tracing Content-Centric Networks", IEEE Communications Magazine, Vol.53, No.3, pp.182-188, March 2015.

[7] Mastorakis, S., Gibson, J., Moiseenko, I., Droms, R., and D. Oran, "ICN Ping Protocol Specification", draft-irtf-icnrg-icnping-00 (work in progress), March 2020.

[8] Mastorakis, S., Gibson, J., Moiseenko, I., Droms, R., and D. Oran, "ICN Traceroute Protocol Specification", draft-irtf-icnrg-icntraceroute-00 (work in progress), March 2020.

[9] Asaeda, H., Mayer, K., and W. Lee, "Mtrace Version 2: Traceroute Facility for IP Multicast", RFC 8487, October 2018.

[10] Li, R. and H. Asaeda, "Hop-by-Hop Authentication in Content-Centric Networking/Named Data Networking", draft-li-icnrg-hopauth-02 (work in progress), February 2020.

[11] Li, R., Matsuzono, K., Asaeda, H., and X. Fu, "Consecutive Caching and Adaptive Retrieval for In-Network Big Data Sharing", Proc. IEEE ICC, Kansas City, USA, May 2018.

[12] Asaeda, H., Ooka, A., Matsuzono, K., and R. Li, "Cefore: Software Platform Enabling Content-Centric Networking and Beyond", IEICE Transaction on Communications, Vol.E102-B, No.9, pp.1792-1803, September 2019.

[13] "Cefore Home Page", <https://cefore.net/>.

Appendix A. ccninfo Command and Options

CCNinfo is implemented in Cefore [12][13]. "ccninfo" is the command invoked by the CCNinfo user (e.g., consumer). The ccninfo command sends the Request message and receives the Reply message(s). There are several options that can be specified with ccninfo, while the content name prefix (e.g., ccnx:/news/today) is the mandatory parameter.

The usage of ccninfo command is as follows:

```
Usage: ccninfo [-c] [-f] [-o] [-V] [-r hop_count] [-s hop_count] [-v
       algo] name_prefix
```

name_prefix
   Prefix name of content (e.g., ccnx:/news/today) or exact name of content (e.g., ccnx:/news/today/Chunk=10) the CCNinfo user wants to trace.

c option
   This option can be specified if a CCNinfo user needs the cache information as well as the routing path information for the specified content/cache and RTT between the CCNinfo user and content forwarder.

f option
   This option enables the "full discovery request"; routers send CCNinfo Requests to multiple upstream faces based on their FIBs simultaneously. The CCNinfo user can then trace all possible forwarding paths.

o option

This option enables to trace the path to the content publisher. Each router along the path to the publisher inserts each Report block and forwards the Request message. It does not send Reply even if it caches the specified content. FHR that attaches the publisher (who has the complete set of content and is not a caching router) sends the Reply message.

V option
This option requests the Reply sender to validate the Reply message with the Reply sender's signature. The Reply message will then include the CCNx ValidationPayload TLV. The validation algorithm is selected by the Reply sender.

r option
Number of traced routers. This value is set in the "HopLimit" field located in the fixed header of the Request. For example, when the CCNinfo user invokes the CCNinfo command with this option, such as "-r 3", only three routers along the path examine their path and cache information.

s option
Number of skipped routers. This value is set in the "SkipHop" field located in the Request block TLV. For example, when the CCNinfo user invokes the CCNinfo command with this option, such as "-s 3", three upstream routers along the path only forward the Request message but do not append their Report blocks in the hop-by-hop header and do not send Reply messages despite having the corresponding cache.

v option
This option enables the CCNinfo user to validate the Request message with his/her signature. The Request message will include the CCNx ValidationPayload TLV. The validation algorithm is specified by the CCNinfo user.

Authors' Addresses

Hitoshi Asaeda
National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi
Koganei, Tokyo 184-8795
Japan

Email: asaeda@nict.go.jp

Atsushi Ooka
National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi
Koganei, Tokyo 184-8795
Japan

Email: a-ooka@nict.go.jp

Xun Shao
Kitami Institute of Technology
165 Koen-cho
Kitami, Hokkaido 090-8507
Japan

Email: x-shao@mail.kitami-it.ac.jp

ICN Research Group
Internet-Draft
Intended status: Informational
Expires: November 3, 2019

R. Ravindran, Ed.
Huawei Technologies
Y. Zhang
WINLAB, Rutgers University
L. Grieco
Politecnico di Bari (DEI)
A. Lindgren
RISE SICS
J. Burke
UCLA REMAP
B. Ahlgren
RISE SICS
A. Azgin
Huawei Technologies

May 2, 2019

Design Considerations for Applying ICN to IoT
draft-irtf-icnrg-icniot-03

Abstract

The Internet of Things (IoT) promises to connect billions of objects to the Internet. After deploying many stand-alone IoT systems in different domains, the current trend is to develop a common, "thin waist" of protocols to enable a horizontally unified IoT architecture. The objective of such an architecture is to make resource objects securely accessible to applications across organizations and domains. Towards this goal, quite a few proposals have been made to build an application-layer based unified IoT platform on top of today's host-centric Internet. However, there is a fundamental mismatch between the host-centric nature of today's Internet and the mostly information-centric nature of the IoT domain. To address this mismatch, the common set of protocols and network services offered by an information-centric networking (ICN) architecture can be leveraged to realize an ICN-based IoT (or ICN-IoT) architecture that can take advantage of the salient features of ICN such as naming, security, mobility, compute and efficient content and service delivery support offered by it.

In this draft, we summarize the general IoT demands, and ICN features that support these requirements, and then discuss the challenges to realize an ICN-based IoT framework. Beyond this, the goal of this draft is not to offer any specific ICN-IoT architectural proposal.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

During the past decade, many Internet of Things (IoT) systems have been developed, and deployed in different domains. The recent trend, however, is to evolve these systems towards a unified IoT architecture, in which a large number of objects hosted by non-interoperable protocol domains can connect to the Internet to enable secure interactions with a diverse set of applications across administrative domain. Note that, here, 'unified' is used to imply a scenario, where all the IoT applications, services, and network functions use a common set of transport APIs and network protocols to interact with each other. Typical IoT applications involve sensing, actuation, processing, and secure content distribution, each of which can occur at different timescales and hierarchical levels that depend on the application requirements. To adapt to different scenarios, IoT systems need to adopt an architecture that can provide (i) pull/push- and publish/subscribe-based application abstractions, (ii) a common naming framework, (iii) support for payload encryption and signature schemes, and (iv) open APIs as opposed to proprietary APIs that are common in today's systems. These requirements can pose great challenges for the underlying network and systems. To name a few, the IoT system needs to support 50-100 Billion networked objects [1], many of which are mobile. These objects are expected to have extremely heterogeneous means of connecting to the Internet, often with severe resource constraints. Interactions between the applications and the objects are often real-time and dynamic, requiring strong security and privacy protections. In addition, the IoT system design should offer efficient data exchange schemes that take into consideration the application behavior. For instance, in many IoT applications, data consumers usually need the data sensed from the environment without any reference to the subset of sensors that can provide the requested information.

In short, adopting a general IoT perspective, we first motivate the discussion of ICN for IoT by focusing on well known scenarios. We then discuss the IoT requirements that are generally applicable to many of these well known IoT scenarios. Next we discuss how the current application-layer unified IoT architectures are inefficient to meet the above requirements, and how the key ICN features can make it a better candidate to realize a unified IoT framework. Finally, adopting an ICN perspective, we address the main IoT design challenges and requirements posed towards an ICN-based-IoT system design.

2. Motivating ICN for IoT

ICN offers many features that include name-based networking, content object security, in-network caching, compute, and storage, active mobility support, context-aware networking (see Section 3.6), and support for ad hoc networking. Within the context of an IP based IoT design (IP-IoT), all of these features offered by the ICN have to be realized in an application-specific way demonstrating the compelling nature of ICN to design IoT systems.

To be specific, the features offered by ICN can be used to enable a distributed and intelligent data distribution platform that supports heterogeneous IoT services requiring minimal configuration for device bootstrapping, carrying simpler protocols to aid self-organizing among the IoT elements, and offering natural support for compute and caching logic at strategic points in the network. We outline general advantages of using an ICN-based IoT system design and discuss these from the perspective of the several service scenarios that are difficult to realize over IP today, and whose characteristics arguably match the features offered by ICN.

2.1. Advantages of using ICN for IoT

A key concept of ICN is the ability to name data and services independently from its original location (at which it is stored) and this simplifies caching, and enables decoupling of consumers and producers. Therefore, using ICN to design an architecture for IoT data potentially provides many such advantages compared to using traditional host-centric networks and other new architectures. This section highlights the general benefits that the ICN can provide to an IoT network.

- Naming of Devices, Data and Services: The heterogeneity of the deployed network equipment and offered services by an IoT network leads to a large variety of data, services, and devices. While using a traditional host-centric architecture, only devices or their network interfaces are named at the network level, leaving the task to name data and services to the application layer. This can cause different applications to use different naming schemes, and, as a result, no consistent mapping from application layer names to network names may exist. In many applications common to an IoT network, data and services represent the main objective, and ICN provides an intuitive way to name them in a way that can be utilized at the network layer as well. Communication with a specific device is often secondary, but when needed, the same ICN naming mechanisms can also be used. In such case, network distributes content and provides a service at the same time, instead of only sending data between two named devices. In this context, content and services can be provided by several devices, or a group of devices, hence naming data and services is often more important than naming the devices. This naming mechanism also enables self-configuration of the IoT system.

- Security and privacy: ICN advocates the object security model to secure data in the network. This concept is based on the idea of securing information objects, unlike the session-based security mechanisms, which secure the communication channel between a pair of nodes. ICN provides data integrity through name-data integrity, i.e., the guarantee that the given data corresponds to the name with which it was addressed. Signature-based schemes can additionally provide data authenticity, meaning establishing the origin, or provenance, of the data, for example, by cryptographically linking a data object to the identity of a publisher. Confidentiality can be handled on a per-object basis based on the keys established at the application level. All of this means that the actual transmission of data does not have to be secured, since the same security mechanisms protect the data starting with its generation until its consumption, regardless of its mobility/location (i.e., whether it is in transit over a communication channel or stored in an intermediate cache). In an ICN network, each individual object within a stream of immutable objects can potentially be retrieved from a cache in a different location. However, having a trust relationship with each of these different caches is not realistic. Through name-data integrity, ICN automatically guarantees data integrity to the requesting client regardless of the location, from where it is delivered. The object security model also ensures that the content is readily available in a secure state, and if the device constraints are severe enough that it is not able to perform the required cryptographic operations for object security, then it may be possible to offload this operation to a trusted gateway, to which only a single secure channel needs to be established. ICN can also derive a name from a public key, as the cryptographic hash of a public key also enables it to be self-certifying, in which case, authenticating the resource object does not require an external authority [27][28].

- Distributed Caching and Processing: While caching mechanisms are already used by other types of overlay networks, IoT networks can potentially benefit even more from caching and in-network processing, because of the resource constraints imposed on the devices. Furthermore, wireless bandwidth and power supply can be limited for multiple devices sharing a communication channel, and especially for small mobile devices powered by batteries. In this case, avoiding unnecessary transmissions to retrieve and distribute IoT data from/to multiple places becomes important, hence processing and storing such content in the network can save wireless bandwidth and battery power. Moreover, as for other types of networks, IoT-driven applications requiring shorter delays can also benefit from local caches and services to reduce the delays between content request and delivery.

- Sender/Receiver Decoupling: IoT devices may be mobile and face intermittent network connectivity issues. When a specific data is requested, such data can often be delivered by ICN without any consistent direct connectivity between devices. Apart from using structured caching systems as described previously, information can also be spread by forwarding data opportunistically.

2.2. Service Scenarios

- Smart Mobility: Smart end-user devices and machine-to-machine (M2M) connections are undergoing a significant growth. By 2021, there will be more than 10 billion mobile devices and connections, including smartphones, tablets, wearables, and vehicles [1]. The involved fields for these devices range from medicine and health care to fitness, from clothing to environmental monitoring [42].

In particular, one of the most affected domains is transportation and the so-called Intelligent Transport Systems (ITS) [44]. The objective of ITS is to provide a multi-modal transportation system that embraces public and private municipal, regional, national, and trans-national vehicles and fleets. This extremely heterogeneous ecosystem of transportation means is made available to the users through advanced services that can fulfill the usability requirements, while pursuing system level objectives, and which include: (i) the reduction of CO2 footprint, (ii) the real-time delivery of specific goods, (iii) the reduction of traffic within urban areas, (iv) the provisioning of pleasant journeys to tourists, and (v) the general commitment of satisfactory travel time and experience [121]. Within this context, IoT technologies can play a pivotal role. For instance, they enable advanced design paradigms (e.g., Mobility as a Service (MaaS) [41]) with significant implications on the system architecture [50] or lead to novel approaches to traffic modeling [49]. As a consequence, smart mobility support can be a significant use case scenario for ICN-IoT, where the important ICN features that corroborate mobility support are listed as follows:

* ICN is unique in that it supports both infrastructure- and ad hoc-based communications. This makes it suitable to support communication in vehicular ad hoc networks (VANETS) [19][126], along with supporting communication with the infrastructure components like the road side units to serve the needs of several smart mobility applications. ICN's name based network APIs along with its caching feature enable the system to simultaneously operate over multiple heterogeneous radio interfaces using broadcast, unicast or anycast communication modes.

* ICN offers location independence of content, which allows one to manage consumer mobility in a simpler way than it is with IP. Furthermore, different from Mobile IP, which needs 'triangular routing' to locate moving hosts, ICN envisions a mobile consumer to only re-issue content requests or use network based late-binding functions once the mobile entity handoffs from one attachment point to another [45];

* In ICN, since the content is not bound to a specific location, it can be cached anywhere in the network, thereby adding redundancy to the system. In doing so, if a producer loses connectivity while it is moving, a request for its content can be resolved to an intermediate node en-route to or routed towards a nearby off-path caching node [45];

* The name based request-response communication paradigm considered for ICN decouples publish/subscribe operations in time and space. Therefore, the involved entities (i.e., publishers and subscribers) do not need to be aware of each other or be connected at the same time [46];

* The use of an in-network Name Resolution Service (NRS) design allows to identify the current location of or associated with a content name in the network, thanks to its network function, which is responsible for updating the location information of a named entity [58].

From a technological perspective, we can list the open challenges as follows: (i) support for ad hoc communications and interoperability across different IoT technologies, (ii) namespace design that is able to harmonize different ITS standards, (iii) scalable data-sharing model(s) across real-time (and non real-time) traffic sources, (iv) design of travel-centric services based on ICN-IoT, (v) seamless support to mobility, and (vi) content authentication and cryptography.

o Smart Building: Buildings are gaining smart capabilities that allow for enhanced comfort, increased safety and security, and improved energy efficiency [105]. In particular, smart buildings are no longer simple consumer(s) (for energy), but can also be prosumers with on-site energy generation systems. These systems can improve a building's usability towards (i) smart heating, ventilation, and air conditioning (HVAC), (ii) smart lightings, (iii) plug loads, and (iv) smart windows. We can list the main requirements for these sub-systems as follows [105]: (i) context awareness, (ii) support for resource-constrained devices, (iii) interoperability across heterogeneous technologies, and (iv) security and privacy protection. The ICN paradigm can ease the fulfillment of these requirements for one simple reason: smart building services are typically information centric by design. To be specific, any time an autonomic management loop is established within a smart building to control a set of physical variables of interest, the information exchanged between the entities (e.g., users, sensors, actuators, and controllers) do not immediately translate to specific nodes within the building, but can be provided by multiple sensors or gateways. The relevance of ICN in a smart building setting is recognized in the literature as well with reference to the several frameworks deployed in various environments. For instance, in [63], nodes are distributed to different rooms, floors, and buildings of a campus university, and their energy consumption and individual behaviors are monitored. A smart home application is investigated in [107] by evaluating the retrieval delay and packet loss statistics for data.

Moreover, [108] designed and tested lighting control over NDN in a theater setting. In short, within the smart building context, we can list the ICN-specific challenges as follows: (i) design of a scalable namespace for uniquely identifying the information of interest and also host services for actuation, (ii) data-sharing model across heterogeneous systems, (iii) self-organizing functionalities for improving network connections between end-nodes, utilities and the control center, (iv) authentication procedures to grant data confidentiality and integrity.

- Smart Grid: Smart Grid systems are increasingly transforming into cyber-physical systems [18] with the goals of maximum automation towards efficiency and minimal human intervention. The system is a very complex one comprising of power distribution grids, end user applications (e.g. Electric Vehicle (EV) charging systems and appliances), smart monitoring systems (spanning the end users and the power grids), heterogeneous energy producing sources (including prosumers), and load distribution/balancing systems. Current smart grid systems are managed using the centralized Supervisory Control and Data Acquisition (SCADA) frameworks with highly restrictive unidirectional communication support [20]. These systems typically have the following requirements: (i) improved flexibility in distributing energy from the feeder, through real-time reconfiguration of multiple monitoring devices (e.g., phasor measurement units or PMUs) and management operations requiring an efficient data delivery infrastructure; (ii) a large scale data delivery infrastructure capable of supporting latency sensitive applications and inter-connecting heterogeneous end user devices that produce, monitor, and/or consume; (iii) resiliency, which is critical to the operation and protection of the grid; (iv) security, to protect mission critical grid applications from network intrusions; and (v) understanding machine-to-machine traffic patterns for optimal placement of storage and computing to maximize efficiency. Smart grid systems can benefit from ICN in the following ways [21][22]:

  * ICN approach of naming content rather than hosts can ensure that the data generated by one subsystem would be useful for multiple entities. Furthermore, naming content can enable the many-to-many communications model, which is very inefficient in the case of host-centric architectures.

  * ICN features such as in-network computing, storage, and caching enable better use of network resources and can benefit diverse application scenarios that vary from latency tolerant applications with low data rates (e.g., smart grid and energy pricing) to applications observing high data rates with stringent delay/disruption requirements (e.g., synchrophasor

measurements). Also, it is typical for smart grid systems to have applications that consume the same data at different rates, in which case in-network caching and computing can be of significant use.

* Host-centric networking exposes a mission critical infrastructure like the smart grid infrastructure to intrusion and Denial of Service (DOS) attacks, which are directly related to exposing the IP addresses of critical applications and subsystems. Naming contents, services, or devices, on the other hand, de-couples them from the location, thereby reducing the exposure to being targeted based on a geographical context.

* ICN's name based networking offers the potential for self-configuration during both the bootstrapping phase and the regular operation of the grid, allowing scalable operation with self-recovery during faults or maintenance tasks in the system.

o Smart Industrial Automation: In a smart and connected industrial environment, equipment with sensors generate large volumes of data during normal operation. This range from the highly time-critical data for real-time control of production processes, to the less time-critical data that is collected by a central cloud environment for control room monitoring, and to pure log data without latency requirements that is mainly kept for a posteriori analysis. Industrial wireless networks are difficult environments with many potential interferences occurring at the same time even as hard reliability and real-time requirements are placed by many applications. This means that the available network capacity is not always high, so it becomes likely for traffic with less stringent delay requirements to experience congestion. One such example is, when errors occur in the production process, a mobile workforce is expected to investigate the problem on-site and they will need high resolution data from the faulty machine(s) as well as other process data from the other parts of the plant. The mobile workforces typically perform their diagnostics or maintenance locally, and they rely on the information acquired from the production system both for safety purposes and to solve any other or related issues in the plant. Furthermore, they rely on both the historical data flow (to pinpoint the root cause of the problems) and the current data flow (to assess the present state of the equipment under control). High resolution measurements are typically generated close to the mobile workforce, while the historic data has to be retrieved from the historian servers. In this scenario, multiple workers involved in the process typically access the same data, possibly with a slight time-shift. The network thus needs to support mobile users to get access to the data flows in a way suitable for their physical location and the task requirements. Introducing ICN functionality into the system can lead to several benefits that enhance the working experience and productivity for the mobile workforce.

* When using ICN, naming of data can be done in a way that corresponds well to the current names often used in industrial scenarios as the hierarchical names defined by the OPC Foundation [10] can be easily mapped to the CCN/NDN name space.

* ICN provides the possibility to get the newest data without knowing the location of the caching nodes or whether a particular piece of data is available locally or in a central repository. ICN also gives the possibility to get either the local high-resolution data or the remote low-resolution data (as there is no need to store all the data centrally, which may not even be possible due to the large data volume). However, it may require well-defined naming conventions or routing policies that can route interests to the right location.

* ICN can reduce the network utilization as unnecessary data is not transmitted, and data accessed by multiple workers is only sent once.

* Workforce mobility between different access points in the factory can be inherently supported, without the need to maintain a connection state.

* Use of ICN can help with removing tedious configurations in clients, since that would be provided by the infrastructure.

* ICN allows the sharing of large volumes of data between users that are in physical proximity, without introducing additional traffic on the backbone network.

* Caching of data in ICN means avoiding additional accesses to a distributed redundant database in the central infrastructure with consistency requirements.

3. IoT Architectural Requirements

Future IoT platforms have to support secure interactions among a large number of heterogeneous, constrained, static or mobile resources across organization/domain boundaries. As a result, it naturally poses stringent requirements in every aspect of the system design. Below, we outline the important requirements that a future IoT platform has to address.

3.1. Naming

An important step towards realizing a unified IoT architecture is the ability to assign names that are unique to (i) each device, (ii) each data item generated by each of these devices, and (iii) each service hosted in a device or a group of devices, towards a common objective. We can assume the naming to have the following requirements. First, names need to be persistent against dynamic features that are common to IoT systems, such as lifetime, mobility, or migration. Second, names that are derived from the keys need to be self-certifying, for both device-centric and content-centric communications. For device-centric communications, binding between device names and the device must be secure. For content-centric communications, binding between the names and the content has to be secure. Third, names usually serve multiple purposes, i.e., routing, security (self-certifying), or human-readability. For IoT applications, the choice of flat versus human readable names needs to be made considering the application and network requirements such as privacy and network level scalability, resource constrained networking requirements, and the name space explosion that may occur because of the complex relationship between name hierarchies [124] that may confound application logic.

One of the challenges in naming is to ensure the trustworthiness of the names. A general approach would require a name certificate service. Such a service acts as a certificate authority in assigning names, which are themselves public keys or appropriately bound to the name for verification at the consumer end.

3.2. Security and Privacy

A variety of security and privacy concerns exist in IoT systems as they are infrastructure typically owned by private entities. For example, the unified IoT architecture makes physical objects accessible to applications across organizations and domains. Furthermore, it often integrates with a critical infrastructure and an industrial system with life safety implications, bringing with it significant security challenges and regulatory requirements [13], as will be discussed in Section 5.3. Security and privacy thus become a serious concern, as does the flexibility and usability of the design approaches. Beyond the overarching trust management challenge, security includes data integrity, authentication, and access control at different layers of the IoT architecture. Privacy includes several aspects: (i) privacy of the data producer/consumer that is directly related to each individual vertical domain such as health, electricity, etc., (ii) privacy of data content, and (iii) privacy of contextual information such as time and location of data transmission [68].

3.3. Scalability

Cisco [1] predicts that there will be around 50 Billion IoT devices on the Internet by 2020 (and these devices include sensors, Radio-Frequency IDentification (RFID) tags, and actuators), and a unified IoT platform needs to name every entity within, which includes these devices, and data and services accessed by/through them. Scalability has to be addressed at multiple levels of the IoT architecture including naming and name resolution, routing and forwarding, and security. Mobility adds further challenges in terms of scalability. Particularly, with respect to name resolution, the system should be able to register/update/resolve a name within a short latency. Additionally, scalability is also affected by the specific IoT system features such as IoT resource object count, state and rate of information update generated by the sensing devices.

3.4. Resource Constraints

IoT devices can be broadly classified as type 1, type 2, and type 3 devices, with type 1 being the most resource-constrained and type 3 being the most resource-rich [47], where the following are considered as the most typical resource types: power, computing, storage, bandwidth, and user interface.

Power constraints of IoT devices limit how much data these devices can communicate, as it has been shown that communications consume more power than other activities for embedded devices [48]. Flexible techniques to collect the relevant information are required, and uploading every single produced data to a central server is not desirable.

Computing constraints limit the type and amount of processing these devices can perform. As a result, more complex processing needs to be done at the cloud servers or at opportunistic points, for instance at the network edge, hence it is important to balance local computation versus communication costs.

Storage constraints of the IoT devices limit the amount of data that can be stored on these devices. This constraint means that unused sensor data may need to be discarded or stored in an aggregated compact form from time to time.

Bandwidth constraints of the IoT devices limit the amount of communication, hence, impose similar restrictions on the system architecture as the power constraints, i.e., one cannot afford to collect every single sensor data generated by the device and/or use complex control plane protocols. It is also worth mentioning that, this constraint also has implications on maintaining idle chatter in the background to maintain connectivity or other volatile service state.

User interface constraints refer to whether the device is itself capable of directly interacting with a user. Possible mechanisms include, via a display and keypad ,LED indicators or requires network connectivity, either locally or globally, to enable human interaction.

The above discussed resources constraints also impact application performance with respect to the end-to-end latency towards sensing or executing control loop based actuation functions.

3.5. Traffic Characteristics

IoT traffic can be broadly classified into local area traffic and wide area traffic. Local area traffic takes place among the nearby devices. For example, neighboring cars may work together to detect potential hazards on the highway, or sensors deployed in a room may collaborate to determine how to adjust the heating level in the room. These local area communications often involve data aggregation and filtering, carry real time constraints, and require fast discovery and association (for the device, data, or service). At the same time, IoT platform has to also support wide area communications. For example, in the case of Intelligent Transportation Systems, realtime video and sensor feeds from the concerned IoT entities can be used towards re-routing operations based on system state, traffic load, availability of freights, weather forecasts, and so on. Wide area communications also require efficient discovery and resolution services for data/services.

While traffic characteristics for different IoT systems are expected to be different, certain IoT systems have been analyzed and shown to have comparable uplink and downlink traffic volumes for some applications such as [2], which means that we have to optimize the bandwidth use and energy consumption in both directions. Furthermore, IoT traffic demonstrates certain periodicity and burstiness [2]. As a result, traffic characteristics of the IoT services have to be properly accounted for during system planning and provisioning.

3.6. Contextual Communication

Many IoT applications rely on dynamic contexts in the IoT system to initiate, maintain, and terminate communication among the IoT devices. Here, we refer to a context as attributes applicable to a group of devices that share some common features, such as their owners may have a certain social relationship or belong to the same administrative group, or the devices may be present near the same proximity. For example, cars traveling on the highway may form a "cluster" based upon their temporal physical proximity to one another as well as the detection of the same event. These temporary groups are referred to as contexts. There are two types of contexts: (i) long-term quasi-static contexts (i.e., contexts based on social contexts as well as stationary physical locations, such as sensors inside a car or a building) and (ii) short-term dynamic contexts (i.e., contexts based on temporary proximity). Between these two classes, short-term contexts are more challenging to support as they require fast formation, update, lookup and association. Therefore, in this draft, our focus will be on the more challenging latter class. In general, IoT applications need to support not only the interactions among the members of a context, but also the interactions across contexts.

3.7. Handling Mobility

There are several degrees of mobility corresponding to different IoT scenarios, ranging from static (as in fixed assets) to highly dynamic (as in vehicle-to-vehicle environments). Furthermore, mobility in an IoT architecture can refer to: (i) data producer mobility, (ii) data consumer mobility, (iii) IoT network mobility (e.g., a body-area network in motion as a person is walking), and/or (iv) disconnection between a source/destination pair (e.g., due to unreliable wireless links). The requirement on mobility support is to deliver IoT data earlier than an application's acceptable delay constraints for all the above considered cases, and if necessary, to negotiate different connectivity or security constraints specific to each mobile context. More detailed discussions on this issue are presented in Section 5.7.

3.8. Storage and Caching

Storage and caching plays a very significant role depending on the type of IoT ecosystem, which is also a function subjected to privacy and security guidelines. Caching is usually needed to increase data availability in the network and for reliability purposes, which is especially useful for wireless access scenarios and with devices experiencing intermittent connectivity to the infrastructure network. Storage is more important for an IoT system, as data is typically stored for long term analysis. Specifically, data is stored at strategic locations in the network to reduce control and computation related overheads. Depending on the application requirements, caching will strictly be driven by application level policies, considering also the privacy requirements. If, for certain type of IoT data, pervasive caching is allowed, then intermediate nodes may not need to always forward a content request to its original creator. Instead, receiving a cached copy would be sufficient for the IoT applications. This approach may greatly reduce the content access latencies.

Considering the hierarchical nature of the IoT systems, ICN architectures can enable a flexible, heterogeneous, and potentially fault-tolerant approach to storage and caching, thereby providing contextual persistence at multiple levels. Within the context of IoT and considering the application requirements, while offering resolution to replicated stored copies, ICN can efficiently support tradeoffs between content security/privacy and regulations.

3.9. Communication Reliability

IoT applications can be broadly categorized into mission critical and non-mission critical applications. For mission critical applications, reliable communication is one of the most important features, as these applications have strong QoS requirements such as low latency and low error rates during information transfer. To support the objective of reliable communications, it is essential for an underlying system to have the following capabilities: (i) seamless mobility support under normal operating conditions, (i) efficient routing in the presence of intermittent connection loss, (iii) QoS aware routing, (iv) support for redundancy at every system level (i.e., device, service, network, storage, etc.), and (v) support for rich and diverse communication patterns, both within an IoT domain (consisting of multiple IoT nodes and one or more gateway nodes to the Internet) and across multiple such domains.

3.10. Self-Organization

Considering the scalability and efficiency requirements, the unified IoT architecture should be able to self-organize to meet various application requirements, e.g., context-driven discovery, which refers to the capability to quickly discover heterogeneous and relevant local/global devices/data/services based on the context. A publish-subscribe service, or a private trust-driven community grouping or clustering scheme, can be used to support this discovery process. For the former case, the publish-subscribe service must be implemented in a way to efficiently support seamless mobility using techniques such as in-network caching and name-based routing. For the latter case, the IoT architecture should be able to discover the private community groups/clusters in a resource efficient way.

Another aspect of self-organization is the decoupling of the sensing infrastructure from the applications. In a typical IoT deployment, various applications run on top of a vast number of IoT devices. It is not an easy task to upgrade the firmware of the IoT devices, and it is also not practical to re-program these IoT devices to accommodate every change in these applications. Therefore, infrastructure and application specific logics need to be decoupled, and a common interface is required (i) to dynamically configure the interactions among the IoT devices and (ii) to easily modify these application logics on top of the sensing/actuating infrastructure [32] [33].

3.11. Ad hoc and Infrastructure Mode

Depending on the presence of a communication infrastructure, an IoT system can operate in an ad-hoc mode or an infrastructure mode, (or use a combination of two). For example, a vehicle may determine to report its location and status information to a server periodically through a cellular connection, or, a group of vehicles may form an ad-hoc network that collectively detects the road conditions around them. In cases, where an infrastructure is sparse, one of the participating nodes may choose to become a temporary gateway node.

The unified IoT architecture needs to design a common protocol that serves both of these modes. Such a protocol should address the challenges that may arise in them: (i) scalability and low latency for the infrastructure mode and (ii) efficient neighbor discovery and ad-hoc communication for the ad-hoc mode. Finally, we note that hybrid modes are very common in realistic IoT systems.

3.12. IoT Platform Management

Service, control and data planes for an IoT platform will be governed by its own management infrastructure, which includes (i) distributed and centralized middleware, (ii) discovery, naming, self-configuring, and analytic functions, and (iii) information dissemination, to achieve the specific IoT system objectives [27][28][29]. Towards this, new IoT management mechanisms and service metrics need to be developed to measure the success of an IoT deployment. Considering an IoT system's defining characteristics (such as the potential to carry a large number of IoT devices, the objective to save power, mobility, and ad hoc communications), autonomous self-management schemes become very critical. Furthermore, considering its hierarchical information processing deployment model, the platform needs to orchestrate computational tasks based on the involved sensors and the available computation resources, which may change over time. An efficient resource discovery and management protocol is required to facilitate this process. The trade-off between information transmission and processing is another challenge.

4. State of the Art

Over the years, many stand-alone IoT systems have been deployed in various domains. These systems usually adopt a vertical silo architecture and support a small set of pre-designated applications. A recent trend, however, is to move away from this approach, and towards a unified IoT architecture, in which the existing silo IoT systems, as well as the new systems that are rapidly deployed, can coexist. Here, a unified architecture refers to the case, where all the application and network functions use common APIs and network protocols to interact with each other. This will make their data and services accessible to general Internet applications (which is the case for ETSI-M2M [3] and oneM2M [4] standards). In such a unified architecture, resources can be accessed over the Internet and shared across the physical boundaries of an enterprise. However, current approaches to achieve this objective are mostly based on service overlays over the Internet, whose inherent inefficiencies caused by the use of the IP protocol [58] hinders the architecture from satisfying the IoT requirements outlined earlier, particularly in terms of scalability, security, mobility, and self-organization, which are discussed in more details in Section 4.2.

4.1. Silo IoT Architecture

```
                          [IoT Server]
                               |
                               |
                        _______|________
  ________             {                }
 {        }            {                }
 {IoT Dev}\            {    Internet    }---[IoT Application]
 {________}  [IoTGW]---{                }
                       {                }
                       {________________}
```

Figure 1:Silo architecture of standalone IoT systems

A typical standalone IoT system is illustrated in Figure 1, which include the devices, applications, gateway and server nodes. Many IoT devices have limited power and computing resources, unable to directly run the normal IP-based access network protocols (i.e., Ethernet, WiFi, 3G/LTE, etc.). Consequently, these devices operate over non-IP protocols to connect to the Internet servers using an IoT gateway. Through the IoT server, applications can subscribe to the data collected by these devices, or interact with them.

There have been quite a few popular protocols for standalone IoT systems, such as DF-1, MelsecNet, Honeywell SDS, BACnet, etc. However, these protocols are operating at a device-level abstraction, rather than an information driven one, leading to a fragmented information and protocol space that requires application level solutions to achieve interoperability.

4.2. Application-Layer Unified IoT Solutions

The current approach to create a unified IoT architecture is to make IoT gateways and servers adopt standard APIs. IoT devices connect to the Internet through standard APIs and IoT applications subscribe/receive data through standard control/data APIs. Built on top of today's Internet, this application-layer unified IoT architecture is the most practical approach towards a unified IoT platform. Towards this, there are ongoing standardization efforts including ETSI[3] and oneM2M[4]. IoT service providers can then use such frameworks to build common IOT gateways and servers for their customers. In addition, IETF's Constrained RESTful Environments (CORE) working group [5] is developing a set of protocols like Constrained Application Protocol (CoAP) [81], that is a lightweight protocol modeled after HTTP [82] and adapted specifically for the IoT. CoAP adopts the Representational State Transfer (REST) architecture with Client-Server interactions. It uses UDP as the underlying transport protocol with reliability and multicast support. Both CoAP and HTTP are considered as the suitable application level protocols for M2M communications, as well as for IoT. For example, oneM2M (which is one of the leading standards for a unified M2M architecture) has protocol bindings to both HTTP and CoAP for its primitives. Figure 2 shows the architecture adopted in this approach.

```
              Publishing----[IoT Server]----Subscribing--
                  |        /    |       \                |
                  |       /     |        \               |
                  |      /______|________ \              |
 ___________      |     /{              }  publishing    |
{           }     |    | {              }     |          |
{Smart Homes}\    |    | {   Internet   }---------[IoT Application]
{___________}  [IoTGW]---{              }\    |     ________________
                       | {              } \   |    {                }
                       | {______________}  [IoTGW]-{Smart Healthcare}
                       |        |                  {________________}
              Publishing [IoTGW]
                       |    ____|_____
                       |   {          }
                        ---{Smart Grid}
                           {__________}
```

Figure 2: Implementing an open IoT architecture through standardized APIs on the IoT gateways and the server

4.2.1. Weaknesses of the Application-Layer Approach

The above application-layer approach can work with many different protocols, but the system is built upon today's IP network, which has inherent weaknesses towards supporting a unified IoT system. As a result, it cannot satisfy some of the requirements outlined in Section 3, and the reasoning for that is explained as follows:

- o Naming: In current application-layer IoT systems, naming scheme is a host-centric one, that is, the name of a given resource/service is linked to the device that can provide it. In turn, device names are coupled to the IP addresses, which are not persistent in mobile scenarios. On the other side, in IoT systems, the same service/resource can be offered by different devices.

- o Security and Trust: In IP, security and trust model is based on the session established between two hosts. Session-based protocols rely on the exchange of several messages to establish a secure session. Use of such protocols in constrained IoT devices can have serious consequences in terms of energy efficiency, because transmission and reception of messages are often more costly than the cryptographic operations. This problem may be amplified with the number of nodes that a constrained device has to interact with, due to increase in both the computation cost and the per-session key state managed by the constrained device. Furthermore, because of focusing on securing communication channels rather than managing the data that needs to be secured directly, current trust management schemes can be considered to be relatively weak.

- Mobility: The application-layer approach uses IP addresses as names at the network layer, which hinders the support for device/service mobility or flexible name resolution. Furthermore, the orthogonal Layer 2/3 management, and application-layer addressing and forwarding required to deploy current IoT solutions limit the scalability and management of these systems.

- Resource Constraints: The application-layer approach requires every device to send data to an aggregator, to a gateway or to the IoT server. Resource constraints of the IoT devices, especially in power and bandwidth, can seriously limit the performance of this approach.

- Traffic Characteristics: In this approach, applications are written in a host-centric manner suitable for point-to-point communication. IoT, however, requires multicast support that is challenging for the application-layer based IoT systems today, which have only limited deployment in the current Internet.

- Contextual Communications: The application-layer based IoT approach may not react to dynamic contextual changes in a timely fashion. The main reason is that the context lists are usually kept at the IoT server and they cannot help with efficient routing of requests at the network layer.

- Storage and Caching: The application-layer approach supports application-centric storage and caching but not what ICN envisions at the network layer, or flexible storage that is enabled via name-based routing or lookups.

- Self-Organization: As the application-layer approach is bound to IP semantics, it is considered as topology-based, and, as a result, it cannot sufficiently satisfy the requirement on self-organization. In addition to the topological self-organization, IoT also requires self-organization at the data and service levels [101], which are also not supported by this approach.

- Ad hoc and Infrastructure Mode: As mentioned above, the overlay-based approach lacks self-organization and adaptation to dynamic topology changes, and, therefore, it cannot provide efficient support for the ad hoc mode of communication.

4.2.2. Relation to Delay Tolerant Networking (DTN) architecture and its suitability for IoT

In [23][24], delay-tolerant networking (DTN) has been considered to support future IoT architectures. DTN was initially developed to support information delivery in the presence of network disruptions and disconnections, but it has also been extended to support heterogeneous networks and name-based routing. The DTN Bundle Protocol is able to achieve some of these same advantages and could be beneficially used in an IoT network to, for example, decouple sender and receiver. The DTN architecture is however centered around named endpoints (or endpoint IDs), each of which usually corresponds to a host or a service, and is mainly a way to transport data, while ICN generalizes this notion to named data, hosts and services and offers ways to address IoT application [25] challenges through features such as (information) naming, discovery, request and dissemination. However, endpoint IDs can also be used to identify named content, enabling the use of the bundle protocol as a transport mechanism for an information-centric system. Such a use of the bundle protocol as a transport would still require other components from an ICN architecture such as naming conventions. However, since the exact transport is not a major focus of the issues addressed by this draft, most of the provided discussions are applicable to a generic ICN architecture.

5. ICN Design Considerations for IoT

This section outlines some of the ICN specific design considerations and challenges that must be considered when adopting an ICN design for IoT applications and systems, and describes some of the trade-offs involved to support large scale IoT deployments with diverse application requirements.

Though ICN integrates (i) abstractions at the content, service, and host levels, (ii) name-based routing, and (iii) computation, caching, and storage as part of the network infrastructure, IoT requires special considerations given the heterogeneity of devices and interfaces such as for constrained networking [63][123][125], data processing, and content distribution models to meet specific application requirements, which we identify as challenges in this section.

5.1. Naming Devices, Data, and Services

Even though the ICN approach of named data and services (i.e., device independent naming) is typically desirable when retrieving IoT data, such data-centric naming may also pose certain challenges.

o Naming of devices: Naming devices [127] [128]can be useful in an IoT system. For example, actuators may require clients to act on a specific node of the deployed network (to switch it on or off), or it could be necessary to access a particular device for administration purposes. This can only be achieved through a specific name that uniquely identifies the targeted network entity. Moreover, a persistent name allows a device to change its attachment point without loosing its identity. A friendly way to address a device is to use a contextual hierarchical name, which is of the same type as one that is used for data objects. Also note that, through disabling of caching and request aggregation on names associated with a device, it is possible to ensure that the requests targeting that device always reach the device.

o Size of data/service name: Content names can have variable lengths. Since each name has to uniquely identify the content and can also include self-certifying properties (e.g., the hash of the content is bound to the name), their lengths can be quite long in relation to the size of the content itself. In particular, for specific application, content name size can even exceed the Data size. This can be the case for IoT networks with sensed values that usually consist only of a few bytes (i.e., data can be as small as a short integer in case of temperature values, or one-byte in case of control messages corresponding to an actuator state as on/off). Moreover, a name that is too long is likely to trigger fragmentation at the link layer, and create additional problems (i.e., several transmissions, increased delay and security issues). Various approaches have been investigated to handle fragmentation and reassembly issues associated with ICN packets. For instance, the work in [109] proposes to perform hop-by-hop operations, i.e., each hop fragments the packet that has to be forwarded and reassembles the packet received for further processing. This mechanism allows to efficiently handle the recovery of lost or corrupted fragments locally, thereby reducing packet delivery failures that require application-level retransmissions.

o Hash-based content name: Hash algorithms are commonly used to name content, in order to verify that the received content is the one requested. This is only possible in contexts, where the requested object already exists, and where there is a directory service to look up names or the names are learned through a manifest service. This approach is suitable for systems with large sized data objects, where it is important to verify the content.

o Hierarchical names: The use of hierarchical names, as is the case with the CCN and NDN architectures, makes it easier to create names a priori based on a predefined naming convention. It also provides a convenient way to use the same naming scheme for device names. However, since names are not self-certifying, this will require other mechanisms for verification of object integrity. If routing is also performed on the hierarchical names, the system will lose some of its location independence and caching will mostly be done on the path towards the publisher.

- o Semantic and metadata-based content name: A semantic-based naming approach can allow for successful retrieval of name through a set of keywords (for example, 'noise level at position X'), even if a perfect matching of the name is not available [65]. Moreover, enriching contents with metadata allows to better describe the names and to establish association between similar ones. However, this mechanism requires more advanced functionality to match such metadata in the data object to the semantics of the name (e.g., comparing the position information of an object with the position information of the requested name). The need for such (potentially) computationally heavy tasks at the intermediate nodes in the network may be considered to understand the trade-offs between application and network performance. [64] proposes a metadata-based naming approach to support ICN-IoT networking with service function identification and processing of IoT data at some vantage points in the local IoT network, before returning the processed result to the consumers.

- o Naming of services: Similar to naming of devices or data, services can also be referred to with a unique identifier, provided by a specific device or by an authorized entity (i.e., someone assigned by a central authority as the service provider). It can also be a service provided by anyone meeting certain metadata conditions. Example of services may include content retrieval, which takes a content name or description as an input and returns the value of that content, and actuation, which takes an actuation command as an input and possibly returns a status code.

- o Trust: Names can be used to verify the authenticity and the integrity of the data. Multiple approaches can be used to provide security functionalities through names. For instance, hierarchical, schematized, Web-of-Trust models can enable public key verification, whereas self-certifying names can enable in-network integrity checks of the name-key or name-content binding without the need of a Public Key Infrastructure (PKI) or another third party to establish whether the key is trustworthy or not. This can be realized either directly or indirectly. In the former case, the hash of the content is bound to the name. In the latter case, first, the hash of the content is signed with the secret key of the publisher, and then the public key of the publisher and the signed hash are bound to the name [46]. The hash algorithm can be applied to the already existing contents and where there is a directory service or manifest to look up names. In case of yet-to-be-published but on-demand generated contents, the hash cannot be known a-priori, hence different trust mechanisms should be investigated. Furthermore, self-certified naming approach can hide the content semantics, thus making names less human friendly. Since trends show that users prefer to find contents through a search engine using keywords, having non-human-friendly names can be a barrier, unless the content is enriched with keywords. However, this problem does not concern M2M applications, as human-readable names may not be useful in the context of just communicating machines.

o Flexibility: Further challenges may arise for the hierarchical naming schema, associated with the requirements on "constructible names" and "on-demand publishing" [37][38]. The former entails that each user is able to construct the name of a desired data item through specific algorithms and that it is possible to retrieve information using partially specified names. The latter refers to the possibility of requesting a not-yet-published content, thereby triggering its creation.

o Scoping: From an application's point of view, scopes are used to gather related data, whereas from the network's perspective, scopes are used to mark where the content is available [68]. For instance, nodes that are involved with caching coordination can vary according to scope [69]. As a result, scoping can be used (i) to limit propagation of requests, thereby improving resource usage efficiency by reducing bandwidth and energy consumption, and (ii) to control content dissemination thanks to access control rules, which can be different for each scope [67]. Note that, relying on scoping for security/privacy has been shown to not work all that well for IP, and is unlikely to work well for ICN either. However, scoping may be useful in certain scenarios, for instance, to limit propagation of requests and provide a simple means to attain context-sensitive communications. Finally, perimeter- and channel-based access control is often violated by the current networks to enable over-the-wire updates and cloud-based services, so scoping is unlikely to replace a need for data-centric security in ICN.

o Confidentiality: As names can reveal information about the nature of the communication (which may also violate the privacy requirements), mechanisms for name confidentiality should be available in the ICN-IoT architecture. To grant confidentiality protection, some approaches have been proposed in order to handle access control in an ICN naming scheme such as Attribute-Based Encryption [66] and access control delegation [67]. In the first solution, a trusted third party assigns a set of attributes to each network entity. Then, a publisher performs the following operations in order: (i) encrypting the data with a random key, (ii) generating the metadata for the decryption phase, (iii) creating an access policy that is used to encrypt the random key, and (iv) appending the encrypted key to the content name. When the consumer receives the packet, if its attributes satisfy the hidden policy in the name, it can get the random key protected in the name and decrypt the data. The second solution introduces a new trusted network entity (i.e., Access Control Provide). In this case, when a publisher generates a content, it also creates an access control policy and send it to an Access Control Provider. This network entity stores the access control policy, to which it associates a Uniform Resource Identifier (URI). This URI is sent to the publisher and included in the advertisements of the content. Then, when a subscriber tries to access a protected content, it can authenticate himself and request authorization for the particular policy to the Access Control Provider through the URI.

5.2. Name Resolution

Inter-connecting numerous IoT entities, as well as establishing reachability to them, requires a scalable name resolution system considering several dynamic factors like mobility of end points, service replication, in-network caching, failure or migration [59] [72] [73] [95]. The objective is to achieve scalable name resolution handling static and dynamic ICN entities with low complexity and control overhead. In particular, the main requirements/challenges of a name space (and the corresponding Name Resolution System where necessary) are [52] [54]:

- Scalability: The first challenge faced by ICN-IoT name resolution system is its scalability. Firstly, the approach has to support billions of objects and devices that are connected to the Internet, many of which are crossing administrative domain boundaries. Second of all, in addition to objects/devices, the name resolution system is also responsible for mapping IoT services to their network addresses. Many of these services are based upon contexts, hence dynamically changing, as pointed out in [59]. As a result, the name resolution should be able to scale gracefully to cover a large number of names/services with wide variations (e.g., hierarchical names, flat names, names with limited scope, etc.). Notice that, if hierarchical names are used, scalability can be also supported by leveraging the inherent aggregation capabilities of the hierarchy. Advanced techniques such as hyperbolic routing [89] may offer further scalability and efficiency.

- o Deployability and inter-operability: Graceful deployability and interoperability with existing platforms is a must to ensure a naming schema to gain success on the market [7]. As a matter of fact, besides the need to ensure coexistence between IP-centric and ICN-IoT systems, it is required to make different ICN-IoT realms, each one based on a different ICN architecture, to inter-operate.

- o Latency: For real-time or delay sensitive M2M application, the name resolution should not affect the overall QoS. With reference to this issue it becomes important to circumvent too centralized resolution schema (whatever the naming style, i.e, hierarchical or flat) by enforcing in-network cooperation among the different entities of the ICN-IoT system, when possible [99]. In addition, fast name lookup are necessary to ensure soft/hard real time services [110][111][112]. This challenge is especially important for applications with stringent latency requirements, such as health monitoring, emergency handling and smart transportation [113].

- o Locality and network efficiency: During name resolution the named entities closer to the consumer should be easily accessible (subject to the application requirements). This requirement is true in general because, whatever the network, if the edges are able to satisfy the requests of their consumers, the load of the core and content seek time decrease, and the overall system scalability is improved. This facet gains further relevance in those domains where an actuation on the environment has to be executed, based on the feedbacks of the ICN-IoT system, such as in robotics applications, smart grids, and industrial plants [101].

- o Agility: Some data items could disappear while some other ones are created so that the name resolution system should be able to effectively take care of these dynamic conditions. In particular, this challenge applies to very dynamic scenarios (e.g., VANETs) in which data items can be tightly coupled to nodes that can appear and disappear very frequently.

5.3. Security and Privacy

Security and privacy is crucial to all the IoT applications including the use cases discussed in Section 2 and subjected to the information context. To exemplify this, in one recent demonstration, it was shown that passive tire pressure sensors in cars could be hacked adversely affecting the automotive system [77], while at the same time this and other car information can be used by a public traffic management system to improve road safety. The ICN paradigm is information-centric as opposed to state-of-the-art host-centric

Internet. Besides aspects like naming, content retrieval and caching this also has security implications. ICN advocates the model of trust in content rather than a direct trust in network host mode. This brings in the concept of Object Security which is contrary to session-based security mechanisms such as Transport Layer Security (TLS)/Datagram Transport Layer Security (DTLS) prevalent in the current host-centric Internet. Object Security is based on the idea of securing information objects unlike session-based security mechanisms which secure the communication channel between a pair of nodes for unicast, (or among a set of nodes for multicast/broadcast). This reinforces an inherent characteristic of ICN networks i.e. to decouple senders and receivers. Even session based trust association can be realized in ICN [86], that offers host-independence allowing authentication and authorization to be separated from session encryption, allowing multiple end points to meet specific service objectives. In the context of IoT, the Object Security model has several concrete advantages. Many IoT applications have as its main objective generating data and providing some services, while the communication between two devices is a secondary task. Therefore, it makes more sense to secure IoT objects instead of securing the session between communicating endpoints. Though ICN includes data-centric security features the mechanisms have to be generic enough to satisfy multiplicity of policy requirements for different applications. Furthermore security and privacy concerns have to be dealt in a scenario-specific manner with respect to network function perspective spanning naming, name-resolution, routing, caching, and ICN-APIs. The work by the JOSE WG [83] provides solution approaches to address some of these concerns for object security for constrained devices and should be considered to see what can be applied to an ICN architecture. In general, we feel that security and privacy protection in IoT systems should mainly focus on the following aspects: confidentiality, integrity, authentication and non-repudiation, and availability. Even though, implementing security and privacy methods in IOT systems faces different challenges than in other systems, like IP. Specifically, below we discuss the challenges in the constrained and infrastructure part of the network.

- In resource-constrained nodes, energy limitation is the biggest challenge. Moreover, a node it has to deliver its data over a wireless link for a reasonable period of time on a coin cell battery. As a result, traditional security/privacy measures are impractical to be implemented in the constrained part. In this case, one possible solution might be utilizing the physical wireless signals as security measures [78] [57].

- In the infrastructure part, we have several new threats introduced by ICN-IoT [88] particularly in architectures employing name resolution service [123]. Below we list several possible attacks to a name resolution service that is critical to ICN-IoT:

1. Each IoT device is given an ICN name. The name spoofing attack is a masquerading threat, where a malicious user A claims another user B's name and attempts to associate it with A's own network address NA-A, by announcing the mapping (ID-B, NA-A). The consequence of this attack is a denial of service as it can cause traffic directed for B to be directed to A's network address.

2. The stale mapping attack is a message manipulation attack involving a malicious name resolution server. In this attack, if a device moves and issues an update, the malicious name resolution server can purposely ignore the update and claim it still has the most recent mapping. Perhaps worse, a name resolution server can selectively choose which (possibly stale) mapping to give out during queries. The result is a denial of service.

3. The third potential attack, false announcement attack, is an information modification attack that results in illegitimate resource consumption. User A, which is in network NA1, claims its ID-A binds to a different network address, (ID-A, NA2). Thus A can direct its traffic to network NA2, which causes NA2's network resources to be consumed.

4. The collusion attack is an example of an information modification attack in which a malicious user, its network and the location where the mapping is stored collude with each other. The objective behind the malicious collusion is to allow for a fake mapping involving a false network address to pass the verification and become be stored in the storage place.

5. An intruder may insert fake/false sensor data into the network. The consequence might be an increase in delay and performance degradation for network services and applications.

- IoT data is collected and stored on such servers, which usually run learning algorithms to extract patterns from such data. In this case, it is important to adopt a framework that enables privacy-preserving learning techniques. The framework defines how data is collected, modified (to satisfy the privacy requirement), and transmitted to application developers.

5.4. Caching

In-network caching helps bring data closer to the consumers, but its usage differs in constrained and infrastructure parts of the IoT network. Furthermore, caching in ICN-IoT faces several challenges:

- Which nodes on the routing path should cache the data: According to [54], caching the data on a subset of nodes can achieve a better gain than caching it on every en-route routers. In particular, the authors propose a "selective caching" scheme to locate those routers with better hit probabilities to cache data. According to [55], selecting a random router to cache data is as good as caching the content everywhere. In [91], the authors suggest that edge caching provides most of the benefits of in-network caching but with simpler deployment. However, the existing research on this topic typically consider workloads that are analogous to today's CDNs, rather than the workload that can be attributed to IoT applications considered here. Therefore, further work is needed to understand the appropriate caching approach for IoT applications.

- What to cache for the IoT applications: In many IoT applications, customers often access a stream of sensor data, and as a result, caching a particular sensor data for longer periods may not be beneficial. In [93], a caching scheme is proposed to ensure that older instances of the same sensor stream were first to be evicted from the cache when needed. In [57], the authors suggest to cache IoT services at the intermediate routers, and in [59], the authors suggest to cache the control information such as pub/sub lists at the intermediate nodes. In addition, it is not yet clear what caching means in the context of actuation in an IoT system. For example, it could mean caching the result of a previous actuation request (using other ICN mechanisms to suppress the repeated actuation requests within a given time period), or it could have little meaning at all if the actuation uses authenticated requests as in [92].

- Efficiency of distributed caching may be application dependent: When content popularity is heterogeneous, some content is often requested repeatedly. In this case, the network can definitely benefit from caching. Another case where caching would be beneficial is when devices with low duty cycle are present in the network and when the access to the cloud infrastructure is limited. In [93], it is also shown that there are benefits to caching in the network when edge links are lossy, in particular if the losses occur close to the content producer, as is common for the wireless IoT networks. Furthermore, IoT devices can collaborate to cache content in a manner that optimizes energy efficiency and content availability [94]. However, using distributed caching mechanisms in the network is not useful when each object is only requested at most once, as a cache hit can only occur for the second and subsequent requests. It may also be less beneficial to have caches distributed throughout the ICN network, especially in cases when there are overlays of distributed repositories, e.g., a cloud or a Content Distribution Network (CDN), from which all clients can retrieve the data. Using ICN to retrieve data from such services may add some efficiency, but in case of dense occurrence of overlay CDN servers the additional benefit of caching in ICN nodes would be lower. Another example is when the name refers to an object with dynamic content/state. For example, when the last value for a sensor reading is requested or desired, the returned data may change any time the sensor reading is updated. In such case, in-network caching may increase the risk of returning old or stale data.

5.5. Storage

Storage is useful for IoT systems regardless of its type, be it as a long-term storage or as a short-term storage.

In the case of long-term in-network storage, resources can be distributed among vantage points, which include the network edge and the main IoT service aggregation points such as in the data centers. The main differences, in regards to IoT-driven storage, between the two locations are the size of data, processing intelligence and heterogeneity of information that has to be dealt at these locations. Specifically, the purpose of long term storage at the edge is to analyze, filter, aggregate, and re-publish IoT data for consumption either by the parent service components or directly by the consumers. At the aggregation service points, the purpose is to re-publish the data that will be presented as part of the global pub/sub service to the interested consumers. Long term storage for IoT data also serves the purpose of backup and replication of data, which come with additional caveats. First, we need to decide on the number of replicas needed for each IoT data stream, and the storage locations for these replicas. Also note that, given that many IoT applications consume data locally, storage locations should be kept near the data sources. However, since IoT data is mostly appended to the end of a stream, instead of being updated, it becomes easier to manage these multiple replicas. Second, we need to adopt a mechanism that can efficiently route traffic to the nearest data replica. ICN provides several solutions to this problem, e.g., global name resolution service (GNRS), which can keep track of each replica's location [58].

In the case of short-term in-network storage (where the term storage refers to a temporary buffer, when an outgoing link is not available), the objective is to improve communication reliability, especially when network links are unreliable, such as wireless links. ICN-IoT can adopt a generalized storage-aware routing algorithm to support delay and disruption tolerant packet forwarding. In such case, each router can employ the in-network storage to facilitate store vs. forward decisions in response to varying link conditions and potential network interruptions [115]. These decisions can be based on both short-term and long-term path quality metrics. Additionally, packets along disconnected paths can be handled using a disruption tolerant networking (DTN) based approach to offer delayed delivery and replication features. In particular, each router maintains two types of topology information: (i) an intra-partition graph that is formed by collecting flooded link state advertisements, which carry fine-grained, time-sensitive information about the intra-network links, and (ii) a DTN graph that is maintained via epidemically disseminated link-state advertisements, which carry connection probabilities among all the network nodes. However, for this scenario, we observe the following challenges: (i) when and how long to store the data, and (ii) the next step after the short-term storage. In [93] the authors show that it is beneficial to store data even for shorter periods of time (and even if only a single requester exist), if the network is lossy such that retransmissions and error recovery can be done locally instead of end-to-end.

5.6. Routing and Forwarding

ICN-IoT supports both device-to-device (D2D) and device-to-infrastructure (D2I) communications. D2D communications may occur within a single IoT domain, or across IoT domains, and may involve data forwarding within the source IoT domain, in the infrastructure network, and within the destination IoT domain. D2I communications involve data forwarding within the source IoT domain and in the infrastructure network. Data forwarding within an IoT domain can adopt routing protocols such as RPL [84], AODV[85], etc, with the main challenge being the resource constraints of the IoT nodes. In order to address this challenge, we can adopt a light-weight protocol using much shorter ICN names for each communicating party within an IoT domain (see Section 5.12 for details). In such case, before a packet leaves the IoT domain, gateway node translates this short ICN name associated with the source device to its original ICN name.

At the ICN infrastructure, data forwarding can adopt one of two approaches: (i) direct name-based routing or (ii) indirect name resolution service (NRS) driven routing.

- In direct name-based routing, packets are forwarded using the name corresponding to either the data itself [95][63][74] or the name of the destination node [75]. Here, the main challenge is to keep the state information required for data routing/forwarding at the ICN router small. This can become an especially challenging issue, if the architecture uses a flat naming scheme due to lack of aggregation capabilities.

- In indirect routing, packets are forwarded based on the locator of the destination node, which is obtained through a name resolution service. Here, name-locator binding can be done either before routing (i.e., assuming static binding) or during routing (i.e., assuming dynamic binding). In the case of static binding, router state is the same as that in traditional routers, and the main challenge is to perform name resolution fast, especially with mobile IoT devices. In the case of dynamic binding, ICN routers need to maintain a name-based routing table, and the challenge becomes keeping the state information small, while at the same time performing fast name resolution.

5.7. Mobility Management

Considering the diversity of IoT applications, mobility scenarios range from tracking sensor data from mobile human beings to large fleets of diverse mobile elements such as drones, vehicles, trucks, trains (each of which may be associated with a transport infrastructure). These mobility scenarios can take place over heterogeneous access infrastructure that ranges from short range 802.15.4 communications to cellular radios. It is therefore expected that handling information delivery in an ad hoc setting, which involves vehicles, road side units (RSU), and the corresponding infrastructure based services, shall offer more challenges. ICN architectures have been generally shown to handle consumer and producer mobility scenarios efficiently [61][129], and to be suitable for V2V scenarios [62]. Networking tools to handle mobility varies based on application requirements, which vary from delay and loss tolerant to mission critical (with stringent delay and loss requirements).

Therefore, the challenge becomes to quantify the cost associated with mobility management in both the control and the forwarding planes, to handle both static binding and dynamic binding (which enables seamless mobility) of a named resource to its location when either or both of consumer and/or producer is mobile.

During a network transaction, either the producer or the consumer may move away, and thus we need mechanisms that can handle the mobility of either or both to avoid information loss. ICN differentiates the mobility of a consumer (Case I) from that of a producer (Case II):

- Case I: When a consumer moves to a new location after sending out a request for data, the data may traverse the path towards the previous point of attachment (PoA), and in doing so, leaving copies of it along that path. The data can then be retrieved by the consumer by simply reissuing its request for the data, which is a technique used by the direct routing approach. Conversely, indirect routing approach does not differentiate between consumer and producer mobility [95], as the indirect routing approach only requires an update on the NRS, which can then update the routers to re-bind the named resource to its new location, while using late-binding to route the packet from the previous PoA to the new one.

- Case II: In the case of a producer that has moved, the challenge becomes managing the control overhead while searching for a new data producer (or for re-locating the initial producer) [60]. For this purpose, flooding techniques can be used to re-discover the producer, or direct routing techniques can be employed after enhancing them with the late-binding feature that enables seamless mobility [61].

5.8. Contextual Communication

ICN enables contextualized communications by allowing metadata to be included within control or application payload. Doing so can help IoT applications to adapt to different environments, thereby enabling intelligent networks that are self-configurable and intelligent networking among consumers and producers [57]. For example, let us look at the following smart transportation scenario: "James walks on an NYC street and wants to find an empty taxi closest to his location." In this example, the context is the location information corresponding to James and the taxi drivers. A context service, as an IoT middleware, processes the contextual information and bridges the gap between raw sensor information and application requirements. Alternatively, we can use naming conventions that allow applications to request content in namespaces related to their local context without requiring a specific service, such as /local/geo/mgrs/4QFJ/123/678 to retrieve objects published within a 100m grid area of 4QFJ 123 678 based on the military grid reference system (MGRS). In both cases, trust providers may emerge that can vouch for an application's local knowledge.

However, extracting contextual information on a real-time basis can become very challenging:

- First, we need to have a fast context resolution service, through which the subscribed IoT devices can continuously update their contextual information to the application (e.g., for the example above, that would be the locations of James and the taxis). Or, in the case of a namespace driven approach, we need to have mechanisms that can query the nearest neighbor based on a given namespace on a continual basis.

- o The difficulty of this challenge grows rapidly as the number of involved devices as well as the number of contexts increase.

5.9. In-network Computing

In-network computing enables ICN routers to host heterogeneous services catering to various network functions and applications needs. Contextual services for IoT networks require in-network computing, with each sensor node or ICN router implementing context reasoning [57]. Another major target for in-network computing is to filter (and cleanse) the sensed data for IoT applications, as the sensed data can be noisy [76].

Within this framework, Named Function Networking (NFN) [117] is proposed as an extension of the ICN concept to named functions, which are processed in the network, and which can be used to generate data flow processing applications (for instance, one that is well-suited to time series data processing by IoT sensing applications). Related to this is the need to support efficient function naming, with functions, input parameters, and the output result can all be encapsulated within the packet header, the packet body, or a mixture of the two (e.g. [33]). If functions are encapsulated within the packet header, naming scheme can impact (i) how a computation task is routed within the network, (ii) which IoT devices are involved with the computation task (e.g. [56]), and (iii) how a name is decomposed into smaller computation tasks and deployed in the network to achieve better performance.

Another challenge is related to how to support compute-aware routing. Default routing is typically used for forwarding requests towards the nearest cache (or source/repository) and return the matching data to the requester. Compute-aware routing, on the other hand, has a different purpose. For instance, if the computation task is for aggregating the sensed data, then the routing strategy becomes routing the data to achieve a better aggregation performance [53].

In-network computing also includes synchronization challenges. Some computation tasks, for instance, may need synchronization among sub-tasks or IoT devices. For instance, a device may not send the generated data as soon as it is available, because waiting for data from the neighbouring devices can lead to better aggregation performance. Or, some devices may choose to sleep to save energy, while waiting for the results from the neighbours. Furthermore, while aggregating the computation results along the path, intermediate IoT devices may need to choose the results generated within a certain time window.

5.10. Self-Organization

General IoT deployments involve heterogeneous IoT systems that consist of embedded systems, aggregators and service gateways in an IoT domain. To scale the IoT deployments to a large scale, scope-based self-organization is typically required. This specifically relates to the IoT system middleware functions [122] that include (i) device bootstrapping and discovery, (ii) assigning local/global names to device and/or content, and (iii) security and trust management functions towards device authentication and data privacy. ICN based on-boarding protocols have been studied [100] and has been shown to offer significant savings compared to the existing approaches. These challenges span both the constrained devices as well as the interactions among the aggregators and the service gateways, which may need to contact external services like the authentication servers to on-board these devices. A critical performance optimization metric for these functions, while operating at scale, is to have low control/data overhead in order to maximize the energy efficiency. Furthermore, within the infrastructure part of the network, scalable name-based resolution mechanisms, pub/sub services, storage and caching, and in-network computing techniques should be studied to meet the scope-based content dissemination needs of an ICN-IoT system.

5.11. Communications Reliability

ICN offers many ingredients for reliable communication, such as multi-home interest anycast over heterogeneous interfaces, caching, and forwarding intelligence for multi-path routing that leverage state-based forwarding in protocols like CCN/NDN. However, these features have not been analyzed from the QoS perspective, when heterogeneous traffic patterns are multiplexed at a router. In general, QoS for ICN is an open area of research [125]. In-network reliability comes at the cost of a complex network layer, hence a research challenge here is to build redundancy and reliability at the network layer to handle a wide range of disruption scenarios, such as congestion, short/long term connection loss, or wireless impairments along the last mile. An ICN network should allow features such as opportunistic store-and-forward mechanisms to be enabled only at certain points in the network, as these mechanisms entail additional control/forwarding plane overheads that can adversely affect the application throughput. For additional details, see Section 5.5, for the discussion on in-network storage.

5.12. Resource Constraints and Heterogeneity

An IoT architecture should take into consideration the resource constraints of the often embedded IoT nodes. Having globally unique IDs (GUID in short) is a key feature in ICN, and these IDs may consist of tens of bytes. Each device would then have a persistent and unique ID no matter when and where it moves. It is also important for ICN-IoT to keep this feature. However, always carrying the long ID in the packet header may not be always feasible, for instance, for transmissions over a low-rate layer-2 protocol such as 802.15.4. To solve this issue, ICN can operate using a lighter-weight packet header and a much shorter locally unique ID (LUID in short). In this way, we can map a device's long GUID to its short LUID when the packet targeting the device reaches the local area IoT domain. To cope with collisions that may occur with this mapping process, we let each domain to have its own GUID-to-LUID mapping scheme, which can be managed by a gateway deployed at the edge of the domain. Different from NAT and other existing domain- or gateway-based solutions, ICN-IoT does not change the identity of an application. The applications, either on the constrained IoT devices or on the infrastructure nodes, continue to use the long GUIDs to identify one another, while the network performs the translation, which is transparent to these applications. An IoT node carries its GUID no matter where it moves, even when it is relocated to another local IoT domain and assigned a new LUID. This ensures the global reachability under mobility, while taking into consideration the resource constraints of the embedded devices.

In addition, optimizations for the other components of the ICN-IoT system (described in earlier subsections) can lead to optimization of the energy consumption as well.

6. Differences from T2TRG

Thing-to-Thing Research Group (T2TRG) [9] is an IoT research group under IRTF, which focuses on the research challenges of realizing IoT solutions assuming IP as the narrow waist. As IP-IoT has been a research topic for over a decade and with active industry solutions, this group provides a venue to study the advanced issues related to its security, provisioning, configuration and inter-operability considering the various heterogeneous application environments. ICN-IoT, on the other hand, is a recent research effort, where the objective is to exploit the ICN features of name based routing and security, caching, multicasting, mobility, etc. in an end-to-end manner to enable IoT services spanning all kind of networking scenarios, i.e., ad hoc, infrastructure, and hybrid scenarios. More detailed comparison of IP-IoT versus ICN-IoT is presented in Section 4.

7. Security Considerations

ICN puts security in the forefront of its design, which the ICN-IoT designs can leverage to build applications with varying security requirements. This issue has been discussed quite elaborately in this draft. However, as this is an informational draft and it does not create new considerations beyond what has been discussed.

8. Conclusions

This draft offers a comprehensive view of the benefits and design challenges of using ICN to deliver IoT services, not only because of its suitability for constraint networks but also for ad hoc and infrastructure environments. The draft begins by motivating the need for ICN-IoT by considering popular IoT scenarios and then delves into understanding the IoT requirements from both application and networking perspectives. We then discuss why the current IP-based application layer unified IoT solutions fall short of meeting these requirements, and how an ICN architecture is more suitable towards addressing the IoT service needs. We then elaborate on the design challenges in realizing an ICN-IoT architecture at scale and one that offers reliability, security, energy efficiency, mobility, self-organization among others to accommodate these varying IoT service needs.

9. Acknowledgements

We thank all the contributors, reviewers and the valuable comments offered by the chairs to improve this draft.

10. Informative References

[1] Cisco System Inc., CISCO., "Cisco visual networking index: Global mobile data traffic forecast update.", 2016-2021.

[2] Shafig, M., Ji, L., Liu, A., Pang, J., and J. Wang, "A first look at cellular machine-to-machine traffic: large scale measurement and characterization.", Proceedings of the ACM Sigmetrics , 2012.

[3] The European Telecommunications Standards Institute, ETSI., "http://www.etsi.org/.", 1988.

[4] Global Intiative for M2M Standardization, oneM2M., "http://www.onem2m.org/.", 2012.

[5] Constrained RESTful Environments, CoRE., "https://datatracker.ietf.org/wg/core/charter/.", 2013.

[6] Ghodsi, A., Shenker, S., Koponen, T., Singla, A., Raghavan, B., and J. Wilcox, "Information-Centric Networking: Seeing the Forest of the Trees.", Hot Topics in Networking , 2011.

[7] Dong, L., Zhang, Y., and D. Raychaudhuri, "Enhance Content Broadcast Efficiency in Routers with Integrated Caching.", Proceedings of the IEEE Symposium on Computers and Communications (ISCC) , 2011.

[8] NSF FIA project, MobilityFirst., "http://mobilityfirst.winlab.rutgers.edu/", 2010.

[9] Thing-to-Thing Research Group, T2TRG., "https://datatracker.ietf.org/rg/t2trg/about/", 2017.

[10] OPC Foundation, OPC., "https://opcfoundation.org/", 2017.

[11] Kim, B., Lee, S., Lee, Y., Hwang, I., and Y. Rhee, "Mobiscape: Middleware Support for Scalable Mobility Pattern Monitoring of Moving Objects in a Large-Scale City.", Journal of Systems and Software, Elsevier, 2011.

[12] Dietrich, D., Bruckne, D., Zucker, G., and P. Palensky, "Communication and Computation in Buildings: A Short Introduction and Overview", IEEE Transactions on Industrial Electronics, 2010.

[13] Keith, K., Falco, F., and K. Scarfone, "Guide to Industrial Control Systems (ICS) Security", NIST, Technical Report 800-82 Revision 1, 2013.

[14] Darianian, M. and Martin. Michael, "Smart home mobile RFID-based Internet-of-Things systems and services.", IEEE, ICACTE, 2008.

[15] Zhu, Q., Wang, R., Chen, Q., Chen, Y., and W. Qin, "IOT Gateway: Bridging Wireless Sensor Networks into Internet of Things", IEEE/IFIP, EUC, 2010.

[16] Biswas, T., Chakrabort, A., Ravindran, R., Zhang, X., and G. Wang, "Contextualized information-centric home network", ACM, Sigcomm, 2013.

[17] Huang, R., Zhang, J., Hu, Y., and J. Yang, "Smart Campus: The Developing Trends of Digital Campus", 2012.

[18] Yan, Y., Qian, Y., Hu, Y., and J. Yang, "A Survey on Smart Grid Communication Infrastructures: Motivations, Requirements and Challenges", IEEE Communications Survey and Tutorials, 2013.

[19] Grassi, G., Pesavento, D., Pau, G., Vayyuru, R., Wakikawa, Ryuji., Wakikawa, Ryuji., and Lixia. Zhang, "Vehicular Inter-Networking via Named Data", ACM Hot Mobile (Poster), 2013.

[20] Chai, W., Katsaros, K., Strobbe, M., and P. Romano, "Enabling Smart Grid Applications with ICN", ICN Sigcomm, 2015.

[21] Katsaros, K., Chai, W., Wang, N., and G. Pavlou, "Information-centric Networking for Machine-to-Machine Data Delivery: A Case Study in Smart Grid Applications", IEEE Network, 2014.

[22] Dong, X., "Event-trigger particle filter for smart grids with limited communication bandwidth infrastructure", IEEE Transactions on Smart Grid, 2017.

[23] Mael, A., Maheo, Y., and F. Raimbault, "CoAP over BP for a delay-tolerant internet of things", Future Internet of Things and Cloud (FiCloud), IEEE, 2015.

[24] Patrice, R. and H. Rivano, "Tests Scenario on DTN for IOT III Urbanet collaboration", Dissertation, INRIA, 2015.

[25] Kevin, F., "Comparing Information-Centric and Delay-Tolerant Networking", Local Computer Networks (LCN), 2012 IEEE 37th Conference on. IEEE, 2012..

[26] Miao, Y. and Y. Bu, "Research on the Architecture and Key Technology of Internet of Things (loT) Applied on Smart Grid", IEEE, ICAEE, 2010.

[27] Castro, M. and A. Jara, "An analysis of M2M platforms: challenges and opportunities for the Internet of Things", IMIS, 2012.

[28] Gubbi, J., Buyya, R., and S. Marusic, "Internet of Things (IoT): A vision, architectural elements, and future directions", Future Generation Computer Systems, 2013.

[29] Vandikas, K. and V. Tsiatsis, "Performance Evaluation of an IoT Platform. In Next Generation Mobile Apps, Services and Technologies(NGMAST)", Next Generation Mobile Apps, Services and Technologies (NGMAST), 2014.

[30] Zhang, Y., Yu, R., Nekovee, M., Liu, Y., Xie, S., and S. Gjessing, "Cognitive Machine-to-Machine Communications: Visions and Potentials for the Smart Grid", IEEE, Network, 2012.

[31] Zhou, H., Liu, B., and D. Wang, "Design and Research of Urban Intelligent Transportation System Based on the Internet of Things", Springer Link, 2012.

[32] Alessandrelli, D., Petracca, M., and P. Pagano, "T-Res: enabling reconfigurable in-network processing in IoT-based WSNs.", International Conference on Distributed Computing in Sensor Systems (DCOSS) , 2013.

[33] Kovatsch, M., Mayer, S., and B. Ostermaier, "Moving application logic from the firmware to the Cloud: towards the thin server architecture for the internet of things.", in Proc. 6th Int. Conf. on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS) , 2012.

[34] Zhang, M., Yu, T., and G. Zhai, "Smart Transport System Based on the Internet of Things", Applied Mechanics and Materials, 2012.

[35] Zhang, A., Yu, R., Nekovee, M., and S. Xie, "The Internet of Things for Ambient Assisted Living", IEEE, ITNG, 2010.

[36] Savola, R., Abie, H., and M. Sihvonen, "Towards metrics-driven adaptive security management in E-health IoT applications.", ACM, BodyNets, 2012.

[37] Jacobson, V., Smetters, D., Plass, M., Stewart, P., Thornton, J., and R. Braynard, "VoCCN: Voice-over Content-Centric Networks", ACM, ReArch, 2009.

[38] Piro, G., Cianci, I., Grieco, L., Boggia, G., and P. Camarda, "Information Centric Services in Smart Cities", ACM, Journal of Systems and Software, 2014.

[39] Gaur, A., Scotney, B., Parr, G., and S. McClean, "Smart City Architecture and its Applications Based on IoT - Smart City Architecture and its Applications Based on IoT", Procedia Computer Science, Volume 52, 2015, Pages 1089-1094.

[40] Herrera-Quintero, L., Banse, K., Vega-Alfonso, J., and A. Venegas-Sanchez, "Smart ITS sensor for the transportation planning using the IoT and Bigdata approaches to produce ITS cloud services", 8th Euro American Conference on Telematics and Information Systems (EATIS), Cartagena, 2016, pp. 1-7.

[41] Melis, A., Pardini, M., Sartori, L., and F. Callegati, "Public Transportation, IoT, Trust and Urban Habits", Internet Science: Third International Conference, INSCI 2016, Florence, Italy, September 12-14, 2016, Proceedings.

[42] Tonneau, A., Mitton, N., and J. Vandaele, "A Survey on (mobile) Wireless Sensor Network Experimentation Testbeds", 2014 IEEE International Conference on Distributed Computing in Sensor Systems, Marina Del Rey, CA, 2014, pp. 263-268.

[43] Zhilin, Y., "Mobile phone location determination and its impact on intelligent transportation systems", IEEE Transactions on Intelligent Transportation Systems, vol. 1, no. 1, pp. 55-64, Mar 2000.

[44] Papadimitratos, P., La Fortelle, A., Evenssen, K., Brignolo, R., and S. Cosenza, "Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation", IEEE Communications Magazine, vol. 47, no. 11, pp. 84-95, November 2009.

[45] Zhang, Yu., Afanasyev, A., Burke, J., and L. Zhang, "A survey of mobility support in named data networking", Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on. IEEE, 2016.

[46] Xylomenos, G., Ververidis, C., Siris, V., and N. Fotiou et al, "A survey of information-centric networking research", IEEE Communications Surveys and Tutorials, Volume: 16, Issue: 2, Second Quarter 2014 .

[47] Mavromoustakis, C., Mastorakis, G., and J. Batalla, "Internet of Things (IoT) in 5G Mobile Technologies", ISBN,3319309137,Springer.

[48] Firner, S., Medhekar, S., and Y. Zhang, "PIP Tags: Hardware Design and Power Optimization", in Proceedings of HotEmNets, 2008.

[49] Masek, P., Masek, J., Frantik, P., and R. Fujdiak, "A Harmonized Perspective on Transportation Management in Smart Cities: The Novel IoT-Driven Environment for Road Traffic Modeling", Sensors, Volume 16, Issue 11, 2016.

[50] Abreu, D., Velasquez, K., Curado, M., and E. Monteiro, "A resilient Internet of Things architecture for smart cities", Annals of Telecommunications, Volume 72, Issue 1, Pages 19-30, 2017.

[51] Ravindran, R., Biswas, T., Zhang, X., Chakrabort, A., and G. Wang, "Information-centric Networking based Homenet", IEEE/IFIP, 2013.

[52] Dannewitz, C., D' Ambrosio, M., and V. Vercellone, "Hierarchical DHT-based name resolution for information-centric networks", 2013.

[53] Fasoloy, E., Rossiy, M., and M. Zorziy, "In-network Aggregation Techniques for Wireless Sensor Networks: A Survey", IEEE Wireless Communications, 2007.

[54] Chai, W., He, D., and I. Psaras, "Cache "less for more" in information-centric networks", ACM, IFIP, 2012.

[55] Eum, S., Nakauchi, K., Murata, M., Shoji, Yozo., and N. Nishinaga, "Catt: potential based routing with content caching for icn", IEEE Communication Magazine, 2012.

[56] Drira, W. and F. Filali, "Catt: An NDN Query Mechanism for Efficient V2X Data Collection", Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking Workshops (SECON Workshops), 2014.

[57] Eum, S., Shvartzshnaider, Y., Francisco, J., Martini, R., and D. Raychaudhuri, "Enabling internet-of-things services in the mobilityfirst future internet architecture", IEEE, WoWMoM, 2012.

[58] Raychaudhuri, D., Nagaraj, K., and A. Venkatramani, "Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet.", ACM SIGMOBILE Mobile Computing and Communications Review 16.3 (2012): 2-13.

[59] Sun, Y., Qiao, X., Cheng, B., and J. Chen, "A low-delay, lightweight publish/subscribe architecture for delay-sensitive IOT services", IEEE, ICWS, 2013.

[60] Azgin, A., Ravindran, R., and GQ. Wang, "Mobility study for Named Data Networking in wireless access networks", IEEE, ICC, 2014.

[61] Azgin, A., Ravindran, R., Chakraborti, A., and GQ. Wang, "Seamless Producer Mobility as a Service in Information Centric Networks", ACM ICN Sigcomm, IC5G Workshop, 2016.

[62] Wang, L., Wakikawa, R., Kuntz, R., and R. Vuyyuru, "Data Naming in Vehicle-to-Vehicle Communications", IEEE, Infocm, Nomen Workshop, 2012.

[63] Baccelli, E., Mehlis, C., Hahm, O., Schmidt, T., and M. Wahlisch, "Information Centric Networking in the IoT:Experiments with NDN in the Wild", ACM, ICN Siggcomm, 2014.

[64] Ascigil, O., Rene, S., Xylomenos, G., Psaras, I., and G. Pavlou, "A Keyword-based ICN-IoT Platform", ACM, ICN Sigcomm, 2017..

[65] Simona, C. and M. Mongiello, "Pushing the role of information in ICN", Telecommunications (ICT), 2016 23rd International Conference on. IEEE, 2016..

[66] Li, B., Huang, D., Wang, Z., and Y. Zhu, "Attribute-based Access Control for ICN Naming Scheme", IEEE Transactions on Dependable and Secure Computing, vol.PP, no.99, pp.1-1..

[67] Polyzos, G. and N. Fotiou, "Building a reliable Internet of Things using Information-Centric Networking", Journal of Reliable Intelligent Environments, vol.1, no.1, 2015..

[68] Pandurang, K., Xu, W., Trappe, W., and Y. Zhang, "Temporal privacy in wireless sensor networks: Theory and practice", ACM Transactions on Sensor Networks (TOSN) 5, no. 4 (2009): 28..

[69] Trossen, D., Sarela, M., and K. Sollins, "Arguments for an information-centric internetworking architecture.", ACM SIGCOMM Computer Communication Review 40.2 (2010): 26-33.

[70] Zhang, G., Li, Y., and T. Lin, "Caching in information centric networking: A survey.", Computer Networks 57.16 (2013): 3128-3141.

[71] Gronbaek, I., "Architecture for the Internet of Things (IoT): API and interconnect", IEEE, SENSORCOMM, 2008.

[72] Tian, Y., Liu, Y., Yan, Z., Wu, S., and H. Li, "RNS-A Public Resource Name Service Platform for the Internet of Things", IEEE, GreenCom, 2012.

[73] Roussos, G. and P. Chartier, "Scalable id/locator resolution for the iot", IEEE, iThings,CPSCom, 2011.

[74] Amadeo, M. and C. Campolo, "Potential of information-centric wireless sensor and actuator networking", IEEE, ComManTel, 2013.

[75] Nelson, S., Bhanage, G., and D. Raychaudhuri, "GSTAR: generalized storage-aware routing for mobilityfirst in the future mobile internet", ACM, MobiArch, 2011.

[76] Trappe, W., Zhang, Y., and B. Nath, "MIAMI: methods and infrastructure for the assurance of measurement information", ACM, DMSN, 2005.

[77] Rouf, I., Mustafa, H., Taylor, T., Oh, S., Xu, W., Gruteser, M., Trappe, W., and I. Seskar, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study", USENIX, 2010.

[78] Liu, R. and W. Trappe, "Securing Wireless Communications at the Physical Layer", Springer, 2010.

[79] Xiao, L., Greenstein, L., Mandayam, N., and W. Trappe, "Using the physical layer for wireless authentication in time-variant channels", IEEE Transactions on Wireless Communications, 2008.

[80] Sun, S., Lannom, L., and B. Boesch, "Handle system overview", IETF, RFC3650, 2003.

[81] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[82] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[83] Barnes, R., "Use Cases and Requirements for JSON Object Signing and Encryption (JOSE)", RFC 7165, DOI 10.17487/RFC7165, April 2014, <https://www.rfc-editor.org/info/rfc7165>.

[84] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <https://www.rfc-editor.org/info/rfc6550>.

[85] Perkins, C., Belding-Royer, E., and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, DOI 10.17487/RFC3561, July 2003, <https://www.rfc-editor.org/info/rfc3561>.

[86] Mosko, M., Uzun, E., and C. Wood, "CCNx Key Exchange Protocol Version 1.0", draft-wood-icnrg-ccnxkeyexchange-02 (work in progress), July 2017.

[87] Sun, S., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", 2014.

[88] Liu, X., Trappe, W., and Y. Zhang, "Secure Name Resolution for Identifier-to-Locator Mappings in the Global Internet", IEEE, ICCCN, 2013.

[89] Boguna, M., Fragkiskos, P., and K. Dmitri, "Sustaining the internet with hyperbolic mapping", Nature Communications, 2010.

[90] Shang, W., "Securing building management systems using named data networking", IEEE Network 2014.

[91] Fayazbakhsh, S. and et al, "Less pain, most of the gain: Incrementally deployable icn", ACM, Siggcomm, 2013.

[92] Burke, J. and et. et al, "Securing instrumented environments over Content-Centric Networking: the case of lighting control", INFOCOM, Computer Communications Workshop, 2013.

[93] Rao, A., Schelen, O., and A. Lindgren, "Performance Implications for IoT over Information Centric Networks", Performance Implications for IoT over Information Centric Networks, ACM CHANTS 2016.

[94] Hahm, O., Baccelli, E., Schmidt, T., Wahlisch, M., Adjih, C., and L. Massoulie, "Low-power Internet of Things with NDN and Cooperative Caching", ICN, Sigcomm, 2017.

[95] Li, S., Zhang, Y., Dipankar, R., and R. Ravindran, "A comparative study of MobilityFirst and NDN based ICN-IoT architectures", IEEE, QShine, 2014.

[96] Chen, J., Li, S., Yu, H., Zhang, Y., and R. Ravindran, "Exploiting icn for realizing service-oriented communication in iot", IEEE, Communication Magazine, 2016.

[97] Quevedo, J., Corujo, D., and R. Aguiar, "A Case for ICN usage in IoT environments", Global Communications Conference GLOBECOM, IEEE, Dec 2014, Pages 2770-2775.

[98] Lindgren, A., Ben Abdesslem, F., Ahlgren, B., and O. Schelen, "Design Choices for the IoT in Information-Centric Networks", IEEE Annual Consumer Communications and Networking Conference (CCNC) 2016.

[99] Grieco, L., Alaya, M., and K. Drira, "Architecting Information Centric ETSI-M2M systems", IEEE, Pervasive and Computer Communications Workshop (PERCOM), 2014.

[100] Compagno, A., Conti, M., and R. Dorms, "OnboardICNg: a Secure Protocol for On-boarding IoT Devices in ICN", ICN, Sigcomm, 2016.

[101] Grieco, L., Rizzo, A., Colucci, R., Sicari, S., Piro, G., Di Paola, D., and G. Boggia, "IoT-aided robotics applications: technological implications, target domains and open issues", Elsevier Computer Communications, Volume 54, 1 December, 2014.

[102] InterDigital, WhitePaper., "Standardized M2M Software Development Platform", 2011.

[103] Boswarthick, D., "M2M Communications: A Systems Approach", 2012.

[104] Swetina, J., Lu, G., Jacobs, P., Ennesser, F., and J. Song, "Toward a standardized common M2M service layer platform: Introduction to oneM2M", IEEE Wireless Communications, Volume 21, Number 3, June 2014.

[105] Wang, L., Wang, Z., and R. Yang, "Intelligent Multiagent Control System for Energy and Comfort Management in Smart and Sustainable Buildings", IEEE Transactions on Smart Grid, vol. 3, no. 2, pp. 605-617, June 2012..

[106] Lawrence, T., Boudreau, M., and L. Helsen, "Ten questions concerning integrating smart buildings into the smart grid, Building and Environment", Building and Environment, Volume 108, 1 November 2016, Pages 273-283..

[107] Hassan, A. and D. Kim, "Named data networking-based smart home", ICT Express 2.3 (2016): 130-134..

[108] Burke, J., Horn, A., and A. Marianantoni, "Authenticated lighting control using named data networking", UCLA, NDN Technical Report NDN-0011 (2012)..

[109] Afanasyev, A., "Packet fragmentation in ndn: Why ndn uses hop-by-hop fragmentation.", UCLA, NDN Technical Report NDN-0032 (2015)..

[110] Quan, Wei., Xu, C., Guan, J., Zhang, H., and L. Grieco, "Scalable Name Lookup with Adaptive Prefix Bloom Filter for Named Data Networking", IEEE Communications Letters, 2014.

[111] Wang, Yi., Pan, T., Mi, Z., Dai, H., Guo, X., Zhang, T., Liu, B., and Q. Dong, "NameFilter: Achieving fast name lookup with low memory cost via applying two-stage Bloom filters", INFOCOM, 2013.

[112] So, W., Narayanan, A., Oran, D., and Y. Wang, "Toward fast NDN software forwarding lookup engine based on Hash tables", ACM, ANCS, 2012.

[113] Amadeo, M., Campolo, C., Iera, A., and A. Molinaro, "Named data networking for IoT: An architectural perspective", IEEE, EuCNC, 2014.

[114] Amadeo, M., Campolo, C., Iera, A., and A. Molinaro, "Information centric networking in iot scenarios: The case of a smart home", IEEE ICC, June 2015.

[115] Somani, N., Chanda, A., Nelson, S., and D. Raychaudhuri, "Storage- Aware Routing for Robust and Efficient Services in the Future Mobile Internet", Proceedings of ICC FutureNet V, 2012.

[116] Blefari Melazzi, N., Detti, A., Arumaithurai, M., and K. Ramakrishnan, "Internames: A name-to-name principle for the future internet", QShine, August 2014.

[117] Sifalakis, M., Kohler, B., Christopher, C., and C. Tschudin, "An information centric network for computing the distribution of computations", ACM, ICN Sigcomm, 2014.

[118] Lu, R., Lin, X., Zhu, H., and X. Shen, "SPARK: a new VANET-based smart parking scheme for large parking lots", INFOCOM, 2009.

[119] Wang, H. and W. He, "A reservation-based smart parking system", The First International Workshop on Cyber-Physical Networking Systems, 2011.

[120] Qian, L., "Constructing Smart Campus Based on the Cloud Computing and the Internet of Things", Computer Science 2011.

[121] Project, BonVoyage., "European Unions - Horizon 2020, http://bonvoyage2020.eu/", 2016.

[122] Li, S., Zhang, Y., Raychaudhuri, D., Ravindran, R., Zheng, Q., Wang, GQ., and L. Dong, "IoT Middleware over Information-Centric Network", Global Communications Conference (GLOBECOM) ICN Workshop, 2015.

[123] Li, S., Chen, J., Yu, H., Zhang, Y., Raychaudhuri, D., Ravindran, R., Gao, H., Dong, L., Wang, GQ., and H. Liu, "MF-IoT: A MobilityFirst-Based Internet of Things Architecture with Global Reachability and Communication Diversity", IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI), 2016.

[124] Adhatarao, S., Chen, J., Arumaithurai, M., and X. Fu, "Comparison of naming schema in ICN", IEEE LANMAN, June , 2016.

[125] Campolo, C., Corujo, D., Iera, A., and R. Aguiar, "Information-centric Networking for Internet-of-things: Challenges and Opportunities", IEEE Networks, Jan , 2015.

[126] Hussain, R., Bouk, S., Javaid, N., and Adil. Khan, "Realization of VANET-Based Cloud Services through Named Data Networking", IEEE Communication Magazine, 2018.

[127] Sobia, A. and et al., "Hierarchical and Flat-Based Hybrid Naming Scheme in Content-Centric Networks of Things", IEEE Internet of Things Journal, 2018.

[128] Sobia, A. and et al., "Towards Information-Centric Networking (ICN) naming for Internet of Things (IoT): the case of smart campus.", Proceedings of the International Conference on Future Networks and Distributed Systems. ACM, 2017.

[129] Agnese, V. and et al., "Publish subscribe in mobile information centric networks: Modeling and performance evaluation.", Computer Networks, 2017.

Authors' Addresses

Ravishankar Ravindran
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: ravi.ravindran@huawei.com

Yanyong Zhang
WINLAB, Rutgers University
671, U.S 1
North Brunswick, NJ 08902
USA

Email: yyzhang@winlab.rutgers.edu

Luigi Alfredo Grieco
Politecnico di Bari (DEI)
Via Orabona 4
Bari 70125
Italy

Email: alfredo.grieco@poliba.it

Anders Lindgren
RISE SICS
Box 1263
Kista SE-164 29
SE

Email: anders.lindgren@ri.se

Jeff Burke
UCLA REMAP
102 East Melnitz Hall
Los Angeles, CA 90095
USA

Email: jburke@ucla.edu

Bengt Ahlgren
RISE SICS
Box 1263
Kista, CA SE-164 29
SE

Email: bengt.ahlgren@ri.se

Aytac Azgin
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: aytac.azgin@huawei.com

ICN Research Group C. Gundogan
Internet-Draft TC. Schmidt
Intended status: Experimental HAW Hamburg
Expires: May 3, 2021 M. Waehlisch
link-lab & FU Berlin
C. Scherb
C. Marxer
C. Tschudin
University of Basel
October 30, 2020

ICN Adaptation to LoWPAN Networks (ICN LoWPAN)
draft-irtf-icnrg-icnlowpan-09

Abstract

This document defines a convergence layer for CCNx and NDN over IEEE 802.15.4 LoWPAN networks. A new frame format is specified to adapt CCNx and NDN packets to the small MTU size of IEEE 802.15.4. For that, syntactic and semantic changes to the TLV-based header formats are described. To support compatibility with other LoWPAN technologies that may coexist on a wireless medium, the dispatching scheme provided by 6LoWPAN is extended to include new dispatch types for CCNx and NDN. Additionally, the fragmentation component of the 6LoWPAN dispatching framework is applied to ICN chunks. In its second part, the document defines stateless and stateful compression schemes to improve efficiency on constrained links. Stateless compression reduces TLV expressions to static header fields for common use cases. Stateful compression schemes elide state local to the LoWPAN and replace names in data packets by short local identifiers.

This document is a product of the IRTF Information-Centric Networking Research Group (ICNRG).

Table of Contents

1. Introduction

The Internet of Things (IoT) has been identified as a promising deployment area for Information Centric Networks (ICN), as infrastructureless access to content, resilient forwarding, and in-network data replication demonstrated notable advantages over the traditional host-to-host approach on the Internet [NDN-EXP1], [NDN-EXP2]. Recent studies [NDN-MAC] have shown that an appropriate mapping to link layer technologies has a large impact on the practical performance of an ICN. This will be even more relevant in the context of IoT communication where nodes often exchange messages via low-power wireless links under lossy conditions. In this memo, we address the base adaptation of data chunks to such link layers for the ICN flavors NDN [NDN] and CCNx [RFC8569], [RFC8609].

The IEEE 802.15.4 [ieee802.15.4] link layer is used in low-power and lossy networks (see "LLN" in [RFC7228]), in which devices are typically battery-operated and constrained in resources. Characteristics of LLNs include an unreliable environment, low bandwidth transmissions, and increased latencies. IEEE 802.15.4 admits a maximum physical layer packet size of 127 bytes. The maximum frame header size is 25 bytes, which leaves 102 bytes for the payload. IEEE 802.15.4 security features further reduce this payload length by up to 21 bytes, yielding a net of 81 bytes for CCNx or NDN packet headers, signatures and content.

6LoWPAN [RFC4944], [RFC6282] is a convergence layer that provides frame formats, header compression and adaptation layer fragmentation for IPv6 packets in IEEE 802.15.4 networks. The 6LoWPAN adaptation introduces a dispatching framework that prepends further information to 6LoWPAN packets, including a protocol identifier for payload and meta information about fragmentation.

Prevalent Type-Length-Value (TLV) based packet formats such as in CCNx and NDN are designed to be generic and extensible. This leads to header verbosity which is inappropriate in constrained environments of IEEE 802.15.4 links. This document presents ICN LoWPAN, a convergence layer for IEEE 802.15.4 motivated by 6LoWPAN. ICN LoWPAN compresses packet headers of CCNx as well as NDN and allows for an increased effective payload size per packet. Additionally, reusing the dispatching framework defined by 6LoWPAN enables compatibility between coexisting wireless deployments of competing network technologies. This also allows to reuse the adaptation layer fragmentation scheme specified by 6LoWPAN for ICN LoWPAN.

ICN LoWPAN defines a more space efficient representation of CCNx and NDN packet formats. This syntactic change is described for CCNx and NDN separately, as the header formats and TLV encodings differ notably. For further reductions, default header values suitable for constrained IoT networks are selected in order to elide corresponding TLVs. Experimental evaluations of the ICN LoWPAN header compression schemes in [ICNLOWPAN] illustrate a reduced message overhead, a shortened message airtime, and an overall decline in power consumption for typical Class 2 [RFC7228] devices compared to uncompressed ICN messages.

In a typical IoT scenario (see (Figure 1)), embedded devices are interconnected via a quasi-stationary infrastructure using a border router (BR) that connects the constrained LoWPAN network by some Gateway with the public Internet. In ICN based IoT networks, non-local Interest and Data messages transparently travel through the BR up and down between a Gateway and the embedded devices situated in the constrained LoWPAN.

```
                             |Gateway Services|
                             -------------------------
                                    |
                               ,--------,
                               |        |
                               |   BR   |
                               |        |
                               '--------'
                                              LoWPAN
                             O            O
                                    O
                           O                O   embedded
                             O      O     O     devices
                              O         O
```

Figure 1: IoT Stub Network

The document has received fruitful reviews by members of the ICN community and the research group (see Acknowledgments) for a period of two years. It is the consensus of ICNRG that this document should be published in the IRTF Stream of the RFC series. This document does not constitute an IETF standard.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119]. The use of the term, "silently ignore" is not defined in RFC 2119. However, the term is used in this document and can be similarly construed.

This document uses the terminology of [RFC7476], [RFC7927], and [RFC7945] for ICN entities.

The following terms are used in the document and defined as follows:

ICN LoWPAN: Information-Centric Networking over Low-power Wireless Personal Area Network

LLN: Low-Power and Lossy Network

CCNx: Content-Centric Networking Architecture

NDN: Named Data Networking Architecture

byte: synonym for octet

nibble: synonym for 4 bits

time-value: a time offset measured in seconds

time-code: an 8-bit encoded time-value

3. Overview of ICN LoWPAN

3.1. Link-Layer Convergence

ICN LoWPAN provides a convergence layer that maps ICN packets onto constrained link-layer technologies. This includes features such as link-layer fragmentation, protocol separation on the link-layer level, and link-layer address mappings. The stack traversal is visualized in Figure 2.

```
      Device 1                                         Device 2
,------------------,          Router          ,------------------,
|  Application   . |    __________________    | ,-> Application  |
|----------------|-|   |    NDN / CCNx    |   |-|----------------|
|  NDN / CCNx    | |   | ,--------------, |   | |    NDN / CCNx  |
|----------------|-|   |-|--------------|-|   |-|----------------|
|  ICN LoWPAN    | |   | |  ICN LoWPAN  | |   | |    ICN LoWPAN  |
|----------------|-|   |-|--------------|-|   |-|----------------|
|  Link-Layer    | |   | |  Link-Layer  | |   | |    Link-Layer  |
'----------------|-'  '-|--------------|-'   '-|----------------'
                 '________'              '__________'
```

Figure 2: ICN LoWPAN convergence layer for IEEE 802.15.4

Section 4 of this document defines the convergence layer for IEEE 802.15.4.

3.2. Stateless Header Compression

ICN LoWPAN also defines a stateless header compression scheme with the main purpose of reducing header overhead of ICN packets. This is of particular importance for link-layers with small MTUs. The stateless compression does not require pre-configuration of global state.

The CCNx and NDN header formats are composed of Type-Length-Value (TLV) fields to encode header data. The advantage of TLVs is its native support of variably structured data. The main disadvantage of TLVs is the verbosity that results from storing the type and length of the encoded data.

The stateless header compression scheme makes use of compact bit fields to indicate the presence of optional TLVs in the uncompressed packet. The order of set bits in the bit fields corresponds to the order of each TLV in the packet. Further compression is achieved by specifying default values and reducing the range of certain header fields.

Figure 3 demonstrates the stateless header compression idea. In this example, the first type of the first TLV is removed and the corresponding bit in the bit field is set. The second TLV represents a fixed-length TLV (e.g., the Nonce TLV in NDN), so that the type and the length fields are removed. The third TLV represents a boolean TLV (e.g., the MustBeFresh selector in NDN) for which the type, length and the value fields are elided.

```
   Uncompressed:

      Variable-length TLV      Fixed-length TLV      Boolean TLV
   ,-----------------------,-----------------------,-------------,
   +-------+-------+-------+-------+-------+-------+------+------+
   |  TYP  |  LEN  |  VAL  |  TYP  |  LEN  |  VAL  |  TYP | LEN  |
   +-------+-------+-------+-------+-------+-------+------+------+

   Compressed:

     +---+---+---+---+---+---+---+---+
     | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |  Bit field
     +---+---+---+---+---+---+---+---+
       |       |                   |
    ,--'       '----,              '- Boolean Value
    |               |
   +-------+-------+-------+
   |  LEN  |  VAL  |  VAL  |
   +-------+-------+-------+
   '---------------'-------'
     Var-len Value  Fixed-len Value
```

Figure 3: Compression using a compact bit field - bits encode the inclusion of TLVs.

Stateless TLV compression for NDN is defined in Section 5. Section 6 defines the stateless TLV compression for CCNx.

The extensibility of this compression is described in Section 4.1.1 and allows future documents to update the compression rules outlined in this manuscript.

3.3. Stateful Header Compression

ICN LoWPAN further employs two orthogonal stateful compression schemes for packet size reductions which are defined in Section 8. These mechanisms rely on shared contexts that are either distributed and maintained in the entire LoWPAN, or are generated on-demand hop-wise on a particular Interest-data path.

The shared context identification is defined in Section 8.1. The hop-wise name compression "en-route" is specified in Section 8.2.

4. IEEE 802.15.4 Adaptation

4.1. LoWPAN Encapsulation

The IEEE 802.15.4 frame header does not provide a protocol identifier for its payload. This causes problems of misinterpreting frames when several network layers coexist on the same link. To mitigate errors, 6LoWPAN defines dispatches as encapsulation headers for IEEE 802.15.4 frames (see Section 5 of [RFC4944]). Multiple LoWPAN encapsulation headers can precede the actual payload and each encapsulation header is identified by a dispatch type.

[RFC8025] further specifies dispatch pages to switch between different contexts. When a LoWPAN parser encounters a "Page switch" LoWPAN encapsulation header, then all following encapsulation headers are interpreted by using a dispatch table as specified by the "Page switch" header. Page 0 and page 1 are reserved for 6LoWPAN. This document uses page TBD1 ("1111 TBD1 (0xFTBD1)") for ICN LoWPAN.

The base dispatch format (Figure 4) is used and extended by CCNx and NDN in Section 5 and Section 6.

```
  0   1   2  ...
+---+---+-----------
| C | P | M |
+---+---+-----------
```

Figure 4: Base dispatch format for ICN LoWPAN

C: Compression

0: The message is compressed.

1: The message is uncompressed.

P: Protocol

0: The included protocol is NDN.

1: The included protocol is CCNx.

M: Message Type

0: The payload contains an Interest message.

1: The payload contains a Data message.

ICN LoWPAN frames with compressed CCNx and NDN messages (C=0) use the extended dispatch format in Figure 5.

```
  0   1   2   3   4  ...
+---+---+---+---+---+---
| 0 | P | M |CID|EXT|
+---+---+---+---+---+---
```

Figure 5: Extended dispatch format for compressed ICN LoWPAN

CID: Context Identifier

0: No context identifiers are present.

1: Context identifier(s) are present (see Section 8.1).

EXT: Extension

0: No extension bytes are present.

1: Extension byte(s) are present (see Section 4.1.1).

The encapsulation format for ICN LoWPAN is displayed in Figure 6.

```
+------...------+------...-----+--------+-------...-------+-----...
| IEEE 802.15.4 | RFC4944 Disp.|  Page  | ICN LoWPAN Disp.| Payl. /
+------...------+------...-----+--------+-------...-------+-----...
```

Figure 6: LoWPAN Encapsulation with ICN-LoWPAN

IEEE 802.15.4: The IEEE 802.15.4 header.

RFC4944 Disp.: Optional additional dispatches defined in Section 5.1 of [RFC4944]

Page: Page Switch. TBD1 for ICN LoWPAN.

ICN LoWPAN: Dispatches as defined in Section 5 and Section 6.

Payload: The actual (un-)compressed CCNx or NDN message.

4.1.1. Dispatch Extensions

Extension bytes allow for the extensibility of the initial compression rule set. The base format for an extension byte is depicted in Figure 7.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| - | - | - | - | - | - | - |EXT|
+---+---+---+---+---+---+---+---+
```

Figure 7: Base format for dispatch extensions.

EXT: Extension

- 0: No other extension byte follows.
- 1: A further extension byte follows.

Extension bytes are numbered according to their order. Future documents MUST follow the naming scheme "EXT_0, EXT_1, ...", when updating or referring to a specific dispatch extension byte. Amendments that require an exchange of configurational parameters between devices SHOULD use manifests to encode structured data in a well-defined format, as, e.g., outlined in [I-D.irtf-icnrg-flic].

4.2. Adaptation Layer Fragmentation

Small payload sizes in the LoWPAN require fragmentation for various network layers. Therefore, Section 5.3 of [RFC4944] defines a protocol-independent fragmentation dispatch type, a fragmentation header for the first fragment, and a separate fragmentation header for subsequent fragments. ICN LoWPAN adopts this fragmentation handling of [RFC4944].

The Fragmentation LoWPAN header can encapsulate other dispatch headers. The order of dispatch types is defined in Section 5 of [RFC4944]. Figure 8 shows the fragmentation scheme. The reassembled ICN LoWPAN frame does not contain any fragmentation headers and is depicted in Figure 9.

```
+------...------+----...----+--------+------...-------+--------...
| IEEE 802.15.4 | Frag. 1st |  Page  |   ICN LoWPAN   | Payload  /
+------...------+----...----+--------+------...-------+--------...

+------...------+----...----+--------...
| IEEE 802.15.4 | Frag. 2nd | Payload  /
+------...------+----...----+--------...

                .
                .
                .

+------...------+----...----+--------...
| IEEE 802.15.4 | Frag. Nth | Payload  /
+------...------+----...----+--------...
```

Figure 8: Fragmentation scheme

```
+------...------+--------+------...-------+--------...
| IEEE 802.15.4 |  Page  |   ICN LoWPAN   | Payload  /
+------...------+--------+------...-------+--------...
```

Figure 9: Reassembled ICN LoWPAN frame

The 6LoWPAN Fragment Forwarding (6FF) [I-D.ietf-6lo-minimal-fragment] is an alternative approach that enables forwarding of fragments without reassembling packets on every intermediate hop. By reusing the 6LoWPAN dispatching framework, 6FF integrates into ICN LoWPAN as seamless as the conventional hop-wise fragmentation. Experimental evaluations [SFR-ICNLOWPAN], however, suggest that a more refined integration can increase the cache utilization of forwarders on a request path.

5. Space-efficient Message Encoding for NDN

5.1. TLV Encoding

The NDN packet format consists of TLV fields using the TLV encoding that is described in [NDN-PACKET-SPEC]. Type and length fields are of variable size, where numbers greater than 252 are encoded using multiple bytes.

If the type or length number is less than "253", then that number is encoded into the actual type or length field. If the number is greater or equals "253" and fits into 2 bytes, then the type or length field is set to "253" and the number is encoded in the next following 2 bytes in network byte order, i.e., from the most significant byte (MSB) to the least significant byte (LSB). If the number is greater than 2 bytes and fits into 4 bytes, then the type or length field is set to "254" and the number is encoded in the subsequent 4 bytes in network byte order. For larger numbers, the type or length field is set to "255" and the number is encoded in the subsequent 8 bytes in network byte order.

In this specification, compressed NDN TLVs make use of a different TLV encoding scheme that reduces size. Instead of using the first byte as a marker for the number of following bytes, the compressed NDN TLV scheme uses a method to chain a variable number of bytes together. If a byte equals "255 (0xFF)", then the following byte will also be interpreted. The actual value of a chain equals the sum of all constituents.

If the type or length number is less than "255", then that number is encoded into the actual type or length field (Figure 10 a). If the type or length number (X) fits into 2 bytes, then the first byte is set to "255" and the subsequent byte equals "X mod 255" (Figure 10 b). Following this scheme, a variable-sized number (X) is encoded using multiple bytes of "255" with a trailing byte containing "X mod 255" (Figure 10 c).

```
    0 1 2 3 4 5 6 7
   +-+-+-+-+-+-+-+-+
a) |   < 255 (X)   | = X
   +-+-+-+-+-+-+-+-+

    0                   1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
b) |      255      |   < 255 (X)   | = 255 + X
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

    0
    0 1 2 3 4 5 6 7
   +-+-+-+-+-+-+-+-+-+-+-.....-+-+-+-+-+-+-+-+-+-+-+
c) |      255      |      255      |   < 255 (X)   | = (N * 255) + X
   +-+-+-+-+-+-+-+-+-+-+-.....-+-+-+-+-+-+-+-+-+-+-+
                           (N)
```

Figure 10: Compressed NDN TLV encoding scheme

5.2. Name TLV Compression

This Name TLV compression encodes length fields of two consecutive NameComponent TLVs into one byte, using a nibble for each. The most significant nibble indicates the length of an immediately following NameComponent TLV. The least significant nibble denotes the length of a subsequent NameComponent TLV. A length of 0 marks the end of the compressed Name TLV. The last length field of an encoded NameComponent is either 0x00 for a name with an even number of components, and 0xYF (Y > 0) if an odd number of components are present. This process limits the length of a NameComponent TLV to 15 bytes, but allows for an unlimited number of components. An example for this encoding is presented in Figure 11.

```
                 Name: /HAW/Room/481/Humid/99

     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 1 1|0 1 0 0|       H       |       A       |       W       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |       R       |       o       |       o       |       m       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 1 1|0 1 0 1|       4       |       8       |       1       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |       H       |       u       |       m       |       i       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |       d       |0 0 1 0|0 0 0 0|       9       |       9       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Name TLV compression for /HAW/Room/481/Humid/99

5.3. Interest Messages

5.3.1. Uncompressed Interest Messages

An uncompressed Interest message uses the base dispatch format (see Figure 4) and sets the C flag to "1" and the P as well as the M flag to "0" (Figure 12). The Interest message is handed to the NDN network stack without modifications.

```
                     0   1   2   3   4   5   6   7
                   +---+---+---+---+---+---+---+---+
                   | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
                   +---+---+---+---+---+---+---+---+
```

Figure 12: Dispatch format for uncompressed NDN Interest messages

5.3.2. Compressed Interest Messages

The compressed Interest message uses the extended dispatch format (Figure 5) and sets the P flag to "0", the C flag to "0" and the M flag to "0". If an Interest message contains TLVs that are not mentioned in the following compression rules, then this message MUST be sent uncompressed.

This specification assumes that a HopLimit TLV is part of the original Interest message. If such HopLimit TLV is not present, it will be inserted with a default value of DEFAULT_NDN_HOPLIMIT prior to the compression.

In the default use case, the Interest message is compressed with the following minimal rule set:

1. The "Type" field of the outermost MessageType TLV is removed.

2. The Name TLV is compressed according to Section 5.2. For this, all NameComponents are expected to be of type GenericNameComponent with a length greater than 0. An ImplicitSha256DigestComponent or ParametersSha256DigestComponent MAY appear at the end of the name. In any other case, the message MUST be sent uncompressed.

3. The Nonce TLV and InterestLifetime TLV are moved to the end of the compressed Interest as illustrated in Figure 13. The InterestLifetime is encoded as described in Section 7. If a lifetime is not a valid time-value, then the lifetime is rounded up to the nearest valid time-value as specified in Section 7.

4. The Type and Length fields of Nonce TLV, HopLimit TLV and InterestLifetime TLV are elided. The Nonce value has a length of 4 bytes and the HopLimit value has a length of 1 byte. The compressed InterestLifetime (Section 7) has a length of 1 byte. The presence of a Nonce and InterestLifetime TLV is deduced from the remaining length to parse. A remaining length of "1" indicates the presence of an InerestLifetime, a length of "4" indicates the presence of a nonce, and a length of "5" indicates the presence of both TLVs.

The compressed NDN LoWPAN Interest message is visualized in Figure 13.

```
        T = Type, L = Length, V = Value
        Lc = Compressed Length, Vc = Compressed Value
        : = optional field, | = mandatory field

        +---------+---------+                 +---------+
        | Msg T   | Msg L   |                 | Msg Lc  |
        +---------+---------+---------+       +---------+
        | Name T  | Name L  | Name V  |       | Name Vc |
        +---------+---------+---------+       +---------+---------+
        : CBPfx T : CBPfx L :                 : FWDH Lc : FWDH Vc :
        +---------+---------+                 +---------+---------+
        : MBFr T  : MBFr L  :                 | HPL V   |
        +---------+---------+---------+  ==>  +---------+---------+
        : FWDH T  : FWDH L  : FWDH V  :       : APM Lc  : APM Vc  :
        +---------+---------+---------+       +---------+---------+
        : NONCE T : NONCE L : NONCE V :       : NONCE V :
        +---------+---------+---------+       +---------+
        :  ILT T  :  ILT L  :  ILT V  :       :  ILT Vc :
        +---------+---------+---------+       +---------+
        :  HPL T  :  HPL L  :  HPL V  :
        +---------+---------+---------+
        :  APM T  :  APM L  :  APM V  :
        +---------+---------+---------+
```

Figure 13: Compression of NDN LoWPAN Interest Message

Further TLV compression is indicated by the ICN LoWPAN dispatch in Figure 14.

```
      0                                       1
      0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    | 0 | 0 | 0 |CID|EXT|PFX|FRE|FWD|APM|DIG|          RSV          |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 14: Dispatch format for compressed NDN Interest messages

CID: Context Identifier See Figure 5.

EXT: Extension

- 0: No extension byte follows.
- 1: Extension byte "EXT_0" follows immediately. See Section 5.3.3.

PFX: CanBePrefix TLV

0: The uncompressed message does not include a CanBePrefix TLV.

1: The uncompressed message does include a CanBePrefix TLV and is removed from the compressed message.

FRE: MustBeFresh TLV

0: The uncompressed message does not include a MustBeFresh TLV.

1: The uncompressed message does include a MustBeFresh TLV and is removed from the compressed message.

FWD: ForwardingHint TLV

0: The uncompressed message does not include a ForwardingHint TLV.

1: The uncompressed message does include a ForwardingHint TLV. The Type field is removed from the compressed message. Further, all link delegation types and link preference types are removed. All included names are compressed according to Section 5.2. If any name is not compressible, the message MUST be sent uncompressed.

APM: ApplicationParameters TLV

0: The uncompressed message does not include an ApplicationParameters TLV.

1: The uncompressed message does include an ApplicationParameters TLV. The Type field is removed from the compressed message.

DIG: ImplicitSha256DigestComponent TLV

0: The name does not include an ImplicitSha256DigestComponent as the last TLV.

1: The name does include an ImplicitSha256DigestComponent as the last TLV. The Type and Length fields are omitted.

RSV: Reserved Must be set to 0.

5.3.3. Dispatch Extension

The "EXT_0" byte follows the description in Section 4.1.1 and is illustrated in Figure 15.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
|  NCS  |          RSV      |EXT|
+---+---+---+---+---+---+---+---+
```

Figure 15: EXT_0 format

NCS: Name Compression Strategy

00: Names are compressed with the default name compression strategy (see Section 5.2).

01: Reserved.

10: Reserved.

11: Reserved.

RSV: Reserved Must be set to 0.

EXT: Extension

0: No extension byte follows.

1: A further extension byte follows immediately.

5.4. Data Messages

5.4.1. Uncompressed Data Messages

An uncompressed Data message uses the base dispatch format and sets the C as well as the M flag to "1" and the P flag to "0" (Figure 16). The Data message is handed to the NDN network stack without modifications.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+
```

Figure 16: Dispatch format for uncompressed NDN Data messages

5.4.2. Compressed Data Messages

The compressed Data message uses the extended dispatch format (Figure 5) and sets the C as well as the P flag to "0". The M flag is set to "1". If a Data message contains TLVs that are not mentioned in the following compression rules, then this message MUST be sent uncompressed.

By default, the Data message is compressed with the following base rule set:

1. The "Type" field of the outermost MessageType TLV is removed.

2. The Name TLV is compressed according to Section 5.2. For this, all NameComponents are expected to be of type GenericNameComponent and to have a length greater than 0. In any other case, the message MUST be sent uncompressed.

3. The MetaInfo TLV Type and Length fields are elided from the compressed Data message.

4. The FreshnessPeriod TLV MUST be moved to the end of the compressed Data message. Type and Length fields are elided and the value is encoded as described in Section 7 as a 1-byte time-code. If the freshness period is not a valid time-value, then the message MUST be sent uncompressed in order to preserve the security envelope of the Data message. The presence of a FreshnessPeriod TLV is deduced from the remaining one byte length to parse.

5. The Type fields of the SignatureInfo TLV, SignatureType TLV and SignatureValue TLV are removed.

The compressed NDN LoWPAN Data message is visualized in Figure 17.

```
        T = Type, L = Length, V = Value
        Lc = Compressed Length, Vc = Compressed Value
        : = optional field, | = mandatory field

        +---------+---------+                 +---------+
        |  Msg T  |  Msg L  |                 |  Msg Lc |
        +---------+---------+---------+       +---------+
        | Name T  | Name L  | Name V  |       | Name Vc |
        +---------+---------+---------+       +---------+---------+
        : Meta T  : Meta L  :                 : CTyp Lc : CType V :
        +---------+---------+---------+       +---------+---------+
        : CTyp T  : CTyp L  : CTyp V  :       : FBID V  :
        +---------+---------+---------+  ==>  +---------+---------+
        : FrPr T  : FrPr L  : FrPr V  :       : CONT Lc : CONT V  :
        +---------+---------+---------+       +---------+---------+
        : FBID T  : FBID L  : FBID V  :       |  Sig Lc |
        +---------+---------+---------+       +---------+---------+
        : CONT T  : CONT L  : CONT V  :       | SInf Lc | SInf Vc |
        +---------+---------+---------+       +---------+---------+
        |  Sig T  |  Sig L  |                 | SVal Lc | SVal Vc |
        +---------+---------+---------+       +---------+---------+
        | SInf T  | SInf L  | SInf V  |       : FrPr Vc :
        +---------+---------+---------+       +---------+
        | SVal T  | SVal L  | SVal V  |
        +---------+---------+---------+
```

Figure 17: Compression of NDN LoWPAN Data Message

Further TLV compression is indicated by the ICN LoWPAN dispatch in Figure 18.

```
      0                                       1
      0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    | 0 | 0 | 1 |CID|EXT|FBI|CON|KLO|              RSV              |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 18: Dispatch format for compressed NDN Data messages

CID: Context Identifier See Figure 5.

EXT: Extension

- 0: No extension byte follows.
- 1: Extension byte "EXT_0" follows immediately. See Section 5.4.3.

FBI: FinalBlockId TLV

0: The uncompressed message does not include a FinalBlockId TLV.

1: The uncompressed message does include a FinalBlockId and it is encoded according to Section 5.2. If the FinalBlockId TLV is not compressible, then the message MUST be sent uncompressed.

CON: ContentType TLV

0: The uncompressed message does not include a ContentType TLV.

1: The uncompressed message does include a ContentType TLV. The Type field is removed from the compressed message.

KLO: KeyLocator TLV

0: If the included SignatureType requires a KeyLocator TLV, then the KeyLocator represents a name and is compressed according to Section 5.2. If the name is not compressible, then the message MUST be sent uncompressed.

1: If the included SignatureType requires a KeyLocator TLV, then the KeyLocator represents a KeyDigest. The Type field of this KeyDigest is removed.

RSV: Reserved Must be set to 0.

5.4.3. Dispatch Extension

The "EXT_0" byte follows the description in Section 4.1.1 and is illustrated in Figure 19.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
|  NCS  |          RSV      |EXT|
+---+---+---+---+---+---+---+---+
```

Figure 19: EXT_0 format

NCS: Name Compression Strategy

00: Names are compressed with the default name compression strategy (see Section 5.2).

01: Reserved.

10: Reserved.

11: Reserved.

RSV: Reserved Must be set to 0.

EXT: Extension

0: No extension byte follows.

1: A further extension byte follows immediately.

6. Space-efficient Message Encoding for CCNx

6.1. TLV Encoding

The generic CCNx TLV encoding is described in [RFC8609]. Type and Length fields attain the common fixed length of 2 bytes.

The TLV encoding for CCNx LoWPAN is changed to the more space efficient encoding described in Section 5.1. Hence NDN and CCNx use the same compressed format for writing TLVs.

6.2. Name TLV Compression

Name TLVs are compressed using the scheme already defined in Section 5.2 for NDN. If a Name TLV contains T_IPID, T_APP, or organizational TLVs, then the name remains uncompressed.

6.3. Interest Messages

6.3.1. Uncompressed Interest Messages

An uncompressed Interest message uses the base dispatch format (see Figure 4) and sets the C as well as the P flag to "1". The M flag is set to "0" (Figure 20). The Interest message is handed to the CCNx network stack without modifications.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+
```

Figure 20: Dispatch format for uncompressed CCNx Interest messages

6.3.2. Compressed Interest Messages

The compressed Interest message uses the extended dispatch format (Figure 5) and sets the C and M flags to "0". The P flag is set to "1". If an Interest message contains TLVs that are not mentioned in the following compression rules, then this message MUST be sent uncompressed.

In the default use case, the Interest message is compressed with the following minimal rule set:

1. The Type and Length fields of the CCNx Message TLV are elided and are obtained from the Fixed Header on decompression.

The compressed CCNx LoWPAN Interest message is visualized in Figure 21.

```
   T = Type, L = Length, V = Value
   Lc = Compressed Length, Vc = Compressed Value
   : = optional field, | = mandatory field

   +-----------------------------+          +-------------------------+
   |    Uncompr. Fixed Header    |          |   Compr. Fixed Header   |
   +-----------------------------+          +-------------------------+
   +---------+---------+---------+          +---------+
   : ILT T   : ILT L   : ILT V   :          : ILT Vc  :
   +---------+---------+---------+          +---------+
   : MSGH T  : MSGH L  : MSGH V  :          : MSGH Vc :
   +---------+---------+---------+          +---------+
   +---------+---------+                    +---------+
   | MSGT T  | MSGT L  |                    | Name Vc |
   +---------+---------+---------+          +---------+
   | Name T  | Name L  | Name V  |    ==>   : KIDR Vc :
   +---------+---------+---------+          +---------+
   : KIDR T  : KIDR L  : KIDR V  :          : OBHR Vc :
   +---------+---------+---------+          +---------+---------+
   : OBHR T  : OBHR L  : OBHR V  :          : PAYL Lc : PAYL V  :
   +---------+---------+---------+          +---------+---------+
   : PAYL T  : PAYL L  : PAYL V  :          : VALG Lc : VALG Vc :
   +---------+---------+---------+          +---------+---------+
   : VALG T  : VALG L  : VALG V  :          : VPAY Lc : VPAY V  :
   +---------+---------+---------+          +---------+---------+
   : VPAY T  : VPAY L  : VPAY V  :
   +---------+---------+---------+
```

Figure 21: Compression of CCNx LoWPAN Interest Message

Further TLV compression is indicated by the ICN LoWPAN dispatch in Figure 22.

```
     0                                       1
     0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
   | 0 | 1 | 0 |CID|EXT|VER|FLG|PTY|HPL|FRS|PAY|ILT|MGH|KIR|CHR|VAL|
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 22: Dispatch format for compressed CCNx Interest messages

CID: Context Identifier See Figure 5.

EXT: Extension

0:      No extension byte follows.

1: Extension byte "EXT_0" follows immediately. See Section 6.3.3.

VER: CCNx protocol version in the fixed header

0: The Version field equals 1 and is removed from the fixed header.

1: The Version field appears in the fixed header.

FLG: Flags field in the fixed header

0: The Flags field equals 0 and is removed from the Interest message.

1: The Flags field appears in the fixed header.

PTY: PacketType field in the fixed header

0: The PacketType field is elided and assumed to be "PT_INTEREST".

1: The PacketType field is elided and assumed to be "PT_RETURN".

HPL: HopLimit field in the fixed header

0: The HopLimit field appears in the fixed header.

1: The HopLimit field is elided and assumed to be "1".

FRS: Reserved field in the fixed header

0: The Reserved field appears in the fixed header.

1: The Reserved field is elided and assumed to be "0".

PAY: Optional Payload TLV

0: The Payload TLV is absent.

1: The Payload TLV is present and the type field is elided.

ILT: Optional Hop-By-Hop InterestLifetime TLV

See Section 6.3.2.1 for further details on the ordering of hop-by-hop TLVs.

0: No InterestLifetime TLV is present in the Interest message.

1: An InterestLifetime TLV is present with a fixed length of 1 byte and is encoded as described in Section 7. The type and length fields are elided. If a lifetime is not a valid time-value, then the lifetime is rounded up to the nearest valid time-value (see Section 7).

MGH: Optional Hop-By-Hop MessageHash TLV

See Section 6.3.2.1 for further details on the ordering of hop-by-hop TLVs.

This TLV is expected to contain a T_SHA-256 TLV. If another hash is contained, then the Interest MUST be sent uncompressed.

0: The MessageHash TLV is absent.

1: A T_SHA-256 TLV is present and the type as well as the length fields are removed. The length field is assumed to represent 32 bytes. The outer Message Hash TLV is omitted.

KIR: Optional KeyIdRestriction TLV

This TLV is expected to contain a T_SHA-256 TLV. If another hash is contained, then the Interest MUST be sent uncompressed.

0: The KeyIdRestriction TLV is absent.

1: A T_SHA-256 TLV is present and the type as well as the length fields are removed. The length field is assumed to represent 32 bytes. The outer KeyIdRestriction TLV is omitted.

CHR: Optional ContentObjectHashRestriction TLV

This TLV is expected to contain a T_SHA-256 TLV. If another hash is contained, then the Interest MUST be sent uncompressed.

0: The ContentObjectHashRestriction TLV is absent.

1: A T_SHA-256 TLV is present and the type as well as the length fields are removed. The length field is assumed to represent 32 bytes. The outer ContentObjectHashRestriction TLV is omitted.

VAL: Optional ValidationAlgorithm and ValidationPayload TLVs

0: No validation related TLVs are present in the Interest message.

1: Validation related TLVs are present in the Interest message. An additional byte follows immediately that handles validation related TLV compressions and is described in Section 6.3.2.2.

6.3.2.1. Hop-By-Hop Header TLVs Compression

Hop-By-Hop Header TLVs are unordered. For an Interest message, two optional Hop-By-Hop Header TLVs are defined in [RFC8609], but several more can be defined in higher level specifications. For the compression specified in the previous section, the Hop-By-Hop TLVs are ordered as follows:

1. Interest Lifetime TLV

2. Message Hash TLV

Note: Other Hop-By-Hop Header TLVs than those two remain uncompressed in the encoded message and they appear in the same order as in the original message, but after the Interest Lifetime TLV and Message Hash TLV.

6.3.2.2. Validation

```
 0       1       2       3       4       5       6       7       8
 +-------+-------+-------+-------+-------+-------+-------+-------+
 |          ValidationAlg        |     KeyID     |      RSV      |
 +-------+-------+-------+-------+-------+-------+-------+-------+
```

Figure 23: Dispatch for Interset Validations

ValidationALg: Optional ValidationAlgorithm TLV

0000: An uncompressed ValidationAlgorithm TLV is included.

0001: A T_CRC32C ValidationAlgorithm TLV is assumed, but no ValidationAlgorithm TLV is included.

0010: A T_CRC32C ValidationAlgorithm TLV is assumed, but no ValidationAlgorithm TLV is included. Additionally, a Sigtime TLV is inlined without a type and a length field.

0011: A T_HMAC-SHA256 ValidationAlgorithm TLV is assumed, but no ValidationAlgorithm TLV is included.

0100: A T_HMAC-SHA256 ValidationAlgorithm TLV is assumed, but no ValidationAlgorithm TLV is included. Additionally, a Sigtime TLV is inlined without a type and a length field.

0101: Reserved.

0110: Reserved.

0111: Reserved.

1000: Reserved.

1001: Reserved.

1010: Reserved.

1011: Reserved.

1100: Reserved.

1101: Reserved.

1110: Reserved.

1111: Reserved.

KeyID: Optional KeyID TLV within the ValidationAlgorithm TLV

00: The KeyId TLV is absent.

01: The KeyId TLV is present and uncompressed.

10: A T_SHA-256 TLV is present and the type field as well as the length fields are removed. The length field is assumed to represent 32 bytes. The outer KeyId TLV is omitted.

11: A T_SHA-512 TLV is present and the type field as well as the length fields are removed. The length field is assumed to represent 64 bytes. The outer KeyId TLV is omitted.

RSV: Reserved Must be set to 0.

The ValidationPayload TLV is present if the ValidationAlgorithm TLV is present. The type field is omitted.

6.3.3. Dispatch Extension

The "EXT_0" byte follows the description in Section 4.1.1 and is illustrated in Figure 24.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
|  NCS  |          RSV          |EXT|
+---+---+---+---+---+---+---+---+
```

Figure 24: EXT_0 format

NCS: Name Compression Strategy

- 00: Names are compressed with the default name compression strategy (see Section 5.2).
- 01: Reserved.
- 10: Reserved.
- 11: Reserved.

RSV: Reserved Must be set to 0.

EXT: Extension

- 0: No extension byte follows.
- 1: A further extension byte follows immediately.

6.4. Content Objects

6.4.1. Uncompressed Content Objects

An uncompressed Content object uses the base dispatch format (see Figure 4) and sets the C, P and M flags to "1" (Figure 25). The Content object is handed to the CCNx network stack without modifications.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+
```

Figure 25: Dispatch format for uncompressed CCNx Content objects

6.4.2. Compressed Content Objects

The compressed Content object uses the extended dispatch format (Figure 5) and sets the P as well as the M flag to "1". The C flag is set to "0". If a Content object contains TLVs that are not mentioned in the following compression rules, then this message MUST be sent uncompressed.

By default, the Content object is compressed with the following base rule set:

1. The PacketType field is elided from the Fixed Header.

2. The Type and Length fields of the CCNx Message TLV are elided and are obtained from the Fixed Header on decompression.

The compressed CCNx LoWPAN Data message is visualized in Figure 26.

```
T = Type, L = Length, V = Value
Lc = Compressed Length, Vc = Compressed Value
: = optional field, | = mandatory field

+-------------------------------+           +---------------------------+
|    Uncompr. Fixed Header      |           |   Compr. Fixed Header     |
+-------------------------------+           +---------------------------+
+---------+---------+---------+             +---------+
: RCT T   : RCT L   : RCT V   :             : RCT Vc  :
+---------+---------+------.--+             +---------+
: MSGH T  : MSGH L  : MSGH V  :             : MSGH Vc :
+---------+---------+---------+             +---------+
+---------+---------+                       +---------+
| MSGT T  | MSGT L  |                       | Name Vc |
+---------+---------+---------+             +---------+
| Name T  | Name L  | Name V  |     ==>     : EXPT Vc :
+---------+---------+---------+             +---------+---------+
: PTYP T  : PTYP L  : PTYP V  :             : PAYL Lc : PAYL V  :
+---------+---------+---------+             +---------+---------+
: EXPT T  : EXPT L  : EXPT V  :             : VALG Lc : VALG Vc :
+---------+---------+---------+             +---------+---------+
: PAYL T  : PAYL L  : PAYL V  :             : VPAY Lc : VPAY V  :
+---------+---------+---------+             +---------+---------+
: VALG T  : VALG L  : VALG V  :
+---------+---------+---------+
: VPAY T  : VPAY L  : VPAY V  :
+---------+---------+---------+
```

Figure 26: Compression of CCNx LoWPAN Data Message

Further TLV compression is indicated by the ICN LoWPAN dispatch in Figure 27.

```
    0                                       1
    0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
  | 0 | 1 | 1 |CID|EXT|VER|FLG|FRS|PAY|RCT|MGH| PLTYP |EXP|VAL|RSV|
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 27: Dispatch format for compressed CCNx Content objects

CID: Context Identifier See Figure 5.

EXT: Extension

0:      No extension byte follows.

1: Extension byte "EXT_0" follows immediately. See Section 6.4.3.

VER: CCNx protocol version in the fixed header

0: The Version field equals 1 and is removed from the fixed header.

1: The Version field appears in the fixed header.

FLG: Flags field in the fixed header See Section 6.3.2.

FRS: Reserved field in the fixed header See Section 6.3.2.

PAY: Optional Payload TLV See Section 6.3.2.

RCT: Optional Hop-By-Hop RecommendedCacheTime TLV

0: The Recommended Cache Time TLV is absent.

1: The Recommended Cache Time TLV is present and the type as well as the length fields are elided.

MGH: Optional Hop-By-Hop MessageHash TLV

See Section 6.4.2.1 for further details on the ordering of hop-by-hop TLVs.

This TLV is expected to contain a T_SHA-256 TLV. If another hash is contained, then the Content Object MUST be sent uncompressed.

0: The MessageHash TLV is absent.

1: A T_SHA-256 TLV is present and the type as well as the length fields are removed. The length field is assumed to represent 32 bytes. The outer Message Hash TLV is omitted.

PLTYP: Optional PayloadType TLV

00: The PayloadType TLV is absent.

01: The PayloadType TLV is absent and T_PAYLOADTYPE_DATA is assumed.

10: The PayloadType TLV is absent and T_PAYLOADTYPE_KEY is assumed.

11: The PayloadType TLV is present and uncompressed.

EXP: Optional ExpiryTime TLV

0: The ExpiryTime TLV is absent.

1: The ExpiryTime TLV is present and the type as well as the length fields are elided.

VAL: Optional ValidationAlgorithm and ValidationPayload TLVs See Section 6.3.2.

RSV: Reserved Must be set to 0.

6.4.2.1. Hop-By-Hop Header TLVs Compression

Hop-By-Hop Header TLVs are unordered. For a Content Object message, two optional Hop-By-Hop Header TLVs are defined in [RFC8609], but several more can be defined in higher level specifications. For the compression specified in the previous section, the Hop-By-Hop TLVs are ordered as follows:

1. Recommended Cache Time TLV

2. Message Hash TLV

Note: Other Hop-By-Hop Header TLVs than those two remain uncompressed in the encoded message and they appear in the same order as in the original message, but after the Recommended Cache Time TLV and Message Hash TLV.

6.4.3. Dispatch Extension

The "EXT_0" byte follows the description in Section 4.1.1 and is illustrated in Figure 28.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
|  NCS  |         RSV       |EXT|
+---+---+---+---+---+---+---+---+
```

Figure 28: EXT_0 format

NCS: Name Compression Strategy

00: Names are compressed with the default name compression strategy (see Section 5.2).

01: Reserved.

10: Reserved.

11: Reserved.

RSV: Reserved Must be set to 0.

EXT: Extension

0: No extension byte follows.

1: A further extension byte follows immediately.

7. Compressed Time Encoding

This document adopts the compact time representation [I-D.gundogan-icnrg-ccnx-timetlv] for relative time values. Exponent and mantissa values are encoded in a 1-byte wide representation as depicted in Figure 29. The exponent occupies the most significant bits, while the mantissa uses the least significant bits.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
|      exponent     | mantissa  |
+---+---+---+---+---+---+---+---+
```

Figure 29: A time-code with exponent and mantissa to encode a logarithmic range time representation.

The mantissa size is set to 3 bits, the exponent size to 5 bits, and a bias of -5 is applied. This allows for a time representation that ranges from tens of milliseconds with high precision to days with low precision. The base unit for time values are seconds. A time-value is calculated using the following formula, where (e) represents the exponent, (m) the mantissa, (m_max = 8) the maximum mantissa value, and (b) the bias.

Subnormal (e == 0): (0 + m/m_max) * 2^(1+b)

Normalized (e > 0): (1 + m/m_max) * 2^(e+b)

The subnormal form provides a gradual underflow from the smallest normalized number towards zero.

This configuration allows for the following ranges:

- Minimum subnormal number: 0 seconds
- Maximum subnormal number: ~0.054688 seconds
- Minimum normalized number: ~0.062500 seconds
- Maximum normalized number: ~3.987284 years

Valid time-values are positive numbers, including 0. An invalid time-value (t, in seconds) MUST be rounded down to the nearest valid time-value using this algorithm, where (e) represents the number of bits for the exponent, (m) the number of bits for the mantissa, and (m_max = 8) the maximum mantissa value. The bias (b) is set to -5 as before.

- e := floor(log2(t/(2^-b)))
- m := floor(8 * (t / 2^(e+b) - 1))

8. Stateful Header Compression

Stateful header compression in ICN LoWPAN enables packet size reductions in two ways. First, common information that is shared throughout the local LoWPAN may be memorized in context state at all nodes and omitted from communication. Second, redundancy in a single Interest-data exchange may be removed from ICN stateful forwarding on a hop-by-hop bases and memorized in en-route state tables.

8.1. LoWPAN-local State

A context identifier (CID) is a byte that refers to a particular conceptual context between network devices and MAY be used to replace frequently appearing information, such as name prefixes, suffixes, or meta information, such as Interest lifetime.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| X |         ContextID         |
+---+---+---+---+---+---+---+---+
```

Figure 30: Context Identifier.

The 7-bit ContextID is a locally-scoped unique identifier that represents contextual state shared between sender and receiver of the corresponding frame (see Figure 30). If set the most significant bit indicates the presence of another, subsequent ContextID byte (see Figure 35).

Context state shared between senders and receivers is removed from the compressed packet prior to sending, and reinserted after reception prior to passing to the upper stack.

The actual information in a context and how it is encoded are out of scope of this document. The initial distribution and maintenance of shared context is out of scope of this document. Frames containing unknown or invalid CIDs MUST be silently discarded.

8.2. En-route State

In CCNx and NDN, Name TLVs are included in Interest messages, and they return in data messages. Returning Name TLVs either equal the original Name TLV, or they contain the original Name TLV as a prefix. ICN LoWPAN reduces this redundancy in responses by replacing Name TLVs with single bytes that represent link-local HopIDs. HopIDs are carried as Context Identifiers (see Section 8.1) of link-local scope as shown in Figure 31.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| X |           HopID           |
+---+---+---+---+---+---+---+---+
```

Figure 31: Context Identifier as HopID.

A HopID is valid if not all ID bits are set to zero and invalid otherwise. This yields 127 distinct HopIDs. If this range (1...127) is exhausted, the messages MUST be sent without en-route state compression until new HopIDs are available. An ICN LoWPAN node that forwards without replacing the name by a HopID (without en-route compression) MUST invalidate the HopID by setting all ID-bits to zero.

While an Interest is traversing, a forwarder generates an ephemeral HopID that is tied to a PIT entry. Each HopID MUST be unique within the local PIT and only exists during the lifetime of a PIT entry. To maintain HopIDs, the local PIT is extended by two new columns: HIDi (inbound HopIDs) and HIDo (outbound HopIDs).

HopIDs are included in Interests and stored on the next hop with the resulting PIT entry in the HIDi column. The HopID is replaced with a newly generated local HopID before the Interest is forwarded. This new HopID is stored in the HIDo column of the local PIT (see Figure 32).

```
       PIT of B      PIT Extension          PIT of C      PIT Extension
   +--------+------++------+------+     +--------+------++------+------+
   | Prefix | Face || HIDi | HIDo |     | Prefix | Face || HIDi | HIDo |
   +========+======++======+======+     +========+======++======+======+
   |  /p0   | F_A  || h_A  | h_B  |     |  /p0   | F_A  || h_A  |      |
   +--------+------++------+------+     +--------+------++------+------+
                       ^       |                            ^
                 store |       '----------------------, ,---' store
                       |                 send         v |
   ,---,         /p0, h_A          ,---,         /p0, h_B          ,---,
   | A | ------------------------> | B | ------------------------> | C |
   '---'                           '---'                           '---'
```

Figure 32: Setting compression state en-route (Interest).

Responses include HopIDs that were obtained from Interests. If the returning Name TLV equals the original Name TLV, then the name is entirely elided. Otherwise, only the matching name prefix is elided and the distinct name suffix is included along with the HopID. When a response is forwarded, the contained HopID is extracted and used to match against the correct PIT entry by performing a lookup on the HIDo column. The HopID is then replaced with the corresponding HopID from the HIDi column prior to forwarding the response (Figure 33).

```
       PIT of B      PIT Extension          PIT of C      PIT Extension
   +--------+------++------+------+     +--------+------++------+------+
   | Prefix | Face || HIDi | HIDo |     | Prefix | Face || HIDi | HIDo |
   +========+======++======+======+     +========+======++======+======+
   |  /p0   | F_A  || h_A  | h_B  |     |  /p0   | F_A  || h_A  |      |
   +--------+------++------+------+     +--------+------++------+------+
                       |       ^                            |
                  send |       '----------------------, ,---' send
                       v                 match        | v
   ,---,                h_A        ,---,                h_B        ,---,
   | A | <------------------------ | B | <------------------------ | C |
   '---'                           '---'                           '---'
```

Figure 33: Eliding Name TLVs using en-route state (data).

It should be noted that each forwarder of an Interest in an ICN LoWPAN network can individually decide whether to participate in en-route compression or not. However, an ICN LoWPAN node SHOULD use en-route compression whenever the stateful compression mechanism is activated.

Note also that the extensions of the PIT data structure are required only at ICN LoWPAN nodes, while regular NDN/CCNx forwarders outside of an ICN LoWPAN domain do not need to implement these extensions.

8.3. Integrating Stateful Header Compression

A CID appears whenever the CID flag is set (see Figure 5). The CID is appended to the last ICN LoWPAN dispatch byte as shown in Figure 34.

```
...-------+--------+-------...-------+--...-+-------...
/  ...    |  Page  | ICN LoWPAN Disp.| CIDs | Payload /
...-------+--------+-------...-------+--...-+-------...
```

Figure 34: LoWPAN Encapsulation with ICN LoWPAN and CIDs

Multiple CIDs are chained together, with the most significant bit indicating the presence of a subsequent CID (Figure 35). This allows to use multiple shared contexts in compressed messages.

The HopID is always included as the very first CID.

```
+-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+-+-+
|1| CID / HopID | --> |1|     CID     | --> |0|     CID     |
+-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+-+-+
```

Figure 35: Chaining of context identifiers.

9. ICN LoWPAN Constants and Variables

This is a summary of all ICN LoWPAN constants and variables.

DEFAULT_NDN_HOPLIMIT: 255

10. Implementation Report and Guidance

The ICN LoWPAN scheme defined in this document has been implemented as an extension of the NDN/CCNx software stack [CCN-LITE] in its IoT version on RIOT [RIOT]. An experimental evaluation for NDN over ICN LOWPAN with varying configurations has been performed in [ICNLOWPAN]. Energy profilings and processing time measurements indicate significant energy savings, while amortized costs for processing show no penalties.

10.1. Preferred Configuration

The header compression performance depends on certain aspects and configurations. It works best for the following cases:

o Relative time values use a compressible encoding as per Section 7.

- o Contextual state (e.g., prefixes) is distributed, such that long names can be elided from Interest and data messages.

- o Frequently used TLV type numbers for CCNx and NDN stay in the lower range (< 255).

Name components are of GenericNameComponent type and are limited to a length of 15 bytes to enable compression for all messages.

10.2. Further Experimental Deployments

An investigation of ICN LoWPAN in large-scale deployments with varying traffic patterns using larger samples of the different board types available remains as future work. This document will be revised to progress it to the Standards Track, once sufficient operational experience has been acquired. Experience reports are encouraged, particularly in the following areas:

- o The name compression scheme (Section 5.2) is optimized for short name components of GenericNameComponent type. An empirical study on name lengths in different deployments of selected use cases, such as smart home, smart city, and industrial IoT can provide meaningful reports on necessary name component types and lengths. A conclusive outcome helps to understand whether and how extension mechanisms are needed (Section 5.3.3). As a preliminary analysis, [ICNLOWPAN] investigates the effectiveness of the proposed compression scheme with URLs obtained from the WWW. Studies on CoAP [RFC7252] deployments can offer additional insights on naming schemes in the IoT.

- o The fragmentation scheme (Section 4.2) inherited from 6LoWPAN allows for a transparent, hop-wise reassembly of CCNx or NDN packets. Fragment forwarding [I-D.ietf-6lo-minimal-fragment] with selective fragment recovery [I-D.ietf-6lo-fragment-recovery] can improve the end-to-end latency and reliability, while it reduces buffer requirements on forwarders. Initial evaluations ([SFR-ICNLOWPAN]) show that a naive integration of these upcoming fragmentation features into ICN LoWPAN renders the hop-wise content replication inoperative, since Interest and data messages are reassembled end-to-end. More deployment experiences are necessary to gauge the feasibility of different fragmentation schemes in ICN LoWPAN.

- o Context state (Section 8.1) holds information that is shared between a set of devices in a LoWPAN. Fixed name prefixes and suffixes are good candidates to be distributed to all nodes in order to elide them from request and response messages. More experience and a deeper inspection of currently available and upcoming protocol features is necessary to identify other protocol fields.

- The distribution and synchronization of contextual state can potentially be adopted from Section 7.2 of [RFC6775], but requires further evaluations. While 6LoWPAN uses the Neighbor Discovery protocol to disseminate state, CCNx and NDN deployments are missing out on a standard mechanism to bootstrap and manage configurations.

- The stateful en-route compression (Section 8.2) supports a limited number of 127 distinct HopIDs that can be simultaneously in use on a single node. Complex deployment scenarios that make use of multiple, concurrent requests can provide a better insight on the number of open requests stored in the Pending Interest Table of memory-constrained devices. This number can serve as an upper-bound and determines whether the HopID length needs to be resized to fit more HopIDs to the cost of additional header overhead.

- Multiple implementations that generate and deploy the compression options of this memo in different ways will also add to the experience and understanding of the benefits and limitations of the proposed schemes. Different reports can help to illuminate on the complexity of implementing ICN LoWPAN for constrained devices, as well as on maintaining interoperability with other implementations.

11. Security Considerations

Main memory is typically a scarce resource of constrained networked devices. Fragmentation as described in this memo preserves fragments and purges them only after a packet is reassembled, which requires a buffering of all fragments. This scheme is able to handle fragments for distinctive packets simultaneously, which can lead to overflowing packet buffers that cannot hold all necessary fragments for packet reassembly. Implementers are thus urged to make use of appropriate buffer replacement strategies for fragments. The upcoming minimal fragment forwarding [I-D.ietf-6lo-minimal-fragment] can potentially prevent fragment buffer saturation in forwarders.

The stateful header compression generates ephemeral HopIDs for incoming and outgoing Interests and consumes them on returning Data packets. Forged Interests can deplete the number of available HopIDs, thus leading to a denial of compression service for subsequent content requests.

To further alleviate the problems caused by forged fragments or Interest initiations, proper protective mechanisms for accessing the link-layer should be deployed. IEEE 802.15.4, e.g., provides capabilities to protect frames and restrict them to a point-to-point link, or a group of devices.

12. IANA Considerations

12.1. Reserving Space in the 6LoWPAN Dispatch Type Field Registry

IANA has assigned dispatch values of the "6LoWPAN Dispatch Type Field" registry [RFC4944][RFC8025] with Page TBD1 for ICN LoWPAN. Table 1 represents updates to the registry.

Bit Pattern	Page	Header Type
00 000000	TBD1	Uncompressed NDN Interest messages
00 100000	TBD1	Uncompressed NDN Data messages
01 000000	TBD1	Uncompressed CCNx Interest messages
01 100000	TBD1	Uncompressed CCNx Content Object messages
10 0xxxxx	TBD1	Compressed NDN Interest messages
10 1xxxxx	TBD1	Compressed NDN Data messages
11 0xxxxx	TBD1	Compressed CCNx Interest messages
11 1xxxxx	TBD1	Compressed CCNx Content Object messages

Table 1: Dispatch types for NDN and CCNx with page TBD1.

13. References

13.1. Normative References

[ieee802.15.4]
"IEEE Std. 802.15.4-2015", April 2016, <https://standards.ieee.org/findstds/standard/802.15.4-2015.html>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <https://www.rfc-editor.org/info/rfc4944>.

[RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <https://www.rfc-editor.org/info/rfc6282>.

[RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <https://www.rfc-editor.org/info/rfc6775>.

13.2. Informative References

[CCN-LITE]
"CCN-lite: A lightweight CCNx and NDN implementation", <http://ccn-lite.net/>.

[I-D.gundogan-icnrg-ccnx-timetlv]
Gundogan, C., Schmidt, TC., Oran, D., and M. Waehlisch, "An Alternative Delta Time encoding for CCNx using Interval Time from RFC5497", draft-gundogan-icnrg-ccnx-timetlv-00 (work in progress), November 2019.

[I-D.ietf-6lo-fragment-recovery]
Thubert, P., "6LoWPAN Selective Fragment Recovery", draft-ietf-6lo-fragment-recovery-21 (work in progress), March 2020.

[I-D.ietf-6lo-minimal-fragment]
Watteyne, T., Thubert, P., and C. Bormann, "On Forwarding 6LoWPAN Fragments over a Multihop IPv6 Network", draft-ietf-6lo-minimal-fragment-15 (work in progress), March 2020.

[I-D.irtf-icnrg-flic]
Tschudin, C., Wood, C., Mosko, M., and D. Oran, "File-Like ICN Collections (FLIC)", draft-irtf-icnrg-flic-02 (work in progress), November 2019.

[ICNLOWPAN]
Gundogan, C., Kietzmann, P., Schmidt, TC., and M. Waehlisch, "ICNLoWPAN -- Named-Data Networking in Low Power IoT Networks", Proc. of 18th IFIP Networking Conference, May 2019, <https://doi.org/10.23919/IFIPNetworking.2019.8816850>.

[NDN] Jacobson, V., Smetters, D., Thornton, J., and M. Plass, "Networking Named Content", 5th Int. Conf. on emerging Networking Experiments and Technologies (ACM CoNEXT), 2009, <https://doi.org/10.1145/1658939.1658941>.

[NDN-EXP1] Baccelli, E., Mehlis, C., Hahm, O., Schmidt, TC., and M. Waehlisch, "Information Centric Networking in the IoT: Experiments with NDN in the Wild", Proc. of 1st ACM Conf. on Information-Centric Networking (ICN-2014) ACM DL, pp. 77-86, September 2014, <http://dx.doi.org/10.1145/2660129.2660144>.

[NDN-EXP2] Gundogan, C., Kietzmann, P., Lenders, M., Petersen, H., Schmidt, TC., and M. Waehlisch, "NDN, CoAP, and MQTT: A Comparative Measurement Study in the IoT", Proc. of 5th ACM Conf. on Information-Centric Networking (ICN-2018) ACM DL, pp. 159-171, September 2018, <https://doi.org/10.1145/3267955.3267967>.

[NDN-MAC] Kietzmann, P., Gundogan, C., Schmidt, TC., Hahm, O., and M. Waehlisch, "The Need for a Name to MAC Address Mapping in NDN: Towards Quantifying the Resource Gain", Proc. of 4th ACM Conf. on Information-Centric Networking (ICN-2017) ACM DL, pp. 36-42, September 2017, <https://doi.org/10.1145/3125719.3125737>.

[NDN-PACKET-SPEC] "NDN Packet Format Specification", <https://named-data.net/doc/NDN-packet-spec/0.3/>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC7476] Pentikousis, K., Ed., Ohlman, B., Corujo, D., Boggia, G., Tyson, G., Davies, E., Molinaro, A., and S. Eum, "Information-Centric Networking: Baseline Scenarios", RFC 7476, DOI 10.17487/RFC7476, March 2015, <https://www.rfc-editor.org/info/rfc7476>.

[RFC7927] Kutscher, D., Ed., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlisch, "Information-Centric Networking (ICN) Research Challenges", RFC 7927, DOI 10.17487/RFC7927, July 2016, <https://www.rfc-editor.org/info/rfc7927>.

[RFC7945] Pentikousis, K., Ed., Ohlman, B., Davies, E., Spirou, S., and G. Boggia, "Information-Centric Networking: Evaluation and Security Considerations", RFC 7945, DOI 10.17487/RFC7945, September 2016, <https://www.rfc-editor.org/info/rfc7945>.

[RFC8025] Thubert, P., Ed. and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch", RFC 8025, DOI 10.17487/RFC8025, November 2016, <https://www.rfc-editor.org/info/rfc8025>.

[RFC8569] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", RFC 8569, DOI 10.17487/RFC8569, July 2019, <https://www.rfc-editor.org/info/rfc8569>.

[RFC8609] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format", RFC 8609, DOI 10.17487/RFC8609, July 2019, <https://www.rfc-editor.org/info/rfc8609>.

[RIOT] Baccelli, E., Gundogan, C., Hahm, O., Kietzmann, P., Lenders, MS., Petersen, H., Schleiser, K., Schmidt, TC., and M. Waehlisch, "RIOT: an Open Source Operating System for Low-end Embedded Devices in the IoT", IEEE Internet of Things Journal Vol. 5, No. 6, p. 4428-4440, December 2018, <https://doi.org/10.1109/JIOT.2018.2815038>.

[SFR-ICNLOWPAN]
Lenders, M., Gundogan, C., Schmidt, TC., and M. Waehlisch, "Connecting the Dots: Selective Fragment Recovery in ICNLoWPAN", Proc. of 7th ACM Conf. on Information-Centric Networking (ICN-2020) ACM DL, pp. 70-76, September 2020, <https://doi.org/10.1145/3405656.3418719>.

[TLV-ENC-802.15.4]
"CCN and NDN TLV encodings in 802.15.4 packets", <https://datatracker.ietf.org/meeting/interim-2015-icnrg-01/materials/slides-interim-2015-icnrg-1-2>.

[WIRE-FORMAT-CONSID]
"CCN/NDN Protocol Wire Format and Functionality Considerations", <https://datatracker.ietf.org/meeting/interim-2015-icnrg-01/materials/slides-interim-2015-icnrg-1-8>.

Appendix A. Estimated Size Reduction

In the following a theoretical evaluation is given to estimate the gains of ICN LoWPAN compared to uncompressed CCNx and NDN messages.

We assume that "n" is the number of name components, "comps_n" denotes the sum of n name component lengths. We also assume that the length of each name component is lower than 16 bytes. The length of the content is given by "clen". The lengths of TLV components is specific to the CCNx or NDN encoding and outlined below.

A.1. NDN

The NDN TLV encoding has variable-sized TLV fields. For simplicity, the 1 byte form of each TLV component is assumed. A typical TLV component therefore is of size 2 (type field + length field) + the actual value.

A.1.1. Interest

Figure 36 depicts the size requirements for a basic, uncompressed NDN Interest containing a CanBePrefix TLV, a MustBeFresh TLV, a InterestLifetime TLV set to 4 seconds and a HopLimit TLV set to 6. Numbers below represent the amount of bytes.

```
------------------------------------,
Interest TLV           = 2          |
  ---------------------,            |
  Name                 |  2 +       |
    NameComponents     = 2n +       |
                       |  comps_n   |
  ---------------------'            = 21 + 2n + comps_n
  CanBePrefix          = 2          |
  MustBeFresh          = 2          |
  Nonce                = 6          |
  InterestLifetime     = 4          |
  HopLimit             = 3          |
------------------------------------'
```

Figure 36: Estimated size of an uncompressed NDN Interest

Figure 37 depicts the size requirements after compression.

```
         ------------------------------------,
         Dispatch Page Switch    = 1         |
         NDN Interset Dispatch   = 2         |
         Interest TLV            = 1         |
         -----------------------,            |
         Name                   |            |
           NameComponents       = n/2 +      = 10 + n/2 + comps_n
                                |  comps_n   |
         -----------------------'            |
         Nonce                  = 4          |
         HopLimit               = 1          |
         InterestLifetime       = 1          |
         ------------------------------------'
```

Figure 37: Estimated size of a compressed NDN Interest

The size difference is:
11 + 1.5n bytes.

For the name "/DE/HH/HAW/BT7", the total size gain is 17 bytes, which is 43% of the uncompressed packet.

A.1.2. Data

Figure 38 depicts the size requirements for a basic, uncompressed NDN Data containing a FreshnessPeriod as MetaInfo. A FreshnessPeriod of 1 minute is assumed and the value is encoded using 1 byte. An HMACWithSha256 is assumed as signature. The key locator is assumed to contain a Name TLV of length klen.

```
      ------------------------------------,
      Data TLV                = 2         |
        ----------------------,           |
        Name                  |  2 +      |
          NameComponents      = 2n +      |
                              |  comps_n  |
        ----------------------'           |
        ----------------------,           |
        MetaInfo              |           |
          FreshnessPeriod     = 6         |
                              |           = 53 + 2n + comps_n +
        ----------------------'           |  clen + klen
        Content               = 2 + clen  |
        ----------------------,           |
        SignatureInfo         |           |
          SignatureType       |           |
            KeyLocator        = 41 + klen |
        SignatureValue        |           |
          DigestSha256        |           |
        ----------------------'           |
      ------------------------------------'
```

Figure 38: Estimated size of an uncompressed NDN Data

Figure 39 depicts the size requirements for the compressed version of the above Data packet.

```
      ------------------------------------,
      Dispatch Page Switch    = 1         |
      NDN Data Dispatch       = 2         |
      ------------------------,           |
      Name                    |           |
        NameComponents        = n/2 +     |
                              |  comps_n  = 38 + n/2 + comps_n +
      ------------------------'           |  clen + klen
      Content                 = 1 + clen  |
      KeyLocator              = 1 + klen  |
      DigestSha256            = 32        |
      FreshnessPeriod         = 1         |
      ------------------------------------'
```

Figure 39: Estimated size of a compressed NDN Data

The size difference is:
15 + 1.5n bytes.

For the name "/DE/HH/HAW/BT7", the total size gain is 21 bytes.

A.2. CCNx

The CCNx TLV encoding defines a 2-byte encoding for type and length fields, summing up to 4 bytes in total without a value.

A.2.1. Interest

Figure 40 depicts the size requirements for a basic, uncompressed CCNx Interest. No Hop-By-Hop TLVs are included, the protocol version is assumed to be 1 and the reserved field is assumed to be 0. A KeyIdRestriction TLV with T_SHA-256 is included to limit the responses to Content Objects containing the specific key.

```
------------------------------------,
Fixed Header            = 8         |
Message                 = 4         |
  ---------------------,            |
  Name                 |  4 +       |= 56 + 4n + comps_n
    NameSegments        = 4n +      |
                       |  comps_n   |
  ---------------------'            |
  KeyIdRestriction      = 40        |
------------------------------------'
```

Figure 40: Estimated size of an uncompressed CCNx Interest

Figure 41 depicts the size requirements after compression.

```
------------------------------------,
Dispatch Page Switch    = 1         |
CCNx Interest Dispatch  = 2         |
Fixed Header            = 3         |
-----------------------,            |
Name                   |            |= 38 + n/2 + comps_n
  NameSegments          = n/2 +     |
                       |  comps_n   |
-----------------------'            |
T_SHA-256               = 32        |
------------------------------------'
```

Figure 41: Estimated size of a compressed CCNx Interest

The size difference is:
18 + 3.5n bytes.

For the name "/DE/HH/HAW/BT7", the size is reduced by 53 bytes, which is 53% of the uncompressed packet.

A.2.2. Content Object

Figure 42 depicts the size requirements for a basic, uncompressed CCNx Content Object containing an ExpiryTime Message TLV, an HMAC_SHA-256 signature, the signature time and a hash of the shared secret key. In the fixed header, the protocol version is assumed to be 1 and the reserved field is assumed to be 0

```
------------------------------------,
Fixed Header            = 8         |
Message                 = 4         |
  ---------------------,            |
  Name                 |  4 +       |
    NameSegments       = 4n +       |
                       |  comps_n   |
  ---------------------'            |
  ExpiryTime           = 12         = 124 + 4n + comps_n + clen
  Payload              = 4 + clen   |
  ---------------------,            |
  ValidationAlgorithm  |            |
    T_HMAC-256         = 56         |
      KeyId            |            |
    SignatureTime      |            |
  ---------------------'            |
  ValidationPayload    = 36         |
------------------------------------'
```

Figure 42: Estimated size of an uncompressed CCNx Content Object

Figure 43 depicts the size requirements for a basic, compressed CCNx Data.

```
------------------------------------,
Dispatch Page Switch    = 1         |
CCNx Content Dispatch   = 3         |
Fixed Header            = 2         |
-----------------------,            |
Name                   |            |
  NameSegments         = n/2 +      |
                       |  comps_n   = 89 + n/2 + comps_n + clen
-----------------------'            |
ExpiryTime             = 8          |
Payload                = 1 + clen   |
T_HMAC-SHA256          = 32         |
SignatureTime          = 8          |
ValidationPayload      = 34         |
------------------------------------'
```

Figure 43: Estimated size of a compressed CCNx Data Object

The size difference is:
35 + 3.5n bytes.

For the name "/DE/HH/HAW/BT7", the size is reduced by 70 bytes, which is 40% of the uncompressed packet containing a 4-byte payload.

Acknowledgments

This work was stimulated by fruitful discussions in the ICNRG research group and the communities of RIOT and CCNlite. We would like to thank all active members for constructive thoughts and feedback. In particular, the authors would like to thank (in alphabetical order) Peter Kietzmann, Dirk Kutscher, Martine Lenders, Colin Perkins, Junxiao Shi. The hop-wise stateful name compression was brought up in a discussion by Dave Oran, which is gratefully acknowledged. Larger parts of this work are inspired by [RFC4944] and [RFC6282]. Special mentioning goes to Mark Mosko as well as G.Q. Wang and Ravi Ravindran as their previous work in [TLV-ENC-802.15.4] and [WIRE-FORMAT-CONSID] provided a good base for our discussions on stateless header compression mechanisms. Many thanks also to Carsten Bormann, who contributed in-depth comments during the IRSG review. This work was supported in part by the German Federal Ministry of Research and Education within the projects I3 and RAPstore.

Authors' Addresses

Cenk Gundogan
HAW Hamburg
Berliner Tor 7
Hamburg D-20099
Germany

Phone: +4940428758067
EMail: cenk.guendogan@haw-hamburg.de
URI: http://inet.haw-hamburg.de/members/cenk-gundogan

Thomas C. Schmidt
HAW Hamburg
Berliner Tor 7
Hamburg D-20099
Germany

EMail: t.schmidt@haw-hamburg.de
URI: http://inet.haw-hamburg.de/members/schmidt

Matthias Waehlisch
link-lab & FU
Berlin
Hoenower Str. 35
Berlin D-10318
Germany

EMail: mw@link-lab.net
URI: http://www.inf.fu-berlin.de/~waehl

Christopher Scherb
University of
Basel
Spiegelgasse 1
Basel CH-4051
Switzerland

EMail: christopher.scherb@unibas.ch

Claudio Marxer
University of
Basel
Spiegelgasse 1
Basel CH-4051
Switzerland

EMail: claudio.marxer@unibas.ch

Christian Tschudin
University of
Basel
Spiegelgasse 1
Basel CH-4051
Switzerland

EMail: christian.tschudin@unibas.ch

icnrg J. Auge, Ed.
Internet-Draft G. Carofiglio
Intended status: Informational L. Muscariello
Expires: 11 December 2020 M. Papalini
Cisco Systems Inc.
June 2020

MAP-Me : Managing Anchorless Mobility in Content Centric Networking
draft-irtf-icnrg-mapme-05

Abstract

Consumer mobility is supported in ICN by design, in virtue of its connectionless pull-based communication model; producer mobility though is not natively supported. This document describes MAP-Me, an anchor-less solution to manage micro-mobility of content producers in the CCN (Content Centric Networking) and NDN (Named Data Networking) architectures, with support for latency-sensitive applications. MAP-Me consists in the combination of two data plane protocols, triggered by producer movements, and leveraging ICN named-based data plane. The main protocol consists in a lightweight FIB update process, complemented by a mechanism of local notification and scoped discovery suitable for low latency applications and fast mobility.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

With the phenomenal spread of portable user devices, mobility has become a basic requirement for almost any communication network as well as a compelling feature to integrate in the next generation networks (5G). The need for a mobility-management paradigm to apply within IP networks has striven a lot of efforts in research and standardization bodies (IETF, 3GPP among others), all resulting in a complex access-dependent set of mechanisms implemented via a dedicated control infrastructure. The complexity and lack of flexibility of such approaches (e.g. Mobile IP) calls for a radically new solution dismantling traditional assumptions like tunneling and anchoring of all mobile communications into the network core. This is particularly important with the increase in rates and mobile nodes (IoT), a vast amount of which never moves.

The Information Centric Network (ICN) paradigm brings native support for mobility, security, and storage within the network architecture, hence emerging as a promising 5G technology candidate. Specifically on mobility management, ICN has the potential to relieve limitations of the existing approaches by leveraging its primary feature, the redefinition of packet forwarding based on "names" rather than "network addresses". Removing the dependence on location identifiers is a first step in the direction of removing the need for any anchoring of communications into fixed network nodes, which may considerably simplify and improve mobility management. Within the ICN paradigm, several architectures have been proposed, as reported in [SURVEY12] and [SURVEY14].

As a direct result of CCN/NDN design principles, consumer mobility is natively supported: a change in physical location for the consumer does not translate into a change in the data plane like for IP. The retransmission of requests for data not yet received by the consumer takes place without involving any signaling to the network. Producer mobility and realtime group communications present more challenges, depending on the frequency of movements, latency requirements, and content lifetime. The topology does not reflect the naming structure, and the mobility management process has to preserve key functionalities such as multipath, caching, etc. In all cases, beyond providing connectivity guarantees, additional transport-level mechanisms might be required to protect the flow performance (see [WLDR] for instance).

MAP-Me aims at tackling such problems by exploiting key CCN/NDN characteristics. Previous attempts have been made in CCN/NDN (and ICN in general) literature to go beyond the traditional IP approaches, by using the existing CCN/NDN request/data packet structures to trace producer movements and to dynamically build a reverse-forwarding path (see [SURVEY16b] for a survey). They still rely on a stable home address to inform about producer movements or on buffering of incoming requests at the producer's previous point of attachment (PoA), which prevents support for latency-sensitive streaming applications. The approach presented in this document makes a particular focus on this class of applications (e.g. live streaming or videoconferencing) as they have the most stringent performance requirements: negligible per-packet loss-rate and delays. In addition, they typically originate from a single producer and don't allow for the use of caching.

MAP-Me defines a name-based mechanism operating in the forwarding plane and completely removing any anchoring, while aiming at latency minimization. Its performance and guarantees of correctness, stability and bounded stretch are analyzed in [MAPME].

2. MAP-Me overview

2.1. Anchor-less mobility management

Many efforts have been made to define mobility-management models for IP networks in the last two decades, resulting in a variety of complex, often not implemented, proposals. A survey of these approaches is proposed in [RFC6301]. Likewise, within ICN, different approaches to mobility management have been presented [SURVEY13]. Specifically for the CCN/NDN solutions, several surveys of mobility-management approaches can be found [SURVEY16a] [SURVEY16b].

We follow here the classification presented in [MAPME] which highlights their reliance on indirection/rendez-vous points. In particular, a new class of anchor-less approaches is introduced, in which the present proposal fits. Such solutions are less common and have been introduced in ICN to remove the need for anchor points in the data plane, but also in the control plane in the form of resolution or mapping services. These solutions completely remove the use of locators and extend the ICN forwarding mechanisms with mobility support.

2.2. Design principles

* *Micro-Mobility* : MAP-Me addresses micro (e.g. intra Autonomous Systems) producer mobility. Addressing macro-mobility is a non-goal of the proposal. We are focusing here on complementary mechanisms able to provide a fast and lightweight handover, preserving the performance of flows in progress.

- *Control Plane Agnostic* : MAP-Me _is control-plane agnostic as it does not rely on routing updates or path computation_, which would be too slow and too costly, but rather works at a faster timescale propagating forwarding updates on a single path. It also leverages real-time notifications left as breadcrumbs by the producer to enable live tracking of its content prefixes and avoid buffering at intermediate nodes. MAP-Me shares the use of data plane mechanisms for ensuring connectivity with [DATAPLANE] which was originally proposed for link failures. This enables the support of high-speed mobility and real-time group applications. In addition, MAP-Me mobility updates are issued at prefix granularity, rather than content or chunk/packet granularity, to minimize signaling overhead and temporary state kept by in-network nodes, and scale to large and dynamic mobile networks.

* *Access-agnostic* : MAP-Me handles mobility at Layer 3 and is designed to be access-agnostic, to cope with highly heterogeneous wireless access and multi-homed/mobile users.

* *Decentralized and localized* : MAP-Me is designed to be fully _decentralized_, to enhance robustness w.r.t. centralized mobility management proposals subject to single point-of-passage problem. MAP-Me updates are _localized_ and affect a minimum number of routers at the edge of the network to restore connectivity. This effectively realizes traffic off-load close to the end-users.

* *Transparent* : MAP-Me does not involve any name nor modifications to basic request/reply operations to be compatible with standard CCN/NDN design and to avoid issues caused by name modifications like triangular routing, caching degradation, or security vulnerabilities. It does not require consumers or producers to be aware of the mobility of the remote endpoint, nor to perform any handover prediction.

* *Robust* : to network conditions (e.g. routing failure, wireless or congestion losses, and delays), by implementing hop-by-hop retransmissions of mobility updates.

2.3. MAP-Me protocols

As a data plane protocol, MAP-Me handles producer mobility events by means of dynamic FIB updates with the objective of minimizing unreachability of the producer. It relies on the existence of a routing protocol responsible for creating/updating the FIB of all routers, possibly with multipath routes, and for managing network failures (eg. [NLSR]).

MAP-Me is composed of:

* an Update protocol, detailed in Section 3, which is the central component of the proposal;

* a Notification/Discovery protocol, presented in Section 4, which is coupled with the Update protocol to enhance reactivity for realtime/latency-sensitive application, and reduce overhead during fast mobility events.

3. Update protocol

3.1. Rationale

The rationale behind MAP-Me is that the producer announces its movements to the network for all served prefixes, by sending a special Interest packet - named Interest Update (IU) - to "itself" after it reattaches to the network. Such a message looks like a regular Interest packet named with the prefix advertised by the producer. As such, it is forwarded according to the information stored in the FIBs of traversed routers towards all previous locations of the producer known by router FIBs. A special flag carried in the header of the IU enables all routers on the path to identify the Interest as a mobility update and to process it accordingly to update their FIBs (a detailed description of the IU processing is provided in Section 5.3).

The key aspect of the proposal is that it removes the need for a stable home address by directly leveraging name-based forwarding information created by CCN/NDN routing protocols, and eventually further updated due to mobility. FIB updates are triggered by the reception of mobility updates in a fully decentralized way and allow an on-the-fly modification to point to the latest known location of the producer.

3.2. Update propagation

The role of the update process is to quickly restore global reachability of mobile prefixes with low signaling overhead, while introducing a bounded maximum path stretch (the ratio between the selected and the shortest path in terms of hops).

Let us illustrate its behavior through an example where a single producer serving prefix /p moves from position P0 to P1 and so on. Figure 1 (a) shows the initial tree formed by the forwarding paths to the name prefix /p, and on which any IU initiated by the producer will propagate.

```
                            +---+                        +---+
                            | 0 | P0                      | 0 | P0
                            +---+                         +---+
                             ^ ^                           ^ ^
                            /   \                         /   \
                        +---+    +---+                +---+    +---+
                        | 0 |    | 0 |                | 0 |    | 0 |
                        +---+    +---+                +---+    +---+
                         ^ ^                           ^ ^
                        /   \                         /   \
                    +---+    +---+                +---+    +---+
                    | 0 |    | 0 |                | 0 |    | 0 |
                    +---+    +---+             A  +---+    +---+
                     ^ ^                  IU1 /    ^ ^
                    /   \                    /    /   \
                +---+    +---+          .... .+---+.   +---+
                | 0 |    | 0 |         .   P1 | 1 | .  | 0 |
                +---+    +---+        .       +---+ .  +---+
                 ^ ^                  .        ^ ^    .
                /   \                 .       /   \     .
            +---+    +---+            .   +---+    +---+  .
            | 0 |    | 0 |            .   | 0 |    | 0 |  .
            +---+    +---+            .   +---+    +---+ .
                                       ..................

                     (a)                       (b)
```

Figure 1: IU propagation example

```
                                          .................
                   +---+               ...       +---+     ..
                   | 0 | P0           .          | 1 | P0     .
                   +---+             .        A  +---+        .
                    ^ ^             .  IU(1) /    / ^         .
                   /   \            .       /    V   \        .
               +---+    +---+      .         +---+    +---+   .
               | 0 |    | 0 |     .          | 1 |    | 0 |   .
               +---+    +---+    .        A  +---+    +---+   .
                ^ ^              . IU(1) /    / ^           .
               /   \            .       /    V   \         .
   ........+---+.   +---+      .         +---+    +---+   .
  .        | 1 | .  | 0 |     .          | 1 |    | 0 |   .
 .  FIB    +---+ .  +---+    .        A  +---+    +---+   .
. updated   / ^   .          . IU(1) /    / ^            .
.          V   \   ....      .      /    V   \          .
.      +---+    +---+  .     .       +---+    +---+   .
.  P1  | 1 |    | 0 |  .     .    P1 | 1 |    | 0 |   .
.      +---+    +---+  .     .       +---+    +---+   .
.       ^ ^            .     .        ^ ^            .
.      /   \          .       .      /   \         .
.  +---+    +---+    .        .  +---+    +---+  .
.  | 0 |    | 0 |   .         .  | 0 |    | 0 |  .
.  +---+    +---+  .          .  +---+    +---+  .
 ..................            ..................

            (a)                       (b)
```

Figure 2: IU propagation example

```
                            +---+
                            | 1 | P1
                            +---+
                           ^  A  ^
                         /    |    \
                    +---+   +---+   +---+
                    | 0 |   | 0 |   | 1 |
                    +---+   +---+   +---+
                                     ^ ^
                                    /   \
                                +---+    +---+
                                | 0 |    | 1 |
                                +---+    +---+
                                          ^ ^
                                         /   \
                                     +---+    +---+
                                 P0  | 1 |    | 0 |
                                     +---+    +---+
                                      ^
                                     /
                                 +---+
                                 | 0 |
                                 +---+
```

Figure 3: IU propagation example

Network FIBs are assumed to be populated with routes towards P0 by a name-based routing protocol. After the relocation of the producer from P0 to P1, once the layer-2 attachment is completed, the producer issues an IU carrying the prefix /p and this is forwarded by the network toward P0 (in general, toward one of its previous locations according to the FIB state of traversed routers).

Figure 1 (b) illustrates the propagation of the IU. As the IU progresses, FIBs at intermediate hops are updated with the ingress face of the IU (Figure 2 (a) and (b)). IU propagation stops when the IU reaches P0 and there is no next hop to forward it to. The result is that the original tree rooted in P0 becomes re-rooted in P1 (Figure 3). Looking at the different connected regions (represented with dotted lines), we see that IU propagation and consequent FIB updates have the effect of extending the newly connected subtree : at every step, an additional router and its predecessors are included in the connected subtree.

3.3. Concurrent updates

Frequent mobility of the producer may lead to the propagation of concurrent updates. To prevent inconsistencies in FIBs, MAP-Me maintains a sequence number at the producer end that is incremented at each handover and associated to all sent IU packets. Network routers also keep track of such sequence number in their FIBs to validate the relative freshness of received updates. The modification of FIB entries is only triggered when the received IU carries a higher sequence number than the locally stored one, while the reception of a less recent update triggers the transmission of a more up-to-date IU backwards in order to fix the not-yet-updated path.

An example reconciliation of concurrent updates is illustrated in Figure 4 (a), when the producer has moved successively to P1 and then to P2 before the first update could complete.

```
                            +---+                        +---+
                            | 0 | P0                     | 2 | P0
                            +---+                     A  +---+
                             ^ ^               IU(2) /     / ^
                            /   \                    /    V   \
                        +---+    +---+               +---+    +---+
                        | 0 |    | 0 |          <!>  | 2 |    | 0 |
                        +---+ A  +---+             A +---+ A  +---+
                         ^ ^   \           IU(1) /     ^ \   \
                        /   \   \  IU(2)         /    /   V   \ IU(2)
                    +---+    +---+               +---+    +---+
                    | 0 |    | 2 | P2            | 1 |    | 2 |
                 A  +---+    +---+             A  +---+    +---+
          IU(1) /    ^ ^                    IU1 /    / ^
               /    /   \                      /    V   \
                +---+    +---+                 +---+    +---+
             P1 | 1 |    | 0 |              P1 | 1 |    | 0 |
                +---+    +---+                 +---+    +---+
                 ^ ^                             ^ ^
                /   \                           /   \
            +---+    +---+                   +---+    +---+
            | 0 |    | 0 |                   | 0 |    | 0 |
            +---+    +---+                   +---+    +---+

                     (a)                       (b)
```

Figure 4

```
                            +---+                      +---+
                            | 2 | P0                   | 2 | P2
                            +---+                      +---+
                             / ^                         ^
                            V   \                        |
                        +---+    +---+                 +---+
                    <!> | 2 |    | 2 |                 | 2 |
             IU(2)  /   +---+    +---+                 +---+
                   /     ^ \                           ^   ^
                  V     /   V                         /     \
                    +---+    +---+                +---+     +---+
                    | 2 |    | 2 | P2         P0  | 2 |     | 2 |
          IU(2)  /  +---+    +---+                +---+     +---+
                /    / ^                           ^         ^ ^
               V    V   \                         /    P1   /   \
                +---+    +---+                 +---+    +---+   +---+
             P1 | 1 |    | 0 |                 | 0 |    | 2 |   | 0 |
                +---+    +---+                 +---+    +---+   +---+
                 ^ ^                                     ^ ^
                /   \                                   /   \
            +---+    +---+                          +---+    +---+
            | 0 |    | 0 |                          | 0 |    | 0 |
            +---+    +---+                          +---+    +---+

                     (a)                                 (b)
```

Figure 5

Both updates propagate concurrently until the one with sequence number 1 (IU(1)) crosses a router that has been updated with fresher information. In the example shown in Figure 4 (b), the junction router has already received an IU with higher sequence number (IU(2)). In this case, the router stops the propagation of IU(1) and sends back along its path a new IU with an updated sequence number (Figure 5 (a)). The update proceeds until the whole network has ultimately converged towards P2 (Figure 5 (b)).

MAP-Me protocol reacts at a faster timescale than routing - allowing more frequent and numerous mobility events - and over a localized portion of the network edge between current and previous producer locations. This allows to minimize disconnectivity time and reduce link load, which are the main factors affecting user flow performance, as shown in [MAPME] evaluations.

4. Notification protocol and scoped discovery

IU propagation in the data plane is designed to accelerate forwarding state re-convergence w.r.t. routing or resolution-based approaches operating at control plane, and w.r.t. anchor-based approaches requiring traffic tunneling through an anchor node. Still, network latency makes IU completion not instantaneous and before an update completes, it may happen that a portion of the traffic is forwarded to the previous PoA and dropped because of the absence of a valid output face leading to the producer.

Previous work in the Anchor-Less category has suggested the buffering of Interests at previous producer location to prevent those losses. However, such a solution is not suitable for applications with stringent latency requirements (e.g. real-time) and may be incompatible with IU completion times. Moreover, the negative effects on latency performance might be further exacerbated by IU losses and consequent retransmissions in case of wireless medium. To alleviate such issues, we introduce two enhancements to the previously described behavior, namely (i) an "Interest Notification" mechanism for frequent, yet lightweight, signaling of producer movements to the network and (ii) a scoped "Producer Discovery" mechanism for consumer requests to proactively search for the producer's recently visited locations.

4.1. Interest Notification

An Interest Notification (IN) is a breadcrumb left by producers at every encountered PoA. It looks like a normal Interest packet carrying a special identification flag and a sequence number, like IUs. Both IU and IN share the same sequence number (producers indistinctly increase it for every sent message) and follow the same FIB lookup and update processes. However, unlike IU packets, the trace left by INs at the first hop router does not propagate further. It is rather used by the discovery process to route consumer requests to the producer even before an update process is completed.

It is worth observing that updates and notifications serve the same purpose of informing the network of a producer movement. The IU process restores connectivity and as such has higher latency/signaling cost than the IN process, due to message propagation. The IN process provides information to track producer movements before update completion when coupled with a scoped discovery. The combination of both IU and IN allows to control the trade-off between protocol reactivity and stability of forwarding re-convergence.

4.2. Scoped discovery

The extension of MAP-Me with notifications relies on a local discovery phase: when a consumer Interest reaches a PoA with no valid output face in the corresponding entry, the Interest is tagged with a "discovery" flag and labeled with the latest sequence number stored in FIB (to avoid loops). From that point on, it is broadcasted with hop limit equal to one to all neighbors and discarded unless it finds a breadcrumb left by the producer with a higher sequence number. The notifications can either allow to forward consumer Interests directly to the producer or give rise to a repeated broadcast in case of no valid output face. The latter is the case of a breadcrumb left by the producer with no associated forwarding information because the producer has already left that PoA as well. A detailed description of the process is reported in Section 5.3.

The notification/discovery mechanism proves important to preserve the performance of flows in progress, especially when latency-sensitive.

4.3. Full approach

The full MAP-Me approach consists in the combination of Updates and Notifications through a heuristic allowing the producer or its PoA to select which type of packet to send. One such heuristic consist in sending a IN immediately after an attachment and a IU at most every Tu seconds, which allows to reduce signaling overhead during periods of high-mobility. The Tu parameter allows to tune the timescale at which Updates occur, and leads to a trade-off between signaling and discovery overhead [MAPME]. The definition of more advanced heuristics is out of scope for the present draft.

5. Implementation

In this section we describe the changes to a regular CCN/NDN architecture required to implement MAP-ME and detail the above-described algorithms. This requires to specify a special Interest message, additional temporary information associated to the FIB entry and additional operations to update such entry.

5.1. MAP-Me messages

MAP-Me signaling messages are carried within user plane as special Interest messages corresponding to "update" and "notification", and their corresponding acknowledgements.

Two new optional fields are introduced in a CCN/NDN Interest header:

* an "Interest Type" (T) used to specify one of the four types of messages: Interest Update (IU), Interest Notification (IN), and as well as their associated acknowledgment (Ack) messages (IU_Ack and IN_Ack). Those flags are recognized by the forwarding pipeline to trigger special treatment;

* a "sequence number" to handle concurrent updates and prevent forwarding loops during signaling, and to control discovery Interests' propagation;

5.2. Data structures and temporary state

FIB entries are augmented with information required for mobility management, that we denote as Transient FIB buffer, or simply TFIB, and sketch in Figure 6:

* a "sequence number" which is incremented upon reception of IU/IN messages. It can be assumed this counter is set to 0 by the routing protocol.

* a list of so-called "previous next hop(s)" (further denoted as PrevHops), similar to the list of NextHops in the original FIB, which temporarily stores information about faces that were previously next hops, and should still be memorized to allow for retransmissions and thus ensure the consistency of MAP-Me operations. They typically correspond to nodes for which an IU has been sent, but no acknowledgement (ACK) has yet been received (upon which they are cleared). In case of notifications, no ACK is expected, and those entries serve as a memory of the former tree structure that will be restored upon producer departure. We flag those entries with a boolean marker indicating if they correspond to an IU (and thus should be monitored for retransmissions) or an IN (in which case they just serve as memory for further use).

```
      IU (IN) input face(s)           IU (IN) output face

+-----------+-------------------+.......+..........................+
|  /prefix  |  { next hop(s) }  |  seq  | { previous next hop(s) } |
+-----------+-------------------+.......+..........................+
 \______________ _______________/ \_______________ ________________/
                V                                 V
          original FIB                       TFIB section
```

Figure 6: MAP-Me FIB/TFIB description

5.3. Algorithm description

5.3.1. Producer attachment and face creation

MAP-Me operations are triggered by a change of adjacencies in the network, reflected in the forwarder by the creation or removal of a face. This can be for instance the layer 2 detachment and attachment following a mobility/handover event, but also any other mechanism such as point-to-point IP link or UDP tunnel for instance, as allowed by the forwarder implementation.

One realization of this architecture is to delegate face management to a third party agent, keeping the ICN forwarder state synchronized with the underlying topology, and having MAP-Me only react to changes in the face table.

5.3.2. IU/IN transmission at producer

The creation of a new face on the producer triggers the increase of MAP-Me sequence number and the transmission for every locally served prefix, of an IU or IN carrying the updated sequence number.

5.3.3. IU/IN transmission at network routers

At the reception of IU/IN packets, each router performs a name-based Longest Prefix Match lookup in FIB to compare sequence number from IU/IN and from FIB. According to that comparison:

* if the IU/IN packet carries a higher sequence number, the existing next hops associated to the lower sequence number in FIB are used to forward further the IU (INs are not propagated) and temporarily copied into TFIB to avoid loss of such information before completion of the IU/IN acknowledgement process. The ingress face of the IU/IN is then added to FIB to route consumer requests to the latest known location of the producer.

* If the IU/IN packet carries the same sequence number as in the FIB, the originating face of the IU/IN is added to the existing ones in FIB without additional packet processing or propagation. This may occur in presence of multiple forwarding paths.

* If the IU/IN packet carries a lower sequence number than the one in the FIB, FIB entry is not updated as it already stores 'fresher information'. To advertise the latest update through the path followed by the IU/IN packet, this one is re-sent through the originating face after having updated its sequence number with the value stored in FIB.

The operations in the forwarding pipeline for IU/IN processing are reported in Figure 7, where we make use of the following primitives: - Send(Interest, Face) is used to send the specified Interest on the specified Face. - ProcessTFIB() sends an IU for all flagged entries in the TFIB, using the latest sequence number stored in the FIB entry, and schedule the entry to be checked for retransmissions.

```
| Alg. 1:ForwardSpecialInterest(SpecialInterest SI,IngressFace F)
|
|  CheckValidity()
|  // Acknowledge reception
|  s <- e.seq
|  e.seq <- SI.seq
|  Send(IU_Ack(e.seq), F)
   flag <- (SI.type == IU)
|  // Retrieve the FIB entry associated to the prefix
|  e <- FIB.LongestPrefixMatch(SI.name)
|  if SI.seq >= e.seq then
|  .   //Process special interest
|  .   e.TFIB = e.TFIB \ { F }
|  .   if SI.seq > s then
|  .   .   e.TFIB = e.TFIB U { (f, flag) | f in (e.NextHops \ F) }
   .   .   ProcessTFIB()
|  .   .   e.NextHops = {}
|  .   e.NextHops = e.NextHops U { F }
|  else
|  .   // Send updated IU backwards
|  .   SI.seq = e.seq
|  .   e.TFIB = e.TFIB U { (F, flag) }
|  .   ProcessTFIB()
```

Figure 7

5.3.4. Reliable transmission

MAP-Me ensure the reliable delivery of signaling messages thanks to a retransmission timer which reissue Interest Updates (eventually carrying updated sequence number as found in the FIB), if no corresponding ACK has been received in a predefined interval, and whose sequence number has to match the one stored in the FIB.

A slotted implementation of such scheme is possible by using a single timer, and keeping a list of FIB entries that require to be checked for pending retransmissions in the next slot. Upon timer expiration, if all required ACKs have been received, the TFIB will be empty and the entry does not have to be tracked anymore. Otherwise, necessary retransmissions are performed and the entry will be checked again in the next slot. When no entry has to be monitored, the process can sleep until the next mobility event.

5.3.5. Consumer request forwarding in case of producer discovery

The forwarding of regular Interests is mostly unaffected in MAP-Me, except in the case of discovery Interests that we detail in Figure 8. The function SendToNeighbors(I) is responsible for broadcasting the Interest I to all neighboring PoAs.

```
Alg. 2:  InterestForward(Interest I, Origin face F)

 // Regular PIT and CS lookup
 e <- FIB.LongestPrefixMatch(I.name)
 if e = 0 then
 .   return
 if I.seq = 0 then
 .   // Regular interest
 .   if hasValidFace(e.NextHops) or DiscoveryDisabled then
 .   .   ForwardingStrategy.process(I, e)
 .   else
 .   .   // Enter discovery mode
 .   .   I.seq <- e.seq
 .   .   SendToNeighbors(I)
 else
 .   // Discovery interest: forward if producer is connnected
 .   if hasProducerFace(e.NextHops) then
 .   .   ForwardingStrategy.process(I, e)
 .   // Otherwise iterate iif higher seq and breadcrumb
 .   else if e.seq >= I.seq and \
             EXISTS f | (f -> NULL) in e.TFIB then
 .   .   I.seq <- e.seq
 .   .   SendToNeighbors(I)
```

Figure 8

When an Interest arrives to a PoA which has no valid next hop for it (because the producer left and the face got destroyed), it enters a discovery phase where the Interest is flagged as a Discovery Interest and with the local sequence number, then broadcasted to neighboring PoAs.

Upon reception of a Discovery Interest, the PoA forwards it directly to the producer if still attached, otherwise it repeats the one-hop brodcast discovery to neighboring PoAs if it stores a recent notification of the producer presence, i.e. an entry in TFIB having higher sequence number than the one in the Discovery Interest. Otherwise, the Discovery Interest is discarded.

It is worth observing that the discovery process is initiated only in the case of no valid next hop, and not every time a notification is found in a router. This is important to guarantee that the notification/discovery process does not affect IU propagation and completion.

5.3.6. Producer departure and face destruction

Upon producer departures from a PoA, the corresponding face is destroyed. If this leads to the removal of the last next hop, then faces in TFIB corresponding to IN are restored as next hops in the FIB so as to preserve the original forwarding tree and thus global connectivity.

6. Security considerations

All mobility management protocols share the same critical need for securing their control messages which have a direct impact on the forwarding of users' traffic. [SEC] reviews standard approaches from the literature and proposes a fast, lightweight and decentralized approach based on hash chains that can be applied to MAP-Me and fits its design principles.

7. Acknowledgements

The authors would like to thank Giulio Grassi (UPMC/UCLA), Giovanni Pau (UPMC/UCLA) and Xuan Zeng (UPMC/SystemX) for their contribution to the work that has led to this document.

8. IANA Considerations

This memo includes no request to IANA.

9. References

9.1. Normative References

[RFC6301] Zhu, Z., Wakikawa, R., and L. Zhang, "A Survey of Mobility Support in the Internet", RFC 6301, DOI 10.17487/RFC6301, July 2011, <https://www.rfc-editor.org/info/rfc6301>.

9.2. Informative References

[DATAPLANE]
J, ., A, ., A, ., B, ., M, ., and . S, "Ensuring connectivity via data plane mechanisms.", 2013.

[MAPME] Auge, J., Carofiglio, G., Grassi, G., Muscariello, L., Pau, G., and X. Zeng, "MAP-Me: Managing Anchor-Less Producer Mobility in Content-Centric Networks", DOI 10.1109/tnsm.2018.2796720, IEEE Transactions on Network and Service Management Vol. 15, pp. 596-610, June 2018, <https://doi.org/10.1109/tnsm.2018.2796720>.

[NLSR] Hoque, A., Amin, S., Alyyan, A., Zhang, B., Zhang, L., and L. Wang, "NISR", DOI 10.1145/2491224.2491231, Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking - ICN '13, 2013, <https://doi.org/10.1145/2491224.2491231>.

[SEC] Compagno, A., Zeng, X., Muscariello, L., Carofiglio, G., and J. Auge, "Secure producer mobility in information-centric network", DOI 10.1145/3125719.3125725, Proceedings of the 4th ACM Conference on Information-Centric Networking, September 2017, <https://doi.org/10.1145/3125719.3125725>.

[SURVEY12] Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., and B. Ohlman, "A survey of information-centric networking", DOI 10.1109/mcom.2012.6231276, IEEE Communications Magazine Vol. 50, pp. 26-36, July 2012, <https://doi.org/10.1109/mcom.2012.6231276>.

[SURVEY13] Tyson, G., Sastry, N., Rimac, I., Cuevas, R., and A. Mauthe, "A survey of mobility in information-centric networks", DOI 10.1145/2248361.2248363, Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications - NoM '12, 2012, <https://doi.org/10.1145/2248361.2248363>.

[SURVEY14] Xylomenos, G., Ververidis, C., Siris, V., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K., and G. Polyzos, "A Survey of Information-Centric Networking Research", DOI 10.1109/surv.2013.070813.00063, IEEE Communications Surveys & Tutorials Vol. 16, pp. 1024-1049, 2014, <https://doi.org/10.1109/surv.2013.070813.00063>.

[SURVEY16a]
Feng, B., Zhou, H., and Q. Xu, "Mobility support in Named Data Networking: a survey", DOI 10.1186/s13638-016-0715-0, EURASIP Journal on Wireless Communications and Networking Vol. 2016, September 2016, <https://doi.org/10.1186/s13638-016-0715-0>.

[SURVEY16b]
Zhang, Y., Afanasyev, A., Burke, J., and L. Zhang, "A survey of mobility support in Named Data Networking", DOI 10.1109/infcomw.2016.7562050, 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), April 2016, <https://doi.org/10.1109/infcomw.2016.7562050>.

[WLDR] Carofiglio, G., Muscariello, L., Papalini, M., Rozhnova, N., and X. Zeng, "Leveraging ICN In-network Control for Loss Detection and Recovery in Wireless Mobile networks", DOI 10.1145/2984356.2984361, Proceedings of the 2016 conference on 3rd ACM Conference on Information-Centric Networking - ACM-ICN '16, 2016, <https://doi.org/10.1145/2984356.2984361>.

Authors' Addresses

Jordan Auge (editor)
Cisco Systems Inc.
11, rue Camille Desmoulins
92130 Issy-les-Moulineaux
France

Email: augjorda@cisco.com

Giovanna Carofiglio
Cisco Systems Inc.
11, rue Camille Desmoulins
92130 Issy-les-Moulineaux
France

Email: gcarofig@cisco.com

Luca Muscariello
Cisco Systems Inc.
11, rue Camille Desmoulins
92130 Issy-les-Moulineaux
France

Email: lumuscar@cisco.com

Michele Papalini
Cisco Systems Inc.
11, rue Camille Desmoulins
92130 Issy-les-Moulineaux
France

Email: micpapal@cisco.com