

ICNRRG
Internet-Draft
Intended status: Experimental
Expires: July 28, 2019

M. Mosko
PARC, Inc.
I. Solis
LinkedIn
C. Wood
University of California Irvine
January 24, 2019

CCNx Messages in TLV Format
draft-irtf-icnrg-ccnxmessages-09

Abstract

This document specifies the encoding of CCNx messages in a TLV packet format, including the TLV types used by each message element and the encoding of each value. The semantics of CCNx messages follow the encoding-independent CCNx Semantics specification.

This document is a product of the Information Centric Networking research group (ICNRRG).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 28, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	4
2. Definitions	4
3. Type-Length-Value (TLV) Packets	5
3.1. Overall packet format	6
3.2. Fixed Headers	7
3.2.1. Interest Fixed Header	8
3.2.1.1. Interest HopLimit	9
3.2.2. Content Object Fixed Header	9
3.2.3. InterestReturn Fixed Header	9
3.2.3.1. InterestReturn HopLimit	10
3.2.3.2. InterestReturn Flags	10
3.2.3.3. Return Code	10
3.3. Global Formats	10
3.3.1. Pad	11
3.3.2. Organization Specific TLVs	11
3.3.3. Hash Format	11
3.3.4. Link	13
3.4. Hop-by-hop TLV headers	13
3.4.1. Interest Lifetime	14
3.4.2. Recommended Cache Time	14
3.4.3. Message Hash	15
3.5. Top-Level Types	16
3.6. CCNx Message	16
3.6.1. Name	17
3.6.1.1. Name Segments	18
3.6.1.2. Interest Payload ID	19
3.6.2. Message TLVs	20
3.6.2.1. Interest Message TLVs	20
3.6.2.2. Content Object Message TLVs	21
3.6.3. Payload	23
3.6.4. Validation	23
3.6.4.1. Validation Algorithm	23
3.6.4.2. Validation Payload	29
4. IANA Considerations	29
4.1. Packet Type Registry	30
4.2. Interest Return Code Registry	30
4.3. Hop-by-Hop Type Registry	31
4.4. Top-Level Type Registry	32
4.5. Name Segment Type Registry	33

4.6. Message Type Registry	34
4.7. Payload Type Registry	35
4.8. Validation Algorithm Type Registry	36
4.9. Validation Dependent Data Type Registry	37
4.10. Hash Function Type Registry	39
5. Security Considerations	40
6. References	43
6.1. Normative References	43
6.2. Informative References	43
Authors' Addresses	45

1. Introduction

This document specifies a Type-Length-Value (TLV) packet format and the TLV type and value encodings for CCNx messages. A full description of the CCNx network protocol, providing an encoding-free description of CCNx messages and message elements, may be found in [CCNSemantics]. CCNx is a network protocol that uses a hierarchical name to forward requests and to match responses to requests. It does not use endpoint addresses, such as Internet Protocol. Restrictions in a request can limit the response by the public key of the response's signer or the cryptographic hash of the response. Every CCNx forwarder along the path does the name matching and restriction checking. The CCNx protocol fits within the broader framework of Information Centric Networking (ICN) protocols [RFC7927].

This document describes a TLV scheme using a fixed 2-byte T and a fixed 2-byte L field. The rational for this choice is described in Section 5. Briefly, this choice avoids multiple encodings of the same value (aliases) and reduces the work of a validator to ensure compliance. Unlike some uses of TLV in networking, the each network hop must evaluate the encoding, so even small validation latencies at each hop could add up to a large overall forwarding delay. For very small packets or low throughput links, where the extra bytes may become a concern, one may use a TLV compression protocol, for example [compress] and [CCNxz].

This document specifies:

- o The TLV packet format.
- o The overall packet format for CCNx messages.
- o The TLV types used by CCNx messages.
- o The encoding of values for each type.
- o Top level types that exist at the outermost containment.

- o Interest TLVs that exist within Interest containment.
- o Content Object TLVs that exist within Content Object containment.

This document is supplemented by this document:

- o Message semantics: see [CCN semantics] for the protocol operation regarding Interest and Content Object, including the Interest Return protocol.
- o URI notation: see [CCNxURI] for the CCNx URI notation.

The type values in Section 4 represent the values in common usage today. These values may change pending IANA assignments. All type values are relative to their parent containers. For example, each level of a nested TLV structure might define a "type = 1" with a completely different meaning. In the following, we use the symbolic names defined in that section.

Packets are represented as 32-bit wide words using ASCII art. Due to the nested levels of TLV encoding and the presence of optional fields and variable sizes, there is no concise way to represent all possibilities. We use the convention that ASCII art fields enclosed by vertical bars "|" represent exact bit widths. Fields with a forward slash "/" are variable bit widths, which we typically pad out to word alignment for picture readability.

The document represents the consensus of the ICN RG. It is the first ICN protocol from the RG, created from the early CCNx protocol [nnc] with significant revision and input from the ICN community and RG members. The draft has received critical reading by several members of the ICN community and the RG. The authors and RG chairs approve of the contents. The document is sponsored under the IRTF and is not issued by the IETF and is not an IETF standard. This is an experimental protocol and may not be suitable for any specific application and the specification may change in the future.

1.1. Requirements Language

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].

2. Definitions

- o Name: A hierarchically structured variable length identifier. It is an ordered list of path segments, which are variable length octet strings. In human-readable form, it is represented in URI

format as `ccnx:/path/part`. There is no host or query string. See [CCNxURI] for complete details.

- o Interest: A message requesting a Content Object with a matching Name and other optional selectors to choose from multiple objects with the same Name. Any Content Object with a Name and attributes that matches the Name and optional selectors of the Interest is said to satisfy the Interest.
- o Content Object: A data object sent in response to an Interest request. It has an optional Name and a content payload that are bound together via cryptographic means.

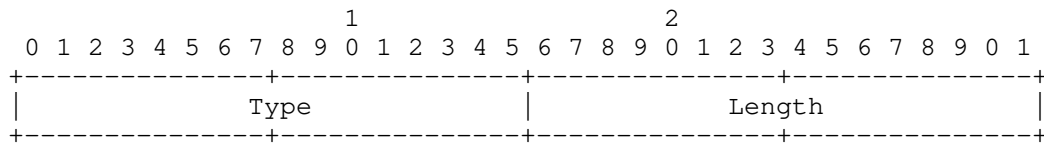
3. Type-Length-Value (TLV) Packets

We use 16-bit Type and 16-bit Length fields to encode TLV based packets. This provides 64K different possible types and value field lengths of up to 64KiB. With 64K possible types at each level of TLV encoding, there should be sufficient space for basic protocol types, while also allowing ample room for experimentation, application use, vendor extensions, and growth. This encoding does not allow for jumbo packets beyond 64 KiB total length. If used on a media that allows for jumbo frames, we suggest defining a media adaptation envelope that allows for multiple smaller frames.

There are several global TLV definitions that we reserve at all hierarchical contexts. The TLV types in the range 0x1000 - 0x1FFF are reserved for experimental use. The TLV type T_ORG is also reserved for vendor extensions (see Section 3.3.2). The TLV type T_PAD is used to optionally pad a field out to some desired alignment.

Abbrev	Name	Description
T_ORG	Vendor Specific Information (Section 3.3.2)	Information specific to a vendor implementation (see below).
T_PAD	Padding (Section 3.3.1)	Adds padding to a field (see below).
n/a	Experimental	Experimental use.

Table 1: Reserved TLV Types



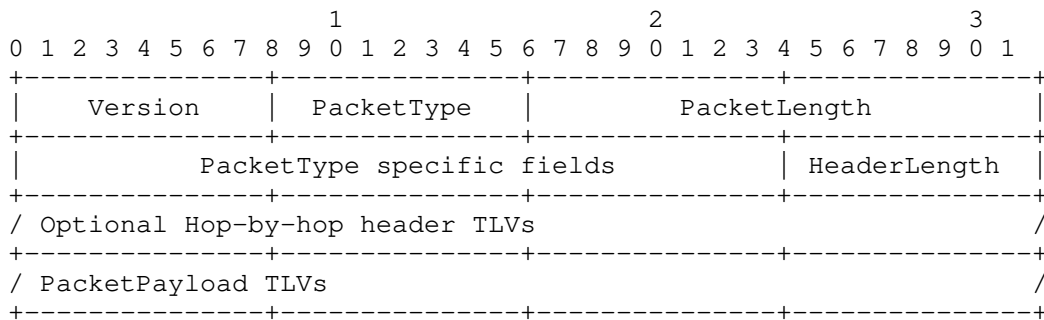
The Length field contains the length of the Value field in octets. It does not include the length of the Type and Length fields. The length MAY be zero.

TLV structures are nestable, allowing the Value field of one TLV structure to contain additional TLV structures. The enclosing TLV structure is called the container of the enclosed TLV.

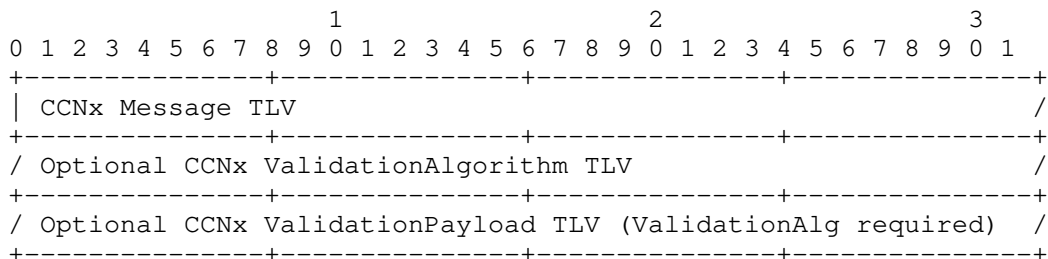
Type values are context-dependent. Within a TLV container, one may re-use previous type values for new context-dependent purposes.

3.1. Overall packet format

Each packet includes the 8 byte fixed header, described below, followed by a set of TLV fields. These fields are optional hop-by-hop headers and the Packet Payload.



The packet payload is a TLV encoding of the CCNx message, followed by optional Validation TLVs.



This document describes the Version "1" TLV encoding.

After discarding the fixed and hop-by-hop headers the remaining PacketPayload should be a valid protocol message. Therefore, the PacketPayload always begins with 4 bytes of type-length that specifies the protocol message (whether it is an Interest, Content Object, or other message type) and its total length. The embedding of a self-sufficient protocol data unit inside the fixed and hop-by-hop headers allows a network stack to discard the headers and operate only on the embedded message. It also de-couples the PacketType field -- which specifies how to forward the packet -- from the PacketPayload.

The range of bytes protected by the Validation includes the CCNx Message and the ValidationAlgorithm.

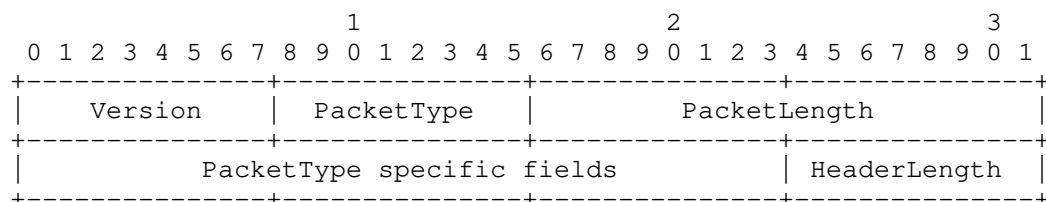
The ContentObjectHash begins with the CCNx Message and ends at the tail of the packet.

3.2. Fixed Headers

CCNx messages begin with an 8 byte fixed header (non-TLV format). The HeaderLength field represents the combined length of the Fixed and Hop-by-hop headers. The PacketLength field represents the entire Packet length from the first byte of Version to the last byte of the packet.

A specific PacketType may assign meaning to the "PacketType specific fields," which are otherwise reserved. For the three defined PacketTypes (Interest, ContentObject, and InterestReturn), we define those values in this document.

The PacketPayload of a CCNx packet is the protocol message itself. The Content Object Hash is computed over the PacketPayload only, excluding the fixed and hop-by-hop headers as those might change from hop to hop. Signed information or Similarity Hashes should not include any of the fixed or hop-by-hop headers. The PacketPayload should be self-sufficient in the event that the fixed and hop-by-hop headers are removed.



- o Version: defines the version of the packet.
- o HeaderLength: The length of the fixed header (8 bytes) and hop-by-hop headers. The minimum value MUST be "8".
- o PacketType: describes forwarder actions to take on the packet.
- o PacketLength: Total octets of packet including all headers (fixed header plus hop-by-hop headers) and protocol message.
- o PacketType Specific Fields: specific PacketTypes define the use of these bits.

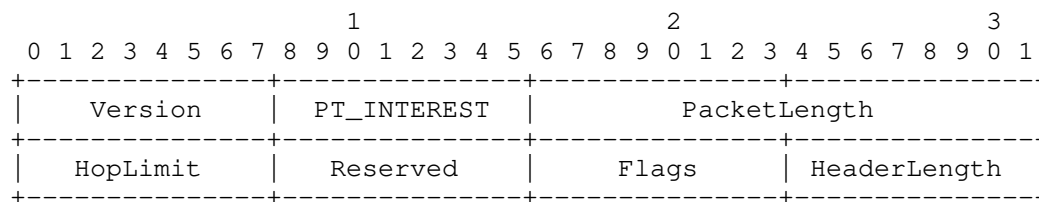
The PacketType field indicates how the forwarder should process the packet. A Request Packet (Interest) has PacketType PT_INTEREST, a Response (Content Object) has PacketType PT_CONTENT, and an InterestReturn has PacketType PT_RETURN.

HeaderLength is the number of octets from the start of the packet (Version) to the end of the hop-by-hop headers. PacketLength is the number of octets from the start of the packet to the end of the packet. Both lengths have a minimum value of 8 (the fixed header itself).

The PacketType specific fields are reserved bits whose use depends on the PacketType. They are used for network-level signaling.

3.2.1. Interest Fixed Header

If the PacketType is PT_INTEREST, it indicates that the PacketPayload should be processed as an Interest message. For this type of packet, the Fixed Header includes a field for a HopLimit as well as Reserved and Flags fields. The Reserved field MUST be set to 0 in an Interest - this field will be set to a return code in the case of an Interest Return. There are currently no Flags defined, so this field MUST be set to 0.



3.2.1.1. Interest HopLimit

For an Interest message, the HopLimit is a counter that is decremented with each hop. It limits the distance an Interest may travel on the network. The node originating the Interest MAY put in any value - up to the maximum of 255. Each node that receives an Interest with a HopLimit decrements the value upon reception. If the value is 0 after the decrement, the Interest MUST NOT be forwarded off the node.

It is an error to receive an Interest with a 0 hop-limit from a remote node.

3.2.2. Content Object Fixed Header

If the PacketType is PT_CONTENT, it indicates that the PacketPayload should be processed as a Content Object message. A Content Object defines a Flags field, however there are currently no flags defined, so the Flags field must be set to 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								
-----										-----										-----										-----									
Version										PT_CONTENT										PacketLength																			
-----										-----										-----										-----									
Reserved										Flags										HeaderLength																			
-----										-----										-----										-----									

3.2.3. InterestReturn Fixed Header

If the PacketType is PT_RETURN, it indicates that the PacketPayload should be processed as a returned Interest message. The only difference between this InterestReturn message and the original Interest is that the PacketType is changed to PT_RETURN and a ReturnCode is put into the ReturnCode field. All other fields are unchanged from the Interest packet. The purpose of this encoding is to prevent packet length changes so no additional bytes are needed to return an Interest to the previous hop. See [CCNSemantics] for a protocol description of this packet type.

										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										PT_RETURN										PacketLength																				
-----										-----										-----										-----										
HopLimit										ReturnCode										Flags										HeaderLength										
-----										-----										-----										-----										

3.2.3.1. InterestReturn HopLimit

This is the original Interest's HopLimit, as received. It is the value before being decremented at the current node (i.e. the received value).

3.2.3.2. InterestReturn Flags

These are the original Flags as set in the Interest.

3.2.3.3. Return Code

The numeric value assigned to the return types is defined below. This value is set by the node creating the Interest Return.

A return code of "0" MUST NOT be used, as it indicates that the returning system did not modify the Return Code field.

Type	Return Type
T_RETURN_NO_ROUTE	No Route
T_RETURN_LIMIT_EXCEEDED	Hop Limit Exceeded
T_RETURN_NO_RESOURCES	No Resources
T_RETURN_PATH_ERROR	Path Error
T_RETURN_PROHIBITED	Prohibited
T_RETURN_CONGESTED	Congested
T_RETURN_MTU_TOO_LARGE	MTU too large
T_RETURN_UNSUPPORTED_HASH_RESTRICTION	Unsupported ContentObjectHashRestriction
T_RETURN_MALFORMED_INTEREST	Malformed Interest

Table 2: Return Codes

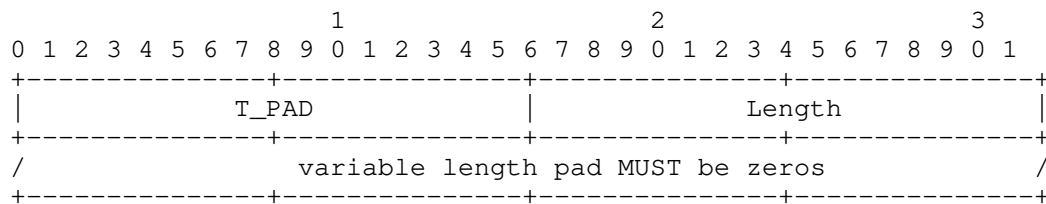
3.3. Global Formats

This section defines global formats that may be nested within other TLVs.

3.3.1. Pad

The pad type may be used by protocols that prefer word-aligned data. The size of the word may be defined by the protocol. Padding 4-byte words, for example, would use a 1-byte, 2-byte, and 3-byte Length. Padding 8-byte words would use a (0, 1, 2, 3, 5, 6, 7)-byte Length.

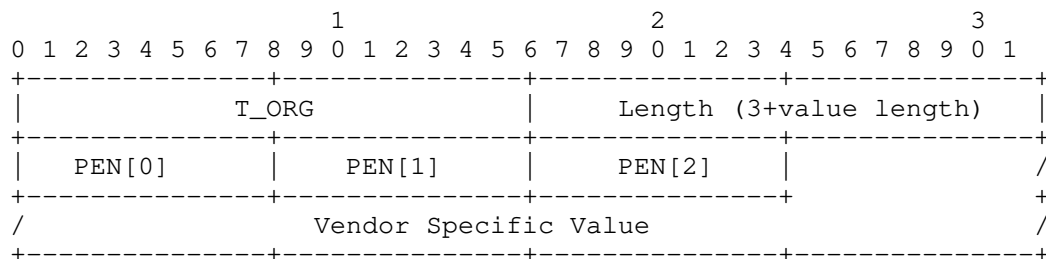
One MUST NOT pad inside a Name. Apart from that, a pad MAY be inserted after any other TLV in the CCNx Message or in the Validation Dependent Data. In the remainder of this document, we will not show optional pad TLVs.



3.3.2. Organization Specific TLVs

Organization specific TLVs (also known as Vendor TLVs) MUST use the T_ORG type. The Length field is the length of the organization specific information plus 3. The Value begins with the 3 byte organization number derived from the last three digits of the IANA Private Enterprise Numbers [EpriseNumbers], followed by the organization specific information.

A T_ORG MAY be used as a path segment in a Name, in which case it is a regular path segment and is part of the regular name matching.



3.3.3. Hash Format

Hash values are used in several fields throughout a packet. This TLV encoding is commonly embedded inside those fields to specify the specific hash function used and it's value. Note that the reserved

TLV types are also reserved here for user-defined experimental functions.

The LENGTH field of the hash value MUST be less than or equal to the hash function length. If the LENGTH is less than the full length, it is taken as the left LENGTH bytes of the hash function output. Only specified truncations are allowed, not arbitrary truncations.

This nested format is used because it allows binary comparison of hash values for certain fields without a router needing to understand a new hash function. For example, the `KeyIdRestriction` is bit-wise compared between an `Interest`'s `KeyIdRestriction` field and a `ContentObject`'s `KeyId` field. This format means the outer field values do not change with differing hash functions so a router can still identify those fields and do a binary comparison of the hash TLV without need to understand the specific hash used. An alternative approach, such as using `T_KEYID_SHA512-256`, would require each router keep an up-to-date parser and supporting user-defined hash functions here would explode the parsing state-space.

A CCNx entity **MUST** support the hash type T_SHA-256. An entity **MAY** support the remaining hash types.

Abbrev	Lengths (octets)
T_SHA-256	32
T_SHA-512	64, 32
n/a	Experimental TLV types

Table 3: CCNx Hash Functions

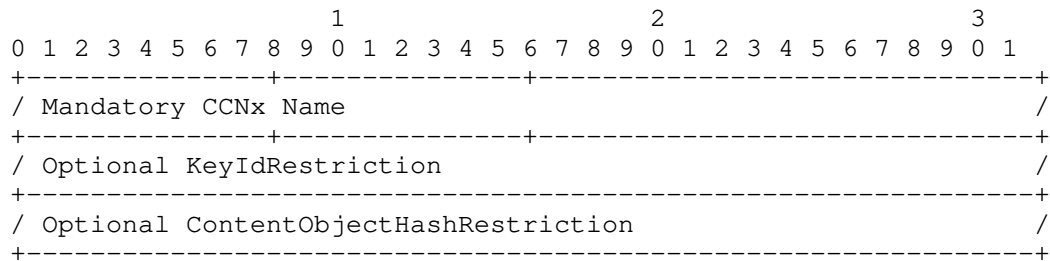
1																2																3															
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1																
T_FOO																36																															
T_SHA512																32																															
/ 32-byte hash value /																																															

Example nesting inside type T_FOO

3.3.4. Link

A Link is the tuple: {Name, [KeyIdRestr], [ContentObjectHashRestr]}.

It is a general encoding that is used in both the payload of a Content Object with PayloadType = "Link" and in the KeyLink field in a KeyLocator. A Link is essentially the body of an Interest.



3.4. Hop-by-hop TLV headers

Hop-by-hop TLV headers are unordered and meaning MUST NOT be attached to their ordering. Three hop-by-hop headers are described in this document:

Abbrev	Name	Description
T_INTLIFE	Interest Lifetime (Section 3.4.1)	The time an Interest should stay pending at an intermediate node.
T_CACHETIME	Recommended Cache Time (Section 3.4.2)	The Recommended Cache Time for Content Objects.
T_MSGHASH	Message Hash (Section 3.4.3)	The hash of the CCNx Message to end of packet using Section 3.3.3 format.

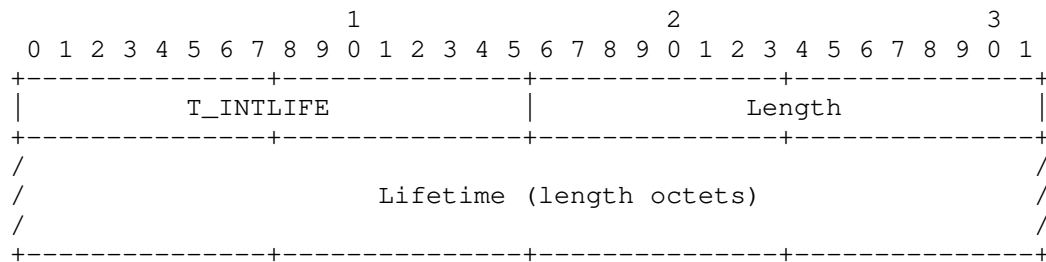
Table 4: Hop-by-hop Header Types

Additional hop-by-hop headers are defined in higher level specifications such as the fragmentation specification.

3.4.1. Interest Lifetime

The Interest Lifetime is the time that an Interest should stay pending at an intermediate node. It is expressed in milliseconds as an unsigned, network byte order integer.

A value of 0 (encoded as 1 byte %x00) indicates the Interest does not elicit a Content Object response. It should still be forwarded, but no reply is expected and a forwarder could skip creating a PIT entry.

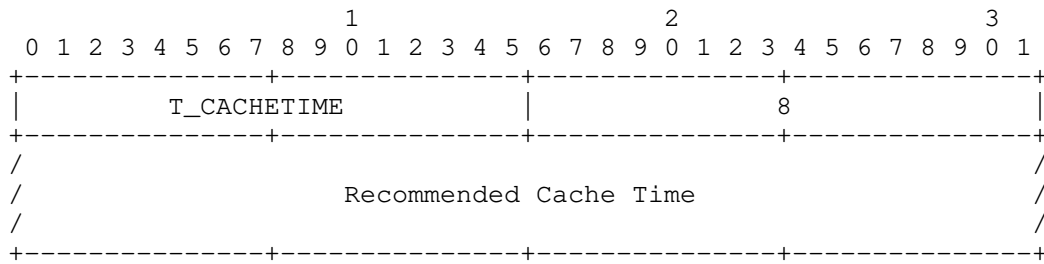


3.4.2. Recommended Cache Time

The Recommended Cache Time (RCT) is a measure of the useful lifetime of a Content Object as assigned by a content producer or upstream node. It serves as a guideline to the Content Store cache in determining how long to keep the Content Object. It is a recommendation only and may be ignored by the cache. This is in contrast to the ExpiryTime (described in Section 3.6.2.2.2) which takes precedence over the RCT and must be obeyed.

Because the Recommended Cache Time is an optional hop-by-hop header and not a part of the signed message, a content producer may re-issue a previously signed Content Object with an updated RCT without needing to re-sign the message. There is little ill effect from an attacker changing the RCT as the RCT serves as a guideline only.

The Recommended Cache Time (a millisecond timestamp) is a network byte ordered unsigned integer of the number of milliseconds since the epoch in UTC of when the payload expires. It is a 64-bit field.



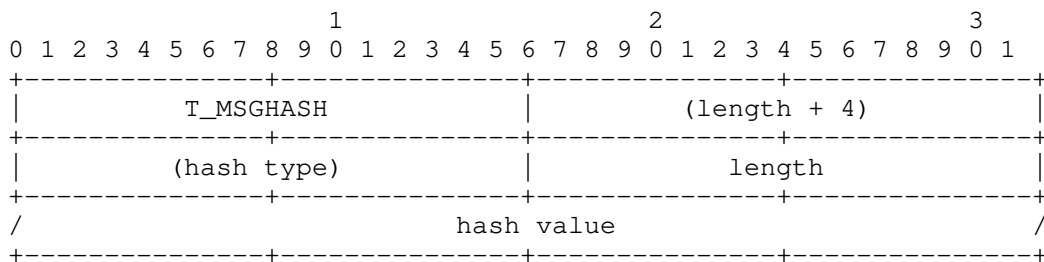
3.4.3. Message Hash

Within a trusted domain, an operator may calculate the message hash at a border device and insert that value into the hop-by-hop headers of a message. An egress device should remove the value. This permits intermediate devices within that trusted domain to match against a ContentObjectHashRestriction without calculating it at every hop.

The message hash is a cryptographic hash from the start of the CCNx Message to the end of the packet. It is used to match against the ContentObjectHashRestriction (Section 3.6.2.1.2). The Message Hash may be of longer length than an Interest's restriction, in which case the device should use the left bytes of the Message Hash to check against the Interest's value.

The Message Hash may only carry one hash type and there may only be one Message Hash header.

The Message Hash header is unprotected, so this header is only of practical use within a trusted domain, such as an operator's autonomous system.



Message Hash Header

3.5. Top-Level Types

The top-level TLV types listed below exist at the outermost level of a CCNx protocol message.

Abbrev	Name	Description
T_INTEREST	Interest (Section 3.6)	An Interest MessageType.
T_OBJECT	Content Object (Section 3.6)	A Content Object MessageType
T_VALIDATION_ALG	Validation Algorithm (Section 3.6.4.1)	The method of message verification such as Message Integrity Check (MIC), a Message Authentication Code (MAC), or a cryptographic signature.
T_VALIDATION_PAYLOAD	Validation Payload (Section 3.6.4.2)	The validation output, such as the CRC32C code or the RSA signature.

Table 5: CCNx Top Level Types

3.6. CCNx Message

This is the format for the CCNx protocol message itself. The CCNx message is the portion of the packet between the hop-by-hop headers and the Validation TLVs. The figure below is an expansion of the "CCNx Message TLV" depicted in the beginning of Section 3. The CCNx message begins with MessageType and runs through the optional Payload. The same general format is used for both Interest and Content Object messages which are differentiated by the MessageType field. The first enclosed TLV of a CCNx Message is always the Name TLV. This is followed by an optional Message TLVs and an optional Payload TLV.

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
-----										-----										-----											
MessageType										MessageLength																					
-----										-----										-----											
Name TLV (Type = T_NAME)																															
-----										-----										-----											
/ Optional Message TLVs (Various Types) /																															
-----										-----										-----											
/ Optional Payload TLV (Type = T_PAYLOAD) /																															
-----										-----										-----											

-----										-----										-----																			
Abbrev										Name										Description																			
-----										-----										-----										-----									
T_NAME										Name (Section 3.6.1)										The CCNx Name requested in an Interest or published in a Content Object.																			
-----										-----										-----										-----									
T_PAYLOAD										Payload (Section 3.6.3)										The message payload.																			
-----										-----										-----										-----									

Table 6: CCNx Message Types

3.6.1. Name

A Name is a TLV encoded sequence of segments. The table below lists the type values appropriate for these Name segments. A Name MUST NOT include PAD TLVs.

As described in CCNx Semantics [CCN semantics], using the CCNx URI [CCNxURI] notation, a T_NAME with 0 length corresponds to ccnx:/ (the default route) and is distinct from a name with one zero length segment, such as ccnx:/NAME=. In the TLV encoding, ccnx:/ corresponds to T_NAME with 0 length, while ccnx:/NAME= corresponds to T_NAME with 4 length and T_NAMESEGMENT with 0 length.

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
T_NAME										Length																					
/ Name segment TLVs																															

Symbolic Name	Name	Description
T_NAMESEGMENT	Name segment (Section 3.6.1.1)	A generic name Segment.
T_IPID	Interest Payload ID (Section 3.6.1.2)	An identifier that represents the Interest Payload field. As an example, the Payload ID might be a hash of the Interest Payload. This provides a way to differentiate between Interests based on their payloads without having to parse all the bytes of the payload itself; instead using only this Payload ID Name segment.
T_APP:00 - T_APP:4096	Application Components (Section 3.6.1.1)	Application-specific payload in a name segment. An application may apply its own semantics to the 4096 reserved types.

Table 7: CCNx Name Types

3.6.1.1. Name Segments

4096 special application payload name segments are allocated. These have application semantics applied to them. A good convention is to put the application's identity in the name prior to using these name segments.

For example, a name like "ccnx:/foo/bar/hi" would be encoded as:

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
(T_NAME)										%x14 (20)																					
(T_NAME_SEGMENT)										%x03 (3)																					
f					o					o					(T_NAME_SEGMENT)																
										%x03 (3)										b											
a					r					(T_NAME_SEGMENT)																					
%x02 (2)										h										i											

3.6.1.2. Interest Payload ID

The InterestPayloadID is a name segment created by the origin of an Interest to represent the Interest Payload. This allows the proper multiplexing of Interests based on their name if they have different payloads. A common representation is to use a hash of the Interest Payload as the InterestPayloadID.

As part of the TLV 'value', the InterestPayloadID contains a one identifier of method used to create the InterestPayloadID followed by a variable length octet string. An implementation is not required to implement any of the methods to receive an Interest; the InterestPayloadID may be treated only as an opaque octet string for purposes of multiplexing Interests with different payloads. Only a device creating an InterestPayloadID name segment or a device verifying such a segment need to implement the algorithms.

It uses the Section 3.3.3 encoding of hash values.

In normal operations, we recommend displaying the InterestPayloadID as an opaque octet string in a CCNx URI, as this is the common denominator for implementation parsing.

The InterestPayloadID, even if it is a hash, should not convey any security context. If a system requires confirmation that a specific entity created the InterestPayload, it should use a cryptographic signature on the Interest via the ValidationAlgorithm and ValidationPayload or use its own methods inside the Interest Payload.

3.6.2. Message TLVs

Each message type (Interest or Content Object) is associated with a set of optional Message TLVs. Additional specification documents may extend the types associated with each.

3.6.2.1. Interest Message TLVs

There are two Message TLVs currently associated with an Interest message: the `KeyIdRestriction` selector and the `ContentObjectHashRestr` selector are used to narrow the universe of acceptable Content Objects that would satisfy the Interest.

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																
Message Type										Message Length																																					
Name TLV																																															
/ Optional KeyIdRestriction TLV																																															
/ Optional ContentObjectHashRestriction TLV																																															

Abbrev	Name	Description
T_KEYIDRESTR	KeyIdRestriction (Section 3.6.2.1.1)	A Section 3.3.3 representation of the KeyId
T_OBJHASHRESTR	ContentObjectHashRestriction (Section 3.6.2.1.2)	A Section 3.3.3 representation of the hash of the specific Content Object that would satisfy the Interest.

Table 8: CCNx Interest Message TLV Types

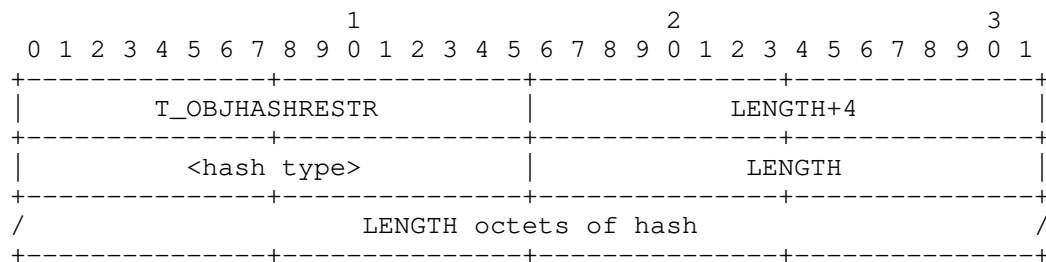
3.6.2.1.1. KeyIdRestriction

An Interest MAY include a `KeyIdRestriction` selector. This ensures that only Content Objects with matching KeyIds will satisfy the Interest. See Section 3.6.4.1.4.1 for the format of a KeyId.

3.6.2.1.2. ContentObjectHashRestriction

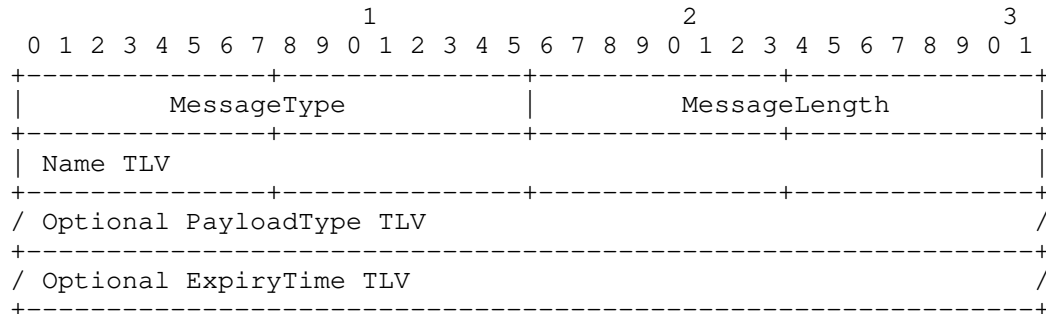
An Interest MAY contain a ContentObjectHashRestriction selector. This is the hash of the Content Object - the self-certifying name restriction that must be verified in the network, if an Interest carried this restriction. It is calculated from the beginning of the CCNx Message to the end of the packet. The LENGTH MUST be from one of the allowed values for that hash (see Section 3.3.3).

The ContentObjectHashRestriction SHOULD be of type T_SHA-256 and of length 32 bytes.



3.6.2.2. Content Object Message TLVs

The following message TLVs are currently defined for Content Objects: PayloadType (optional) and ExpiryTime (optional).



Abbrev	Name	Description
T_PAYLDTYPE	PayloadType (Section 3.6.2.2.1)	Indicates the type of Payload contents.
T_EXPIRY	ExpiryTime (Section 3.6.2.2.2)	The time at which the Payload expires, as expressed in the number of milliseconds since the epoch in UTC. If missing, Content Object may be used as long as desired.

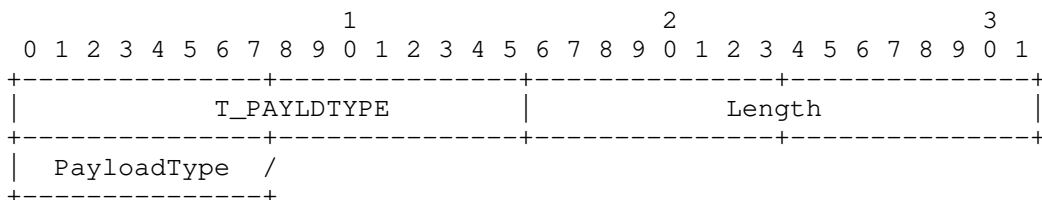
Table 9: CCNx Content Object Message TLV Types

3.6.2.2.1. PayloadType

The PayloadType is a network byte order integer representing the general type of the Payload TLV.

- o T_PAYLOADTYPE_DATA: Data (possibly encrypted)
- o T_PAYLOADTYPE_KEY: Key
- o T_PAYLOADTYPE_LINK: Link

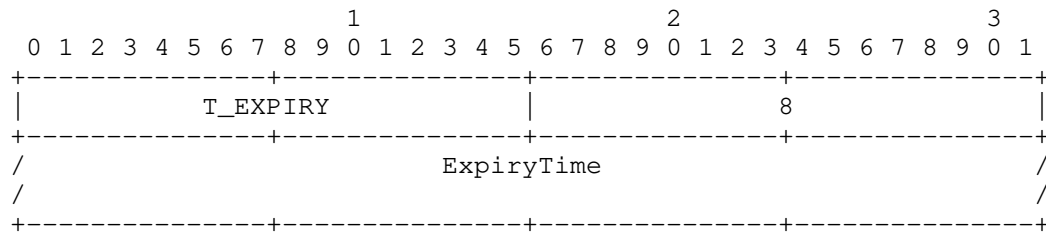
The Data type indicate that the Payload of the ContentObject is opaque application bytes. The Key type indicates that the Payload is a DER encoded public key. The Link type indicates that the Payload is one or more Link (Section 3.3.4). If this field is missing, a "Data" type is assumed.



3.6.2.2.2. ExpiryTime

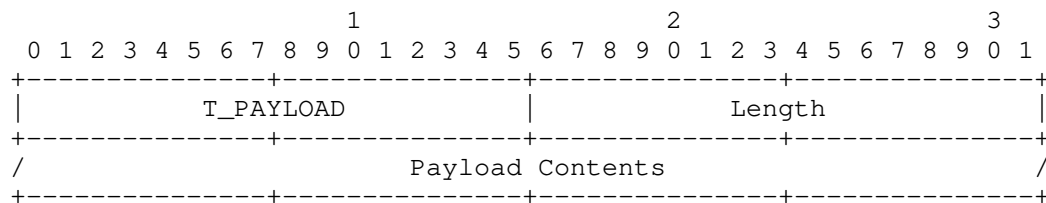
The ExpiryTime is the time at which the Payload expires, as expressed by a timestamp containing the number of milliseconds since the epoch in UTC. It is a network byte order unsigned integer in a 64-bit field. A cache or end system should not respond with a Content Object past its ExpiryTime. Routers forwarding a Content Object do

not need to check the ExpiryTime. If the ExpiryTime field is missing, the Content Object has no expressed expiration and a cache or end system may use the Content Object for as long as desired.



3.6.3. Payload

The Payload TLV contains the content of the packet. It MAY be of zero length. If a packet does not have any payload, this field MAY be omitted, rather than carrying a zero length.



3.6.4. Validation

Both Interests and Content Objects have the option to include information about how to validate the CCNx message. This information is contained in two TLVs: the ValidationAlgorithm TLV and the ValidationPayload TLV. The ValidationAlgorithm TLV specifies the mechanism to be used to verify the CCNx message. Examples include verification with a Message Integrity Check (MIC), a Message Authentication Code (MAC), or a cryptographic signature. The ValidationPayload TLV contains the validation output, such as the CRC32C code or the RSA signature.

An Interest would most likely only use a MIC type of validation - a crc, checksum, or digest.

3.6.4.1. Validation Algorithm

The ValidationAlgorithm is a set of nested TLVs containing all of the information needed to verify the message. The outermost container has type = T_VALIDATION_ALG. The first nested TLV defines the specific type of validation to be performed on the message. The type

is identified with the "ValidationType" as shown in the figure below and elaborated in the table below. Nested within that container are the TLVs for any ValidationType dependent data, for example a Key Id, Key Locator etc.

Complete examples of several types may be found in Section 3.6.4.1.5

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																
T_VALIDATION_ALG																ValidationAlgLength																															
ValidationType																Length																															
/ ValidationType dependent data /																																															
Abbrev																Name																Description															
T_CRC32C																CRC32C (Section 3.6.4.1.1)																Castagnoli CRC32 (iSCSI, ext4, etc.), with normal form polynomial 0x1EDC6F41.															
T_HMAC-SHA256																HMAC-SHA256 (Section 3.6.4.1.2)																HMAC (RFC 2104) using SHA256 hash.															
T_RSA-SHA256																RSA-SHA256 (Section 3.6.4.1.3)																RSA public key signature using SHA256 digest.															
EC-SECP-256K1																SECP-256K1 (Section 3.6.4.1.3)																Elliptic Curve signature with SECP-256K1 parameters (see [ECC]).															
EC-SECP-384R1																SECP-384R1 (Section 3.6.4.1.3)																Elliptic Curve signature with SECP-384R1 parameters (see [ECC]).															

Table 10: CCNx Validation Types

3.6.4.1.1. Message Integrity Checks

MICs do not require additional data in order to perform the verification. An example is CRC32C that has a "0" length value.

3.6.4.1.2. Message Authentication Checks

MACs are useful for communication between two trusting parties who have already shared private keys. Examples include an RSA signature of a SHA256 digest or others. They rely on a KeyId. Some MACs might use more than a KeyId, but those would be defined in the future.

3.6.4.1.3. Signature

Signature type Validators specify a digest mechanism and a signing algorithm to verify the message. Examples include RSA signature og a SHA256 digest, an Elliptic Curve signature with SECP-256K1 parameters, etc. These Validators require a KeyId and a mechanism for locating the publishers public key (a KeyLocator) - optionally a PublicKey or Certificate or KeyLink.

3.6.4.1.4. Validation Dependent Data

Different Validation Algorithms require access to different pieces of data contained in the ValidationAlgorithm TLV. As described above, Key Ids, Key Locators, Public Keys, Certificates, Links and Key Names all play a role in different Validation Algorithms. Any number of Validation Dependent Data containers can be present in a Validation Algorithm TLV.

Following is a table of CCNx ValidationType dependent data types:

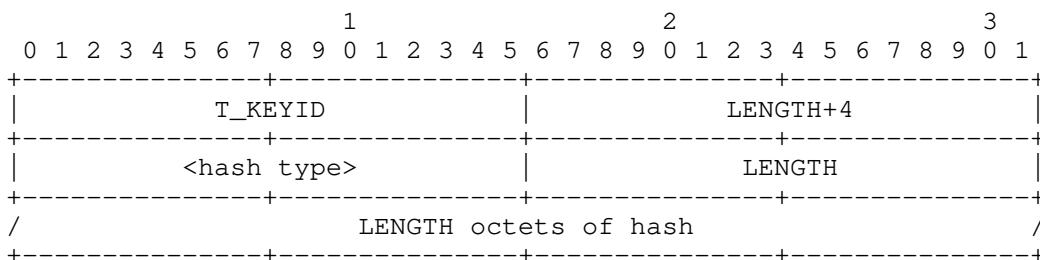
Abbrev	Name	Description
T_KEYID	SignerKeyId (Section 3.6.4.1.4.1)	An identifier of the shared secret or public key associated with a MAC or Signature.
T_PUBLICKEY	Public Key (Section 3.6.4.1.4.2)	DER encoded public key.
T_CERT	Certificate (Section 3.6.4.1.4.3)	DER encoded X509 certificate.
T_KEYLINK	KeyLink (Section 3.6.4.1.4.4)	A CCNx Link object.
T_SIGTIME	SignatureTime (Section 3.6.4.1.4.5)	A millisecond timestamp indicating the time when the signature was created.

Table 11: CCNx Validation Dependent Data Types

3.6.4.1.4.1. KeyId

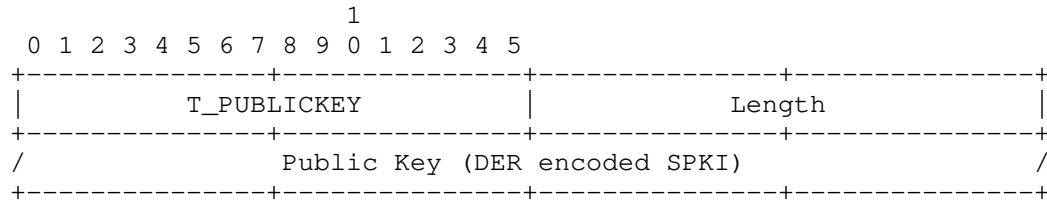
The KeyId is the publisher key identifier. It is similar to a Subject Key Identifier from X509 [RFC 5280, Section 4.2.1.2]. It should be derived from the key used to sign, such as from the SHA-256 hash of the key. It applies to both public/private key systems and to symmetric key systems.

The KeyId is represented using the Section 3.3.3. If a protocol uses a non-hash identifier, it should use one of the reserved values.

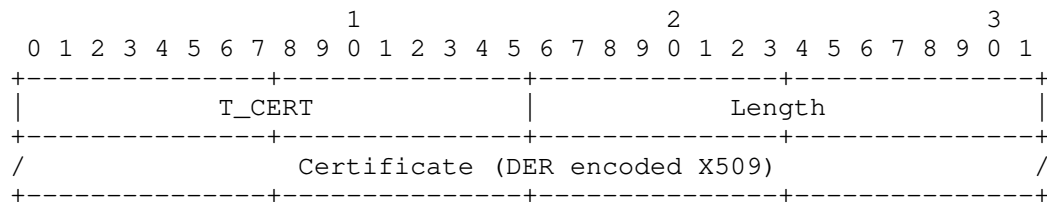


3.6.4.1.4.2. Public Key

A Public Key is a DER encoded Subject Public Key Info block, as in an X509 certificate.



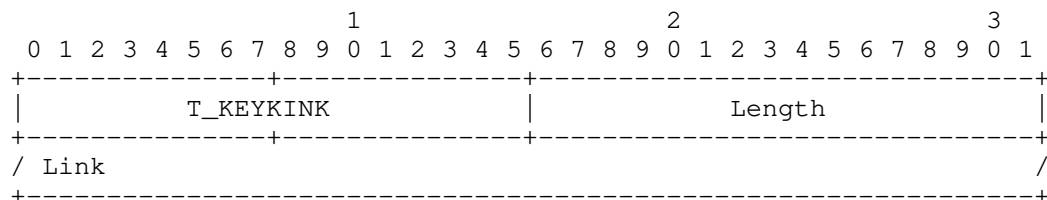
3.6.4.1.4.3. Certificate



3.6.4.1.4.4. KeyLink

A KeyLink type KeyLocator is a Link.

The KeyLink ContentObjectHashRestr, if included, is the digest of the Content Object identified by KeyLink, not the digest of the public key. Likewise, the KeyIdRestr of the KeyLink is the KeyId of the ContentObject, not necessarily of the wrapped key.

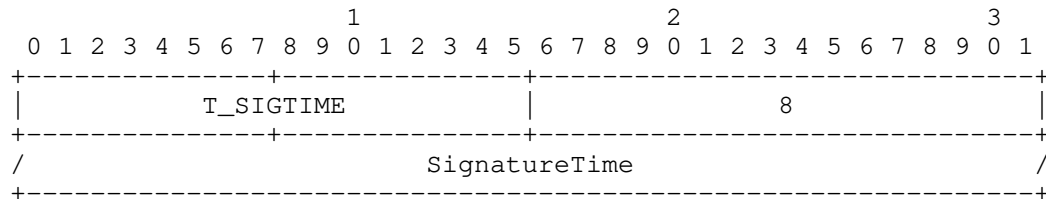


3.6.4.1.4.5. SignatureTime

The SignatureTime is a millisecond timestamp indicating the time at which a signature was created. The signer sets this field to the current time when creating a signature. A verifier may use this time to determine whether or not the signature was created during the validity period of a key, or if it occurred in a reasonable sequence with other associated signatures. The SignatureTime is unrelated to

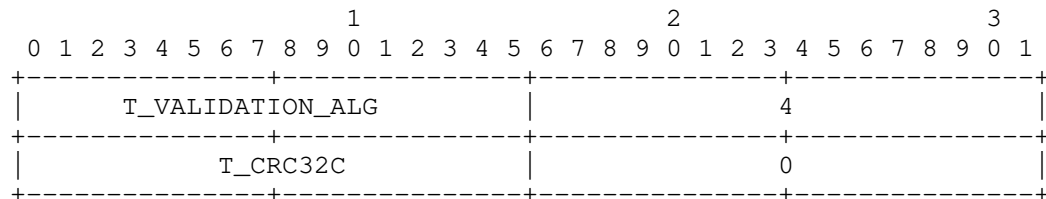
any time associated with the actual CCNx Message, which could have been created long before the signature. The default behavior is to always include a SignatureTime when creating an authenticated message (e.g. HMAC or RSA).

SignatureTime is a network byte ordered unsigned integer of the number of milliseconds since the epoch in UTC of when the signature was created. It is a fixed 64-bit field.

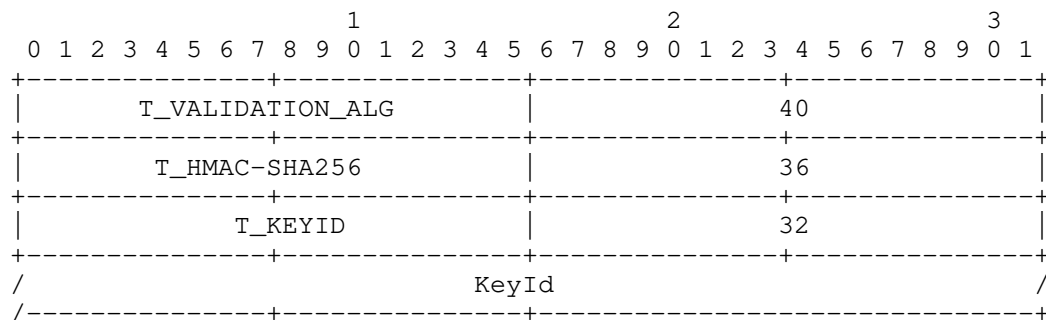


3.6.4.1.5. Validation Examples

As an example of a MIC type validation, the encoding for CRC32C validation would be:



As an example of a MAC type validation, the encoding for an HMAC using a SHA256 hash would be:



As an example of a Signature type validation, the encoding for an RSA public key signing using a SHA256 digest and Public Key would be:

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
T_VALIDATION_ALG										44 + Variable Length																					
T_RSA-SHA256										40 + Variable Length																					
T_KEYID										32																					
										KeyId																					
T_PUBLICKEY										Variable Length (~ 160)																					
										Public Key (DER encoded SPKI)																					

3.6.4.2. Validation Payload

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
T_VALIDATION_PAYLOAD										ValidationPayloadLength																					
/ Type-dependent data																															

The ValidationPayload contains the validation output, such as the CRC32C code or the RSA signature.

4. IANA Considerations

This section details each kind of protocol value that can be registered. Each type registry can be updated by incrementally expanding the type space, i.e., by allocating and reserving new types. As per [RFC5226] this section details the creation of the "CCNx Registry" and several sub-registries.

Property	Value
Name	CCNx Registry
Abbrev	CCNx

Registry Creation

4.1. Packet Type Registry

The following packet types should be allocated. A PacketType MUST be 1 byte. New packet types are allocated via "RFC Required" action.

Property	Value
Name	Packet Type Registry
Parent	CCNx Registry
Review process	RFC Required
Syntax	1 octet

Registry Creation

Type	Name	Reference
%x00	PT_INTEREST	Fixed Header Types (Section 3.2)
%x01	PT_CONTENT	Fixed Header Types (Section 3.2)
%x02	PT_RETURN	Fixed Header Types (Section 3.2)

Packet Type Namespace

4.2. Interest Return Code Registry

The following InterestReturn code types should be allocated.

Property	Value
Name	Interest Return Code
Parent	CCNx Registry
Review process	Specification Required
Syntax	1 octet

Registry Creation

Type	Name	Reference
%x00	Reserved	
%x01	T_RETURN_NO_ROUTE	Fixed Header Types (Section 3.2.3.3)
%x02	T_RETURN_LIMIT_EXCEEDED	Fixed Header Types (Section 3.2.3.3)
%x03	T_RETURN_NO_RESOURCES	Fixed Header Types (Section 3.2.3.3)
%x04	T_RETURN_PATH_ERROR	Fixed Header Types (Section 3.2.3.3)
%x05	T_RETURN_PROHIBITED	Fixed Header Types (Section 3.2.3.3)
%x06	T_RETURN_CONGESTED	Fixed Header Types (Section 3.2.3.3)
%x07	T_RETURN_MTU_TOO_LARGE	Fixed Header Types (Section 3.2.3.3)
%x08	T_RETURN_UNSUPPORTED_HASH_RESTRICTION	Fixed Header Types (Section 3.2.3.3)
%x09	T_RETURN_MALFORMED_INTEREST	Fixed Header Types (Section 3.2.3.3)

Interest Return Type Namespace

4.3. Hop-by-Hop Type Registry

The following hop-by-hop types should be allocated.

Property	Value
Name	Hop-by-Hop Type Registry
Parent	CCNx Registry
Review process	RFC Required
Syntax	2 octet TLV type

Registry Creation

Type	Name	Reference
%x0000	Reserved	
%x0001	T_INTLIFE	Hop-by-hop TLV headers (Section 3.4)
%x0002	T_CACHETIME	Hop-by-hop TLV headers (Section 3.4)
%x0003	T_MSGHASH	Hop-by-hop TLV headers (Section 3.4)
%x0004 – %x0007	Reserved	
%x0FFE	T_PAD	Pad (Section 3.3.1)
%x0FFF	T_ORG	Organization-Specific TLVs (Section 3.3.2)
%x1000–%x1FFF	Reserved	Experimental Use (Section 3)

Hop-by-Hop Type Namespace

4.4. Top-Level Type Registry

The following top-level types should be allocated.

Property	Value
Name	Top-Level Type Registry
Parent	CCNx Registry
Review process	RFC Required
Syntax	2 octet TLV type

Registry Creation

Type	Name	Reference
%x0000	Reserved	
%x0001	T_INTEREST	Top-Level Types (Section 3.5)
%x0002	T_OBJECT	Top-Level Types (Section 3.5)
%x0003	T_VALIDATION_ALG	Top-Level Types (Section 3.5)
%x0004	T_VALIDATION_PAYLOAD	Top-Level Types (Section 3.5)

Top-Level Type Namespace

4.5. Name Segment Type Registry

The following name segment types should be allocated.

Property	Value
Name	Name Segment Type Registry
Parent	CCNx Registry
Review process	Specification Required
Syntax	2 octet TLV type

Registry Creation

Type	Name	Reference
%x0000	Reserved	
%x0001	T_NAMESEGMENT	Name (Section 3.6.1)
%x0002	T_IPID	Name (Section 3.6.1)
%x0010 – %x0013	Reserved	Used in other drafts
%x0FFF	T_ORG	Organization-Specific TLVs (Section 3.3.2)
%x1000 – %x1FFF	T_APP:00 – T_APP:4096	Application Components (Section 3.6.1)

Name Segment Type Namespace

4.6. Message Type Registry

The following CCNx message segment types should be allocated.

Property	Value
Name	Message Type Registry
Parent	CCNx Registry
Review process	RFC Required
Syntax	2 octet TLV type

Registry Creation

Type	Name	Reference
%x0000	T_NAME	Message Types (Section 3.6)
%x0001	T_PAYLOAD	Message Types (Section 3.6)
%x0002	T_KEYIDRESTR	Message Types (Section 3.6)
%x0003	T_OBJHASHRESTR	Message Types (Section 3.6)
%x0005	T_PAYLDTYPE	Content Object Message Types (Section 3.6.2.2)
%x0006	T_EXPIRY	Content Object Message Types (Section 3.6.2.2)
%x0007 – %x000C	Reserved	Used in other RFC drafts
%x0FFE	T_PAD	Pad (Section 3.3.1)
%x0FFF	T_ORG	Organization-Specific TLVs (Section 3.3.2)
%x1000–%x1FFF	Reserved	Experimental Use (Section 3)

CCNx Message Type Namespace

4.7. Payload Type Registry

The following payload types should be allocated.

Property	Value
Name	PayloadType Registry
Parent	CCNx Registry
Review process	Specification Required
Syntax	Variable length unsigned integer

Registry Creation

Type	Name	Reference
%x00	T_PAYLOADTYPE_DATA	Payload Types (Section 3.6.2.2.1)
%x01	T_PAYLOADTYPE_KEY	Payload Types (Section 3.6.2.2.1)
%x02	T_PAYLOADTYPE_LINK	Payload Types (Section 3.6.2.2.1)

Payload Type Namespace

4.8. Validation Algorithm Type Registry

The following validation algorithm types should be allocated. Note: registration requires public specification of the algorithm.

Property	Value
Name	Validation Algorithm Type Registry
Parent	CCNx Registry
Review process	Specification Required
Syntax	2 octet TLV type

Registry Creation

Type	Name	Reference
%x0000	Reserved	
%x0001	Unassigned	
%x0002	T_CRC32C	Validation Algorithm (Section 3.6.4.1)
%x0003	Unassigned	
%x0004	T_HMAC-SHA256	Validation Algorithm (Section 3.6.4.1)
%x0005	T_RSA-SHA256	Validation Algorithm (Section 3.6.4.1)
%x0006	EC-SECP-256K1	Validation Algorithm (Section 3.6.4.1)
%x0007	EC-SECP-384R1	Validation Algorithm (Section 3.6.4.1)
%x0FFE	T_PAD	Pad (Section 3.3.1)
%x0FFF	T_ORG	Organization-Specific TLVs (Section 3.3.2)
%x1000-%x1FFF	Reserved	Experimental Use (Section 3)

Validation Algorithm Type Namespace

4.9. Validation Dependent Data Type Registry

The following validation dependent data types should be allocated.

Property	Value
Name	Validation Dependent Data Type Registry
Parent	CCNx Registry
Review process	RFC Required
Syntax	2 octet TLV type

Registry Creation

Type	Name	Reference
%x0000	Reserved	
%x0001 – %x0008	Unassigned	
%x0009	T_KEYID	Validation Dependent Data (Section 3.6.4.1.4)
%x000A	T_PUBLICKEYLOC	Validation Dependent Data (Section 3.6.4.1.4)
%x000B	T_PUBLICKEY	Validation Dependent Data (Section 3.6.4.1.4)
%x000C	T_CERT	Validation Dependent Data (Section 3.6.4.1.4)
%x000D	T_LINK	Validation Dependent Data (Section 3.6.4.1.4)
%x000E	T_KEYLINK	Validation Dependent Data (Section 3.6.4.1.4)
%x000F	T_SIGTIME	Validation Dependent Data (Section 3.6.4.1.4)
%x0FFF	T_ORG	Organization-Specific TLVs (Section 3.3.2)
%x1000–%x1FFF	Reserved	Experimental Use (Section 3)

Validation Dependent Data Type Namespace

4.10. Hash Function Type Registry

The following CCNx hash function types should be allocated. Note: registration requires public specification of the algorithm.

Property	Value
Name	Hash Function Type Registry
Parent	CCNx Registry
Review process	Specification Required
Syntax	2 octet TLV type

Registry Creation

Type	Name	Reference
%x0000	Reserved	
%x0001	T_SHA-256	Hash Format (Section 3.3.3)
%x0002	T_SHA-512	Hash Format (Section 3.3.3)
%x0FFF	T_ORG	Organization-Specific TLVs (Section 3.3.2)
%x1000-%x1FFF	Reserved	Experimental Use (Section 3)

CCNx Hash Function Type Namespace

5. Security Considerations

The CCNx protocol is a layer 3 network protocol, which may also operate as an overlay using other transports, such as UDP or other tunnels. It includes intrinsic support for message authentication via a signature (e.g. RSA or elliptic curve) or message authentication code (e.g. HMAC). In lieu of an authenticator, it may instead use a message integrity check (e.g. SHA or CRC). CCNx does not specify an encryption envelope, that function is left to a high-layer protocol (e.g. [esic]).

The CCNx message format includes the ability to attach MICs (e.g. SHA-256 or CRC), MACs (e.g. HMAC), and Signatures (e.g. RSA or ECDSA) to all packet types. This does not mean that it is a good idea to use an arbitrary ValidationAlgorithm, nor to include computationally expensive algorithms in Interest packets, as that could lead to computational DoS attacks. Applications should use an

explicit protocol to guide their use of packet signatures. As a general guideline, an application might use a MIC on an Interest to detect unintentionally corrupted packets. If one wishes to secure an Interest, one should consider using an encrypted wrapper and a protocol that prevents replay attacks, especially if the Interest is being used as an actuator. Simply using an authentication code or signature does not make an Interests secure. There are several examples in the literature on how to secure ICN-style messaging [mobile] [ace].

As a layer 3 protocol, this document does not describe how one arrives at keys or how one trusts keys. The CCNx content object may include a public key embedded in the object or may use the PublicKeyLocator field to point to a public key (or public key certificate) that authenticates the message. One key exchange specification is CCNxKE [ccnxke] [mobile], which is similar to the TLS 1.3 key exchange except it is over the CCNx layer 3 messages. Trust is beyond the scope of a layer-3 protocol protocol and left to applications or application frameworks.

The combination of an ephemeral key exchange (e.g. CCNxKE [ccnxke]) and an encapsulating encryption (e.g. [esic]) provides the equivalent of a TLS tunnel. Intermediate nodes may forward the Interests and Content Objects, but have no visibility inside. It also completely hides the internal names in those used by the encryption layer. This type of tunneling encryption is useful for content that has little or no cache-ability as it can only be used by someone with the ephemeral key. Short term caching may help with lossy links or mobility, but long term caching is usually not of interest.

Broadcast encryption or proxy re-encryption may be useful for content with multiple uses over time or many consumers. There is currently no recommendation for this form of encryption.

The specific encoding of messages will have security implications. This document uses a type-length-value (TLV) encoding. We chose to compromise between extensibility and unambiguous encodings of types and lengths. Some TLVs use variable length T and variable length L fields to accomodate a wide gamut of values while trying to be byte-efficient. Our TLV encoding uses a fixed length 2-byte T and 2-byte L. Using a fixed-length T and L field solves two problems. The first is aliases. If one is able to encode the same value, such as 0x2 and 0x02, in different byte lengths then one must decide if they mean the same thing, if they are different, or if one is illegal. If they are different, then one must always compare on the buffers not the integer equivalents. If one is illegal, then one must validate the TLV encoding -- every field of every packet at every hop. If they are the same, then one has the second problem: how to specify

packet filters. For example, if a name has 6 name components, then there are 7 T's and 7 L's, each of which might have up to 4 representations of the same value. That would be 14 fields with 4 encodings each, or 1001 combinations. It also means that one cannot compare, for example, a name via a memory function as one needs to consider that any embedded T or L might have a different format.

The Interest Return message has no authenticator from the previous hop. Therefore, the payload of the Interest Return should only be used locally to match an Interest. A node should never forward that Interest payload as an Interest. It should also verify that it sent the Interest in the Interest Return to that node and not allow anyone to negate Interest messages.

Caching nodes must take caution when processing content objects. It is essential that the Content Store obey the rules outlined in [CCNSemantics] to avoid certain types of attacks. Unlike NDN, CCNx 1.0 has no mechanism to work around an undesired result from the network (there are no "excludes"), so if a cache becomes poisoned with bad content it might cause problems retrieving content. There are three types of access to content from a content store: unrestricted, signature restricted, and hash restricted. If an Interest has no restrictions, then the requester is not particular about what they get back, so any matching cached object is OK. In the hash restricted case, the requester is very specific about what they want and the content store (and every forward hop) can easily verify that the content matches the request. In the signature verified case (often used for initial manifest discovery), the requester only knows the KeyId that signed the content. It is this case that requires the closest attention in the content store to avoid amplifying bad data. The content store must only respond with a content object if it can verify the signature -- this means either the content object carries the public key inside it or the Interest carries the public key in addition to the KeyId. If that is not the case, then the content store should treat the Interest as a cache miss and let an endpoint respond.

A user-level cache could perform full signature verification by fetching a public key according to the PublicKeyLocator. That is not, however, a burden we wish to impose on the forwarder. A user-level cache could also rely on out-of-band attestation, such as the cache operator only inserting content that it knows has the correct signature.

The CCNx grammar allows for hash algorithm agility via the HashType. It specifies a short list of acceptable hash algorithms that should be implemented at each forwarder. Some hash values only apply to end systems, so updating the hash algorithm does not affect forwarders --

they would simply match the buffer that includes the type-length-hash buffer. Some fields, such as the ConObjHash, must be verified at each hop, so a forwarder (or related system) must know the hash algorithm and it could cause backward compatibility problems if the hash type is updated.

A CCNx name uses binary matching whereas a URI uses a case insensitive hostname. Some systems may also use case insensitive matching of the URI path to a resource. An implication of this is that human-entered CCNx names will likely have case or non-ASCII symbol mismatches unless one uses a consistent URI normalization to the CCNx name. It also means that an entity that registers a CCNx routable prefix, say `ccnx:/example.com`, would need separate registrations for simple variations like `ccnx:/Example.com`. Unless this is addressed in URI normalization and routing protocol conventions, there could be phishing attacks.

For a more general introduction to ICN-related security concerns and approaches, see [RFC7927] and [RFC7945]

6. References

6.1. Normative References

- [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>>.

6.2. Informative References

- [ace] Shang, W., Yu, Y., Liang, T., Zhang, B., and L. Zhang, "NDN-ACE: Access control for constrained environments over named data networking", NDN Technical Report NDN-0036, 2015, <<http://new.named-data.net/wp-content/uploads/2015/12/ndn-0036-1-ndn-ace.pdf>>.
- [CCN semantics] Mosko, M., Solis, I., and C. Wood, "CCNx Semantics (Internet draft)", 2018, <<https://www.ietf.org/id/draft-irtf-icnrg-ccnxsemantics-09.txt>>.
- [ccnxke] Mosko, M., Uzun, E., and C. Wood, "CCNx Key Exchange Protocol Version 1.0", 2017, <<https://www.ietf.org/archive/id/draft-wood-icnrg-ccnxkeyexchange-02.txt>>.

- [CCNxURI] Mosko, M. and C. Wood, "The CCNx URI Scheme (Internet draft)", 2017, <<http://tools.ietf.org/html/draft-mosko-icnrg-ccnxuri-02>>.
- [CCNxz] Mosko, M., "CCNxz TLV Header Compression Experimental Code", 2016-2018, <<https://github.com/PARC/CCNxz>>.
- [compress] Mosko, M., "Header Compression for TLV-based Packets", 2016, <<https://datatracker.ietf.org/meeting/interim-2016-icnrg-02/materials/slides-interim-2016-icnrg-2-7>>.
- [ECC] Certicom Research, "SEC 2: Recommended Elliptic Curve Domain Parameters", 2010, <<http://www.secg.org/sec2-v2.pdf>>.
- [EpriseNumbers] IANA, "IANA Private Enterprise Numbers", 2015, <<http://www.iana.org/assignments/enterprise-numbers/enterprise-numbers>>.
- [esic] Mosko, M. and C. Wood, "Encrypted Sessions In CCNx (ESIC)", 2017, <<https://www.ietf.org/id/draft-wood-icnrg-esic-01.txt>>.
- [mobile] Mosko, M., Uzun, E., and C. Wood, "Mobile Sessions in Content-Centric Networks", IFIP Networking, 2017, <<http://dl.ifip.org/db/conf/networking/networking2017/1570334964.pdf>>.
- [nnc] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and R. Braynard, "Networking Named Content", 2009, <<http://dx.doi.org/10.1145/1658939.1658941>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.
- [RFC7927] Kutscher, D., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlich, "Information-Centric Networking (ICN) Research Challenges", 2016, <<https://trac.tools.ietf.org/html/rfc7927>>.

[RFC7945] Pentikousis, K., Ohlman, B., Davies, E., Spirou, S., and G. Boggia, "Information-Centric Networking: Evaluation and Security Considerations", 2016, <<https://trac.tools.ietf.org/html/rfc7945>>.

Authors' Addresses

Marc Mosko
PARC, Inc.
Palo Alto, California 94304
USA

Phone: +01 650-812-4405
Email: marc.mosko@parc.com

Ignacio Solis
LinkedIn
Mountain View, California 94043
USA

Email: nsolis@linkedin.com

Christopher A. Wood
University of California Irvine
Irvine, California 92697
USA

Phone: +01 315-806-5939
Email: woodcl@uci.edu

ICNRRG
Internet-Draft
Intended status: Experimental
Expires: July 28, 2019

M. Mosko
PARC, Inc.
I. Solis
LinkedIn
C. Wood
University of California Irvine
January 24, 2019

CCNx Semantics
draft-irtf-icnrg-ccnxsemantics-10

Abstract

This document describes the core concepts of the Content Centric Networking (CCNx) architecture and presents a network protocol based on two messages: Interests and Content Objects. It specifies the set of mandatory and optional fields within those messages and describes their behavior and interpretation. This architecture and protocol specification is independent of a specific wire encoding.

The protocol also uses a Control message called an InterestReturn, whereby one system can return an Interest message to the previous hop due to an error condition. This indicates to the previous hop that the current system will not respond to the Interest.

This document is a product of the Information Centric Networking research group (ICNRRG).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 28, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	4
1.2. Architecture	4
1.3. Protocol Overview	5
2. Protocol	9
2.1. Message Grammar	9
2.2. Consumer Behavior	12
2.3. Publisher Behavior	14
2.4. Forwarder Behavior	14
2.4.1. Interest HopLimit	15
2.4.2. Interest Aggregation	16
2.4.3. Content Store Behavior	17
2.4.4. Interest Pipeline	18
2.4.5. Content Object Pipeline	18
3. Names	19
3.1. Name Examples	20
3.2. Interest Payload ID	21
4. Cache Control	21
5. Content Object Hash	22
6. Link	22
7. Hashes	22
8. Validation	23
8.1. Validation Algorithm	23
9. Interest to Content Object matching	24
10. Interest Return	25
10.1. Message Format	25
10.2. ReturnCode Types	26
10.3. Interest Return Protocol	26
10.3.1. No Route	27
10.3.2. HopLimit Exceeded	28
10.3.3. Interest MTU Too Large	28

10.3.4.	No Resources	28
10.3.5.	Path Error	28
10.3.6.	Prohibited	28
10.3.7.	Congestion	29
10.3.8.	Unsupported Content Object Hash Algorithm	29
10.3.9.	Malformed Interest	29
11.	IANA Considerations	29
12.	Security Considerations	29
13.	References	32
13.1.	Normative References	32
13.2.	Informative References	32
	Authors' Addresses	34

1. Introduction

This document describes the principles of the CCNx architecture. It describes a network protocol that uses a hierarchical name to forward requests and to match responses to requests. It does not use endpoint addresses, such as Internet Protocol. Restrictions in a request can limit the response by the publickey of the response's signer or the cryptographic hash of the response. Every CCNx forwarder along the path does the name matching and restriction checking. The CCNx protocol fits within the broader framework of Information Centric Networking (ICN) protocols [RFC7927]. This document concerns the semantics of the protocol and is not dependent on a specific wire format encoding. The CCNx Messages [CCNMessages] document describes a type-length-value (TLV) wire protocol encoding. This section introduces the main concepts of CCNx, which are further elaborated in the remainder of the document.

The CCNx protocol derives from the early ICN work by Jacobson et al. [nnc]. Jacobson's version of CCNx is known as the 0.x version ("CCNx 0.x") and the present work is known as the 1.0 version ("CCNx 1.0"). There are two active implementations of CCNx 1.0. The most complete implementation is Community ICN (CINC) [cicn], a Linux Foundation project hosted at fd.io. Another active implementation is CCN-lite [ccnlite], with support for IoT systems and the RIOT operating system. CCNx 0.x formed the basis of the Named Data Networking [ndn] (NDN) university project.

The current CCNx 1.0 specification diverges from CCNx 0.x in a few significant areas. The most pronounced behavioral difference between CCNx 0.x and CCNx 1.0 is that CCNx 1.0 has a simpler response processing behavior. In both versions, a forwarder uses a hierarchical longest prefix match of a request name against the forwarding information base (FIB) to send the request through the network to a system that can issue a response. A forwarder must then match a response's name to a request's name to determine the reverse

path and deliver the response to the requester. In CCNx 0.x, the Interest name may be a hierarchical prefix of the response name, which allows a form of layer 3 content discovery. In CCNx 1.0, a response's name must exactly equal a request's name. Content discovery is performed by a higher-layer protocol.

CCNx Selectors [selectors] is an example of using a higher-layer protocol on top of the CCNx 1.0 layer-3 to perform content discovery. The selector protocol uses a method similar to the original CCNx 0.x techniques without requiring partial name matching of a response to a request in the forwarder.

The document represents the consensus of the ICN RG. It is the first ICN protocol from the RG, created from the early CCNx protocol [nnc] with significant revision and input from the ICN community and RG members. The draft has received critical reading by several members of the ICN community and the RG. The authors and RG chairs approve of the contents. The document is sponsored under the IRTF and is not issued by the IETF and is not an IETF standard. This is an experimental protocol and may not be suitable for any specific application and the specification may change in the future.

1.1. Requirements Language

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].

1.2. Architecture

We describe the architecture of the network in which CCNx operates and introduce certain terminology from [terminology]. The detailed behavior of each component and message grammars are in Section 2.

A producer (also called a publisher) is an endpoint that encapsulates content in Content Objects for transport in the CCNx network. A producer has a public/private keypair and signs (directly or indirectly) the content objects. Usually, the producer's keyid (hash of the public key) is well-known or may be derived from the producer's namespace via standard means.

A producer operates within one or more namespaces. A namespace is a name prefix that is represented in the forwarding information base (FIB). This allows a request to reach the producer and fetch a response (if one exists).

The forwarding information base (FIB) is a table that tells a forwarder where to send a request. It may point to a local

application, a local cache or content store, or to a remote system. If there is no matching entry in the FIB, a forwarder cannot process a request. The detailed rules on name matching to the FIB are given in Section 2.4.4. An endpoint has a FIB, though it may be a simple default route. An intermediate system (i.e. a router) typically has a much larger FIB. A core CCNx forwarder, for example, would know all the global routes.

A consumer is an endpoint that requests a name. It is beyond the scope of this document to describe how a consumer learns of a name or publisher keyid -- higher layer protocols build on top of CCNx handle those tasks, such as search engines or lookup services or well known names. The consumer constructs a request, called an Interest, and forwards it via the endpoint's FIB. The consumer should get back either a response, called a Content Object, that matches the Interest or a control message, called an InterestReturn, that indicates the network cannot handle the request.

There are three ways to detect errors in Interest handling. An InterestReturn is a network control message that indicates a low-level error like no route or out of resources. If an Interest arrives at a producer, but the producer does not have the requested content, the producer should send an application-specific error message (e.g. a not found message). Finally, a consumer may not receive anything, in which case it should timeout and, depending on the application, retry the request or return an error to the application.

1.3. Protocol Overview

The goal of CCNx is to name content and retrieve the content from the network without binding it to a specific network endpoint. A routing system (specified separately) populates the forwarding information base (FIB) tables at each CCNx router with hierarchical name prefixes that point towards the content producers under that prefix. A request finds matching content along those paths, in which case a response carries the data, or if no match is found a control message indicates the failure. A request may further refine acceptable responses with a restriction on the response's signer and the cryptographic hash of the response. The details of these restrictions are described below.

The CCNx name is a hierarchical series of path segments. Each path segment has a type and zero or more bytes. Matching two names is done as a binary comparison of the type and value, segment by segment. The human-readable form is defined under a URI scheme "ccnx:" [CCNxURI], though the canonical encoding of a name is a series of (type, octet string) pairs. There is no requirement that

any path segment be human readable or UTF-8. The first few segments in a name will be matched against the FIB and a routing protocol may put its own restrictions on the routable name components (e.g. a maximum length or character encoding rules). In principle, path segments and names have unbounded length, though in practice they are limited by the wire format encoding and practical considerations imposed by a routing protocol. Note that in CCNx path segments use binary comparison whereas in a URI the authority uses case-insensitive hostname (due to DNS).

The CCNx name, as used by the forwarder, is purposefully left as a general octet-encoded type and value without any requirements on human readability and character encoding. The reason for this is that we are concerned with how a forwarder processes names. We expect that applications, routing protocols, or other higher layers will apply their own conventions and restrictions on the allowed path segment types and path segment values.

CCNx is a request and response protocol to fetch chunks of data using a name. The integrity of each chunk may be directly asserted through a digital signature or Message Authentication Code (MAC), or, alternatively, indirectly via hash chains. Chunks may also carry weaker message integrity checks (MICs) or no integrity protection mechanism at all. Because provenance information is carried with each chunk (or larger indirectly protected block), we no longer need to rely on host identities, such as those derived from TLS certificates, to ascertain the chunk legitimacy. Data integrity is therefore a core feature of CCNx; it does not rely on the data transmission channel. There are several options for data confidentiality, discussed later.

This document only defines the general properties of CCNx names. In some isolated environments, CCNx users may be able to use any name they choose and either inject that name (or prefix) into a routing protocol or use other information foraging techniques. In the Internet environment, there will be policies around the formats of names and assignments of names to publishers, though those are not specified here.

The key concept of CCNx is that a subjective name is cryptographically bound to a fixed payload. These publisher-generated bindings can therefore be cryptographically verified. A named payload is thus the tuple {Name, ExtraFields, Payload, ValidationAlgorithm}, where all fields in the inner tuple are covered by the validation payload (e.g. signature). Consumers of this data can check the binding integrity by re-computing the same cryptographic hash and verifying the digital signature in ValidationPayload.

In addition to digital signatures (e.g. RSA), CCNx also supports message authentication codes (e.g. HMAC) and message integrity codes (e.g. SHA-256 or CRC). To maintain the cryptographic binding, there should be at least one object with a signature or authentication code, but not all objects require it. For example, a first object with a signature could refer to other objects via a hash chain, a Merkle tree, or a signed manifest. The later objects may not have any validation and rely purely on the references. The use of an integrity code (e.g. CRC) is intended for detecting accidental corruption in an Interest.

CCNx specifies a network protocol around Interests (request messages) and Content Objects (response messages) to move named payloads. An Interest includes the Name -- which identifies the desired response -- and optional matching restrictions. Restrictions limit the possible matching Content Objects. Two restrictions exist: `KeyIdRestr` and `ContentObjectHashRestr`. The first restriction on the `KeyId` limits responses to those signed with a `ValidationAlgorithm` `KeyId` field equal to the restriction. The second is the `ContentObjectHash` restriction, which limits the response to one where the cryptographic hash of the entire named payload is equal to the restriction.

The hierarchy of a CCNx Name is used for routing via the longest matching prefix in a Forwarder. The longest matching prefix is computed name segment by name segment in the hierarchical path name, where each name segment must be exactly equal to match. There is no requirement that the prefix be globally routable. Within a deployment any local routing may be used, even one that only uses a single flat (non-hierarchical) name segment.

Another concept of CCNx is that there should be flow balance between Interest messages and Content Object messages. At the network level, an Interest traveling along a single path should elicit no more than one Content Object response. If some node sends the Interest along more than one path, that node should consolidate the responses such that only one Content Object flows back towards the requester. If an Interest is sent broadcast or multicast on a multiple-access media, the sender should be prepared for multiple responses unless some other media-dependent mechanism like gossip suppression or leader election is used.

As an Interest travels the forward path following the Forwarding Information Base (FIB), it establishes state at each forwarder such that a Content Object response can trace its way back to the original requester(s) without the requester needing to include a routable return address. We use the notional Pending Interest Table (PIT) as

a method to store state that facilitates the return of a Content Object.

The notional PIT table stores the last hop of an Interest plus its Name and optional restrictions. This is the data required to match a Content Object to an Interest (see Section 9). When a Content Object arrives, it must be matched against the PIT to determine which entries it satisfies. For each such entry, at most one copy of the Content Object is sent to each listed last hop in the PIT entries.

An actual PIT table is not mandated by the specification. An implementation may use any technique that gives the same external behavior. There are, for example, research papers that use techniques like label switching in some parts of the network to reduce the per-node state incurred by the PIT table [dart]. Some implementations store the PIT state in the FIB, so there is not a second table.

If multiple Interests with the same {Name, KeyIdRestr, ContentObjectHashRestr} tuple arrive at a node before a Content Object matching the first Interest comes back, they are grouped in the same PIT entry and their last hops aggregated (see Section 2.4.2). Thus, one Content Object might satisfy multiple pending Interests in a PIT.

In CCNx, higher-layer protocols are often called "name-based protocols" because they operate on the CCNx Name. For example, a versioning protocol might append additional name segments to convey state about the version of payload. A content discovery protocol might append certain protocol-specific name segments to a prefix to discover content under that prefix. Many such protocols may exist and apply their own rules to Names. They may be layered with each protocol encapsulating (to the left) a higher layer's Name prefix.

This document also describes a control message called an InterestReturn. A network element may return an Interest message to a previous hop if there is an error processing the Interest. The returned Interest may be further processed at the previous hop or returned towards the Interest origin. When a node returns an Interest it indicates that the previous hop should not expect a response from that node for the Interest, i.e., there is no PIT entry left at the returning node for a Content Object to follow.

There are multiple ways to describe larger objects in CCNx. Aggregating layer-3 content objects in to larger objects is beyond the scope of this document. One proposed method, FLIC [flic], uses a manifest to enumerate the pieces of a larger object. Manifests are, themselves, Content Objects. Another option is to use a convention

in the Content Object name, as in the CCNx Chunking [chunking] protocol where a large object is broken in to small chunks and each chunk receives a special name component indicating its serial order.

At the semantic level, described in this document, we do not address fragmentation. One experimental fragmentation protocol, BeginEnd Fragments [befrags] uses a multipoint-PPP style technique for use over layer-2 interfaces with the CCNx Messages [CCNMessages] TLV wire format specification.

With these concepts, the remainder of the document specifies the behavior of a forwarder in processing Interest, Content Object, and InterestReturn messages.

2. Protocol

CCNx is a request and response protocol. A request is called an Interest and a response is called a Content Object. CCNx also uses a 1-hop control message called InterestReturn. These are, as a group, called CCNx Messages.

2.1. Message Grammar

The CCNx message ABNF [RFC5234] grammar is shown in Figure 1. The grammar does not include any encoding delimiters, such as TLVs. Specific wire encodings are given in a separate document. If a Validation section exists, the Validation Algorithm covers from the Body (BodyName or BodyOptName) through the end of the ValidationAlg section. The InterestLifetime, CacheTime, and Return Code fields exist outside of the validation envelope and may be modified.

The various fields -- in alphabetical order -- are defined as:

- o AbsTime: Absolute times are conveyed as the 64-bit UTC time in milliseconds since the epoch (standard POSIX time).
- o CacheTime: The absolute time after which the publisher believes there is low value in caching the content object. This is a recommendation to caches (see Section 4).
- o ConObjField: These are optional fields that may appear in a Content Object.
- o ConObjHash: The value of the Content Object Hash, which is the SHA256-32 over the message from the beginning of the body to the end of the message. Note that this coverage area is different from the ValidationAlg. This value SHOULD NOT be trusted across domains (see Section 5).

- o **ExpiryTime:** An absolute time after which the content object should be considered expired (see Section 4).
- o **Hash:** Hash values carried in a Message carry a HashType to identify the algorithm used to generate the hash followed by the hash value. This form is to allow hash agility. Some fields may mandate a specific HashType.
- o **HopLimit:** Interest messages may loop if there are loops in the forwarding plane. To eventually terminate loops, each Interest carries a HopLimit that is decremented after each hop and no longer forwarded when it reaches zero. See Section 2.4.
- o **InterestField:** These are optional fields that may appear in an Interest message.
- o **KeyIdRestr:** The KeyId Restriction. A Content Object must have a KeyId with the same value as the restriction.
- o **ContentObjectHashRestr:** The Content Object Hash Restriction. A content object must hash to the same value as the restriction using the same HashType. The ContentObjectHashRestr MUST use SHA256-32.
- o **KeyId:** An identifier for the key used in the ValidationAlg. For public key systems, this should be the SHA-256 hash of the public key. For symmetric key systems, it should be an identifier agreed upon by the parties.
- o **KeyLink:** A Link (see Section 6) that names how to retrieve the key used to verify the ValidationPayload. A message SHOULD NOT have both a KeyLink and a PublicKey.
- o **Lifetime:** The approximate time during which a requester is willing to wait for a response, usually measured in seconds. It is not strongly related to the network round trip time, though it must necessarily be larger.
- o **Name:** A name is made up of a non-empty first segment followed by zero or more additional segments, which may be of 0 length. Path segments are opaque octet strings, and are thus case-sensitive if encoding UTF-8. An Interest MUST have a Name. A Content Object MAY have a Name (see Section 9). The segments of a name are said to be complete if its segments uniquely identify a single Content Object. A name is exact if its segments are complete. An Interest carrying a full name is one which specifies an exact name and the ContentObjectHashRestr of the corresponding Content Object.

- o Payload: The message's data, as defined by PayloadType.
- o PayloadType: The format of the Payload. If missing, assume DataType. DataType means the payload is opaque application bytes. KeyType means the payload is a DER-encoded public key. LinkType means it is one or more Links (see Section 6).
- o PublicKey: Some applications may wish to embed the public key used to verify the signature within the message itself. The PublicKey is DER encoded. A message SHOULD NOT have both a KeyLink and a PublicKey.
- o RelTime: A relative time, measured in milli-seconds.
- o ReturnCode: States the reason an Interest message is being returned to the previous hop (see Section 10.2).
- o SigTime: The absolute time (UTC milliseconds) when the signature was generated.
- o Vendor: Vendor-specific opaque data. The Vendor data includes the IANA Private Enterprise Numbers [EpriseNumbers], followed by vendor-specific information. CCNx allows vendor-specific data in most locations of the grammar.

```

Message      := Interest / ContentObject / InterestReturn
Interest     := IntHdr BodyName [Validation]
IntHdr       := HopLimit [Lifetime] *Vendor
ContentObject := ConObjHdr BodyOptName [Validation]
ConObjHdr    := [CacheTime / ConObjHash] *Vendor
InterestReturn := ReturnCode Interest
BodyName     := Name Common
BodyOptName  := [Name] Common
Common       := *Field [Payload]
Validation   := ValidationAlg ValidatonPayload

Name         := FirstSegment *Segment
FirstSegment := 1* OCTET / Vendor
Segment      := 0* OCTET / Vendor

ValidationAlg := (RSA-SHA256 / HMAC-SHA256 / CRC32C) *Vendor
ValidatonPayload := 1* OCTET
RSA-SHA256     := KeyId [PublicKey] [SigTime] [KeyLink]
HMAC-SHA256    := KeyId [SigTime] [KeyLink]
CRC32C         := [SigTime]

AbsTime        := 8 OCTET ; 64-bit UTC msec since epoch
CacheTime      := AbsTime

```



```

ConObjField    := ExpiryTime / PayloadType
ConObjHash     := Hash ; The Content Object Hash
DataType       := "1"
ExpiryTime     := AbsTime
Field          := InterestField / ConObjField / Vendor
Hash           := HashType 1* OCTET
HashType       := SHA256-32 / SHA512-64 / SHA512-32
HopLimit       := OCTET
InterestField  := KeyIdRestr / ContentObjectHashRestr
KeyId          := 1* OCTET ; key identifier
KeyIdRestr     := 1* OCTET
KeyLink        := Link
KeyType        := "2"
Lifetime       := RelTime
Link           := Name [KeyIdResr] [ContentObjectHashRestr]
LinkType       := "3"
ContentObjectHashRestr := Hash
Payload        := *OCTET
PayloadType    := DataType / KeyType / LinkType
PublicKey      := ; DER-encoded public key
RelTime        := 1* OCTET ; msec
ReturnCode     := ; see Section 10.2
SigTime        := AbsTime
Vendor         := PEN 0* OCTET
PEN            := ; IANA Private Enterprise Number

```

Figure 1

2.2. Consumer Behavior

To request a piece of content for a given {Name, [KeyIdRest], [ContentObjectHashRestr]} tuple, a consumer creates an Interest message with those values. It MAY add a validation section, typically only a CRC32C. A consumer MAY put a Payload field in an Interest to send additional data to the producer beyond what is in the Name. The Name is used for routing and may be remembered at each hop in the notional PIT table to facilitate returning a content object; Storing large amounts of state in the Name could lead to high memory requirements. Because the Payload is not considered when forwarding an Interest or matching a Content Object to an Interest, a consumer SHOULD put an Interest Payload ID (see Section 3.2) as part of the name to allow a forwarder to match Interests to content objects and avoid aggregating Interests with different payloads. Similarly, if a consumer uses a MAC or a signature, it SHOULD also include a unique segment as part of the name to prevent the Interest from being aggregated with other Interests or satisfied by a Content Object that has no relation to the validation.

The consumer SHOULD specify an InterestLifetime, which is the length of time the consumer is willing to wait for a response. The InterestLifetime is an application-scale time, not a network round trip time (see Section 2.4.2). If not present, the InterestLifetime will use a default value (2 seconds).

The consumer SHOULD set the Interest HopLimit to a reasonable value or use the default 255. If the consumer knows the distances to the producer via routing, it SHOULD use that value.

A consumer hands off the Interest to its first forwarder, which will then forward the Interest over the network to a publisher (or replica) that may satisfy it based on the name (see Section 2.4).

Interest messages are unreliable. A consumer SHOULD run a transport protocol that will retry the Interest if it goes unanswered, up to the InterestLifetime. No transport protocol is specified in this document.

The network MAY send to the consumer an InterestReturn message that indicates the network cannot fulfill the Interest. The ReturnCode specifies the reason for the failure, such as no route or congestion. Depending on the ReturnCode, the consumer MAY retry the Interest or MAY return an error to the requesting application.

If the content was found and returned by the first forwarder, the consumer will receive a Content Object. The consumer SHOULD:

- o Ensure the content object is properly formatted.
- o Verify that the returned Name matches a pending request. If the request also had KeyIdRestr or ObjHashRest, it MUST also validate those properties.
- o If the content object is signed, it SHOULD cryptographically verify the signature. If it does not have the corresponding key, it SHOULD fetch the key, such as from a key resolution service or via the KeyLink.
- o If the signature has a SigTime, the consumer MAY use that in considering if the signature is valid. For example, if the consumer is asking for dynamically generated content, it should expect the SigTime to not be before the time the Interest was generated.
- o If the content object is signed, it should assert the trustworthiness of the signing key to the namespace. Such an assertion is beyond the scope of this document, though one may use

traditional PKI methods, a trusted key resolution service, or methods like [trust].

- o It MAY cache the content object for future use, up to the ExpiryTime if present.
- o A consumer MAY accept a content object off the wire that is expired. It may happen that a packet expires while in flight, and there is no requirement that forwarders drop expired packets in flight. The only requirement is that content stores, caches, or producers MUST NOT respond with an expired content object.

2.3. Publisher Behavior

This document does not specify the method by which names populate a Forwarding Information Base (FIB) table at forwarders (see Section 2.4). A publisher is either configured with one or more name prefixes under which it may create content, or it chooses its name prefixes and informs the routing layer to advertise those prefixes.

When a publisher receives an Interest, it SHOULD:

- o Verify that the Interest is part of the publishers namespace(s).
- o If the Interest has a Validation section, verify the ValidationPayload. Usually an Interest will only have a CRC32C unless the publisher application specifically accommodates other validations. The publisher MAY choose to drop Interests that carry a Validation section if the publisher application does not expect those signatures as this could be a form of computational denial of service. If the signature requires a key that the publisher does not have, it is NOT RECOMMENDED that the publisher fetch the key over the network, unless it is part of the application's expected behavior.
- o Retrieve or generate the requested content object and return it to the Interest's previous hop. If the requested content cannot be returned, the publisher SHOULD reply with an InterestReturn or a content object with application payload that says the content is not available; this content object should have a short ExpiryTime in the future or not be cacheable (i.e. an expiry time of 0).

2.4. Forwarder Behavior

A forwarder routes Interest messages based on a Forwarding Information Base (FIB), returns Content Objects that match Interests to the Interest's previous hop, and processes InterestReturn control messages. It may also keep a cache of Content Objects in the

notional Content Store table. This document does not specify the internal behavior of a forwarder -- only these and other external behaviors.

In this document, we will use two processing pipelines, one for Interests and one for Content Objects. Interest processing is made up of checking for duplicate Interests in the PIT (see Section 2.4.2), checking for a cached Content Object in the Content Store (see Section 2.4.3), and forwarding an Interest via the FIB. Content Store processing is made up of checking for matching Interests in the PIT and forwarding to those previous hops.

2.4.1. Interest HopLimit

Interest looping is not prevented in CCNx. An Interest traversing loops is eventually discarded using the hop-limit field of the Interest, which is decremented at each hop traversed by the Interest.

A loop may also terminate because the Interest is aggregated with it's previous PIT entry along the loop. In this case, the Content will be sent back along the loop and eventually return to a node that already forwarded the content, so it will likely not have a PIT entry any more. When the content reaches a node without a PIT entry, it will be discarded. It may be that a new Interest or another looped Interest will return to that same node, in which case the node will either return a cached response to make a new PIT entry, as below.

The HopLimit is the last resort method to stop Interest loops where a Content Object chases an Interest around a loop and where the intermediate nodes, for whatever reason, no longer have a PIT entry and do not cache the Content Object.

Every Interest MUST carry a HopLimit. An Interest received from a local application MAY have a 0 HopLimit, which restricts the Interest to other local sources.

When an Interest is received from another forwarder, the HopLimit MUST be positive, otherwise the forwarder will discard the Interest. A forwarder MUST decrement the HopLimit of an Interest by at least 1 before it is forwarded.

If the decremented HopLimit equals 0, the Interest MUST NOT be forwarded to another forwarder; it MAY be sent to a local publisher application or serviced from a local Content Store.

A RECOMMENDED HopLimit processing pipeline is below:

- o If Interest received from a remote system:

- * If received HopLimit is 0, optionally send InterestReturn (HopLimit Exceeded), and discard Interest.
- * Otherwise, decrement the HopLimit by 1.
- o Process as per Content Store and Aggregation rules.
- o If the Interest will be forwarded:
 - * If the (potentially decremented) HopLimit is 0, restrict forwarding to the local system.
 - * Otherwise, forward as desired to local or remote systems.

2.4.2. Interest Aggregation

Interest aggregation is when a forwarder receives an Interest message that could be satisfied by the response to another Interest message already forwarded by the node, so the forwarder suppresses forwarding the new Interest; it only records the additional previous hop so a Content Object sent in response to the first Interest will satisfy both Interests.

CCNx uses an Interest aggregation rule that assumes the InterestLifetime is akin to a subscription time and is not a network round trip time. Some previous aggregation rules assumed the lifetime was a round trip time, but this leads to problems of expiring an Interest before a response comes if the RTT is estimated too short or interfering with an ARQ scheme that wants to re-transmit an Interest but a prior interest over-estimated the RTT.

A forwarder MAY implement an Interest aggregation scheme. If it does not, then it will forward all Interest messages. This does not imply that multiple, possibly identical, Content Objects will come back. A forwarder MUST still satisfy all pending Interests, so one Content Object could satisfy multiple similar interests, even if the forwarded did not suppress duplicate Interest messages.

A RECOMMENDED Interest aggregation scheme is:

- o Two Interests are considered 'similar' if they have the same Name, KeyIdRestr, and ContentObjectHashRestr.
- o Let the notional value InterestExpiry (a local value at the forwarder) be equal to the receive time plus the InterestLifetime (or a platform-dependent default value if not present).

- o An Interest record (PIT entry) is considered invalid if its InterestExpiry time is in the past.
- o The first reception of an Interest MUST be forwarded.
- o A second or later reception of an Interest similar to a valid pending Interest from the same previous hop MUST be forwarded. We consider these a retransmission requests.
- o A second or later reception of an Interest similar to a valid pending Interest from a new previous hop MAY be aggregated (not forwarded). If this Interest has a larger HopLimit than the pending Interest, it MUST be forwarded.
- o Aggregating an Interest MUST extend the InterestExpiry time of the Interest record. An implementation MAY keep a single InterestExpiry time for all previous hops or MAY keep the InterestExpiry time per previous hop. In the first case, the forwarder might send a Content Object down a path that is no longer waiting for it, in which case the previous hop (next hop of the Content Object) would drop it.

2.4.3. Content Store Behavior

The Content Store is a special cache that is an integral part of a CCNx forwarder. It is an optional component. It serves to repair lost packets and handle flash requests for popular content. It could be pre-populated or use opportunistic caching. Because the Content Store could serve to amplify an attack via cache poisoning, there are special rules about how a Content Store behaves.

1. A forwarder MAY implement a Content Store. If it does, the Content Store matches a Content Object to an Interest via the normal matching rules (see Section 9).
2. If an Interest has a KeyIdRestr, then the Content Store MUST NOT reply unless it knows the signature on the matching Content Object is correct. It may do this by external knowledge (i.e., in a managed network or system with pre-populated caches) or by having the public key and cryptographically verifying the signature. A Content Store is NOT REQUIRED to verify signatures; if it does not, then it treats these cases like a cache miss.
3. If a Content Store chooses to verify signatures, then it MAY do so as follows. If the public key is provided in the Content Object itself (i.e., in the PublicKey field) or in the Interest, the Content Store MUST verify that the public key's SHA-256 hash is equal to the KeyId and that it verifies the signature. A

Content Store MAY verify the digital signature of a Content Object before it is cached, but it is not required to do so. A Content Store SHOULD NOT fetch keys over the network. If it cannot or has not yet verified the signature, it should treat the Interest as a cache miss.

4. If an Interest has an ContentObjectHashRestr, then the Content Store MUST NOT reply unless it knows the the matching Content Object has the correct hash. If it cannot verify the hash, then it should treat the Interest as a cache miss.
5. It must obey the Cache Control directives (see Section 4).

2.4.4. Interest Pipeline

1. Perform the HopLimit check (see Section 2.4.1).
2. Determine if the Interest can be aggregated, as per Section 2.4.2. If it can be, aggregate and do not forward the Interest.
3. If forwarding the Interest, check for a hit in the Content Store, as per Section 2.4.3. If a matching Content Object is found, return it to the Interest's previous hop. This injects the Content Store as per Section 2.4.5.
4. Lookup the Interest in the FIB. Longest prefix match (LPM) is performed name segment by name segment (not byte or bit). It SHOULD exclude the Interest's previous hop. If a match is found, forward the Interest. If no match is found or the forwarder choses to not forward due to a local condition (e.g., congestion), it SHOULD send an InterestReturn message, as per Section 10.

2.4.5. Content Object Pipeline

1. It is RECOMMENDED that a forwarder that receives a content object check that the Content Object came from an expected previous hop. An expected previous hop is one pointed to by the FIB or one recorded in the PIT as having had a matching Interest sent that way.
2. A Content Object MUST be matched to all pending Interests that satisfy the matching rules (see Section 9). Each satisfied pending Interest MUST then be removed from the set of pending Interests.

3. A forwarder SHOULD NOT send more than one copy of the received Content Object to the same Interest previous hop. It may happen, for example, that two Interest ask for the same Content Object in different ways (e.g., by name and by name an KeyId) and that they both come from the same previous hop. It is normal to send the same content object multiple times on the same interface, such as Ethernet, if it is going to different previous hops.
4. A Content Object SHOULD only be put in the Content Store if it satisfied an Interest (and passed rule #1 above). This is to reduce the chances of cache poisoning.

3. Names

A CCNx name is a composition of name segments. Each name segment carries a label identifying the purpose of the name segment, and a value. For example, some name segments are general names and some serve specific purposes, such as carrying version information or the sequencing of many chunks of a large object into smaller, signed Content Objects.

There are three different types of names in CCNx: prefix, exact, and full names. A prefix name is simply a name that does not uniquely identify a single Content Object, but rather a namespace or prefix of an existing Content Object name. An exact name is one which uniquely identifies the name of a Content Object. A full name is one which is exact and is accompanied by an explicit or implicit ConObjHash. The ConObjHash is explicit in an Interest and implicit in a Content Object.

Note that a forwarder does not need to know any semantics about a name. It only needs to be able to match a prefix to forward Interests and match an exact or full name to forward Content Objects. It is not sensitive to the name segment types.

The name segment labels specified in this document are given in the table below. Name Segment is a general name segment, typically occurring in the routable prefix and user-specified content name. Other segment types are for functional name components that imply a specific purpose.

Name	Description
Name Segment	A generic name segment that includes arbitrary octets.
Interest Payload ID	An octet string that identifies the payload carried in an Interest. As an example, the Payload ID might be a hash of the Interest Payload. This provides a way to differentiate between Interests based on the Payload solely through a Name Segment without having to include all the extra bytes of the payload itself.
Application Components	An application-specific payload in a name segment. An application may apply its own semantics to these components. A good practice is to identify the application in a Name segment prior to the application component segments.

Table 1: CCNx Name Segment Types

At the lowest level, a Forwarder does not need to understand the semantics of name segments; it need only identify name segment boundaries and be able to compare two name segments (both label and value) for equality. The Forwarder matches paths segment-by-segment against its forwarding table to determine a next hop.

3.1. Name Examples

This section uses the CCNx URI [CCNxURI] representation of CCNx names. Note that as per the message grammar, an Interest must have a Name with at least one name segment and that name segment must have at least 1 octet of value. A Content Object must have a similar name or no name at all. The FIB, on the other hand, could have 0-length names (a default route), or a first name segment with no value, or a regular name.

Name	Description
<code>ccnx:/</code>	A 0-length name, corresponds to a default route.
<code>ccnx:/NAME=</code>	A name with 1 segment of 0 length, distinct from <code>ccnx:/</code> .
<code>ccnx:/NAME=foo/APP:0=bar</code>	A 2-segment name, where the first segment is of type NAME and the second segment is of type APP:0.

Table 2: CCNx Name Examples

3.2. Interest Payload ID

An Interest may also have a Payload which carries state about the Interest but is not used to match a Content Object. If an Interest contains a payload, the Interest name should contain an Interest Payload ID (IPID). The IPID allows a PIT table entry to correctly multiplex Content Objects in response to a specific Interest with a specific payload ID. The IPID could be derived from a hash of the payload or could be a GUID or a nonce. An optional Metadata field defines the IPID field so other systems could verify the IPID, such as when it is derived from a hash of the payload. No system is required to verify the IPID.

4. Cache Control

CCNx supports two fields that affect cache control. These determine how a cache or Content Store handles a Content Object. They are not used in the fast path, but only to determine if a Content Object can be injected on to the fast path in response to an Interest.

The ExpiryTime is a field that exists within the signature envelope of a Validation Algorithm. It is the UTC time in milliseconds after which the Content Object is considered expired and MUST no longer be used to respond to an Interest from a cache. Stale content MAY be flushed from the cache.

The Recommended Cache Time (RCT) is a field that exists outside the signature envelope. It is the UTC time in milliseconds after which the publisher considers the Content Object to be of low value to cache. A cache SHOULD discard it after the RCT, though it MAY keep it and still respond with it. A cache MAY discard the content object

before the RCT time too; there is no contractual obligation to remember anything.

This formulation allows a producer to create a Content Object with a long ExpiryTime but short RCT and keep re-publishing the same, signed, Content Object over and over again by extending the RCT. This allows a form of "phone home" where the publisher wants to periodically see that the content is being used.

5. Content Object Hash

CCNx allows an Interest to restrict a response to a specific hash. The hash covers the Content Object message body and the validation sections, if present. Thus, if a Content Object is signed, its hash includes that signature value. The hash does not include the fixed or hop-by-hop headers of a Content Object. Because it is part of the matching rules (see Section 9), the hash is used at every hop.

There are two options for matching the content object hash restriction in an Interest. First, a forwarder could compute for itself the hash value and compare it to the restriction. This is an expensive operation. The second option is for a border device to compute the hash once and place the value in a header (ConObjHash) that is carried through the network. The second option, of course, removes any security properties from matching the hash, so SHOULD only be used within a trusted domain. The header SHOULD be removed when crossing a trust boundary.

6. Link

A Link is the tuple {Name, [KeyIdRestr], [ContentObjectHashRestr]}. The information in a Link comprises the fields of an Interest which would retrieve the Link target. A Content Object with PayloadType = "Link" is an object whose payload is one or more Links. This tuple may be used as a KeyLink to identify a specific object with the certificate wrapped key. It is RECOMMENDED to include at least one of KeyIdRestr or Content ObjectHashRestr. If neither restriction is present, then any Content Object with a matching name from any publisher could be returned.

7. Hashes

Several protocol fields use cryptographic hash functions, which must be secure against attack and collisions. Because these hash functions change over time, with better ones appearing and old ones falling victim to attacks, it is important that a CCNx protocol implementation supports hash agility.

In this document, we suggest certain hashes (e.g., SHA-256), but a specific implementation may use what it deems best. The normative CCNx Messages [CCNMessages] specification should be taken as the definition of acceptable hash functions and uses.

8. Validation

8.1. Validation Algorithm

The Validator consists of a ValidationAlgorithm that specifies how to verify the message and a ValidationPayload containing the validation output, e.g., the digital signature or MAC. The ValidationAlgorithm section defines the type of algorithm to use and includes any necessary additional information. The validation is calculated from the beginning of the CCNx Message through the end of the ValidationAlgorithm section. The ValidationPayload is the integrity value bytes, such as a MAC or signature.

Some Validators contain a KeyId, identifying the publisher authenticating the Content Object. If an Interest carries a KeyIdRestr, then that KeyIdRestr MUST exactly match the Content Object's KeyId.

Validation Algorithms fall into three categories: MICs, MACs, and Signatures. Validators using Message Integrity Code (MIC) algorithms do not need to provide any additional information; they may be computed and verified based only on the algorithm (e.g., CRC32C). MAC validators require the use of a KeyId identifying the secret key used by the authenticator. Because MACs are usually used between two parties that have already exchanged secret keys via a key exchange protocol, the KeyId may be any agreed-upon value to identify which key is used. Signature validators use public key cryptographic algorithms such as RSA, DSA, ECDSA. The KeyId field in the ValidationAlgorithm identifies the public key used to verify the signature. A signature may optionally include a KeyLocator, as described above, to bundle a Key or Certificate or KeyLink. MAC and Signature validators may also include a SignatureTime, as described above.

A PublicKeyLocator KeyLink points to a Content Object with a DER-encoded X509 certificate in the payload. In this case, the target KeyId must equal the first object's KeyId. The target KeyLocator must include the public key corresponding to the KeyId. That key must validate the target Signature. The payload is an X.509 certificate whose public key must match the target KeyLocator's key. It must be issued by a trusted authority, preferably specifying the valid namespace of the key in the distinguished name.

9. Interest to Content Object matching

A Content Object satisfies an Interest if and only if (a) the Content Object name, if present, exactly matches the Interest name, and (b) the ValidationAlgorithm KeyId of the Content Object exactly equals the Interest KeyIdRestr, if present, and (c) the computed Content ObjectHash exactly equals the Interest ContentObjectHashRestr, if present.

The matching rules are given by this predicate, which if it evaluates true means the Content Object matches the Interest. N_i = Name in Interest (may not be empty), K_i = KeyIdRestr in the interest (may be empty), H_i = ContentObjectHashRestr in Interest (may be empty). Likewise, N_o , K_o , H_o are those properties in the Content Object, where N_o and K_o may be empty; H_o always exists (it is an intrinsic property of the Content Object). For binary relations, we use $\&$ for AND and $|$ for OR. We use E for the EXISTS (not empty) operator and $!$ for the NOT EXISTS operator.

As a special case, if the ContentObjectHashRestr in the Interest specifies an unsupported hash algorithm, then no Content Object can match the Interest so the system should drop the Interest and MAY send an InterestReturn to the previous hop. In this case, the predicate below will never get executed because the Interest is never forwarded. If the system is using the optional behavior of having a different system calculate the hash for it, then the system may assume all hash functions are supported and leave it to the other system to accept or reject the Interest.

$$(!N_o \mid (N_i=N_o)) \ \& \ (!K_i \mid (K_i=K_o)) \ \& \ (!H_i \mid (H_i=H_o)) \ \& \ (E \ N_o \mid E \ H_i)$$

As one can see, there are two types of attributes one can match. The first term depends on the existence of the attribute in the Content Object while the next two terms depend on the existence of the attribute in the Interest. The last term is the "Nameless Object" restriction which states that if a Content Object does not have a Name, then it must match the Interest on at least the Hash restriction.

If a Content Object does not carry the Content ObjectHash as an expressed field, it must be calculated in network to match against. It is sufficient within an autonomous system to calculate a Content ObjectHash at a border router and carry it via trusted means within the autonomous system. If a Content Object ValidationAlgorithm does not have a KeyId then the Content Object cannot match an Interest with a KeyIdRestr.

10. Interest Return

This section describes the process whereby a network element may return an Interest message to a previous hop if there is an error processing the Interest. The returned Interest may be further processed at the previous hop or returned towards the Interest origin. When a node returns an Interest it indicates that the previous hop should not expect a response from that node for the Interest -- i.e., there is no PIT entry left at the returning node.

The returned message maintains compatibility with the existing TLV packet format (a fixed header, optional hop-by-hop headers, and the CCNx message body). The returned Interest packet is modified in only two ways:

- o The PacketType is set to InterestReturn to indicate a Feedback message.
- o The ReturnCode is set to the appropriate value to signal the reason for the return

The specific encodings of the Interest Return are specified in [CCNMessages].

A Forwarder is not required to send any Interest Return messages.

A Forwarder is not required to process any received Interest Return message. If a Forwarder does not process Interest Return messages, it SHOULD silently drop them.

The Interest Return message does not apply to a Content Object or any other message type.

An Interest Return message is a 1-hop message between peers. It is not propagated multiple hops via the FIB. An intermediate node that receives an InterestReturn may take corrective actions or may propagate its own InterestReturn to previous hops as indicated in the reverse path of a PIT entry.

10.1. Message Format

The Interest Return message looks exactly like the original Interest message with the exception of the two modifications mentioned above. The PacketType is set to indicate the message is an InterestReturn and the reserved byte in the Interest header is used as a Return Code. The numeric values for the PacketType and ReturnCodes are in [CCNMessages].

10.2. ReturnCode Types

This section defines the InterestReturn ReturnCode introduced in this RFC. The numeric values used in the packet are defined in [CCNMessages].

Name	Description
No Route (Section 10.3.1)	The returning Forwarder has no route to the Interest name.
HopLimit Exceeded (Section 10.3.2)	The HopLimit has decremented to 0 and need to forward the packet.
Interest MTU too large (Section 10.3.3)	The Interest's MTU does not conform to the required minimum and would require fragmentation.
No Resources (Section 10.3.4)	The node does not have the resources to process the Interest.
Path error (Section 10.3.5)	There was a transmission error when forwarding the Interest along a route (a transient error).
Prohibited (Section 10.3.6)	An administrative setting prohibits processing this Interest.
Congestion (Section 10.3.7)	The Interest was dropped due to congestion (a transient error).
Unsupported Content Object Hash Algorithm (Section 10.3.8)	The Interest was dropped because it requested a Content Object Hash Restriction using a hash algorithm that cannot be computed.
Malformed Interest (Section 10.3.9)	The Interest was dropped because it did not correctly parse.

Table 3: Interest Return Reason Codes

10.3. Interest Return Protocol

This section describes the Forwarder behavior for the various Reason codes for Interest Return. A Forwarder is not required to generate

any of the codes, but if it does, it MUST conform to this specification.

If a Forwarder receives an Interest Return, it SHOULD take these standard corrective actions. A forwarder is allowed to ignore Interest Return messages, in which case its PIT entry would go through normal timeout processes.

- o Verify that the Interest Return came from a next-hop to which it actually sent the Interest.
- o If a PIT entry for the corresponding Interest does not exist, the Forwarder should ignore the Interest Return.
- o If a PIT entry for the corresponding Interest does exist, the Forwarder MAY do one of the following:
 - * Try a different forwarding path, if one exists, and discard the Interest Return, or
 - * Clear the PIT state and send an Interest Return along the reverse path.

If a forwarder tries alternate routes, it MUST ensure that it does not use same same path multiple times. For example, it could keep track of which next hops it has tried and not re-use them.

If a forwarder tries an alternate route, it may receive a second InterestReturn, possibly of a different type than the first InterestReturn. For example, node A sends an Interest to node B, which sends a No Route return. Node A then tries node C, which sends a Prohibited. Node A should choose what it thinks is the appropriate code to send back to its previous hop

If a forwarder tries an alternate route, it should decrement the Interest Lifetime to account for the time spent thus far processing the Interest.

10.3.1. No Route

If a Forwarder receives an Interest for which it has no route, or for which the only route is back towards the system that sent the Interest, the Forwarder SHOULD generate a "No Route" Interest Return message.

How a forwarder manages the FIB table when it receives a No Route message is implementation dependent. In general, receiving a No Route Interest Return should not cause a forwarder to remove a route.

The dynamic routing protocol that installed the route should correct the route or the administrator who created a static route should correct the configuration. A forwarder could suppress using that next hop for some period of time.

10.3.2. HopLimit Exceeded

A Forwarder MAY choose to send HopLimit Exceeded messages when it receives an Interest that must be forwarded off system and the HopLimit is 0.

10.3.3. Interest MTU Too Large

If a Forwarder receives an Interest whose MTU exceeds the prescribed minimum, it MAY send an "Interest MTU Too Large" message, or it may silently discard the Interest.

If a Forwarder receives an "Interest MTU Too Large" it SHOULD NOT try alternate paths. It SHOULD propagate the Interest Return to its previous hops.

10.3.4. No Resources

If a Forwarder receives an Interest and it cannot process the Interest due to lack of resources, it MAY send an InterestReturn. A lack of resources could be the PIT table is too large, or some other capacity limit.

10.3.5. Path Error

If a forwarder detects an error forwarding an Interest, such as over a reliable link, it MAY send a Path Error Interest Return indicating that it was not able to send or repair a forwarding error.

10.3.6. Prohibited

A forwarder may have administrative policies, such as access control lists, that prohibit receiving or forwarding an Interest. If a forwarder discards an Interest due to a policy, it MAY send a Prohibited InterestReturn to the previous hop. For example, if there is an ACL that says /parc/private can only come from interface e0, but the Forwarder receives one from e1, the Forwarder must have a way to return the Interest with an explanation.

10.3.7. Congestion

If a forwarder discards an Interest due to congestion, it MAY send a Congestion InterestReturn to the previous hop.

10.3.8. Unsupported Content Object Hash Algorithm

If a Content Object Hash Restriction specifies a hash algorithm the forwarder cannot verify, the Interest should not be accepted and the forwarder MAY send an InterestReturn to the previous hop.

10.3.9. Malformed Interest

If a forwarder detects a structural or syntactical error in an Interest, it SHOULD drop the interest and MAY send an InterestReturn to the previous hop. This does not imply that any router must validate the entire structure of an Interest.

11. IANA Considerations

This memo includes no request to IANA.

12. Security Considerations

The CCNx protocol is a layer 3 network protocol, which may also operate as an overlay using other transports, such as UDP or other tunnels. It includes intrinsic support for message authentication via a signature (e.g. RSA or elliptic curve) or message authentication code (e.g. HMAC). In lieu of an authenticator, it may instead use a message integrity check (e.g. SHA or CRC). CCNx does not specify an encryption envelope, that function is left to a high-layer protocol (e.g. [esic]).

The CCNx message format includes the ability to attach MICs (e.g. SHA-256 or CRC), MACs (e.g. HMAC), and Signatures (e.g. RSA or ECDSA) to all packet types. This does not mean that it is a good idea to use an arbitrary ValidationAlgorithm, nor to include computationally expensive algorithms in Interest packets, as that could lead to computational DoS attacks. Applications should use an explicit protocol to guide their use of packet signatures. As a general guideline, an application might use a MIC on an Interest to detect unintentionally corrupted packets. If one wishes to secure an Interest, one should consider using an encrypted wrapper and a protocol that prevents replay attacks, especially if the Interest is being used as an actuator. Simply using an authentication code or signature does not make an Interests secure. There are several examples in the literature on how to secure ICN-style messaging [mobile] [ace].

As a layer 3 protocol, this document does not describe how one arrives at keys or how one trusts keys. The CCNx content object may include a public key embedded in the object or may use the `PublicKeyLocator` field to point to a public key (or public key certificate) that authenticates the message. One key exchange specification is CCNxKE [ccnxke] [mobile], which is similar to the TLS 1.3 key exchange except it is over the CCNx layer 3 messages. Trust is beyond the scope of a layer-3 protocol and left to applications or application frameworks.

The combination of an ephemeral key exchange (e.g. CCNxKE [ccnxke]) and an encapsulating encryption (e.g. [esic]) provides the equivalent of a TLS tunnel. Intermediate nodes may forward the Interests and Content Objects, but have no visibility inside. It also completely hides the internal names in those used by the encryption layer. This type of tunneling encryption is useful for content that has little or no cache-ability as it can only be used by someone with the ephemeral key. Short term caching may help with lossy links or mobility, but long term caching is usually not of interest.

Broadcast encryption or proxy re-encryption may be useful for content with multiple uses over time or many consumers. There is currently no recommendation for this form of encryption.

The specific encoding of messages will have security implications. [CCNMessages] uses a type-length-value (TLV) encoding. We chose to compromise between extensibility and unambiguous encodings of types and lengths. Some TLVs use variable length T and variable length L fields to accommodate a wide gamut of values while trying to be byte-efficient. Our TLV encoding uses a fixed length 2-byte T and 2-byte L. Using a fixed-length T and L field solves two problems. The first is aliases. If one is able to encode the same value, such as 0x2 and 0x02, in different byte lengths then one must decide if they mean the same thing, if they are different, or if one is illegal. If they are different, then one must always compare on the buffers not the integer equivalents. If one is illegal, then one must validate the TLV encoding -- every field of every packet at every hop. If they are the same, then one has the second problem: how to specify packet filters. For example, if a name has 6 name components, then there are 7 T's and 7 L's, each of which might have up to 4 representations of the same value. That would be 14 fields with 4 encodings each, or 1001 combinations. It also means that one cannot compare, for example, a name via a memory function as one needs to consider that any embedded T or L might have a different format.

The Interest Return message has no authenticator from the previous hop. Therefore, the payload of the Interest Return should only be used locally to match an Interest. A node should never forward that

Interest payload as an Interest. It should also verify that it sent the Interest in the Interest Return to that node and not allow anyone to negate Interest messages.

Caching nodes must take caution when processing content objects. It is essential that the Content Store obey the rules outlined in Section 2.4.3 to avoid certain types of attacks. Unlike NDN, CCNx 1.0 has no mechanism to work around an undesired result from the network (there are no "excludes"), so if a cache becomes poisoned with bad content it might cause problems retrieving content. There are three types of access to content from a content store: unrestricted, signature restricted, and hash restricted. If an Interest has no restrictions, then the requester is not particular about what they get back, so any matching cached object is OK. In the hash restricted case, the requester is very specific about what they want and the content store (and every forward hop) can easily verify that the content matches the request. In the signature verified case (often used for initial manifest discovery), the requester only knows the KeyId that signed the content. It is this case that requires the closest attention in the content store to avoid amplifying bad data. The content store must only respond with a content object if it can verify the signature -- this means either the content object carries the public key inside it or the Interest carries the public key in addition to the KeyId. If that is not the case, then the content store should treat the Interest as a cache miss and let an endpoint respond.

A user-level cache could perform full signature verification by fetching a public key according to the PublicKeyLocator. That is not, however, a burden we wish to impose on the forwarder. A user-level cache could also rely on out-of-band attestation, such as the cache operator only inserting content that it knows has the correct signature.

The CCNx grammar allows for hash algorithm agility via the HashType. It specifies a short list of acceptable hash algorithms that should be implemented at each forwarder. Some hash values only apply to end systems, so updating the hash algorithm does not affect forwarders -- they would simply match the buffer that includes the type-length-hash buffer. Some fields, such as the ConObjHash, must be verified at each hop, so a forwarder (or related system) must know the hash algorithm and it could cause backward compatibility problems if the hash type is updated. [CCNMessages] is the authoritative source for per-field allowed hash types in that encoding.

A CCNx name uses binary matching whereas a URI uses a case insensitive hostname. Some systems may also use case insensitive matching of the URI path to a resource. An implication of this is

that human-entered CCNx names will likely have case or non-ASCII symbol mismatches unless one uses a consistent URI normalization to the CCNx name. It also means that an entity that registers a CCNx routable prefix, say `ccnx:/example.com`, would need separate registrations for simple variations like `ccnx:/Example.com`. Unless this is addressed in URI normalization and routing protocol conventions, there could be phishing attacks.

For a more general introduction to ICN-related security concerns and approaches, see [RFC7927] and [RFC7945]

13. References

13.1. Normative References

- [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>>.

13.2. Informative References

- [ace] Shang, W., Yu, Y., Liang, T., Zhang, B., and L. Zhang, "NDN-ACE: Access control for constrained environments over named data networking", NDN Technical Report NDN-0036, 2015, <<http://new.named-data.net/wp-content/uploads/2015/12/ndn-0036-1-ndn-ace.pdf>>.
- [befrags] Mosko, M. and C. Tschudin, "ICN "Begin-End" Hop by Hop Fragmentation", 2017, <<https://www.ietf.org/archive/id/draft-mosko-icnrg-beginendfragment-02.txt>>.
- [ccnlite] Tschudin, C., et al., University of Basel, "CCN-Lite V2", 2011-2018, <<http://www.ccn-lite.net/>>.
- [CCNMessages] Mosko, M., Solis, I., and C. Wood, "CCNx Messages in TLV Format (Internet draft)", 2018, <<https://www.ietf.org/id/draft-irtf-icnrg-ccnxmessages-07.txt>>.
- [ccnxke] Mosko, M., Uzun, E., and C. Wood, "CCNx Key Exchange Protocol Version 1.0", 2017, <<https://www.ietf.org/archive/id/draft-wood-icnrg-ccnxkeyexchange-02.txt>>.
- [CCNxURI] Mosko, M. and C. Wood, "The CCNx URI Scheme (Internet draft)", 2017, <<http://tools.ietf.org/html/draft-mosko-icnrg-ccnxuri-02>>.

- [chunking] Mosko, M., "CCNx Content Object Chunking", 2016, <<https://www.ietf.org/archive/id/draft-mosko-icnrg-ccnxchunking-02.txt>>.
- [cicn] Muscariello, L., et al., Cisco Systems, "Community ICN (CICN)", 2017-2018, <<https://wiki.fd.io/view/Cicn>>.
- [dart] Garcia-Luna-Aceves, J. and M. Mirzazad-Barijough, "A Light-Weight Forwarding Plane for Content-Centric Networks", 2016, <<https://arxiv.org/pdf/1603.06044.pdf>>.
- [EpriseNumbers] IANA, "IANA Private Enterprise Numbers", 2015, <<http://www.iana.org/assignments/enterprise-numbers/enterprise-numbers>>.
- [esic] Mosko, M. and C. Wood, "Encrypted Sessions In CCNx (ESIC)", 2017, <<https://www.ietf.org/id/draft-wood-icnrg-esic-01.txt>>.
- [flic] Tschudin, C. and C. Wood, "File-Like ICN Collection (FLIC)", 2017, <<https://www.ietf.org/archive/id/draft-tschudin-icnrg-flic-03.txt>>.
- [mobile] Mosko, M., Uzun, E., and C. Wood, "Mobile Sessions in Content-Centric Networks", IFIP Networking, 2017, <<http://dl.ifip.org/db/conf/networking/networking2017/1570334964.pdf>>.
- [ndn] UCLA, "Named Data Networking", 2007, <<http://www.named-data.net>>.
- [nnc] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and R. Braynard, "Networking Named Content", 2009, <<http://dx.doi.org/10.1145/1658939.1658941>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.
- [RFC7927] Kutscher, D., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlich, "Information-Centric Networking (ICN) Research Challenges", 2016, <<https://trac.tools.ietf.org/html/rfc7927>>.

- [RFC7945] Pentikousis, K., Ohlman, B., Davies, E., Spirou, S., and G. Boggia, "Information-Centric Networking: Evaluation and Security Considerations", 2016, <<https://trac.tools.ietf.org/html/rfc7945>>.
- [selectors] Mosko, M., "CCNx Selector Based Discovery", 2017, <<https://raw.githubusercontent.com/mmosko/ccnx-protocol-rfc/master/docs/build/draft-mosko-icnrg-selectors-01.txt>>.
- [terminology] Wissingh, B., Wood, C., Afanasyev, A., Zhang, L., Oran, D., and C. Tschudin, "Information-Centric Networking (ICN): CCN and NDN Terminology", 2017, <<https://www.ietf.org/id/draft-irtf-icnrg-terminology-00.txt>>.
- [trust] Tschudin, C., Uzun, E., and C. Wood, "Trust in Information-Centric Networking: From Theory to Practice", 2016, <<https://doi.org/10.1109/ICCCN.2016.7568589>>.

Authors' Addresses

Marc Mosko
PARC, Inc.
Palo Alto, California 94304
USA

Phone: +01 650-812-4405
Email: marc.mosko@parc.com

Ignacio Solis
LinkedIn
Mountain View, California 94043
USA

Email: nsolis@linkedin.com

Christopher A. Wood
University of California Irvine
Irvine, California 92697
USA

Phone: +01 315-806-5939
Email: woodcl@uci.edu

Network Coding Research Group
Internet-Draft
Intended status: Informational
Expires: September 6, 2018

K. Matsuzono
H. Asaeda
NICT
C. Westphal
Huawei
March 5, 2018

Network Coding for Content-Centric Networking / Named Data Networking:
Requirements and Challenges
draft-matsuzono-nwcr-g-nwc-ccn-reqs-01

Abstract

This document describes the current research outcomes regarding Network Coding (NC) for Content-Centric Networking (CCN) / Named Data Networking (NDN), and clarifies the requirements and challenges for applying NC into CCN/NDN.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 6, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
2.1. Definitions	3
2.2. NDN/CCN Background	5
3. Advantage given by NC and CCN/NDN	6
4. Requirements	7
4.1. Content Naming	7
4.2. Transport	8
4.2.1. Scope of Network Coding	9
4.2.2. Consumer Operation	9
4.2.3. Router Operation	10
4.2.4. Publisher Operation	11
4.3. In-network Caching	11
4.4. Seamless Mobility	12
4.5. Security and Privacy	12
5. Challenges	13
5.1. Adopting Convolutional Coding	13
5.2. Rate and Congestion Control	13
5.3. Security and Privacy	14
5.4. Routing Scalability	14
6. Security Considerations	14
7. References	14
7.1. Normative References	14
7.2. Informative References	14
Authors' Addresses	17

1. Introduction

Information-Centric Networks in general, and Content-Centric Networking (CCN) [15] or Named Data Networking (NDN) [16] in particular, have emerged as a novel communication paradigm advocating to retrieve data through their names. This paradigm pushes content awareness into the network layer. It is expected to enable consumers to obtain the content they desire in a straightforward and efficient manner from the heterogeneous networks they may be connected to. The CCN/NDN architecture has introduced innovative ideas and has stimulated research in a variety of areas, such as in-network caching, name-based routing, multi-path transport, content security, and so on. One key benefit of requesting content by name is that it removes the need to establish a session between the client and a specific server, and that content can thereby be retrieved from multiple sources.

In parallel, there has been a growing interest from both academia and industry to better understand fundamental aspects of Network Coding (NC) toward enhancing key system performance metrics such as data throughput, robustness and reduction in the required number of transmissions through connected networks, point-to-multipoint connections, etc. Typically, NC is a technique mainly used to encode packets to recover lost source packets at the receiver, and to effectively get the desired information in a fully distributed manner. In addition, NC can be used for security enhancements [2][3][4][5].

NC aggregates multiple packets with parts of the same content together, and may do this at the source or at other nodes in the network. As such, network coded packets are not connected to a specific server, as they may have evolved within the network. Since NC focuses on what information should be encoded in a network packet, rather than the specific host where it has been generated, it is in line with the CCN/NDN core networking layer (described in more detail later on). NC has already been implemented for information/content dissemination (e.g. [6][7][8]). NC provides CCN/NDN with the highly beneficial potential to effectively disseminate information in a completely independent and decentralized manner. [9] first suggested to exploit NC techniques to enhance key system performances in ICN, and others have considered NC in ICN use cases such as content dissemination [10], seamless mobility [11], joint caching and network coding [12][13], low-latency video streaming [14], etc.

In this document, we consider how NC can be applied to the CCN/NDN architecture and describe the requirements and potential challenges for making CCN/NDN-based communications better using the NC technology. Please note that providing specific solutions (e.g., NC optimization methods) to enhance CCN/NDN performance metrics by exploiting NC is out of scope of this document.

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 [1].

2.1. Definitions

The terminology regarding NC used in this document is described below. It is aligned with RFCs produced by the FEC Framework (FECFRAME) IETF Working Groups as well as recent activities in the Network Coding Research Group [18].

- o Random Linear Coding (RLC): Particular case of Linear Coding using a set of random coding coefficients.
- o Generation, or (IETF) Block: With Block Codes, the set of content data that are logically grouped into a Block, before doing encoding.
- o Generation Size: With Block Codes, the number k of content data belonging to a Block.
- o Encoding Vector: A set of coding coefficients used to generate a certain coded packet through linear coding. The number of nonzero coefficients in the Coding Vector defines its density
- o Finite Field: Finite fields, used in Linear Codes, have the desired property of having all elements (except zero) invertible for $+$ and $*$ and all operations over any elements do not result in an overflow or underflow. Examples of Finite Fields are prime fields $\{0..p^m-1\}$, where p is prime. Most used fields use $p=2$ and are called binary extension fields $\{0..2^m-1\}$, where m often equals 1, 4 or 8 for practical reasons.
- o Finite Field size: The number of elements in a finite field. For example the binary extension field $\{0..2^m-1\}$ has size $q=2^m$.
- o Block Coding: Coding technique where the input Flow(s) must be first segmented into a sequence of blocks, FEC encoding and decoding being performed independently on a per-block basis.
- o Sliding Window Coding or Convolutional Coding: General class of coding techniques that rely on a sliding encoding window. This is an alternative solution to Block Coding.
- o Fixed or Elastic Sliding Window Coding: Coding technique that generates repair data on-the-fly, from the set of source data present in the sliding encoding window at that time, usually by using Linear Coding. The sliding window may be either of fixed size or of variable size over the time (also known as "elastic sliding window").
- o Feedback: Feedback information sent by a decoding node to a node (or from a consumer to a publisher in case of End-to-End Coding). The nature of information contained in a feedback packet varies, depending on the use-case. It can provide reception and/or decoding statistics, or the list of available source packets received or decoded, or the list of lost source packets that should be retransmitted, or a number of additional repair packet needed to have a full rank linear system.

Concerning CCN/NDN, the following terminology and definitions are used.

- o Consumer: A node requesting content. It initiates communication by sending an interest packets.
- o Publisher: A node providing content. It originally creates or owns the content.
- o Forwarding Information Base (FIB): A lookup table in a content router containing the name prefix and corresponding destination interface to forward the interest packets.
- o Pending Interest Table (PIT): A lookup table populated by the interest packets containing the name prefix of the requested data, and the outgoing interface used to forward the received data packets.
- o Content Store (CS): A storage space for a router to cache content objects. It is also known as in-network cache.
- o Content Object: A unit of content data delivered through the CCN/NDN network.
- o Content Flow: A sequence of content objects associated with the unique content name prefix.

2.2. NDN/CCN Background

Armed with the terminology above, we briefly explain the key concepts of CCN/NDN. Both protocols are similar in principle, and different on some implementation choices.

In a CCN network, there are two types of packets at the network level: interest and data. The consumer request a content by sending an "interest" message, that carries the name of the data. On difference to note here in CCN and NDN is that in later versions of CCN, the interest must carry a full name, while in NDN it may carry a name prefix (and receive in return any data with a name matching this prefix).

Once a router receives an "interest" message, it performs a series of look-up: first it checks in the Content Store if it has a copy of the requested content available. If it does, it returns the data and the transaction has successfully completed.

If it does not, it performs a look-up of the PIT to see if there is already an outgoing request for the same data. If there is not, then

it creates an entry in the PIT that lists the name included in the interest, and the interfaces from which it received the interest. This is used later to send the data back, since interest packets do not carry a source field that identifies the requester. If there is already a PIT entry for this name, then it is updated with the incoming interface of this new request and the interest is discarded.

After the PIT look-up, the interest undergoes a FIB lookup to select an outgoing interface. The FIB lists name prefixes and their corresponding forwarding interfaces, to send the interface towards a router that possesses a copy of the requested data.

Once a copy of the data is retrieved, it is send back to the requester(s) using the trail of PIT entries; intermediate node remove the PIT state every time that an interest is satisfied, and may store the data in their content store.

Data packets carry some information to validate the data, in particular that the data is indeed the one that corresponds to the name. This is required since authentication of the object is crucial in CCN/NDN. However, this step is optional at intermediate routers, so as to speed up the processing.

The key aspect of CCN/NDN is that the consumer of the content does not establish a session with a specific server. Indeed, the node that returns the content is not aware of the network location of the requester and the requester is not aware of the network location of the node that provides the content. This in theory allows the interests to follow different paths within a network, or even to be sent over totally different networks.

3. Advantage given by NC and CCN/NDN

Both NC for large scale content dissemination [7] and CCN/NDN can contribute to effective content/information delivery while working jointly. They both bring similar benefits such as throughput/capacity gain and robustness enhancement. The difference between their approaches is that, the former considers content flow as algebraic information to combine [17], while the latter focuses on content/information itself at the networking layer. Because these approaches are complementary, it is natural to combine them. The CCN/NDN core abstraction at networking layer through name makes network stack simple as it enables applications to take maximum advantage of multiple simultaneous connectivities due to its simpler relationship with the layer 2 [15].

CCN/NDN itself, however, cannot provide reliable and robust content dissemination. This requires some specific CCN/NDN transport (i.e.,

strategy layer) [15]. NC can enable the CCN/NDN transport system to effectively distribute and cache data associated with multi-path data retrieval. Furthermore, NC may further enhance CCN/NDN security [23]. In this context, it should be natural that there is much room for considering NC integration into CCN/NDN transport exploiting in-network caching and multi-path transmission [9] and seamless mobility [11] [28].

From the perspective of NC transport mechanism, NC is divided into two major categories: one is coherent NC, and the other is non-coherent NC [30]. In coherent NC, source and destination nodes exactly know network topology and coding operations at intermediate nodes. When multiple consumers are trying to receive the same content such as live video streaming, coherent NC could enable the optimal throughput by making the content flow sent over the constructed optimal multicast trees [24].

However, it requires fully adjustable and specific name-based routing mechanism for CCN/NDN, and an intense computational task for central coordination. In the case of non-coherent NC that often utilizes RLC, they do not need to know network topology and intermediate coding operations. Since non-coherent NC works in a completely independent and decentralized manner, this approach is more feasible especially in the large scale use cases that are intended with CCN/NDN. This document thus focuses on non-coherent NC with RLC.

4. Requirements

This section presents the NC requirements for ICN/CCN in terms of network architecture and protocol. The current document focuses on NC in a block coding manner.

4.1. Content Naming

Naming content objects is as important for CCN/NDN as naming hosts is for today's Internet [19]. Before performing network coding for specified content in CCN/NDN, the overall content should be split into small content objects to avoid packet fragmentation that could cause unnecessary packet processing and degrades throughput. The size of content objects should be within the allowable packet size so as to avoid packet fragmentation in CCN/NDN network, and then network coding should be applied into a set of the content objects.

Each coded packet MAY have a unique name as the original content object has in CCN/NDN, since PIT/FIB/CS operations need a unique name to identify the coded data. As a way of naming coded packet, the encoding vector and the identifier of generation can be used as a part of the content object name [10]. For instance, when the block

size (also called generation size) is k and the encoding vector is $[1,0,0,0]$, the name would be like `/CCN.com/video-A/k/1000`. This naming scheme is simple and can support the delivery of coded packets with exactly the same operations in the FIB/PIT/CS as for original source packets. However, such a naming way requires the consumer to know the naming structure (through a specific name resolution scheme for instance) in order for nodes to specify the exact name of generated coded data packet to retrieve it. From this point of view, it could shift the generation of the encoding vector from the content producer onto the content requester.

If a naming schema such as above is used, it would be valuable to reconsider whether Interest should carry full names (as in CCN) or prefixes (as in NDN) as multiple network coded packets could match a response to a specific prefix for a given generation, such as `/CCN.com/video-A/k`. In the latter case allowing partial name matching, the content requestor may not be able to obtain degrees of freedom. Thus, extensions in the TLV header of the Interest would be used to specify further network coding information so as to limit coded packets to be received (for instance, by specifying the encoded vectors the content requestor receives (also called decoding matrix) as in [9]). However, it may incur a largely increased size of TLV header. Without such coding information, the forwarding node would need to maintain some records regarding interest packets sent before, in order to provide new degrees of freedom.

Coded packet MAY have a name that indicates that it is a coded packet, and move the coding information into a metadata field in the payload (i.e., the name includes only data type, original or coded packet, etc). This however would preclude network coding on packets without prior decoding them (for instance, in the CS of forwarding nodes). It would not be beneficial for applications or services that may not need to understand the packet payload. Due to the possibility that multiple coded packets may have a same name, as described above, some mechanism needs for the content requestor to obtain innovative coded packets. It would also require some mechanism to insert the multiple innovative packets into the CS. If the coding information of coded packet are encrypted together with the payload (for instance, at source coding), the content requestor or forwarding nodes would incur extra computational overhead for decryption of the packet to interpret the coding information.

4.2. Transport

The pull-based request-response feature of CCN/NDN is the fundamental principle of its transport layer; one Interest retrieves at most one Data packet. It is important to not violate this rule, as it would

open denial of service attacks issues, and thus the following basic operation should be considered to apply NC to CCN/NDN.

4.2.1. Scope of Network Coding

It should be discussed whether the network can update data packets that are being received in transit, or if only the data that matches an interest can be subject to network coding operations. In the latter case, the network coding is performed on an end-to-end basis (where one end is the consumer, and the other end is any node that is able to respond to the Interest). In the former case, NC happens anywhere in the network that is able to update the data. As CCN/NDN has mechanisms in place to ensure the integrity of the data during transfer, NC in the network introduce complexities that would require special consideration for the integrity mechanisms to still work.

Similarly, caching of network coded packets at intermediate node may be valuable, but may prevent the node caching the coded content to validate the content.

4.2.2. Consumer Operation

To attain NC benefits associated with in-network caching, consumers need to issue interests directing the router (or publisher) to forward innovative coded packets if available. The reason why this directive is needed is that delay-sensitive applications such as live-video streaming may want to sequentially get original packets rather than coded packets cached in routers due to real-time constraint. Issuing such an interest is possible by using optional TLV (Type Length Value) header contained in Interest TLV packet format which allows network elements to add or modify information on the fly. Consumer can put an instruction into it, and for instance, if routers detect that it is better for consumer to get coded packets rather than original packets, routers can modify it to do so. After receiving interests having the instruction in optional header, the router with useful coded packets forward them.

As another solution, consumer issues interests specifying unique names for each coded packets. In this case, a unified naming scheme considering both original and coded packets is required. Moreover, in the case of NC end-to-end approach, publishers need to get feedback from the corresponding receivers to adjust some coding parameters. To deal with this, a receiver may have to request a specific interest name to reach the corresponding publisher and put required information into the optional header.

4.2.3. Router Operation

Routers need to appropriately handle PIT entries to accommodate interests for coded packets as well as original packets. Moreover, in order to decode as necessary, nodes need to know the coding vector used for each coded packet (note: since all the data for a specific content may not come through the same path/network, intermediate nodes may never be able to decode). In a typical case, the coding vector used for each coded packet is attached to the header of coded data. In regard to this point, the generation size (also called block size) for NC should be set to a reasonable value so that the total coded packet size including header needed for expressing the coding vector information and data message fits into the allowable packet size. It may be useful to use compression techniques for coding vectors [20][21].

Router may try to forward useful independent coded packets toward downstream nodes in order to respond to received interests for coded packets. Routers thus need to determine whether or not they can generate useful coded packets for consumers. Assuming that the size of the Finite Field in use is not relatively small, re-encoding using enough cached packets has a strong probability of making independent coded packets [24]. If router does not have enough cached packets to newly produce independent coded packets, it relays received interests to upstream nodes to receive a new original or independent coded packet and pass it to downstream nodes. In another possible case, when receiving interests for only original packets, routers may try to decode and get all the original packets and store them (if there are fully available cache capacity), enabling faster response to the interests. Since there is a tradeoff between NC encoding/decoding calculation cost and cache capacity, and the usage efficacy of re-encoding or decoding at router, router should need to determine how to response to receiving interests according to the use case (e.g., delay-sensitive or delay-tolerant application) and the router situation such as available cache space and computational capability.

Some proposed schemes [10] require that the router maintain a tally of the interests for a specific name and generation, so as to know how many degrees of freedom have been provided already for the NC packets. Scalability and practicality of maintaining such scheme at intermediate routers should be considered.

To enable fast loss recovery cooperating with in-network caching, a transport mechanism of in-network loss detection and recovery [28][14] at router as well as consumer-driven mechanism should be considered.

4.2.4. Publisher Operation

The procedure for splitting an overall content into small content objects is responsible for the original publisher. When applying NC for the content, the publisher performs NC over the content objects, and naming processing for the coded packets. If the producer takes the lead in determining the used encoding vectors and generating the coded packets, there are the two possible end-to-end cases; 1) content requestors obtain the names of coded packets through a certain mechanism, and send the correspond interests toward the publisher to get the coded packets already generated at the publisher, and 2) the publisher determines the encoding vectors after receiving interests specifying them. In the former case, although content requestors cannot flexibly specify an encoding vector for generating the coded packet to retain, but the latency for getting the coded data can be reduced compared to the latter case where additional NC operations need after receiving interests. According to application requirement for latency, such NC operation strategy should be considered.

4.3. In-network Caching

Caching is an essential technique to improve throughput and latency in various applications. In-network caching CCN/NDN essentially supports at network level is highly beneficial by exploiting NC to enable effective multicast transmission [29], multipath data retrieval [10] [11], fast loss recovery [14], and so on. However, there are several issues to be considered.

As a general issue, there are limitations of cache capacity, and caching policy affects on consumer's performances [22] [25] [26]. It is thus highly significant for routers to determine which packets should be cached and discarded. Since delay-sensitive applications often do not require in-network cache for a long period due to their real-time constraints, routers have to know the necessity for caching received packets to save the caching volume. This could be possible by putting a flag into optional header of data packets at publisher side. When receiving data packets with the flag meaning no necessity for cache, routers just have to forward them to downstream nodes. On the other hand, when receiving original packets or coded packets without the flag, router may cache them based on a specified replacement policy.

One key aspect of in-network caching is whether or not intermediate nodes can cache NC packets without first decoding them. If in-network caches store coded packets, they need to be able to validate that the packets are not compromised, so as to avoid cache pollution attacks. Without having all the packets in a generation, the cache

cannot decode the packets to check if it is authenticated. Caching of coded packets would require some mechanism to validate coded packets. In addition, when coded packets have a same name, it would also require some mechanism to identify them.

4.4. Seamless Mobility

This subsection presents how NC can achieve seamless mobility [11] [28] and clarify the requirements. A key feature of CCN/NDN is that it is sessionless and that multiple interests can be sent to different copies of the content in parallel. CCN/NDN enables a consumer to retrieve the content from multiple sources that are distributed and asynchronous.

In this context, network coding provide a mechanism to ensure that the Interests sent to multiple copies of the content retrieve innovative packets, even in the case of packet losses on some of the paths/networks to these copies. NC adds a reliability layer to CCN in a distributed and asynchronous manner. One key benefit is that the link between the consumer and the multiple copies acts as a virtual logical link, upon which rate adaptation mechanism can be performed.

This naturally applies to mobility event, where the consumer may connect between multiple access points before a mobility event (make-before-break handoff). In such mobility event, the consumer is connected first to the previous access point, then to both the previous and next access points, then finally only to the next access points. With CCN, the consumer only sends interests on the available interfaces. Requesting network coded packets ensures that during the phase where it is connected to the previous and the next APs at the same time, it does not receive duplicate data, but does not miss on any content either. By combining NC with CCN, the consumer receives additional degrees of freedom with any innovative packet it receives on either interface.

Further discussion is [TBD].

4.5. Security and Privacy

This subsection describes the requirement for security and privacy provided by NC in CCN/NDN, such as data integrity especially when intermediate nodes perform re-encoding, as in the case of hash restrictions for original data packets, and so on.

Network coding impacts the security mechanisms of CCN/NDN. In particular, CCN/NDN is designed to prevent modification of the Data packets. Because Data packets for a specific name can be self-

authenticated, they can be validated on the delivery path, and can also be cached at untrusted intermediate nodes. Network coding may bring up issues if intermediate nodes are allowed to modify packets by performing additional network coding operations. Intermediate nodes may also be caching network coded packets without having the ability to perform validation of the content and therefore open themselves to cache pollution attacks.

In CCN/NDN, content objects can be encrypted to support access control or privacy. If the coding information of coded packet is included in the encrypted data payload, extra computational overhead occurs.

5. Challenges

This section presents several primary challenges and research items to be considered when applying NC into CCN/NDN.

5.1. Adopting Convolutional Coding

Several block coding approaches have been proposed so far, but there is still no sufficient discussion and application of convolutional coding approach (e.g., sliding or elastic window coding) in CCN/NDN. Convolutional coding is often appropriate to situations where a fully or partially reliable delivery of continuous data flows is needed, especially when these data flows feature realtime constraints. As in [31] on an end-to-end basis, it would be advantageous for continuous content flow to adopt sliding window coding in CCN/NDN. In this case, the publisher needs to appropriately set coding parameters and let content requestor know the information, and content requestor needs to send interest (i.e., feedback information) about the data reception status. Since CCN/NDN advocates hop-by-hop communication, it would be worth discussing and investigating how convolutional coding can be applied in a hop-by-hop fashion and the benefits. In particular, assuming that NC could occur at intermediate nodes with some useful data packets stored in the CS as described in the previous section, both the encoding window and CS management would be required, and the feasibility and practicality should be considered.

5.2. Rate and Congestion Control

Adding redundancy using coded packets may cause further network congestion and adversely affect overall throughput performance. In particular, in a situation where fair bandwidth sharing is more desirable, each streaming flow must adapt to the network conditions to fairly consume the available link bandwidth. It is thus indispensable that each content flow cooperatively implements congestion control to adjust the consumed bandwidth to stabilize the

network condition (i.e., to achieve low packet loss rate, delay, and jitter).

5.3. Security and Privacy

A variety of security and privacy concerns would exist in NC and CCN/NDN. This subsection focuses on the description of security and privacy challenges related to NC for CCN/NDN. [TBD]

5.4. Routing Scalability

This subsection focuses on the challenges of routing mechanisms such as scalability and protocol overhead, and so on.

6. Security Considerations

This document does not impact the security of the Internet. Security considerations related to NC for CCN/NDN are described in the previous Section.

7. References

7.1. Normative References

- [1] 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>>.

7.2. Informative References

- [2] Cai, N. and R. Yeung, "Secure network coding", Proc. International Symposium on Information Theory (ISIT), IEEE, June 2002.
- [3] Lima, L., Gheorghiu, S., Barros, J., Mdard, M., and A. Toledo, "Secure Network Coding for Multi-Resolution Wireless Video Streaming", IEEE Journal of Selected Area (JSAC), vol. 28, no. 3, April 2002.
- [4] Gkantsidis, C. and P. Rodriguez, "Cooperative Security for Network Coding File Distribution", Proc. Infocom, IEEE, April 2006.
- [5] Vilea, J., Lima, L., and J. Barros, "Lightweight security for network coding", Proc. ICC, IEEE, May 2008.

- [6] Dimarkis, A., Godfrey, P., Wu, Y., Wainwright, M., and K. Ramchandran, "Network Coding for Distributed Storage Systems", IEEE Trans. Information Theory, vol. 56, no.9, September 2010.
- [7] Gkantsidis, C. and P. Rodriguez, "Network coding for large scale content distribution", Proc. Infocom, IEEE, March 2005.
- [8] Seferoglu, H. and A. Markopoulou, "Opportunistic Network Coding for Video Streaming over Wireless", Proc. Packet Video Workshop (PV), IEEE, November 2007.
- [9] Montpetit, M., Westphal, C., and D. Trossen, "Network Coding Meets Information-Centric Networking: An Architectural Case for Information Dispersion Through Native Network Coding", Proc. Workshop on Emerging Name-Oriented Mobile Networking Design (NoM), ACM, June 2012.
- [10] Saltarin, J., Bourtsoulatzé, E., Thomos, N., and T. Braun, "NetCodCCN: a network coding approach for content-centric networks", Proc. Infocom, IEEE, April 2016.
- [11] Ramakrishnan, A., Westphal, C., and J. Saltarin, "Adaptive Video Streaming over CCN with Network Coding for Seamless Mobility", Proc. International Symposium on Multimedia (ISM), IEEE, December 2016.
- [12] Wang, J., Ren, J., Lu, K., Wang, J., Liu, S., and C. Westphal, "An Optimal Cache Management Framework for Information-Centric Networks with Network Coding", Proc. Networking Conference, IFIP/IEEE, June 2014.
- [13] Wang, J., Ren, J., Lu, K., Wang, J., Liu, S., and C. Westphal, "A Minimum Cost Cache Management Framework for Information-Centric Networks with Network Coding", Computer Networks, Elsevier, August 2016.
- [14] Matsuzono, K., Asaeda, H., and T. Turetti, "Low Latency Low Loss Streaming using In-Network Coding and Caching", Proc. Infocom, IEEE, May 2017.
- [15] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and R. Braynard, "Networking Named Content", Proc. CoNEXT, ACM, December 2009.

- [16] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., Claffy, K., Crowley, P., Papadopoulos, C., Wang, L., and B. Zhang, "Named data networking", ACM Comput. Commun. Rev., vol. 44, no. 3, July 2014.
- [17] Koetter, R. and M. Medard, "An Algebraic Approach to Network Coding", IEEE/ACM Trans. on Networking, vol. 11, no 5, Oct. 2003.
- [18] Adamson, B., Adjih, C., Bilbao, J., Firoiu, V., Fitzek, F., Lochin, E., Masucci, A., Montpetit, M., Pedersen, M., Peralta, G., Roca, V., Saxena, P., and S. Sivakumar, "Network Coding Taxonomy", draft-irtf-nwcrd-network-coding-taxonomy-05 (work in progress), September 2017.
- [19] Kutscher, et al., D., "Information-Centric Networking (ICN) Research Challenges", RFC 7927, July 2016.
- [20] Thomos, N. and P. Frossard, "Toward one Symbol Network Coding Vectors", IEEE Communications letters, vol. 16, no. 11, November 2012.
- [21] Lucani, D., Pedersen, M., Heide, J., and F. Fitzek, "Fulcrum Network Codes: A Code for Fluid Allocation of Complexity", available at <http://arxiv.org/abs/1404.6620>, April 2014.
- [22] Perino, D. and M. Varvello, "A reality check for content centric networking", Proc. SIGCOMM Workshop on Information-centric networking (ICN'11), ACM, August 2011.
- [23] Wu, Q., Li, Z., Tyson, G., Uhlig, S., Kaafar, M., and G. Xie, "Privacy-Aware Multipath Video Caching for Content-Centric Networks", IEEE Journal of Selected Area (JSAC) vol. 38, no. 8, June 2016.
- [24] Wu, Y., Chou, P., and K. Jain, "A comparison of network coding and tree packing", Proc. ISIT, IEEE, June 2004.
- [25] Podlipnig, S. and L. Osz, "A Survey of Web Cache Replacement Strategies", Proc. ACM Computing Surveys vol. 35, no. 4, December 2003.
- [26] Rossini, G. and D. Rossi, "Evaluating CCN multi-path interest forwarding strategies", Elsevier Computer Communication, vol.36, no. 7, April 2013.

- [27] Chai, W., He, D., Psaras, I., and G. Pavlou, "Cache Less for More in Information-centric Networks", Journal Computer Communications, vol. 37. no. 7, April 2013.
- [28] 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", Proc. ICN ACM, September 2016.
- [29] Ali, M. and U. Niesen, "Coding for Caching: Fundamental Limits and Practical Challenges", IEEE Communications Magazine vol. 54, no. 8, August 2016.
- [30] Koetter, R. and F. Kschischang, "An algebraic approach to network coding", IEEE Trans. Netw. vol.11, no.5, October 2008.
- [31] Tournoux, P., Lochin, E., Lacan, J., Bouabdallah, A., and V. Roca, "On-the-Fly Erasure Coding for Real-Time Video Applications", IEEE Trans. Multimed. vol.13, no.4, August 2011.

Authors' Addresses

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

Email: matsuzono@nict.go.jp

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

Email: asaeda@nict.go.jp

Cedric Westphal
Huawei
2330 Central Expressway
Santa Clara, California 95050
USA

Email: cedric.westphal@huawei.com

ICNRG
Internet-Draft
Intended status: Informational
Expires: July 19, 2018

A. Rahman
D. Trossen
InterDigital Inc.
D. Kutscher
R. Ravindran
Huawei
January 15, 2018

Deployment Considerations for Information-Centric Networking (ICN)
draft-rahman-icnrg-deployment-guidelines-05

Abstract

Information-Centric Networking (ICN) is now reaching technological maturity after many years of fundamental research and experimentation. This document provides a number of deployment considerations in the interest of helping the ICN community move forward to the next step of live deployments. First, the major deployment configurations for ICN are described including the key overlay and underlay approaches. Then proposed deployment migration paths are outlined to address major practical issues such as network and application migration. Next, selected ICN trial experiences are summarized. Finally, protocol areas that require further standardization are identified to facilitate future interoperable ICN deployments.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 19, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Deployment Configurations	4
3.1. Clean-slate ICN	4
3.2. ICN-as-an-Overlay	5
3.3. ICN-as-an-Underlay	5
3.3.1. Edge Network	6
3.3.2. Core Network	6
3.4. ICN-as-a-Slice	7
4. Deployment Migration Paths	8
4.1. Application and Service Migration	8
4.2. Content Delivery Network Migration	9
4.3. Edge Network Migration	9
4.4. Core Network Migration	10
5. Deployment Trial Experiences	10
5.1. ICN-as-an-Overlay	10
5.1.1. FP7 PURSUIT Efforts	10
5.1.2. FP7 SAIL Trial	11
5.1.3. NDN Testbed	11
5.1.4. ICN2020 Efforts	11
5.2. ICN-as-an-Underlay	12
5.2.1. H2020 POINT and RIFE Efforts	12
5.2.2. H2020 FLAME Efforts	12
5.2.3. CableLabs Content Delivery System	13
5.2.4. NDN IoT Trials	13
5.3. Other Configurations	13
5.3.1. Hybrid ICN Trials	14
5.4. Summary of Deployment Trials	14
6. Deployment Issues Requiring Further Standardization	14
6.1. Protocols for Application and Service Migration	15
6.2. Protocols for Content Delivery Network Migration	15

6.3. Protocols for Edge and Core Network Migration	15
6.4. Summary of ICN Protocol Gaps and Potential Protocol Efforts	16
7. Conclusion	17
8. IANA Considerations	18
9. Security Considerations	18
10. Acknowledgments	19
11. Informative References	19
Appendix A. Change Log	25
Authors' Addresses	25

1. Introduction

The ICNRG charter identifies deployment guidelines as an important topic area for the ICN community. Specifically, the charter states that defining concrete migration paths for ICN deployments which avoid forklift upgrades, and defining practical ICN interworking configurations with the existing Internet paradigm, are key topic areas that require further investigation [ICNRGCharter]. Also, it is well understood that results and conclusions from any mid to large-scale ICN experiments in the live Internet will also provide useful guidance for deployments.

However, so far outside of some preliminary investigations such as [I-D.paik-icn-deployment-considerations], there has not been much progress on this topic. This document attempts to fill some of these gaps by defining clear deployment configurations for ICN, and associated migration pathways for these configurations. Also, selected deployment trial experiences of ICN technology are summarized. Finally, recommendations are made for potential future IETF standardization of key protocol functionality that will facilitate interoperable ICN deployments going forward.

2. Terminology

This document assumes readers are, in general, familiar with the terms and concepts that are defined in [RFC7927] and [I-D.irtf-icnrg-terminology]. In addition, this document defines the following terminology:

Deployment - In the context of this document, deployment refers to the final stage of the process of setting up an ICN network that is (1) ready for useful work (e.g. transmission of end user video and text) in a live environment, and (2) integrated and interoperable with the Internet. We consider the Internet in its widest sense where it encompasses various access networks (e.g. WiFi, Mobile radio network), service edge networks (e.g. for edge computing), transport networks, Content Distribution Networks

(CDNs), core networks (e.g. Mobile core network), and back-end processing networks (e.g. Data Centres). However, through out the document we typically limit the discussion to edge networks, core networks and CDNs for simplicity.

Information-Centric Networking (ICN) - A data-centric network architecture where accessing data by name is the essential network primitive. See [I-D.irtf-icnrg-terminology] for further information.

Network Function Virtualization (NFV): A networking approach where network functions (e.g. firewalls, load balancers) are modularized as software logic that can run on general purpose hardware, and thus are specifically decoupled from the previous generation of proprietary and dedicated hardware. See [I-D.irtf-nfvrg-gaps-network-virtualization] for further information.

Software-Defined Networking (SDN) - A networking approach where the control and data plane for switches are separated, allowing for realizing capabilities such as traffic isolation and programmable forwarding actions. See [RFC7426] for further information.

3. Deployment Configurations

In this section, we present various deployment options for ICN. These are presented as "configurations" that allow for studying these options further. While this document will outline experiences with various of these configurations (in Section 5), we will not provide an in-depth technical or commercial evaluation for any of them - for this we refer to existing literature in this space such as [Tateson].

3.1. Clean-slate ICN

ICN has often been described as a "clean-slate" approach with the goal to renew or replace the complete IP infrastructure of the Internet (e.g., existing applications which are typically tied directly to the TCP/IP protocol stack, IP routers, etc.). As such, existing routing hardware as well as ancillary services are not taken for granted. For instance, a Clean-slate ICN deployment would see existing IP routers being replaced by ICN-specific forwarding and routing elements, such as NFD (Named Data Networking Forwarding Daemon) [NFD], CCN routers [Jacobson] or PURSUIT forwarding nodes [IEEE_Communications].

While such clean-slate replacement could be seen as exclusive for ICN deployments, some ICN approaches (e.g., [POINT]) also rely on the

deployment of general infrastructure upgrades, here SDN switches. Such SDN infrastructure upgrades, while being possibly utilized for a Clean-slate ICN deployment would not necessarily be used exclusively for such deployments. Different proposals have been made for various ICN approaches to enable the operation over an SDN transport [Reed][CONET][C_FLOW].

3.2. ICN-as-an-Overlay

Similar to other significant changes to the Internet routing fabric, particularly the transition from IPv4 to IPv6 or the introduction of IP multicast, this deployment configuration foresees the creation of an ICN overlay. Note that this overlay approach is sometimes, informally, also referred to as a tunneling approach. The overlay approach can be implemented directly such as ICN-over-UDP as described in [CCNx_UDP]. Alternatively, the overlay can be accomplished via ICN-in-L2-in-IP as in [IEEE_Communications] which describes a recursive layering process. Another approach used in the Network of Information (NetInf) is to define a convergence layer to map NetInf semantics to HTTP [I-D.kutscher-icnrg-netinf-proto]. Finally, [Overlay_ICN] describes an incremental approach to deploying an ICN architecture based on segregating ICN user and control plane traffic which is particularly well-suited to being overlaid on SDN based networks.

Regardless of the flavor, however, the overlay approach results in islands of ICN deployments over existing IP-based infrastructure. Furthermore, these ICN islands are typically connected to each other via ICN/IP tunnels. In certain scenarios this requires interoperability between existing IP routing protocols (e.g. OSPF, RIP, ISIS) and ICN based ones. ICN-as-an-Overlay can be deployed over IP infrastructure in either edge or core networks. This overlay approach is thus very attractive for ICN experimentation and testing as it allows rapid and easy deployment of ICN over existing IP networks.

3.3. ICN-as-an-Underlay

Proposals such as [POINT] and [White] outline the deployment option of using an ICN underlay that would integrate with existing (external) IP-based networks by deploying application layer gateways at appropriate locations. The main reasons for such a configuration option is the introduction of ICN technology in given islands (e.g., inside a CDN or edge IoT network) to reap the benefits of native ICN in terms of underlying multicast delivery, mobility support, fast indirection due to location independence, in-network computing and possibly more. The underlay approach thus results in islands of native ICN deployments which are connected to the rest of the

Internet through protocol conversion gateways or proxies. Routing domains are strictly separated. Outside of the ICN island, normal IP routing protocols apply. Within the ICN island, ICN based routing schemes apply. The gateways transfer the semantic content of the messages (i.e., IP packet payload) between the two routing domains.

3.3.1. Edge Network

Native ICN networks may be located at the edge of the network which allows the possibility of introducing new network architectures and protocols, and in this context ICN is an attractive option for newer deployments such as IoT [I-D.irtf-icnrg-icniot]. The integration with the current IP protocol suite takes place at an application gateway/proxy at the edge network boundary, e.g., translating incoming CoAP request/response transactions [RFC7252] into ICN message exchanges or vice versa. Furthermore, ICN will allow enhancement of the role of gateways/proxies as ICN message security should be preserved through the protocol translation function of a gateway/proxy and thus offer a substantial gain.

The work in [VSER] positions ICN as an edge service gateway driven by a generalized ICN based service orchestration system with its own compute and network virtualization controllers to manage an ICN infrastructure. The platform also offers service discovery capabilities to enable user applications to discover appropriate ICN service gateways. To exemplify a use case scenario, the [VSER] platform shows the realization of a multi-party audio/video conferencing service over such a edge cloud deployment of ICN routers realized over commodity hardware platforms. This platform has also been extended to offer seamless mobility and mobility as a service [VSER-Mob] features.

3.3.2. Core Network

In this sub-option, a core network would utilize edge-based protocol mapping onto the native ICN underlay. For instance, [POINT] proposes to map HTTP transactions, or some other IP based transactions such as CoAP, directly onto an ICN-based message exchange. This mapping is realized at the network attachment point, such as realized in access points or customer premise equipment, which in turn provides a standard IP interface to existing user devices. Towards peering networks, such network attachment point turns into a modified border gateway/proxy, preserving the perception of an IP-based core network towards any peering network.

The work in [White] proposes a similar deployment configuration. Here, the target is the use of ICN for content distribution within CDN server farms, i.e., the protocol mapping is realized at the

ingress of the server farm where the HTTP-based retrieval request is served, while the response is delivered through a suitable egress node translation.

3.4. ICN-as-a-Slice

The objective of Network slicing [NGMN] is to multiplex a general pool of compute, storage and bandwidth resources among multiple services with exclusive SLA requirements on transport level QoS and security. From a 5G perspective, this also includes slicing the air interface spectrum resources among different applications. These services could include both connectivity services like LTE-as-a-service or OTT services like VoD or other IoT services through composition of a group of virtual and/or physical network functions. Such a framework can also be used to realize ICN slices with its own control, service and forwarding plane over which one or more end-user services can be delivered.

5G next generation architecture [fiveG-23501] provides the flexibility to deploy the ICN-as-a-Slice over either the edge (RAN) or Mobile core network, or the ICN-as-a-Slice may be deployed end-to-end. Further discussions on extending the architecture presented in [fiveG-23501] and the corresponding procedures in [fiveG-23502] to support ICN has been provided in [I-D.ravi-icnrg-5gc-icn]. Such a generalized network slicing framework should be able to offer service slices to be realized using both IP and ICN. Network slicing will rely heavily on network softwarization and programmability using SDN/NFV technologies for efficient utilization of available resources without compromising on the slice requirements. Coupled with the view of ICN functions as being "chained service functions" [RFC7665], an ICN deployment within such a slice could also be realized within the emerging orchestration plane that is targeted for adoption in future (e.g., 5G Mobile) network deployments. Finally, it should be noted that ICN is not creating the network slice, but instead that the slice is created to run an 5G-ICN instance [Ravindran].

At the level of the specific technologies involved, such as ONAP [ONAP] that can be used to orchestrate slices, the 5G-ICN slice requires compatibility for instance at the level of the forwarding/data plane depending on if it is realized as an overlay or using programmable data planes. With SDN emerging for new network deployments, some ICN approaches will need to integrate with SDN as a data plane forwarding function, as briefly discussed in Section 3.1. Further cross domain ICN slices can also be realized using frameworks such as [ONAP].

4. Deployment Migration Paths

After outlining the various ICN deployment configurations in Section 3, we now focus on the various migration paths that will have importance to the various stakeholders that are usually involved in the deployment of a technology at (ultimately) large scale. We can identify these stakeholders as:

- o Application providers
- o ISPs and service providers, both as core as well as access network providers, and also ICN network providers
- o CDN providers (due to the strong relation of the ICN proposition to content delivery)
- o End device manufacturers and users

Note that our presentation purely focuses on technological aspects of such migration. Economic or regulatory aspects, such as studied in [Tateson], [Techno_Economic] and [Internet_Pricing] are left out of our discussion.

4.1. Application and Service Migration

The internet is full of applications and services, utilizing the innovation capabilities of the many protocols defined over the packet level IP service. HTTP provides one convergence point for these services with many web development frameworks based on the semantics provided by the hypertext transfer protocol. In recent years, even services such as video delivery have been migrating from the traditional RTP-over-UDP delivery to the various HTTP-level streaming solutions, such as DASH [DASH] and others. Nonetheless, many non-HTTP services exist, all of which need consideration when migrating from the IP-based internet to an ICN-based one.

The underlay deployment configuration options presented in Section 3.3.2 and Section 3.3.1 aim at providing some level of backward compatibility to this existing ecosystem through a proxy based message flow mapping mechanism (e.g., mapping of existing HTTP/TCP/IP message flows to HTTP/TCP/IP/ICN message flows). A related approach of mapping TCP/IP to TCP/ICN message flows is described in [Moiseenko]

Alternatively, ICN as an overlay (Section 3.2), as well as ICN-as-a-Slice (Section 3.4), allow for the introduction of the full capabilities of ICN through new application/service interfaces as well as operations in the network. With that, these approaches of

deployment are likely to aim at introducing new application/services capitalizing on those ICN capabilities.

Finally, [I-D.suthar-icnrg-icn-lte-4g] outlines a dual-stack end user device approach that is applicable for all deployment configurations. Specifically, [I-D.suthar-icnrg-icn-lte-4g] introduces middleware layers (called the Transport Convergence Layer, TCL) in the device that will dynamically adapt existing applications to either an underlying ICN protocol stack or standard IP protocol stack. This involves end device signalling with the network to determine which protocol stack instance and associated middleware adaptation layers to utilize for a given application transaction.

4.2. Content Delivery Network Migration

A significant number of services and applications are devoted to content delivery in some form, either as video delivery services, social media platforms, and many others. Content delivery networks (CDNs) are deployed to assist these services through localizing the content requests and therefore reducing latency and possibly increase utilization of available bandwidth as well as reducing the load on origin servers. Similar to the previous sub-section, the underlay deployment configurations presented in Section 3.3.2 and Section 3.3.1 aim at providing a migration path for existing CDNs. This is also highlighted in the BIER WG use case document [I-D.ietf-bier-use-cases], specifically with potential benefits in terms of utilizing multicast in the delivery of content but also reducing load on origin as well as delegation server. We return to this benefit in the trial experiences in Section 5.

4.3. Edge Network Migration

Edge networks often see the deployment of novel network level technology, e.g., in the space of IoT. Such IoT deployments have for many years relied, and often still do, on proprietary protocols for reasons such as increased efficiency, lack of standardization incentives and others. Utilizing the underlay deployment configuration in Section 3.3.1, application gateways/proxies can integrate such edge deployments into IP-based services, e.g., utilizing CoAP [RFC7252] based machine-to-machine (M2M) platforms such as oneM2M [oneM2M] or others.

Another area of increased edge network innovation is that of Mobile (access) networks, particularly in the context of the 5G Mobile networks. With the proliferation of network softwarization (using technologies like service orchestration frameworks leveraging NFV and SDN concepts) access networks and other network segments, the ICN-as-a-Slice deployment configuration in Section 3.4 provides a suitable

migration path for integration non-IP-based edge networks into the overall system through virtue of realizing the relevant (ICN) protocols in an access network slice.

4.4. Core Network Migration

Migrating core networks (e.g., of the Internet or Mobile core network) requires not only significant infrastructure renewal but also the fulfillment of the significant performance requirements, particularly in terms of throughput. For those parts of the core network that would see a migration to an SDN-based optical transport the ICN-as-a-Slice deployment configuration in Section 3.4 could see the introduction of native ICN solutions within slices provided by the SDN-enabled transport network or as virtual network functions, allowing for isolating the ICN traffic while addressing the specific ICN performance benefits and constraints within such isolated slice. For ICN solutions that natively work on top of SDN, the underlay deployment configuration in Section 3.3.2 provides an additional migration path, preserving the IP-based services and applications at the edge of the network, while realizing the core network routing through an ICN solution (possibly itself realized in a slice of the SDN transport network).

5. Deployment Trial Experiences

In this section, we will outline trial experiences, often conducted within international collaborative project efforts. Our focus here is on the realization of the various deployment configurations in Section 3, and we therefore categorize the trial experiences according to these deployment configurations. While a large body of work exists at the simulation or emulation level, we specifically exclude these studies from our presentation to retain the focus on real life experiences.

5.1. ICN-as-an-Overlay

5.1.1. FP7 PURSUIT Efforts

Although the FP7 PURSUIT [IEEE_Communications] efforts were generally positioned as a Clean-slate ICN replacement of IP (Section 3.1), the project realized its experimental test bed as an L2 VPN-based overlay between several European, US as well as Asian sites, i.e., following the overlay deployment configuration presented in Section 3.2. Software-based forwarders were utilized for the ICN message exchange, while native ICN applications, e.g., for video transmissions, were showcased. At the height of the project efforts, about 70+ nodes were active in the (overlay) network with presentations given at several conferences as well as to the ICNRG.

5.1.2. FP7 SAIL Trial

The Network of Information (NetInf) is the approach to Information-Centric Networking developed by the European Union (EU) FP7 SAIL project (<http://www.sail-project.eu/>). NetInf provides both name-based forwarding with CCNx-like semantics and name resolution (for indirection and late-binding). The NetInf architecture supports different deployment options through its convergence layer abstraction. In its first prototypes and trials, NetInf was deployed mostly in an HTTP embedding and in a UDP overlay following the overlay deployment configuration in Section 3.2. Reference [SAIL_NetInf] describes several trials including a stadium environment large crowd scenario and a multi-site testbed, leveraging NetInf's Routing Hint approach for routing scalability.

5.1.3. NDN Testbed

The Named Data Networking (NDN) is one of the research projects funded by the National Science Foundation (NSF) of the USA as part of the Future Internet Architecture Program. The original NDN proposal was positioned as a Clean-slate ICN replacement of IP (Section 3.1). However, in several trials, NDN generally follows the overlay deployment configuration of Section 3.2 to connect institutions over the public Internet across several continents. The use cases covered in the trials include real-time video-conferencing, geo-locating, and interfacing to consumer applications. Typical trials involve up to 100 NDN enabled nodes (<https://named-data.net/ndn-testbed/>) [Jangam].

5.1.4. ICN2020 Efforts

ICN2020 is an ICN related research project funded by the EU and Japan as part of the H2020 research and innovation program and NICT (<http://www.icn2020.org/>). ICN2020 has a specific focus to advance ICN towards real-world deployments through innovative applications and global scale experimentation. Both NDN and CCN approaches are within the scope of the project.

ICN2020 was kicked off in July 2016 and at the end of the first year released a set of public technical reports [ICN2020]. The report titled "Deliverable D4.1: 1st yearly report on Testbed and Experiments (WP4)" contains a detailed description of the progress made in both local testbeds as well as federated testbeds. The plan for the federated testbed includes integrating the NDN testbed, the CUTEi testbed [RFC7945] [CUTEi] and the GEANT testbed (<https://www.geant.org/>) to create an overlay deployment configuration of Section 3.2 over the public Internet.

5.2. ICN-as-an-Underlay

5.2.1. H2020 POINT and RIFE Efforts

POINT and RIFE are two more ICN related research projects funded by the EU as part of the H2020 effort. The efforts in the H2020 POINT+RIFE projects follow the underlay deployment configuration in Section 3.3.2, although this is mixed with utilizing an overlay deployment to provide multi-national connectivity. However, underlay SDN-based deployments do exist at various project partner sites, e.g., at Essex University, without any overlaying being realized. Edge-based network attachment points (NAPs) provide the IP/HTTP-level protocol mapping onto ICN protocol exchanges, while the SDN underlay (or the VPN-based L2 underlay) is used as a transport network.

The multicast as well as service endpoint surrogate benefits in HTTP-based scenarios, such as for HTTP-level streaming video delivery, have been demonstrated in the deployed POINT test bed with 80+ nodes being utilized. Demonstrations of this capability have been given to the ICNRG in 2016, and public demonstrations were also provided at events such as Mobile World Congress in 2016 [MWC_Demo]. The trial has also been accepted by the ETSI MEC group as a proof-of-concept with a demonstration at the ETSI MEC World Congress in 2016.

While the afore-mentioned demonstrations all use the overlay deployment, H2020 also has performed ICN underlay trials. One such trial involved commercial end users located in the Primetel network in Cyprus with the use case centered on IPTV and HLS video dissemination. Another trial was performed in the community network of "guifi.net" in the Barcelona region, where the solution was deployed in 40 households, providing general Internet connectivity to the residents. Standard IPTV STBs as well as HLS video players were utilized in accordance with the aim of this deployment configuration, namely to provide application and service migration.

5.2.2. H2020 FLAME Efforts

The H2020 FLAME efforts concentrate on providing an experimental ground for the aforementioned POINT/RIFE solution in initially two city-scale locations, namely in Bristol and Barcelona. This trial followed the underlay deployment configuration in Section 3.3.2 as per POINT/RIFE approach. Experiments were conducted with the city/university joint venture Bristol-is-Open (BIO), to ensure the readiness of the city-scale SDN transport network for such experiments. Another trial was for the ETSI MEC PoC. This trial showcased operational benefits provided by the ICN underlay for the scenario of a location-based game. These benefits aim at reduced network utilization through improved video delivery performance

(multicast of all captured videos to the service surrogates deployed in the city at six locations) as well as reduced latency through the playout of the video originating from the local NAP instead of a remote server.

Ensuring the technology readiness and the early trialing of the ICN capabilities lays the ground for the goal of the H2020 FLAME efforts to conduct 23 large-scale experiments in the area of Future Media Internet (FMI) throughout 2018 and 2019. Standard media service functions as well as applications will ultimately utilize the ICN underlay in the delivery of their experience. The platform, which includes the ICN capabilities, will utilize concepts of SFC, integrated with NFV and SDN capabilities of the infrastructure. The ultimate goal of these platform efforts is the full integration of ICN into the overall media function platform for the provisioning of advanced (media-centric) internet services.

5.2.3. CableLabs Content Delivery System

The work in [White] proposes an underlay deployment configuration based on Section 3.3.2. The use case is ICN for content distribution within CDN server farms (which can be quite large and complex) to leverage ICN's superior in-network caching properties. This "island of ICN" based CDN is then used to service standard HTTP/IP-based content retrieval request coming from the general Internet. This approach acknowledges that whole scale replacement (see Section 3.1) of existing HTTP/IP end user applications and related Web infrastructure is a difficult proposition. [White] does not yet provide results but indicated that experiments will be forthcoming.

5.2.4. NDN IoT Trials

[Baccelli] summarizes the trial of an NDN system adapted specifically for a wireless IoT scenario. The trial was run with 60 nodes distributed over several multi-story buildings in a university campus environment. The NDN protocols were optimized to run directly over 6LoWPAN wireless link layers. The performance of the NDN based IoT system was then compared to an equivalent system running standard IP based IoT protocols. It was found that the NDN based IoT system was superior in several respects including in terms of energy consumption, and for RAM and ROM footprints [Baccelli] [Anastasiades].

5.3. Other Configurations

This section records deployment trial experiences from systems that do not directly correspond to one of the basic configurations defined in Section 3.

5.3.1. Hybrid ICN Trials

Hybrid ICN [Hybrid_ICN-1] [Hybrid_ICN-2] is an approach where the ICN names are mapped to IPv6 addresses, and other ICN information is carried as payload inside the IP packet. This allows standard (ICN-unaware) IP routers to forward packets based on IPv6 info, but enables ICN-aware routers to apply ICN semantics. A related open source effort was kicked off in 2017 (<https://wiki.fd.io/view/Cicn>). The intent of the trials are to show the routing performance efficiency of the Hybrid ICN router (called the Vector Packet Processor) over existing IP routers. Results have not yet been published but are expected in the near future.

5.4. Summary of Deployment Trials

In summary, there have been significant trials over the years with all the major ICN protocol flavors (e.g., CCN, NDN, POINT) using both the ICN-as-an-Overlay and ICN-as-an-Underlay deployment configurations. The major limitations of the trials include the fact that only a limited number of applications have been tested. However, the tested applications include both native ICN and existing IP based applications (e.g. video-conferencing and IPTV). Another limitation of the trials is that all of them involve less than 1000 users maximum.

The ICN-as-a-Slice configuration still has not been trialled primarily due to the fact that 5G standards are still in flux and not expected to be stable before the mid-2018 time frame. The Clean-slate ICN approach has obviously never been trialled as complete replacement of Internet infrastructure (e.g., existing applications, TCP/IP protocol stack, IP routers, etc.) is no longer considered a viable alternative. Finally, the Hybrid ICN approach offers an interesting alternative as it allows ICN semantics to be embedded in standard IPv6 packets and so the packets can be routed through either IP routers or Hybrid ICN routers. Detailed performance results are still pending for this alternative.

6. Deployment Issues Requiring Further Standardization

The ICN Research Challenges [RFC7927] describes key ICN principles and technical research topics. As the title suggests, [RFC7927] is research oriented without a specific focus on deployment or standardization issues. This section addresses this open area by identifying key protocol functionality that may be relevant for further standardization effort in IETF. The focus is specifically on identifying protocols that will facilitate future interoperable ICN deployments correlating to the scenarios identified in the deployment

migration paths in Section 4. The identified list of potential protocol functionality is not exhaustive.

6.1. Protocols for Application and Service Migration

End user applications and services need a standardized approach to trigger ICN transactions. For example, in Internet and Web applications today, there are established socket APIs, communication paradigms such as REST, common libraries, and best practices. We see a need to study application requirements in an ICN environment further and, at the same time, develop new APIs and best practices that can take advantage of ICN communication characteristics.

6.2. Protocols for Content Delivery Network Migration

A key issue in CDNs is to quickly find a location of a copy of the object requested by an end user. In ICN, a Named Data Object (NDO) is typically defined by its name. There already exists [RFC6920] that is suitable for static naming of ICN data objects. Other ways of encoding and representing ICN names have been described in [I-D.irtf-icnrg-ccnxmessages] and [I-D.mosko-icnrg-ccnxurischeme]. Naming dynamically generated data requires different approaches (for example, hash digest based names would normally not work), and there is lack of established conventions and standards.

Another CDN issue for ICN is related to multicast distribution of content. Existing CDNs have started using multicast mechanisms for certain cases such as for broadcast streaming TV. However, as discussed in Section 5.2.1, certain ICN approaches provide substantial improvements over IP multicast, such as the implicit support for multicast retrieval of content in all ICN flavours.

Caching is an implicit feature in many ICN architectures that can improve performance and availability in several scenarios. The ICN in-network caching can augment managed CDN and improve its performance. The details of the interplay between ICN caching and managed CDN need further consideration.

6.3. Protocols for Edge and Core Network Migration

ICN provides the potential to redesign current edge and core network computing approaches. Leveraging ICN's inherent security and its ability to make name data and dynamic computation results available independent of location, can enable a secure, yet light-weight insertion of traffic into the network without relying on redirection of DNS requests. For this, proxies that translate from commonly used protocols in the general Internet to ICN message exchanges in the ICN domain could be used for the migration of application and services

within deployments at the network edge but also in core networks. This is similar to existing approaches for IoT scenarios where a proxy translates CoAP request/responses to other message formats. For example, [RFC8075] specifies proxy mapping between CoAP and HTTP protocols. However, as mentioned previously, ICN will allow us to evolve the role of gateways/proxies as ICN message security should be preserved through the protocol translation function of a thus offer a substantial gain.

Interaction and interoperability between existing IP routing protocols (e.g., OSPF, RIP, ISIS) and ICN routing approaches (e.g., NFD, CCN routers) are expected especially in the overlay approach. Another important topic is integration of ICN into networks that support virtualized infrastructure in the form of NFV/SDN and most likely utilizing Service Function Chaining (SFC) as a key protocol. Further work is required to validate this idea and document best practices.

Operations and Maintenance (OAM) is a crucial area that has not yet been fully addressed by the ICN research community, but which is obviously critical for future deployments of ICN. Potential areas that need investigation include whether the YANG data modelling approach and associated NETCONF/RESTCONF protocols need any specific updates for ICN support. Another open area is how to measure and benchmark performance of ICN networks comparable to the sophisticated techniques that exist for standard IP networks, virtualized networks and data centers. It should be noted that some initial progress has been made in the area of ICN network path traceroute facility with approaches such as CONTRACE [I-D.asaeda-icnrg-contrace] [Contrace].

6.4. Summary of ICN Protocol Gaps and Potential Protocol Efforts

Without claiming completeness, Table 1 maps the open the open ICN issues identified in this document to potential protocol efforts that could address some aspects of the gap.

ICN Gap	Potential Protocol Effort
1-Support of REST APIs	HTTP/CoAP support of ICN semantics
2-Naming	Dynamic naming of ICN data objects
3-Routing	Interactions between IP and ICN routing protocols
4-Multicast distribution	Multicast enhancements for ICN
5-In-network caching	ICN Cache placement and sharing
6-NFV/SDN support	Integration of ICN with NFV/SDN and including possible impacts to SFC
7-ICN mapping	Mapping of HTTP and other protocols onto ICN message exchanges (and vice-versa) while preserving ICN message security
8-OAM support	YANG models, NETCONF/RESTCONF protocols, and network performance measurements

Table 1: Mapping of ICN Gaps to Potential Protocol Efforts

7. Conclusion

This document provides high level deployment considerations for the ICN community. Specifically, the major configurations of possible ICN deployments are identified as (1) Clean-slate ICN replacement of existing Internet infrastructure; (2) ICN-as-an-Overlay; (3) ICN-as-an-Underlay; and (4) ICN-as-a-Slice. Existing ICN trial systems primarily fall under either the ICN-as-an-Overlay or ICN-as-an-Underlay configuration.

In terms of deployment migration paths, ICN-as-an-Underlay offers a clear migration path for CDN, edge and core networks to go to an ICN paradigm (e.g., for an IoT deployment). ICN-as-an-Overlay is probably the easiest configuration to deploy as it leaves the underlying IP infrastructure essentially untouched. However its applicability for general deployment must be considered on a case by case basis (e.g., based on if it can run all required applications or other similar criteria). ICN-as-a-Slice is an attractive deployment

option for future 5G systems (i.e., for 5G radio and core networks) which will naturally support network slicing, but this still has to be validated through actual trial experiences.

For the crucial issue of existing application and service migration to ICN, various mapping schemes are possible to mitigate impacts. For example, HTTP/TCP/IP flows may be mapped to/from ICN message flows at proxies in the ICN-as-an-Underlay configurations leaving the massive number of existing end point applications/services untouched or minimally impacted. Also dual stack end user devices that include middleware to allow applications to communicate in both ICN mode and standard IP mode are an attractive proposition for gradual and geographically discontinuous introduction for all deployment configurations.

There has been significant trial experience with all the major ICN protocol flavors (e.g., CCN, NDN, POINT). However, only a limited number of applications have been tested so far, and the maximum number of users in any given trial has been less than 1000 users. It is recommended that future ICN deployments scale their users gradually and closely monitor network performance as they go above 1000 users.

Finally, this document describes a set of technical features in ICN that warrant potential future IETF specification work. This will aid initial and incremental deployments to proceed in an interoperable manner. The fundamental details of the potential protocol specification effort, however, are best left for future study by the appropriate IETF WGs and/or BoFs.

8. IANA Considerations

This document requests no IANA actions.

9. Security Considerations

ICN was purposefully designed from the start to have certain intrinsic security properties. The most well known of which are authentication of delivered content and (optional) encryption of the content. [RFC7945] has an extensive discussion of various aspects of ICN security including many which are relevant to deployments. Specifically, [RFC7945] points out that ICN access control, privacy, security of in-network caches, and protection against various network attacks (e.g. DoS) have not yet been fully developed due to the lack of real deployments. [RFC7945] also points out relevant advances occurring in the ICN research community that hold promise to address each of the identified security gaps. Lastly, [RFC7945] points out that as secure communications in the existing Internet (e.g. HTTPS)

becomes the norm, that major gaps in ICN security will inevitably slow down the adoption of ICN.

In addition to the security findings of [RFC7945], this document has highlighted that all anticipated ICN deployment configurations will involve co-existence with existing Internet infrastructure and applications. Thus even the basic authentication and encryption properties of ICN content will need to account for interworking with non-ICN content to preserve end-to-end security. For example, in the edge network underlay deployment configuration described in Section 3.3.1, the gateway/proxy that translates HTTP or CoAP request/responses into ICN message exchanges will need to support a model to preserve end-to-end security.

10. Acknowledgments

The authors want to thank Alex Afanasyev, Mayutan Arumaithurai, Hitoshi Asaeda, Giovanna Carofiglio, Xavier de Foy, Hannu Flinck, Anil Jangam, Luca Muscariello, Dave Oran, Thomas Schmidt, Jan Seedorf, Eve Schooler, Samar Shailendra, Milan Stolic, Prakash Suthar, Atsushi Tagami, and Lixia Zhang for their very useful reviews, comments and input to the document.

11. Informative References

[Anastasiades]

Anastasiades, C., "Information-centric communication in mobile and wireless networks", PhD Dissertation, 2016, <http://boris.unibe.ch/83683/1/16anastasiades_c.pdf>.

[Baccelli]

Baccelli, E. and et al., "Information Centric Networking in the IoT: Experiments with NDN in the Wild", ACM 20164, Paris, France, 2014, <<http://conferences2.sigcomm.org/acm-icn/2014/papers/p77.pdf>>.

[C_FLOW]

Suh, J. and et al., "C_FLOW: Content-Oriented Networking over OpenFlow", Open Networking Summit, April, 2012, <<http://opennetsummit.org/archives/apr12/site/pdf/snu.pdf>>.

[CCNx_UDP]

PARC, "CCNx Over UDP", 2015, <<https://www.ietf.org/proceedings/interim-2015-icnrg-04/slides/slides-interim-2015-icnrg-4-5.pdf>>.

- [CONET] Veltri, L. and et al., "CONET Project: Supporting Information-Centric Functionality in Software Defined Networks", Workshop on Software Defined Networks, , 2012, <http://netgroup.uniroma2.it/Stefano_Salsano/papers/salsano-iccl2-wshop-sdn.pdf>.
- [Contrace] Asaeda, H. and et al., "Contrace: A Tool for Measuring and Tracing Content-Centric Networks", IEEE Communications Magazine, Vol.53, No.3 , 2015.
- [CUTEi] Asaeda, H. and N. Choi, "Container-Based Unified Testbed for Information Centric Networking", IEEE Network, Vol.28, No.6 , 2014.
- [DASH] DASH, "DASH Industry Forum", 2017, <<http://dashif.org/>>.
- [fiveG-23501] 3gpp-23.501, "Technical Specification Group Services and System Aspects; System Architecture for the 5G System (Rel.15)", 3GPP , 2017.
- [fiveG-23502] 3gpp-23.502, "Technical Specification Group Services and System Aspects; Procedures for the 5G System (Rel.15)", 3GPP , 2017.
- [Hybrid_ICN-1] Cisco, "Hybrid ICN: Cisco Announces Important Steps toward Adoption of Information-Centric Networking", 2017, <<http://blogs.cisco.com/sp/cisco-announces-important-steps-toward-adoption-of-information-centric-networking>>.
- [Hybrid_ICN-2] Cisco, "Mobile Video Delivery with Hybrid ICN: IP-Integrated ICN Solution for 5G", 2017, <<https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/ultra-services-platform/mwcl7-hicn-video-wp.pdf>>.
- [I-D.asaeda-icnrg-contrace] Asaeda, H., Shao, X., and T. Turlatti, "Contrace: Traceroute Facility for Content-Centric Network", draft-asaeda-icnrg-contrace-04 (work in progress), October 2017.

[I-D.ietf-bier-use-cases]

Kumar, N., Asati, R., Chen, M., Xu, X., Dolganow, A., Przygienda, T., Gulko, A., Robinson, D., Arya, V., and C. Bestler, "BIER Use Cases", draft-ietf-bier-use-cases-05 (work in progress), July 2017.

[I-D.irtf-icnrg-ccnxmessages]

Mosko, M., Solis, I., and C. Wood, "CCNx Messages in TLV Format", draft-irtf-icnrg-ccnxmessages-06 (work in progress), October 2017.

[I-D.irtf-icnrg-icniot]

Ravindran, R., Zhang, Y., Grieco, L., Lindgren, A., Raychadhuri, D., Baccelli, E., Burke, J., Wang, G., Ahlgren, B., and O. Schelen, "Design Considerations for Applying ICN to IoT", draft-irtf-icnrg-icniot-00 (work in progress), January 2018.

[I-D.irtf-icnrg-terminology]

Wissingh, B., Wood, C., Afanasyev, A., Zhang, L., Oran, D., and C. Tschudin, "Information-Centric Networking (ICN): CCN and NDN Terminology", draft-irtf-icnrg-terminology-00 (work in progress), December 2017.

[I-D.irtf-nfvrg-gaps-network-virtualization]

Bernardos, C., Rahman, A., Zuniga, J., Contreras, L., Aranda, P., and P. Lynch, "Network Virtualization Research Challenges", draft-irtf-nfvrg-gaps-network-virtualization-08 (work in progress), November 2017.

[I-D.kutscher-icnrg-netinf-proto]

Kutscher, D., Farrell, S., and E. Davies, "The NetInf Protocol", draft-kutscher-icnrg-netinf-proto-01 (work in progress), February 2013.

[I-D.mosko-icnrg-ccnxurischeme]

marc.mosko@parc.com, m. and c. cwood@parc.com, "The CCNx URI Scheme", draft-mosko-icnrg-ccnxurischeme-01 (work in progress), April 2016.

[I-D.paik-icn-deployment-considerations]

Paik, E., Yun, W., Kwon, T., and h. hgchoi@mmlab.snu.ac.kr, "Deployment Considerations for Information-Centric Networking", draft-paik-icn-deployment-considerations-00 (work in progress), July 2013.

- [I-D.ravi-icnrg-5gc-icn]
Ravindran, R., suthar, P., and G. Wang, "Enabling ICN in 3GPP's 5G NextGen Core Architecture", draft-ravi-icnrg-5gc-icn-00 (work in progress), October 2017.
- [I-D.suthar-icnrg-icn-lte-4g]
suthar, P., Stolic, M., Jangam, A., and D. Trossen, "Native Deployment of ICN in LTE, 4G Mobile Networks", draft-suthar-icnrg-icn-lte-4g-04 (work in progress), November 2017.
- [ICN2020] ICN2020, "ICN2020 Deliverables", 2017,
<<http://www.icn2020.org/dissemination/deliverables-public/>>.
- [ICNRCCharter]
NDN, "Information-Centric Networking Research Group Charter", 2013,
<<https://datatracker.ietf.org/doc/charter-irtf-icnrg/>>.
- [IEEE_Communications]
Trossen, D. and G. Parisis, "Designing and Realizing an Information-Centric Internet", Information-Centric Networking, IEEE Communications Magazine Special Issue, 2012.
- [Internet_Pricing]
Trossen, D. and G. KBiczok, "Not Paying the Truck Driver: Differentiated Pricing for the Future Internet", ReArch Workshop in conjunction with ACM Context, December, 2010.
- [Jacobson]
Jacobson, V. and et al., "Networking Named Content", Proceedings of ACM Context, , 2009.
- [Jangam]
Jangam, A. and et al., "Porting and Simulation of Named-data Link State Routing Protocol into ndnSIM", ACM DIVANet'17, Miami Beach, USA, 2017,
<<http://symposium.nsercdiva.com/2017/program.html>>.
- [Moiseenko]
Moiseenko, I. and D. Oran, "TCP/ICN : Carrying TCP over Content Centric and Named Data Networks", 2016,
<<http://conferences2.sigcomm.org/acm-icn/2016/proceedings/pl12-moiseenko.pdf>>.

- [MWC_Demo] InterDigital, "InterDigital Demo at Mobile World Congress (MWC)", 2016, <<http://www.interdigital.com/download/56d5c71bd616f892ba001861>>.
- [NFD] NDN, "NFD - Named Data Networking Forwarding Daemon", 2017, <<https://named-data.net/doc/NFD/current/>>.
- [NGMN] NGMN, "NGMN 5g Initiative White Paper", 2015, <https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0.pdf>.
- [ONAP] ONAP, "Open Network Automation Platform", 2017, <<https://www.onap.org/>>.
- [oneM2M] OneM2M, "oneM2M Service Layer Standards for M2M and IoT", 2017, <<http://www.onem2m.org/>>.
- [Overlay_ICN] Shailendra, S. and et al., "A Novel Overlay Architecture for Information Centric Networking", 2016, <https://www.researchgate.net/publication/282779666_A_novel_overlay_architecture_for_Information_Centric_Networking>.
- [POINT] Trossen, D. and et al., "POINT: IP Over ICN - The Better IP?", European Conference on Networks and Communications (EuCNC), , 2015.
- [Ravindran] Ravindran, R., Chakraborti, A., Amin, S., Azgin, A., and G. Wang, "5G-ICN : Delivering ICN Services over 5G using Network Slicing", IEEE Communication Magazine, May, 2016, <<https://arxiv.org/abs/1610.01182>>.
- [Reed] Reed, M. and et al., "Stateless Multicast Switching in Software Defined Networks", ICC 2016, Kuala Lumpur, Malaysia, 2016.
- [RFC6920] Farrell, S., Kutscher, D., Dannewitz, C., Ohlman, B., Keranen, A., and P. Hallam-Baker, "Naming Things with Hashes", RFC 6920, DOI 10.17487/RFC6920, April 2013, <<https://www.rfc-editor.org/info/rfc6920>>.
- [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>>.

- [RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <<https://www.rfc-editor.org/info/rfc7426>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.
- [RFC7927] Kutscher, D., Ed., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlich, "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>>.
- [RFC8075] Castellani, A., Loreto, S., Rahman, A., Fossati, T., and E. Dijk, "Guidelines for Mapping Implementations: HTTP to the Constrained Application Protocol (CoAP)", RFC 8075, DOI 10.17487/RFC8075, February 2017, <<https://www.rfc-editor.org/info/rfc8075>>.
- [SAIL_NetInf] FP7, "SAIL Prototyping and Evaluation", 2013, <http://www.sail-project.eu/wp-content/uploads/2013/05/SAIL_DB4_v1.1_Final_Public.pdf>.
- [Tateson] Tateson, J. and et al., "Final Evaluation Report on Deployment Incentives and Business Models", 2010, <http://www.psirp.org/files/Deliverables/FP7-INFSo-ICT-216173-PSIRP-D4.6_FinalReportOnDeplIncBusinessModels.pdf>.
- [Techno_Economic] Trossen, D. and A. Kostopolous, "Techno-Economics Aspects of Information-Centric Networking", Journal for Information Policy, Volume 2, 2012.

- [VSER] Ravindran, R., Liu, X., Chakraborti, A., Zhang, X., and G. Wang, "Towards software defined ICN based edge-cloud services", CloudNetworking(CloudNet), IEEE International Conference on, IEEE International Conference on CloudNetworking(CloudNet), 2013.
- [VSER-Mob] Azgin, A., Ravindran, R., Chakraborti, A., and G. Wang, "Seamless Mobility as a Service in Information-centric Networks", ACM ICN Sigcomm, IC5G Workshop, 2016.
- [White] White, G. and G. Rutz, "Content Delivery with Content Centric Networking, CableLabs White Paper", 2010, <<http://www.cablelabs.com/wp-content/uploads/2016/02/Content-Delivery-with-Content-Centric-Networking-Feb-2016.pdf>>.

Appendix A. Change Log

[Note to RFC Editor: Please remove this section before publication.]

Changes from rev-04 to rev-05:

- o Added this Change Log in Appendix A.
- o Removed references to Hybrid ICN from section 3.2 (ICN-as-an-Overlay definition). Instead, consolidated all Hybrid ICN info in the Deployment Trial Experiences under a new subsection 5.3 (Other Configurations).
- o Updated ICN2020 description in Section 5.1.4 with text received from Mayutan Arumaithurai and Hitoshi Asaeda.
- o Clarified in ICN-as-a-Slice description (section 3.4) that it may be deployed on either the Edge (RAN) or Core Network, or the ICN-as-a-Slice may be deployed end-to-end through the entire Mobile network.
- o Added several new references in various sections.
- o Various minor editorial updates.

Authors' Addresses

Akbar Rahman
InterDigital Inc.
1000 Sherbrooke Street West, 10th floor
Montreal H3A 3G4
Canada

Email: Akbar.Rahman@InterDigital.com
URI: <http://www.InterDigital.com/>

Dirk Trossen
InterDigital Inc.
64 Great Eastern Street, 1st Floor
London EC2A 3QR
United Kingdom

Email: Dirk.Trossen@InterDigital.com
URI: <http://www.InterDigital.com/>

Dirk Kutscher
Huawei German Research Center
Riesstrasse 25
Munich 80992
Germany

Email: ietf@dkutscher.net
URI: <http://www.Huawei.com/>

Ravi Ravindran
Huawei Research Center
2330 Central Expressway
Santa Clara 95050
USA

Email: ravi.ravindran@huawei.com
URI: <http://www.Huawei.com/>

ICNRG
Internet-Draft
Intended status: Informational
Expires: May 2, 2018

R. Ravindran
Huawei
P. Suthar
Cisco
G. Wang
Huawei
October 29, 2017

Enabling ICN in 3GPP's 5G NextGen Core Architecture
draft-ravi-icnrg-5gc-icn-00

Abstract

The proposed 3GPP's 5G core nextgen architecture (5GC) offers flexibility to introduce new user and control plane function within the context of network slicing that allows greater flexibility to handle heterogeneous devices and applications. In this draft, we provide a short description of the proposed 5GC, followed by extensions to 5GC's control and user plane to support packet data unit (PDU) sessions from information-centric networks. The value of enabling ICN in 5GC is discussed using two service scenarios which include mobile edge computing and support for seamless mobility for ICN sessions.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 2, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	4
3. 5G NextGen Core Design Principles	5
4. 5G NextGen Core Architecture	6
5. 5GC Architecture with ICN Support	8
5.1. Control Plane Extensions	10
5.1.1. Normative Interface Extensions	12
5.2. User Plane Extensions	12
5.2.1. Normative Interface Extensions	13
6. ICN Deployment Use Case Scenarios	14
6.1. Mobile Edge Computing	14
6.1.1. IP-MEC Scenario	14
6.1.2. ICN-MEC Scenario	15
6.2. ICN Session Mobility	16
7. Conclusion	18
8. IANA Considerations	18
9. Security Considerations	18
10. Acknowledgments	18
11. Informative References	18
Authors' Addresses	20

1. Introduction

The objective of this draft is to propose an architecture to enable information-centric networking (ICN) in the proposed 5G Next-generation Core network architecture (5GC) by leveraging its flexibility to allow new user and associated control plane functions. The reference architectural discussions in three core 3GPP specifications [TS23.501][TS23.502][TS23.799] form the basis of our discussions. This draft also complements the discussions related to various ICN deployment opportunities explored in [I-D.rahman-icnrg-deployment-guidelines], where 5G technology is considered as one of the promising alternatives.

Though ICN is a general networking technology, it would benefit 5G particularly from the perspective of mobile edge computing (MEC). Following ICN features shall benefit MEC deployments in 5G:

- o Edge Computing: Multi-access Edge Computing (MEC) is located at edge of network and aids several latency sensitive applications such as augmented and virtual reality (AR/VR) and ultra reliable and low latency class (URLLC) of applications such as autonomous vehicles. Enabling edge computing over an IP converged 5GC comes with the challenge of application level reconfiguration required to re-initialize a session whenever it is being served by a non-optimal service instance. In contrast, named-based networking, as considered by ICN, naturally supports service-centric networking, which minimizes network related configuration for applications and allows fast resolution for named service instances.
- o Edge Storage and Caching : The principal entity for ICN is the secured content (or named-data) object, which allows location independent data replication at strategic storage points in the network, or data dissemination through ICN routers by means of opportunistic caching. These features benefit both realtime and non-realtime applications, where a set of users share the same content, thereby advantageous to both high-bandwidth and/or low-latency applications such as Video-on-Demand (VOD), AR/VR or low bandwidth IoT applications.
- o Session Mobility: Existing long-term evolution (LTE) deployments handle session mobility using IP anchor points at Packet Data Network Gateway (PDN-GW) and service anchor point called Access Point Name (APN) functionality hosted in PDN-GW, and uses a tunnel between radio edge (eNodeB) and PDN-GW for each mobile device attached to network. This design fails when service instances are replicated close to radio access network (RAN) instances, requiring new techniques to handle session mobility. In contrast, application-bound identifier and name resolution split principle considered for the ICN is shown to handle host mobility quite efficiently [mas].

To summarize the draft, we first discuss 5GC's design principals that allows the support of new network architectures. Then we summarily discuss the 5GC proposal, followed by control and user plane extensions required to support ICN PDU sessions. We then discuss specific network services enabled using ICN data networks, specifically MEC and ICN session mobility with aid from 5GC control plane.

2. Terminology

Following are terminologies relevant to this draft:

5G-NextGen Core (5GC) : Refers to the new 5G core network architecture being developed by 3GPP, we specifically refer to the architectural discussions in [TS23.501][TS23.502].

5G-New Radio (5G-NR): This refers to the new radio access interface developed to support 5G wireless interface [TS-5GNNR].

User Plane Function (UPF): UPF is the generalized logical data plane function with context of the UE PDU session. UPFs can play many role, such as, being an flow classifier (UL-CL) (defined next), a PDU session anchoring point, or a branching point.

Uplink Classifier (UL-CL): This is a functionality supported by an UPF that aims at diverting traffic (locally) to local data networks based on traffic matching filters applied to the UE traffic.

Packet Data Network (PDN or DN): This refers to service networks that belong to the operator or third party offered as a service to the UE.

Unified Data Management (UDM): Manages unified data management for wireless, wireline and any other types of subscribers for M2M, IOT application etc. UDM report subscriber related vital information e.g. virtual edge region, list of location visits, sessions active etc. UDM work as subscriber anchor point that means OSS/BSS systems will have central monitoring/access of the system to get/set subscriber information.

Authentication Server Function (AUSF): Provides mechanism for unified authentication for subscribers related to wireless, wireline and any other types of subscribers such as M2M and IOT applications. The functions performed by AUSF are similar to HSS with additional functionalities to related to 5G.

Session Management Function (SMF): Perform session management functions for attached users equipment (UE) in 5G Core. SMF can thus be formed by leveraging the CUPS (discussed in the next section) feature with control plane session management.

Access Mobility Function (AMF): Perform access mobility management for attached user equipment (UE) to the 5G core network. The function includes, network access stratus (NAS) mobility functions such as authentication and authorization.

Application Function (AF): Helps with influencing the user plane routing state in 5GC considering service requirements.

Network Slicing: This conceptualizes the grouping for a set of logical or physical network functions with its own or shared control, data and service plane to meet specific service requirements.

3. 5G NextGen Core Design Principles

The 5GC architecture is based on the following design principles that allow it to support new service networks like ICN efficiently compared to LTE networks:.

- o Control and User plane split (CUPS): This design principle moves away from LTE's vertically integrated control/user plane design (i.e., Serving Gateway, S-GW, and Packet Data Network Gateway, P-GW) to one espousing an NFV framework with network functions separated from the hardware for service-centricity, flexibility and programmability. In doing so, network functions can be implemented both physically and virtually, while allowing each to be customized and scaled based on their individual requirements, also allowing the realization of multi-slice co-existence. This feature also allows the introduction of new user plane functions (UPF). UPFs can play many roles, such as, being an uplink flow classifier (UL-CL), a PDU session anchor point, a branching point function, or one based on new network architectures like ICN with new control functions, or re-using/extending the existing ones to manage the new user plane realizations.
- o Decoupling of RAT and Core Network : Unlike LTE's unified control plane for access and the core, 5GC offers control plane separation of the RAN from the core network. This allows the introduction of new radio access technologies (RAT) along with slices based on new network architectures, offering the ability to map heterogeneous RAN flows to arbitrary core network slices based on service requirements.
- o Non-IP PDU Session Support : A PDU session is defined as the logical connection between the UE and the data network (DN). 5GC offers a scope to support both IP and non-IP PDU (termed as "unstructured" payload), and this feature can potentially allow the support for ICN PDUs by extending or re-using the existing control functions.
- o Service Centric Design: 5GC's service orchestration and control functions, such as naming, addressing, registration/authentication and mobility, will utilize cloud based service APIs. Doing so

enables opening up interfaces for authorized service function interaction and creating service level extensions to support new network architectures. These APIs include the well accepted Get/Response and Pub/Sub approaches, while not precluding the use of procedural approach between functional units (where necessary).

4. 5G NextGen Core Architecture

In this section, for brevity purposes, we restrict the discussions to the control and user plane functions relevant to an ICN deployment. More exhaustive discussions on the various architecture functions, such as registration, connection and subscription management, can be found in[TS23.501][TS23.502].

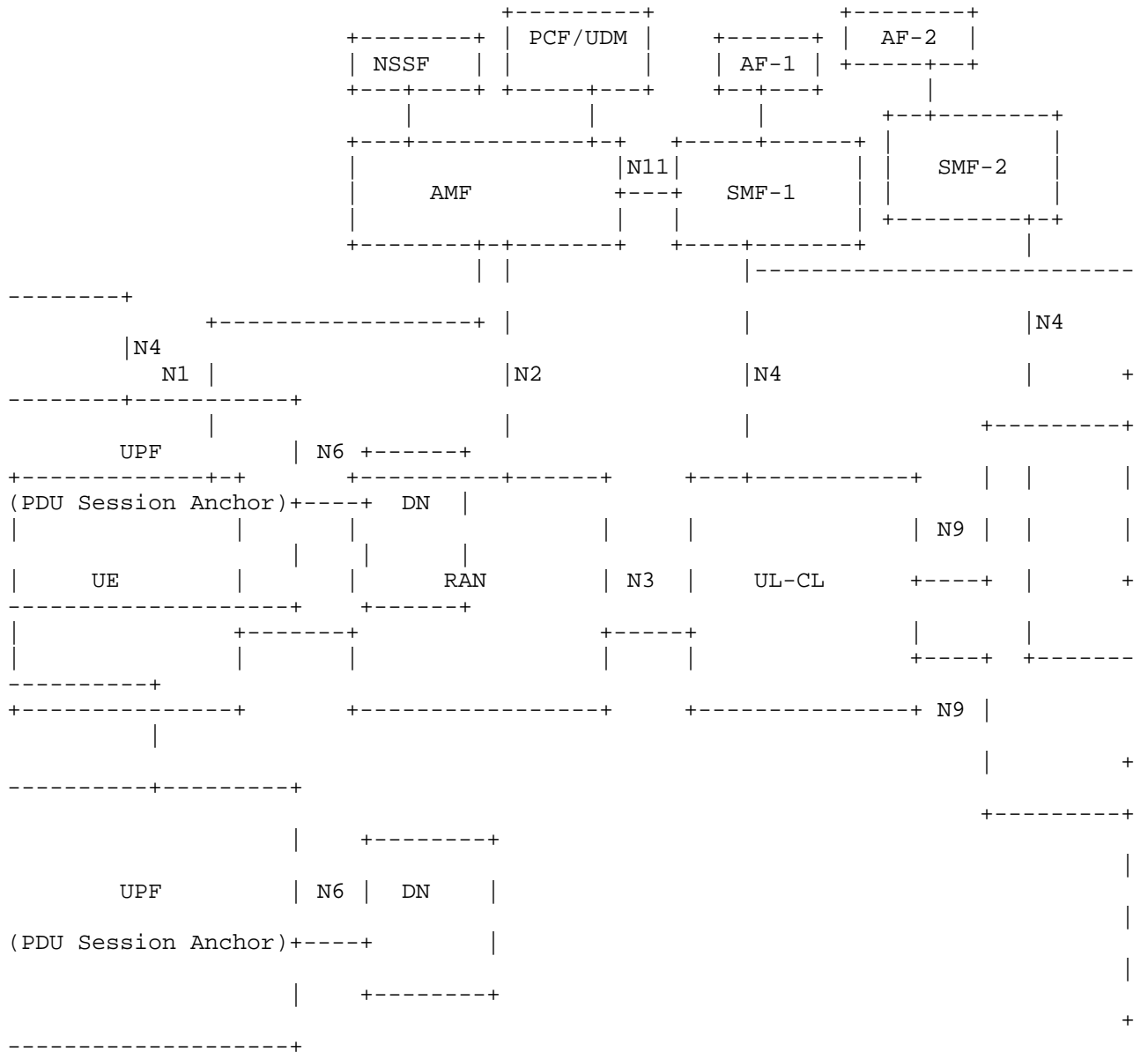


Figure 1: 5G Next Generation Core Architecture

In Figure 1, we show one variant of a 5GC architecture from [TS23.501], for which the functions of UPF's branching point and PDU

session anchoring are used to support inter-connection between a UE and the related service or packet data networks (or PDNs) managed by the signaling interactions with control plane functions. In 5GC, control plane functions can be categorized as follows:

- o Common control plane functions that are common to all slices and which include the Authentication and Mobility Function (AMF), Network Slice and Selection Function (NSSF), Policy Control Function (PCF), and Unified Data Management (UDM) among others.
- o Shared or slice specific control functions, which include the Session and Management Function (SMF) and the Application Function (AF).

AMF serves multiple purposes: (i) device authentication and authorization; (ii) security and integrity protection to non-access stratum (NAS) signaling; (iii) tracking UE registration in the operator's network and mobility management functions as the UE moves among different RANs, each of which might be using different radio access technologies (RAT).

NSSF handles the selection of a particular slice for the PDU session request from the user entity (UE) using the Network Slice Selection Assistance Information (NSSAI) parameters provided by the UE and the configured user subscription policies in PCF and UDM functions. Compared to LTE's evolved packet core (EPC), where PDU session states in RAN and core are synchronized with respect to management, 5GC decouples this using NSSF by allowing PDU sessions to be defined prior to a PDU session request by a UE (for other differences see [lteversus5g]). This de-coupling allows policy based inter-connection of RAN flows with slices provisioned in the core network.

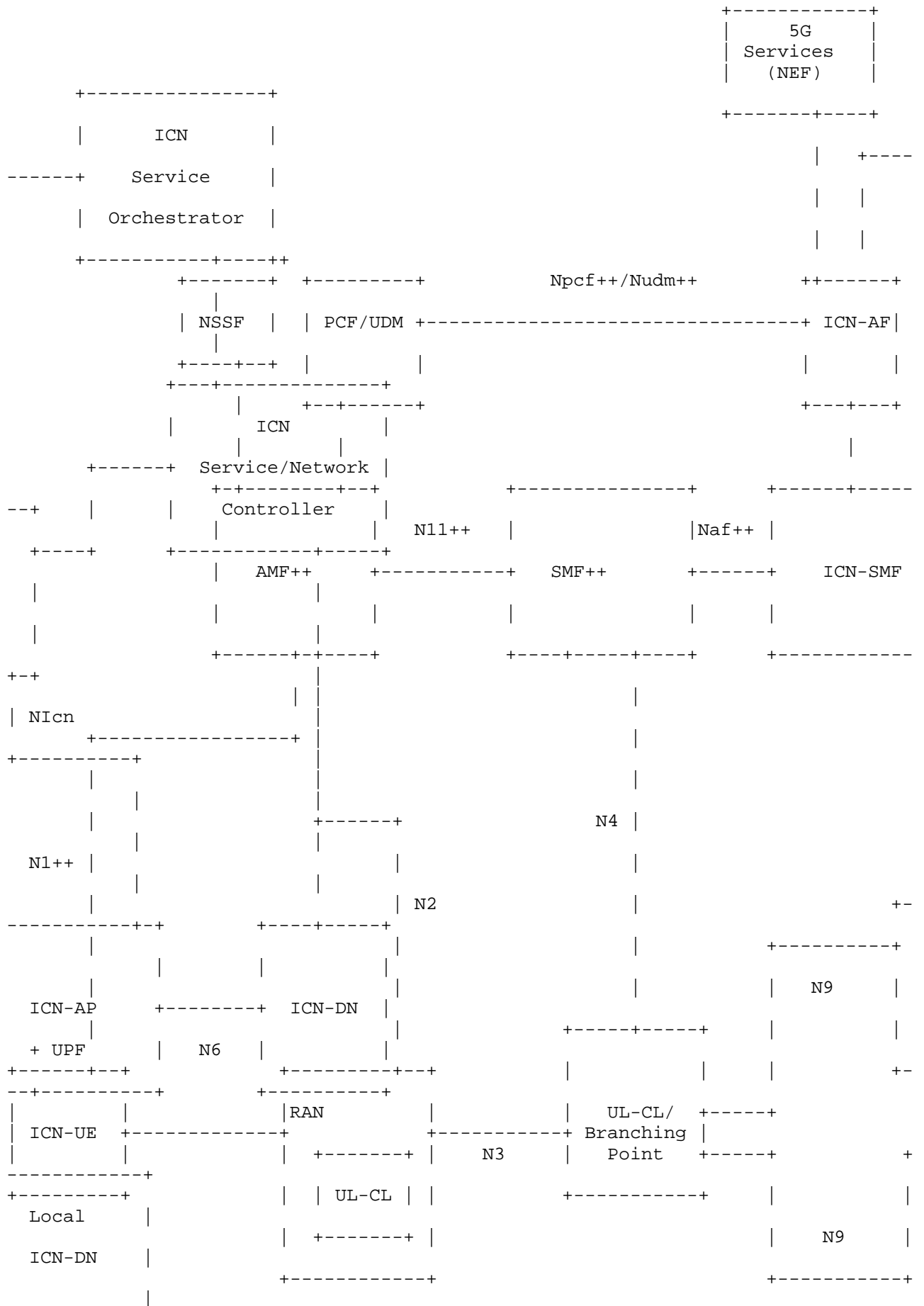
SMF handles session management functions including IP address assignment functionality, policy and service capabilities. Furthermore, it manages the data plane state in the user plane through PDU session establishment, modification and termination, and management of RAN (through the AMF) and UPF states related to a particular service or slice.

In the data plane, UE's PDUs are sent to the RAN using the 5G RAN protocol [TS-5G NR]. From the RAN, the PDU's five tuple header information (IP source/destination, port, protocol etc.) is used to map the flow to an appropriate tunnel from RAN to UPF. The UPF in this case also offers flexibility as a flow classifier and a branching point interconnecting PDUs from diverse services (within UEs) to their respective DNs. Though [TS23.501] follows LTE on using GTP tunnel from NR to the UPF to carry data PDU and another one for

the control messages to serve the control plane functions; there are ongoing discussions to arrive upon efficient alternatives to GTP.

5. 5GC Architecture with ICN Support

In this section, we focus on control and user plane enhancements required to enable ICN within 5GC, and identify the interfaces that require extensions to support ICN PDU sessions. Explicit support for ICN PDU sessions within access and 5GC networks will enable applications to leverage the core ICN features while offering it as a service to 5G users.



-----+

Figure 2: 5G Next Generation Core Architecture with ICN support

For an ICN-enabled 5GC network, the assumption is that the UE may have applications that can run over ICN or IP, for instance, UE's operating system offering applications to operate over ICN [Jacobson] or IP-based networking sockets. There may also be cases where UE is exclusively based on ICN. In either case, we identify an ICN enabled UE as ICN-UE. Different options exist to implement ICN in UE as described in [I-D.suthar-icnrg-icn-lte-4g] which is also applicable

for 5G UE to enable formal ICN session handling, such as, using a transport convergence layer above 5G-NR, through IP address assignment from 5GC or using 5GC provision of using unstructured PDU session mode during the PDU session establishment process. 5G UE can also be non-mobile devices or an IOT device using radio specification which can operate based on [TS-5G NR].

5GC will take advantage of network slicing function to instantiate heterogeneous slices, the same framework can be extended to create ICN slices as well [Ravindran]. This discussion also borrows ideas from [TS23.799], which offers a wide range of architectural discussions and proposals on enabling slices and managing multiple PDU sessions with local networks (with MEC) and its associated architectural support (in the service, control and data planes) and procedures within the context of 5GC.

Figure 2 shows the proposed ICN-enabled 5GC architecture. In the figure, new/modified functional components are identified to interconnect an ICN-DN with 5GC. The interfaces and functions that require extensions to enable ICN as a service in 5GC can be identified in the figure with a '++' symbol. We next summarize the control, user plane and normative interface extensions that help with the formal ICN support.

5.1. Control Plane Extensions

To support interconnection between ICN UEs and the appropriate ICN DN instances, we require five additional control plane extensions, which are discussed as follows.

- o Authentication and Mobility Function (AMF++) : Applications in the UEs have to be authorized to access ICN DNs. For this purpose, as in [TS23.501], operator enables ICN as a service-DN to support ICN PDU session flows. As a network service, ICN-UE should also be subscribed to it and this is imposed using the PCF and User Data and Management (UDM) functions, which may interface with the ICN Application Function (ICN-AF) for policy management of ICN PDU sessions. Hence, if the UE policy profile in the UDM doesn't enable this feature, then the ICN applications in the UE will not be allowed to connect to ICN DNs. To enable ICN stack in the UE, AMF function has to be modified to understand ICN type UE registration request and handle authentication and registration requests. AMF++ will support ICN specific bootstrapping and forwarding functions (such as naming and security) to configure UE's ICN applications. With appropriate enhancements, it should also support forwarding rules for the name prefixes that bind the flows to appropriate 5G-NR logical tunnel or slice interfaces. These functions can also be handled by the ICN-AF after setting

PDU session state in 5GC. Here, we are not recommending modification of 5G UE attach procedures but use existing attach procedures messages to carry ICN capabilities extensions in addition to supporting existing IP based services. 5G UE can request authentication for attaching to 5GC either in ICN, IP or dual-stack (IP and ICN) modes.

- o Session Management Function (SMF++) : Once a UE is authenticated to access ICN service in network, SMF manages to connect UE's ICN PDU sessions to the ICN DN. SMF++ capabilities should be able to manage both IP, ICN or dual stack UE with IP and ICN capabilities. SMF++ creates appropriate PDU session policies in the UPF, which include UL-CL and ICN anchor point (ICN-AP). For centrally delivered services, ICN-AP could also be an IP anchor point for IP applications. If MEC is enabled, these two functions would be distributed, as the UL-CL will re-route the flow to a local ICN-DN. SMF++ interfaces with AMF over N11++ to enable ICN specific user plane function, which includes IP address configuration and associated traffic filter policy to inter-connect UE with the appropriate radio slice. Furthermore, AMF++ sets appropriate state in the RAN that directs ICN flows to chosen ICN UL-CL.
- o ICN Session Management Function (ICN-SMF) : ICN-SMF serves as control plane for the ICN state managed in ICN-AP. This function interacts with SMF++ to obtain and also push ICN PDU session management information for the creation, modification and deletion of ICN PDU sessions in ICN-AP. For instance, when new ICN slices are provisioned by the ICN service orchestrator, ICN-SMF requests a new PDU session to the SMF++ that extends to the RAN and UE. While SMF++ manages the tunnels to interconnect ICN-AP to UL-CL, ICN-SMF creates the appropriate forwarding state in ICN (using the forwarding information base or FIB) to enable ICN flows over appropriate tunnel interfaces managed by the SMF++.
- o ICN Application Function (ICN-AF) : ICN-AF represents the application controller function that interfaces with ICN-SMF and user data management (UDM) function in 5GC. In addition to transferring ICN forwarding rules to ICN-SMF, ICN-AF also interfaces with UDM to transfer user profile and subscription policies required to map UE's ICN PDU session request to an appropriate ICN slice through NSSF. ICN-AF is an extension of the ICN service orchestration function, which can influence both ICN-SMF and UPF to steer traffic based on UE (or changing service) requirements. ICN-AP can also interact with the northbound 5G operator's service functions, such as network exposure function (NEF) that exposes network capabilities, for e.g. location based services, that can be used by ICN-AF for proactive ICN PDU session and slice management and offer additional capabilities to UE.

5.1.1. Normative Interface Extensions

- o Nl++/N11++: This extension enables ICN specific control extensions to support ICN programmability at the UE via AMF++, and also impose QoS requirements to ICN PDU session in 5GC based on service requirements.
- o Nlcn: This extension shall support two functions: (i) control plane programmability to enable ICN PDU sessions applicable to 5GC to map to name based forwarding rules in ICN-AP; (ii) control plane extensions to enable ICN mobility anchoring at ICN-AP, in which case it also acts as POA for ICN flows. Features such as ICN mobility as a service can be supported with this extension [mas].
- o Naf++: This extensions shall support 5GC control functions such as (i.e. naming, addressing, registration/authentication and mobility) for ICN UE and PDU sessions respectively with interaction with the PCF and UDM functions. The PCF and UDM functions interacts with ICN-AF function for service or slice specific configuration.
- o Npcf++/Nudm++: This extension creates an interface to push ICN PDU session requirement to PCF and UDM functions, which is enforced during ICN application registration, authentication, during ICN slice mapping, and provisioning of resources for these PDU sessions in the UPFs.

5.2. User Plane Extensions

As explained in detail in [TS23.501], UPFs are service agnostic functions, hence extensions are not required to operate an ICN-DN. The inter-connection of UE to ICN-DN comprises of two segments, one from RAN to UL-CL and the other from UL-CL to ICN-AP. These segments use IP tunneling constructs, where the service semantic check at UL-CL and ICN-AP is performed using IP's five tuples to determine both UL and DL tunnel mappings. We summarize the relevant UPFs and the interfaces for handling ICN PDU sessions as follows.

- o ICN Anchor Point (ICN-AP): ICN-AP shall host the 5GC PDU sessions and offer inter-connection to the ICN-DNs. It manages multiple logical interfaces with ICN capable UE and relays ICN packets to the appropriate ICN PDU session instances in the DL. ICN-AP shall be logical service anchor point and it should be capable of serving different ICN services. ICN-AP also manages the mobility state of ICN-UE after session is established such as in the case of handover or roaming.

- o ICN Packet Data Network (ICN-(P)DN) : ICN-DN represents a set of ICN nodes used for ICN networking and with heterogeneous service resources such as storage and computing points. An ICN network enables both network and application services, with network services including caching, mobility, multicast, multi-path routing (and possibly network layer computing), and application services including network resources (such as cache, storage, network state resources) dedicated to the application. This UPF requires service, control and data plane mechanisms to understand the application requirements and translate them to control signaling to provision the required state in the data plane.
- o Uplink Classifier (UL-CL) :UL-CL enables classification of flows based on source or destination IP address and steers the traffic to an appropriate network or service function anchor point. Within the current context, with the assumption that ICN-AP is identified based on service IP address associated with the UE's flows, UL-CL checks the source or destination address to direct traffic to an appropriate ICN-AP. As UL-CL is a logical function, it can also reside in RAN, as shown in Figure 2, where traffic classification rules can be applied to forward the ICN payload towards the next ICN-AP, as an extension classification can also be performed over 5G-NR or ICN protocol to determine the next logical hop. For native ICN UE, ICN shall be deployed on layer-2 MAC, hence there may not be any IP association; for such packet flow, new classification schema shall be required.

5.2.1. Normative Interface Extensions

- o N3: Though the current architecture supports heterogeneous service PDU handling, future extensions can include user plane interface extensions to offer explicit support to ICN PDU session traffic, for instance, an incremental caching and computing function in RAN or UL-CL to aid with content distribution.
- o N9: Extensions to this interface can consider UPFs to enable richer service functions, for instance to aid context processing. In addition extensions to enable ICN specific encapsulation to piggyback ICN specific attributes such as traffic characteristics between the UPF branching point and the ICN-AP. The intermediate nodes between the UL-CL and the ICN-AP can also be other caching points.
- o N6: This interface is established between the ICN-AP and the ICN-DN, whose networking elements in this segment can be deployed as an overlay or as a native Layer-3 network.

6. ICN Deployment Use Case Scenarios

Here we discuss two relevant network services enabled using ICN in 5G.

6.1. Mobile Edge Computing

We consider here a radio edge service requiring low latency, high capacity and strict quality of service. For the discussion in this draft, we analyze connected vehicle scenario, where the car's navigation system (CNS) uses data from the edge traffic monitoring (TM-E) service instance to offer rich and critical insights on the road conditions (such as real-time congestion assisted with media feeds). This is aided using traffic sensing (TS) information collected through vehicle-to-vehicle (V2V) communication over dedicated short-range communications (DSRC) radio by the TS-E, or using road-side sensor units (RSU) from which this information can be obtained. The TS-E instances then push this information to a central traffic sensing instance (TS-C). This information is used by the central traffic monitoring service (TM-C) to generate useable navigation information, which can then be periodically pushed to or pulled by the edge traffic monitoring service (TM-E) to respond to requests from vehicle's CNS. For this scenario, our objective is to compare advantages of offering this service over an IP based MEC versus one based on ICN. We can generalize the following discussion to other MEC applications as well.

6.1.1. IP-MEC Scenario

Considering the above scenario, when a vehicle's networking system comes online, it first undergoes an attachment process with the 5G-RAN, which includes authentication, IP address assignment and DNS discovery. The attachment process is followed by PDU session establishment, which is managed by SMF signaling to UL-CL and the UPF instance. When the CNS application initializes, it assumes this IP address as its own ID and tries to discover the closest service instance. Local DNS then resolves the service name to a local MEC service instance. Accordingly, CNS learns the IP service point address and uses that to coordinate between traffic sensing and monitoring applications.

CNS is a mission critical application requiring instant actions which is accurate and reliable all the time. Delay of microsecond or non-response could result in fatalities. Following are main challenges with the IP-MEC design:

- o At the CNS level, non-standardization of the naming schema results in introducing an application level gateway to adapt the sensing

data obtained from DSRC system to IP networks, which becomes mandatory if the applications are from different vendors.

- o As the mobility results in handover between RAN instances, service-level or 5GC networking-level mechanisms need to be initiated to discover a better TM-E instance, which may affect the service continuity and result in session reestablishment that introduces additional control/user plane overheads.
- o Data confidentiality among multiple CNS attached 5G RAN, authentication and privacy control are offered through an SSL/TLS mechanism over the transport channel, which has to be re-established whenever the network layer attributes are reset.

6.1.2. ICN-MEC Scenario

If the CNS application is developed over ICN either natively or as an overlay over IP, ICN shall allow the same named data logic to operate over heterogeneous interfaces (such as DSRC radio, and IP transport-over-5G, unlicensed radio over WiFi etc. link), thereby avoiding the need for application layer adaptations.

We can list the advantages of using ICN-based MEC as follows:

- o As vehicles within a single road segment are likely to seek the same data, ICN-based MEC allows to leverage opportunistic caching and storage enabled at ICN-AP, thereby avoiding service level unicast transmissions.
- o Processed and stored traffic data can be easily contextualized to different user requirements.
- o Appropriate mobility handling functions can be used depending on mobility type (as consumer or producer), specifically, when an ICN-UE moves from one RAN instance to another, the next IP hop, which identifies the ICN-AP function, has to be re-discovered. Unlike the IP-MEC scenario, this association is not exposed to the applications. As discussed earlier, control plane extensions to AMF and SMF can enable re-programmability of the ICN layer in the vehicle to direct it towards a new ICN-AP, or to remain with the same ICN-AP, based on optimization requirements.
- o As ICN offers content-based security, produced content can be consumed while authenticating it at the same time (i.e., allowing any data produced to diffuse to its point of use through named data networking).

6.2. ICN Session Mobility

Mobility scenario assumes a general ICN-UE handover from S-RAN to T-RAN, where each of them is served by different UPFs, i.e., UL-CL-1 and UL-CL-2. We also assume that UL-CL-1 and UL-CL-2 use different ICN-APs as gateways, referred to as ICN-AP-1 and ICN-AP-2. From an ICN perspective, we discuss here the producer mobility case, which can be handled in multiple ways, one of which is proposed in [mas]. However, the details of the ICN mobility solution are orthogonal to this discussion. Here, ICN-UE refers to an application producer (e.g., video conferencing application, from which ICN consumers request real-time content. Here we also assume the absence of any direct physical interface, Xn, between the two RANs. The current scenario follows the handover procedures discussed in [TS23.502], with focus here on integrating it with an ICN-AP and ICN-DN, where mobility state of the ICN sessions are handled.

The overall signaling overhead to handle seamless mobility also depends on the deployment models discussed in Section 4. Here we consider the case when RAN, UL-CL and ICN-AP are physically disjoint; however in the case where RAN and UL-CL are co-located then a part of the signaling to manage the tunnel state between the RAN and UL-CL is localized, which then improves the overall signaling efficiency. This can be further extended to the case when ICN-APs are co-located with the RAN and UL-CL, leading to further simplification of the mobility signaling.

Next, we discuss the high-level steps involved during handover.

- o Step 1: When the ICN-UE decides to handover from S-RAN to T-RAN, ICN-UE signals the S-RAN with a handover-request indicating the new T-RAN it is willing to connect. This message includes the affected PDU session IDs from the 5GC perspective, along with the ICN names that require mobility support.
- o Step 2: S-RAN then signals the AMF serving the ICN-UE about the handover request. The request includes the T-RAN details, along with the affected ICN PDU sessions.
- o Step 3: Here, when SMF receives the ICN-UE's and the T-RAN information, it identifies UL-CL-2 as the better candidate to handle the ICN PDU sessions to T-RAN. In addition, it also identifies ICN-AP-2 as the appropriate gateway for the affected ICN PDU sessions.
- o Step 4: SMF signals the details of the affected PDU sessions along with the traffic filter rules to switch the UL traffic from UL-CL-2 to ICN-AP-2 and DL flows from UL-CL-2 to T-RAN.

- o Step 5: SMF then signals ICN-SMF about the PDU session mobility change along with the information on UL-CL-2 for it to provision the tunnel between ICN-AP-2 and UL-CL-2.
- o Step 6: Based on the signaling received on the ICN PDU session, ICN-SMF identifies the affected gateways, i.e., ICN-AP-1 and ICN-AP-2: (i) ICN-SMF signals ICN-AP-2 about the affected PDU session information to update its DL tunnel information to UL-CL-2. Then, based on the ICN mobility solution, appropriate ICN mobility state to switch the future incoming Interests from ICN-AP-1 to UL-CL-2; (ii) ICN-SMF also signals ICN-AP-1 with the new forwarding label[mas] to forward the incoming Interest traffic to ICN-AP-2. This immediately causes the new Interest payload for the ICN-UE to be send to the new ICN gateway in a proactive manner.
- o Step 7: ICN-SMF then acknowledges SMF about the successful mobility update. Upon this, the SMF then acknowledges AMF about the state changes related to mobility request along with the tunnel information that is required to inter-connect T-RAN with UL-CL-2.
- o Step 8: AMF then updates the T-RAN PDU session state in order to tunnel ICN-UE's PDU sessions from T-RAN to UL-CL-2. This is followed by initiating the RAN resource management functions to reserve appropriate resources to handle the new PDU session traffic from the ICN-UE.
- o Step 9: AMF then signals the handover-ack message to the UE, signaling it to handover to the T-RAN.
- o Step 10: UE then issues a handover-confirm message to T-RAN. At this point, all the states along the new path comprising the T-RAN, UL-CL-2 and ICN-AP-2 is set to handle UL-DL traffic between the ICN-UE and the ICN-DN.
- o Step 11: T-RAN then signals the AMF on its successful connection to the ICN-UE. AMF then signals S-RAN to remove the allocated resources to the PDU session from the RAN and the tunnel state between S-RAN and UL-CL-1.
- o Step 12: AMF then signals SMF about the successful handover, upon which SMF removes the tunnel states from UL-CL-1. SMF then signals the ICN-SMF, which then removes the ICN mobility state related to the PDU session from ICN-AP-1. Also at this point, ICN-SMF can signal the ICN-NRS (directly or through ICN-AP-2) to update the UE-ID resolution information, which now points to ICN-AP-2 [mas].

Note that, inter-RAN handover mapping to the same UL-CL represents a special case of the above scenario.

7. Conclusion

In this draft, we explore the feasibility of realizing future networking architectures like ICN within the proposed 3GPP's 5GC architecture. Towards this, we summarized the design principles that offer 5GC the flexibility to enable new network architectures. We then discuss 5GC architecture along with the user/control plane extensions required to handle ICN PDU sessions formally. We then apply the proposed architecture to two relevant services that ICN networks can enable: first, mobile edge computing over ICN versus the traditional IP approach considering a connected car scenario, and argue based on architectural benefits; second, handling ICN PDU session mobility in ICN-DN rather than using IP anchor points, with minimal support from 5GC.

8. IANA Considerations

This document requests no IANA actions.

9. Security Considerations

This draft proposes extensions to support ICN in 5G's next generation core architecture. ICN being name based networking opens up new security and privacy considerations which have to be studied in the context of 5GC. This is in addition to other security considerations of 5GC for IP or non-IP based services considered in [TS33.899].

10. Acknowledgments

...

11. Informative References

[I-D.rahman-icnrg-deployment-guidelines]

Rahman, A., Trossen, D., Kutscher, D., and R. Ravindran, "Deployment Considerations for Information-Centric Networking (ICN)", draft-rahman-icnrg-deployment-guidelines-04 (work in progress), October 2017.

[I-D.suthar-icnrg-icn-lte-4g]

suthar, P., Stolic, M., Jangam, A., and D. Trossen, "Native Deployment of ICN in LTE, 4G Mobile Networks", draft-suthar-icnrg-icn-lte-4g-03 (work in progress), September 2017.

- [IEEE_Communications]
Trossen, D. and G. Parisis, "Designing and Realizing an Information-Centric Internet", Information-Centric Networking, IEEE Communications Magazine Special Issue, 2012.
- [Jacobson]
Jacobson, V. and et al., "Networking Named Content", Proceedings of ACM Context, , 2009.
- [lteversus5g]
Kim, J., Kim, D., and S. Choi, "3GPP SA2 architecture and functions for 5G mobile communication system.", ICT Express 2017, 2017.
- [mas]
Azgin, A., Ravindran, R., Chakraborti, A., and G. Wang, "Seamless Producer Mobility as a Service in Information Centric Networks.", 5G/ICN Workshop, ACM ICN Sigcomm 2016, 2016.
- [Ravindran]
Ravindran, R., Chakraborti, A., Amin, S., Azgin, A., and G. Wang, "5G-ICN : Delivering ICN Services over 5G using Network Slicing", IEEE Communication Magazine, May, 2016.
- [RFC7927]
Kutscher, D., Ed., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlich, "Information-Centric Networking (ICN) Research Challenges", RFC 7927, DOI 10.17487/RFC7927, July 2016, <<https://www.rfc-editor.org/info/rfc7927>>.
- [TS-5G NR]
3GPP-38-xxx, "Technical Specification series on 5G-NR (Rel.15)", 3GPP , 2017.
- [TS23.501]
3gpp-23.501, "Technical Specification Group Services and System Aspects; System Architecture for the 5G System (Rel.15)", 3GPP , 2017.
- [TS23.502]
3gpp-23.502, "Technical Specification Group Services and System Aspects; Procedures for the 5G System(Rel. 15)", 3GPP , 2017.

- [TS23.799] 3gpp-23.799, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Study on Architecture for Next Generation System (Rel. 14)", 3GPP , 2017.
- [TS33.899] 3gpp-33.899, "Study on the security aspects of the next generation system", 3GPP , 2017.
- [VSER] Ravindran, R., Liu, X., Chakraborti, A., Zhang, X., and G. Wang, "Towards software defined ICN based edge-cloud services", CloudNetworking(CloudNet), IEEE International Conference on, IEEE International Conference on CloudNetworking(CloudNet), 2013.

Authors' Addresses

Ravi Ravindran
Huawei Research Center
2330 Central Expressway
Santa Clara 95050
USA

Email: ravi.ravindran@huawei.com
URI: <http://www.Huawei.com/>

Prakash Suthar
Cisco Systems
9501 Technology Blvd.
Rosemont 50618
USA

Email: psuthar@cisco.com
URI: <http://www.cisco.com/>

Guoqiang Wang
Huawei Research Center
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: gq.wang@huawei.com

ICN Research Group
Internet-Draft
Intended status: Informational
Expires: January 17, 2018

R. Ravindran
A. Chakraborti
S. Amin
Huawei Technologies
J. Chen
Winlab, Rutgers University
July 16, 2017

Support for Notifications in CCN
draft-ravi-icnrg-ccn-notification-01

Abstract

This draft proposes a new packet primitive called Notification for CCN. Notification is a PUSH primitive and can be unicast or multicast to multiple listening points. Notifications do not expect a Content Object response hence only requires the use of FIB state in the CCN forwarder. Emulating Notification as a PULL has performance and routing implications. The draft first discusses the design choices associated with using current Interest/Data abstraction for achieving push and challenges associated with them. We follow this by proposing a new fixed header primitive called Notification and a CCN message encoding using Content Object primitive to transport Notifications. This discussion are presented in the context of CCNx1.0 [1] proposal. The draft also provides discussions on various aspects related to notification such as flow and congestion control, routing and reliability considerations, and use case scenarios.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 17, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Notification Requirements in CCN	3
3. Using Interest/Data Abstraction for PUSH	4
4. Proposed Notification Primitive in CCN	9
5. Notification Message Encoding	10
6. Notification Processing	12
7. Security Considerations	12
8. Annex	13
8.1. Flow and Congestion Control	13
8.1.1. Issues with Basic Notifications	13
8.1.2. Flow and Congestion Control Mechanisms	14
8.1.2.1. End-to-End Approaches	14
8.1.2.2. Hybrid Approaches	15
8.1.3. Receiver Reliability	17
8.2. Routing Notifications	18
8.3. Notification reliability	18
8.4. Use Case Scenarios	19
8.4.1. Realizing PUB/SUB System	19
9. Informative References	20
Authors' Addresses	22

1. Introduction

Notification is a PUSH primitive used in the Internet today by many IoT and social applications. The nature of notifications varies with the application scenario, ranging from being mission critical to one that is best effort. Notifications can be unicast or multicast depending on whether the notification service is aware of all the consumers or not. A notification service is preceded by a consumer subscribing to a specific event such as, subscription to hash-tag feeds, health emergency notification service, or temperature sensor

reading from a room in a building; following this subscription the service pushes notifications to consuming entities. It has to be noted that certain IoT applications expects notification end-to-end latency of few milliseconds [2]. Industrial IoT applications have more stringent requirement in terms of QoS, timeliness, and reliability of message delivery. Though we term it as a Notification, this primitive can also be used for transactional exchange between two points.

CCN optimizes networking around efficiently distributing already published content which the consumers learn through mechanisms like manifests containing the names of published content chunks and their locations. Applications relying on notifications requires event driven data to be pushed from multiple producers to multiple subscribers for which the current Interest/Data primitive is inefficient. This draft proposes to extend CCN's current primitives set with a new notification primitive that can be processed in a new way by the CCN forwarder to serve notification objectives. Notification here implies a PUSH semantic that is available with IP today and supported by other FIA architectures like MobilityFirst [3] and XIA [4].

2. Notification Requirements in CCN

General notification requirements and features have been discussed have been discussed in protocols such as CoAP's Observe proposal [5] to push notifications from the server to the clients. Here we discuss basic notification requirements from CCN's network layer perspective. Other requirements related to reliability, low latency, flow control can be engineered by the application or through more network layer state once the following requirements are met.

- o Supporting PUSH Intent: CCN should provide efficient and scalable support for PUSH, where application's intent is to PUSH content to listening application without expecting any data in return. Efficiency relates to minimizing control and forwarding overhead and scalability refers to support arbitrary number of producers and consumers participating in a general pub/sub or multicast service.
- o Multicast Support: CCN network should be able to handle multicast notifications from a producer to multiple consumers.
- o Security: Just as a content object in the context of Interest/Data primitive provides data authentication and privacy, similar features should also be offered by notification objects too.

- o Routing/Forwarding Support: Name prefixes over which multicast notifications are managed should be handled in a different manner from the name prefixes over which Interest/Data primitive is used for content distribution in order to support the PUSH intent. This differentiation applies to the control as well as the forwarding plane.
- o Minimizing Processing: Notification processing in the forwarder should be minimized considering the application's intent to PUSH data to listening consumers.

3. Using Interest/Data Abstraction for PUSH

Recent CCN and NDN research [6][7] have studied the problem of handling notifications and have proposed several solutions to handle this. Here, we discuss several of them and point out their benefits and issues:

Long-lived Interest v.1: The most intuitive solution makes the assumption that the consumers know exactly the names of the contents that will be published in the future. Yet, it is not easy since the providers can give arbitrary names to each piece of content, even though the contents might share a common prefix (i.e., GROUP_PREFIX). To make it feasible, the providers can publish the contents with sequential ID, e.g., /GROUP_PREFIX/SEQUENTIAL_ID[/SEGMENT_ID], so that the consumers can query the contents with names /GROUP_ID/item_1, /GROUP_ID/item_2, ... (each name represents a content item). The consumers can pipeline the requests (always keep some unsatisfied requests in flight, similar to TCP) to better utilize the network capacity.

However, this solution has several issues, especially in the multi-provider scenario:

- * Since it is unknown to the consumer (and the network) which provider will use which sequential ID, each request has to be forwarded to all the possible providers. This solution might use up a large amount of state (PIT entries) in the network, as each consumer can keep tens of requests (to all providers) in flight for each group.
- * Since each sequential ID should only be used by one provider, many PIT entries will not be consumed until timeout (if there is a timeout mechanism). E.g., P1 and P2 are 2 providers of a group (/GROUP), the consumers have to send requests /GROUP/item_1, and /GROUP/item_2 to both providers. Assume that P1 publishes first so he uses the name /GROUP/item_1. The PIT

entries for /GROUP/item_1 towards P2 will not be consumed since P2 should now publish with name /GROUP/item_2.

- * When the PIT entries form loops in the network (it can happen quite often in the multi-provider, multi-consumer scenario), the data packets can waste network traffic while following the loops and get discarded when redundancy happens.
- * Other than the inefficiencies mentioned above, one major issue with this solution is the difficulty of provider synchronization. It is not easy to make sure that different providers would use different sequential IDs especially when the providers are publishing contents at the same time.

Polling v.1: To eliminate the requirement for a sequential ID when publishing (to address the synchronization issue), the solution Polling v.1 makes the providers publish contents with name format: /GROUP_ID/TIMESTAMP. While querying the contents, the consumer query using name /GROUP_ID/ with "exclude" field <Earliest version after Tx>, where Tx is the latest version the consumer has received. E.g., after receiving a content with name /GROUP_ID/v_1234 (v_1234 is the timestamp of the publication time), the consumer would send a query with name /GROUP_ID/<Earliest after v_1234>. He might get the next piece with name /GROUP_ID/v_2345 (assuming that there is no content published between these two time stamps) without the need to know the exact names of the contents. The content providers do not have to be synchronized on the sequential IDs and use the timestamp instead.

While this solution is similar to the one used in NDN for getting the "latest" version under a prefix, it has several issues when we need to get "all" versions under a prefix:

- * Ambiguity contents will appear when two providers of a same group publish at the same time.
- * Consumers might miss messages when the clocks are not synchronized on the providers. E.g., one provider (with faster clock) might publish a content with name /GROUP_ID/v_2345 after v_1234. When the consumer queries for the earliest version after v_1234, he will get the content. Yet, another provider (with slower clock) would publish a content with name /GROUP_ID/v_2234 after the consumer gets v_2345. The consumer would miss the content with v_2234 as he will query for <Earliest after v_2345>.
- * Consumers might miss messages due to different delivery latency (e.g., cache hit vs. no cache hit) even when the clocks on the

providers are perfectly synchronized (e.g., via GPS signals). E.g., when a client queries for content /GROUP_ID/<Earliest after v_1234>, and there are two pieces of content exist in the network (v_2234, and v_2345). It can happen that v_2345 is returned earlier (either due to a cache hit or because the provider is closer). The consumer would then query for <Earliest after v_2345> and miss v_2234 with this solution.

- * Also just as with the previous approach, this mechanism also requires the producers to sync so that they don't produce content using the same name.

Long-lived Interest v.2: To completely address the issues with multiple providers sharing a same prefix (e.g., synchronization in Long-lived Interest v.1, and clock synchronization in Polling v.1), Long-lived Interest v.2 gives a prefix to each provider. The providers in this solution provide contents with name /GROUP_ID/PROVIDER_ID/SEQUENTIAL_ID, and the consumers query the full names accordingly (similar to Long-lived Interest v.1 but with an extra prefix PROVIDER_ID). The consumer can still use pipelining to improve the throughput.

While this solution can avoid packet losses in the previous solution, it has several other issues:

- * Consumers have to know all the potential providers, which might be difficult in some applications where every user can send messages in any group that he might be interested in.
- * Compared to Long-lived Interest v.1, the consumers in this solution have to keep multiple pending queries per group per provider. It might consume even more states in the network, which makes the solution less scalable.
- * When a provider has more than one device (e.g., laptop and smartphone) that can publish contents under a same name /GROUP_ID/PROVIDER_ID, the solution would have the same synchronization issue as Long-lived Interest v.1. If the solution mandates each device to have a separate provider ID, it will end up with even more PIT entries (states) in the network, and the solution becomes less "information-centric".

Polling v.2: To reduce the states and the control overhead in Long-lived Interest v.2, the solution Polling v.2 allows the provider process the requests in the application layer. Periodically, the consumer would query each provider "if there is any update after Nx" (Nx is name of the last content the consumer has received). The query would be in the format: /GROUP_ID/PROVIDER_ID/Nx/NONCE.

The provider would reply aggregated results in one response (with different segments, but under the same name), and an indication of "no update" if there is no publication after Nx. Since a same query for /GROUP_ID/PROVIDER_ID/Nx can get different responses ("no update", or aggregated publications), a NONCE has to be added in the name to prevent possible cache hits in the network. This solution can be effective in games since the publication rate (actions of the provider in the game) is much higher than the polling rate (refresh rate on the consumer). However, it still has some issues (inefficiencies):

- * There is a tradeoff between timeliness vs. in-network traffic when choosing the polling frequency. The solution can be inefficient when the polling is too frequent: most of the polling will get "no update" responses. This can consume a large amount of traffic in the network and extra computation on both the providers and the consumers. The timeliness can be impaired when the polling is infrequent since the publication can only reach the consumer when the consumer queries. The average delivery time of a publication in such solution is half of the polling period.
- * In-network cache cannot be used since the response to a same query (without nonce) can be different according to the time (and maybe the consumer).
- * Consumers still have to know all the potential providers similar to Long-lived Interest v.2.

Polling with A Server: To relieve the consumers from knowing all potential providers in Polling v.2, solution Polling with A Server introduces a server (or broker) as the delegate of all the providers. The providers would publish data into the server and the consumers would poll for the updates from the server (similar to Twitter and Facebook in IP network). In this solution, the consumers do not have to poll each provider for the updates, which reduces the overhead in the network. With the aggregated response on the server, the network traffic is further reduced. However, it still has several issues:

- * Similar to all the server-based solutions like Facebook and Twitter, the server has to deal with all the polls. This can cause single point of failure.
- * It is not easy for the providers "publish contents to the server". This becomes another notification problem and has to be solved by the other solutions mentioned in this section.

- * Cache is not used in this solution similar to Polling v.2.
- * This solution is not really "information-centric" as the consumers have to get the location of the content rather than the content itself.

Interest Overloading: Since all the aforementioned query/response solutions have issues with efficiency, scalability and/or timeliness, Interest Overloading tries to modify the communication pattern by using Interest packets to deliver publications directly. The consumers in this solution propagate FIB entry of /GROUP_ID to all potential providers (or simply flood the network). When a provider sends a publication, he would send an Interest with name /GROUP_ID/NONCE/<Payload> and the lifetime set to zero. Since the traditional Interest packets do not have payload, the solution has to embed (e.g., URL encode [1]) the payload in the name of the Interest. NONCE is used to prevent PIT aggregation since providers may publish contents with same payload (e.g., sensor readings). This solution can address the timeliness and scalability issues with the Polling and Long-lived Interest solutions, yet there are still some issues:

- * This solution creates ambiguity in the meaning of Interest packets (and the corresponding forwarding behaviors on the routers). For a normal Interest packet, the forwarding engines should perform an anycast (send it to only one of the providers) according to FIB. However, in this solution, the forwarding engines should use multicast logic for prefix /GROUP_ID (and avoid PIT storage). Solution in [8] specifies some multicast prefixes so that the forwarding engines can distinguish the publications from the normal requests. Yet, this places higher overhead on both the forwarding engines and the network management. It also prevents providers to create contents under the /GROUP_ID prefix (since the query will be forwarded using multicast, and not kept in the PIT).
- * The routing is also a concern in this solution. When the consumers propagate FIB, it should reach all potential providers (in most of the time it will flood the network since all the users can be potential providers). Naturally, in a multi-provider, multi-consumer scenario, the FIB entries would form a mesh in the network. It is less scalable compared to the tree-based routing in IP multicast (PIM-SM). The network has to specify another routing policy specifically for these prefixes, which places even higher overhead on network management.

- * As is mentioned in [9], it is not efficient to embed large amount of data into the name of the Interest packets. It adds more computation and storage overhead in the forwarding engines (PITs).

Interest Trigger: Similar to Interest Overloading, Interest Trigger uses an Interest packet as notification. To eliminate the overhead of embedding the content in the Interest, this solution places the name of the publication in the name of the notification (Interest) packet. On receiving the notification, the consumers can extract the content name and send another query (Interest) for the real content. While this solution reduces the overhead of embedding the payload, it still has the ambiguity and routing issues similar to Interest Overloading solution. It also incurs additional round trip delay before the produced data arrives at the listening consumer.

To summarize CCN and NDN operates on PULL primitive optimized for content distribution applications. Emulating PUSH operation over PULL has the following issues:

- o It is a mismatch between an application's intent to PUSH data and the PULL APIs currently available.
- o Unless Interests are marked distinctly, overloading Interests with notification data will undergo PIT/CS processing and are also subjected to similar routing and forwarding policies as regular Interests which is inefficient.
- o Another concern in treating PUSH as PULL is with respect to the effect of local strategy layer routing policies, where the intent to experiment with multiple faces to fetch content is not required for notification messages.

This motivates the need for treating notifications as a separate class of traffic which would allow a forwarder to apply the appropriate routing and forwarding processing in the network.

4. Proposed Notification Primitive in CCN

Notification is a new type of packet hence can be subjected to different processing logic by a forwarder. By definition, a notification message is a PUSH primitive, hence is not subjected to PIT/CS processing. This primitive can also be used by any other transactional or content distribution application towards service authentication or exchanging contextual information between end points and the service.

5. Notification Message Encoding

The wire packet format for a Notification is shown in Fig. 1 and Fig. 2. Fig. 1 shows the Notification fixed header considering the CCNx1.0 encoding, and Fig. 2 shows the format for the CCN Notification message, which is used to transport the notification data. We next discuss these two packet segments of the Notification message.

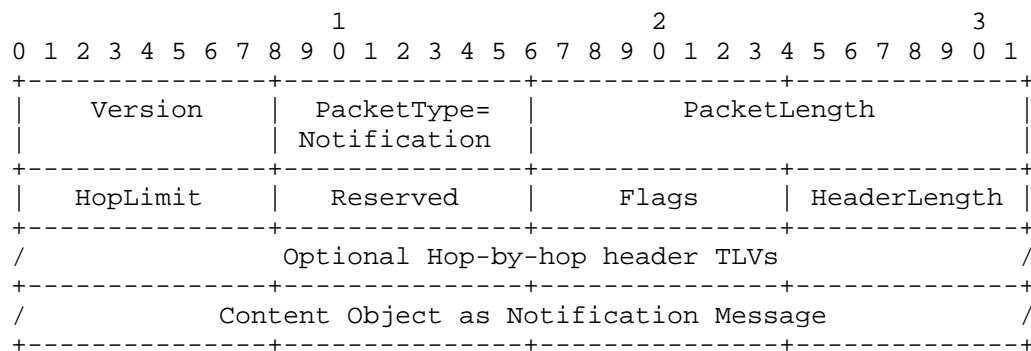


Figure 1: CCN Notification fixed header

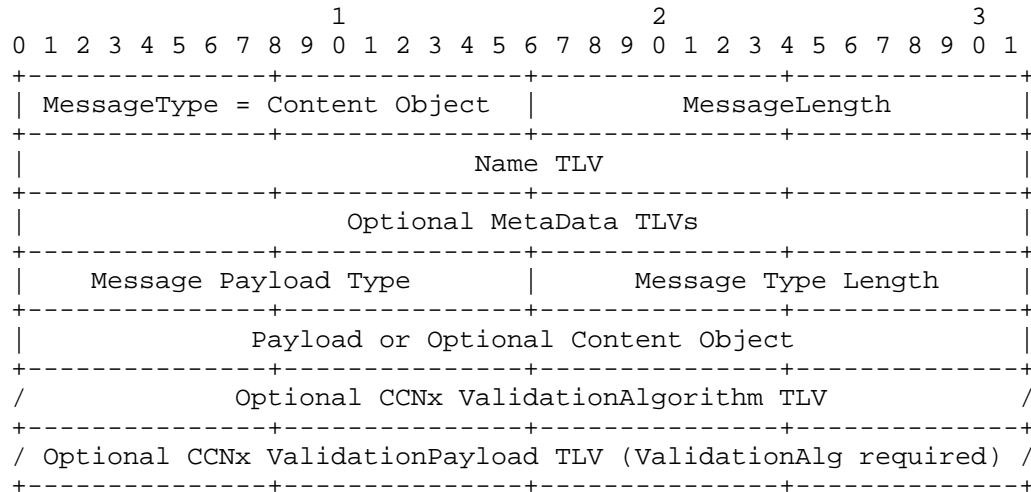


Figure 2: CCN Notification Message

Notification Fixed Header: The fields in the fixed header that have new meaning in the context of notifications are discussed next, while the other fields follow the definition in [1].

- o Packet Type: This new type code identifies that the packet is of type Notification [TBD].
- o Optional Hop-by-hop header TLVs : Encodes any new hop-by-hop headers relevant to notifications [TBD].

CCN Notification message: The CCN Notification message is a Content Object as in [1]. Notifications are always routed on the top level Content Object (outer CO) name. Notification itself can be encoded in two forms depending on the application requirement:

- o Notification with single name: In this case the notification contains a single content object. Here the producer generates notification using the same name used by consumers on which they listen on.
- o Notification with two names: In this case the notification contains a top level Content Object (outer CO), that encapsulates another Content Object (inner CO). With an encapsulated Content Object, the meaning is that notification producers and consumers operate on different name-spaces requiring separate name-data security binding. A good application of the encapsulation format is a PUB/SUB service, where the consumer learns about the notification service name offline, and the producer who is decoupled from the consumer generates a new Content Object using its own name and pushes the notification to the consumer.

The interpretation of the fields shown in Fig. 2 are as follows:

- o MessageType : The CCN message type is of type Content Object.
- o Name TLV : Name TLV in the Content Object is used to route the Notification.
- o Optional Metadata TLV: These TLVs carry metadata used to describe the Notification payload.
- o Message Payload Type: This is of type T_PAYLOADTYPE defined in CCNx.1.0 or a new encapsulation type (T_ENCAP) that indicates the presence of another encapsulated Content Object [TBD].
- o Optional Encapsulated Content Object: This is an optional encapsulated Content Object newly defined for the Notification primitive. The name in the encapsulated Content Object

corresponds to the producer's name-space, or anything else based on the application logic. The rationale for an encapsulated Content Object was discussed earlier.

- o Optional Security Validation data: The Content Object optionally carries security validation payload as per CCNx1.0.

6. Notification Processing

The following steps are followed by a CCN forwarder to process the Notification packet.

- o Notification packet type is identified in the fixed header of a CCN packet with a new type code. The Notification carries a Content Object, whose name is used for routing. This name is matched against the FIB entries to determine the next hop(s). Novel strategy layer routing techniques catering to the notification traffic can be applied here.
- o CCN forwarder also processes the optional metadata associated with the Notification meant for the network to help with the forwarding strategy, for e.g., mission critical notifications can be given priority over all other traffic.
- o As mentioned earlier, CCN forwarder MUST NOT cache the Content Objects in the notifications.

7. Security Considerations

The proposed processing logic of Notifications that bypass the processing of PIT/CS has the following security implications:

Flow Balance : PIT state maintains the per-hop flow balance over all the available faces by enforcing a simple rule, that is, one Content Object is sent over a face for a single Interest. Bypassing PIT processing compromises this flow balancing property. For scenarios where the notification traffic volume is not high such as for IoT applications, the impact may not be significant. However, this may not be the case considering the plethora of social networking and emerging IoT applications in a general Internet scenario. This flow balance tradeoff has to be understood considering an application's intent to PUSH data and the latency introduced by processing such traffic if a PULL primitive is used. Also PIT offers a natural defense mechanism by throttling traffic at the network edge, considering the provisioned PIT size, and bypassing it could exacerbate DDOS attacks on producing end points.

Cache Poisoning: This draft doesn't recommend the caching of the Content Object in the Notification payload, though doing so might help in increasing the availability of notification information in the network. A possible exception would be if the inner CO is a nameless object [10]. as those can only be fetched from CS by hash We leave this possibility of applying policy-based caching of Notification Content Objects for future exploration. The recommendation for not caching these Content objects is that, in a regular Interest/Content Object exchange, content arrives at the forwarder and is cached as a result of per-hop active Interest expression. Unsolicited Content Objects, as in the case of the Notification, violates this rule, which could be exploited by malicious producers to generate DDOS attack against the cache resource of a CCN infrastructure.

8. Annex

8.1. Flow and Congestion Control

8.1.1. Issues with Basic Notifications

As mentioned in the previous sections, one of the main issues with notification is the flow and congestion control. One naive way to solve this issue is the routers drop the packets from aggressive flows. Flow-based fair queueing (and its variation stochastic fairness queueing) maintain queues for flows (or the hash of flows) and try to give a fair share to each flow (or a hash). Flows can be classified by the prefixes in the ICN case. However, according to [11], the overall network throughput will be affected when there are multiple bottlenecks in the network. Therefore, [11] promotes an end-to-end solution for congestion control. Flow balance is a key requirement to an end-to-end (or end-driven) flow and congestion control. In the case of CCN query/response, flow balance entails that an Interest pulls at most one Data object from upstream. The data consumer can therefore control the amount of traffic coming from the data source(s) either it is a data provider or a cache in the network. However, the basic notification does not follow the rule of flow balance (each Subscription can result in more than one Notifications disseminated in the network). In the absence of a proper feedback mechanism to notify the data sender or the network the available bandwidth and local resource the consumer has, the sender can easily congest the bottleneck link of the receivers (causing congestion collapse) and/or overflow the buffer on the receiver side. In the later sections, we will describe the possible congestion control mechanisms in ICN and how to deal with packet loss when both congestion control and reliability are required.

However, the basic notification does not follow the rule of flow balance (each Subscription can result in more than one Notifications disseminated in the network). There is no way a receiver can notify the data sender or the network the available bandwidth and local resource it has. As a result, the sender can easily congest the bottleneck link of the receivers (causing congestion collapse) and/or overflow the buffer on the receiver side.

8.1.2. Flow and Congestion Control Mechanisms

Here we discuss broad approaches towards achieving flow and congestion control in CCN as applied to Notification traffic. Since the forwarding logic of the Notification packets are quite similar to that of IP multicast, existing multicast congestion control solutions can be candidates to solve the flow/congestion control issue with Notification. In addition we also summarize recent ICN research to address this issue.

8.1.2.1. End-to-End Approaches

In the multicast communication, it is not scalable to have direct receiver-to-sender feedback loop similar to TCP since this would result in each receiver sending ACKs (or NACKs) to the data sender and cause ACK (NACK) implosion. To address the ACK implosion issue, two types of solutions have been proposed in multicast congestion control, namely, sender-driven approaches and receiver-driven approaches.

8.1.2.1.1. Sender-driven Multicast

In the first category, the sender controls the sending rate and to ensure the network friendliness, the sender usually align the sending rate to the slowest receiver.

To avoid the ACK implosion issue, TCP-Friendly Multicast Congestion Control (TFMCC [12]) uses rate based solution. This solution uses TCP-Friendly Rate Control (TFRC) to get a proper sending rate based on the RTT between sender and each receiver. The sender only needs to collect the RTTs periodically instead of per-packet ACKs. Similarly, in ICN, the sender can create another channel (namespace) to collect the RTT measurement from the receivers. However, due to the dynamics on each path, it is difficult to calculate the proper sending rate.

To address the rate calculation issue, pgmcc [13], a window-based solution is proposed. It uses NACKs to detect the slowest receiver (the ACKer). The ACKer sends an ACK back to the sender on receiving each multicast packet. A feedback loop similar to TCP is formed

between the sender and the ACKer to control the sending rate. Since the ACKer is the slowest receiver, the sender adapts its sending rate to the available bandwidth of the slowest receiver, the solution can therefore ensure the network friendliness. In the ICN case, the receivers can send NACKs in the form of Notification packets through another namespace, and the ACKer can also use the same mechanism to send ACKs.

However, since the sender is always aligning the sending rate to the slowest receiver to ensure the network friendliness, the performance of the solutions can be dramatically affected by a very slow receiver.

8.1.2.1.2. Receiver-driven Multicast

Unlike the sender-driven solutions, the receiver-driven solutions [14] choose to use layered-multicast to satisfy heterogeneous receivers. The sender first initiates several multicast groups (namespaces in the case of ICN) with different sending rates. Each receiver would choose to join a multicast group with the highest sending rate that it can afford. The sender can also adapt the sending rate of each multicast group according to the receiver status.

These solutions can support applications like video streaming (with layered codecs) efficiently. However, they also have some issues: 1) they complicate the sender and receiver logic, especially for simple applications like file transfer; and 2) the receivers are limited by the sending rates initiated by the provider and would therefore under-utilize the available bandwidth.

8.1.2.2. Hybrid Approaches

In this approach, flow balance of Notification is achieved by the receivers notifying the network (rather than the sender or other receivers) about the capacity it can receive. Here, we take advantage of operating the Notification service through a receiver-driven approach and get support from the network.

A solution based on this approach is proposed in [15], which we summarize next.

To retain flow balance, the consumers in this solution send out one subscription for only one next Notification instead of the original logic (that receives all the Notifications). Similar to the flow and congestion control in query/response, the receivers can now maintain a congestion window to control the amount of traffic coming from upstream.

Here, instead of maintaining a (name, outgoing face) pair in FIB (or subscription table), the routers now adds a third field -- accumulated count -- for each entry. The accumulated count is increased by 1 on receiving such a subscription and decreased by 1 on sending a Notification to that face. The routers should also propagate the maximum accumulated count upstream till the 1st hop router of the provider (or the rendezvous point in the network). The subscribers sends a subscription for every successfully received notification. Here we also assume that, the subscribers operate based on the AIMD scheme.

If the dissemination of Notification follows a tree topology in the network, we define the branching point of a receiver R (BP_R) as the router closest to R which has another outgoing face that can receive data faster than R. For receivers that has bandwidth/resources to receive all the data from the provider, BP_R is the 1st hop router of the provider (or the rendezvous point).

In this solution, we can prove that there is a feedback loop between each receiver and its branching point. Therefore, when a receiver maintains its congestion window size using AIMD, the traffic between the branching point and the receiver is similar to TCP. It can get a fair share at the bottleneck on the path, even if the bottleneck is not directly under the branching point. In the multicast tree, the solution can ensure the fairness with other (TCP-like) flows on each branch.

The solution can thus allow the sender to send at an application-efficient rate rather than being affected by the slowest receiver like pgmcc [13].

It is true that the solution requires more packets and more states in the network compared to the basic notification solution, but the cost is similar to (and smaller than) that of query/response. Since we are using one notification per subscription pattern, the amount of traffic overhead is the same as query/response. As for the states stored in the router, the solution only requires 1 entry per prefix per face, which is smaller than the query/response which requires 1 entry per packet per face. Therefore, the overhead of the solution is acceptable in CCN.

8.1.2.2.1. Other Challenges

- o Sender Rate Control: The sender in the solution does not have to limit the sending rate to the slowest receiver to maintain network friendliness. Therefore, the choice of sending rate is a tradeoff between network traffic and session completion time. In the case where the application does not require a certain sending rate

(like file transfer), the sender can align the sending rate to the slowest receiver (similar to pgmcc) to minimize the repair traffic, but at the cost of longer session completion time. He can also send at the rate of the fastest receiver and try to get peer repair in the network. This allows faster receivers finish the session earlier but causing higher network traffic due to the repair. An ACKer-based solution similar to pgmcc can be adopted to allow the sender align the rate at a proportion of users (e.g., top 30%). The sender can collect feedback (throughput, latency, etc.) from all the receivers periodically and pick an ACKer according to the proportion it desires. On receiving a Notification packet, the ACKer would send an ACK just like TCP. The sender can maintain a congestion window also like TCP. The feedback loop between the sender and the ACKer can align the sending rate at the ACKers's available bandwidth.

- o Receiver Window Control: Slightly different from one-sender one-receiver window control in TCP, the sending rate in the hybrid approach is not controlled by any of the receivers. Receiving intermittent packets can indicate both congestion (similar to TCP) and not enough window size (since the sending rate is higher). In the first case, the receiver should reduce the window size while in the second case, the receiver should increase the window size. An indication of congestion (e.g., Random Early Detection, RED) should be provided directly from the network. The receivers with available bandwidth higher than the sending rate would have too large window size since it does not see any packet loss. Please refer to [15] for a detailed solution on this issue.

8.1.3. Receiver Reliability

The receiver would miss packets when the available bandwidth/resource of the receiver is lower than the sending rate of the Notification provider. Some applications (like gaming and video conferencing) can tolerate such kind of packet loss while the others (like file transfer) cannot. Therefore, another module that ensures the reliability is needed. However, reliability should be separated from the flow and congestion control since it is not a universal requirement.

With the solution described in the receiver-driver or the hybrid approach, the slower consumers would receive intermittent packets since the sending rate can be faster than their fair share. The applications that require reliable transfer can query the missing packets similar to the normal query/response. This also requires that each content in the Notifications should have a unique Content Name (or hash in the nameless scenario). The clients should also be able to detect the missing packets either based on the sequence

number or based on a pre-acquired meta-file. Caching in CCN can be leveraged to achieve availability and reliability.

The network can forward the requests (Interests) of the missing packets towards the data provider, the other consumers and/or the in-network cache to optimize the overall throughput of the consumers. This solution is similar to Scalable Reliable Multicast (SRM [16]). However, as mentioned in [17], solutions like SRM requires the consumers communicate directly with each other and therefore lose the privacy and trust. CCN can ensure the privacy since the providers cannot get the information of the identity of the consumers. Trust (data integrity) is also maintained with the signature in the Data packets.

8.2. Routing Notifications

Appropriate routing policies should be employed to ensure reliable forwarding of a notification to its one or many intended receivers. The name in the notification identifies a host or a multicast service being listened to by the multiple intended receivers. Two types of routing strategies can be adopted to handle notifications, depending on whether or not an explicit pub/sub state is maintained in the forwarder.

- o Stateless forwarding: In this case the notification only relies on the CCN FIB state to route the notification. The FIB entries are populated through a routing control plane, which distinguishes the FIB states for the notification service from the content fetching FIB entries. Through this logical separation, Notifications can be routed by matching its name with the matching FIB policy in the CCN forwarder, hence processed as notification multicast.
- o Stateful forwarding: In this case, specific subscription state is managed in the forwarder to aid notification delivery. This is required to scale notifications at the same time apply notification policies, such as filter notifications or to improve notification reliability and efficiency to subscribing users [18].

8.3. Notification reliability

This proposal doesn't provide any form of reliability. Reliability can be realized by the specific application using the proposed notification primitive, for instance using the following potential approaches:

Caching: This proposal doesn't propose any form of caching. But caching feature can be explored to improve notification reliability, and this is a subject of future study. For instance, consumers,

which expect notifications and use external means (such as periodic updates or by receiving manifests) to track notifications, can recover the lost notifications using the PULL feature of CCN.

Notification Acknowledgment: If the producer maintains per-receiver state, then the consumer can send back notification ACK or NACK to the producer of having received or not received them.

8.4. Use Case Scenarios

Here we provide the discussions related to the use of Notification in different scenarios.

8.4.1. Realizing PUB/SUB System

A PUB/SUB system provides a service infrastructure for subscribers to request update on a set of topics of interest, and with multicast publishers publishing content on those topics. A PUB/SUB system maps the subscribers' interests to published contents and pushes them as Notifications to the subscribers. A PUB/SUB system has many requirements as discussed in [19] which include low latency, reliability, fast recovery, scalability, security, minimizing false (positive/negative) notifications.

Current IP based PUB/SUB systems suffer from interoperability challenges because of application-defined naming approach and lack of support of multicast in the data plane. The proposed Notification primitive can be used to realize large scale PUB/SUB system, as it unifies naming in the network layer and support for name-based multicasting.

Depending on the routing strategy discussed earlier, two kind of PUB/SUB approaches can be realized : 1) Rendezvous style approach ; 2) Distributed approach. Each of these approaches can use the Notification primitive to implement their PUSH service.

In the Rendezvous style approach, a logically centralized service maps subscriber's topic interest with the publisher's content and pushes it as notifications. If stateless forwarding is used, the routing entries contain specific application-ID's requesting a given notification, to handle scalability, a group of these application can share a multicast-ID reducing the state in the FIB.

In the Distributed approach, the CCN/NDN protocol is further enhanced with new subscription primitive for the subscription interested consumers. When a consumer explicitly subscribes to a multicast topic, its subscription request is forwarded to the upstream forwarder which manages this state mapping between subscription names

to the downstream faces which has expressed interest for Notifications being pushed under that prefix. An example of the network layer based approach is the COPSS notification proposal [19]. Here a PUB/SUB multi-cast state state, called the subscribers interest table, is managed in the forwarders. When a Notification arrives at a forwarder, the content descriptor in the notification is matched to the PUB/SUB state in the forwarder to decide the faces over which the Notification has to be forwarded.

9. Informative References

- [1] CCN Wire format, CCNX1., "<http://www.ietf.org/id/draft-mosko-icnrg-ccnxmessages-00.txt>", 2013.
- [2] Osseiran, A., "Scenarios for 5G Mobile and Wireless Communications: The Vision of the METIS Project.", IEEE Communication Magazine , 2014.
- [3] NSF FIA project, MobilityFirst., "<http://www.nets-fia.net/>", 2010.
- [4] NSF FIA project, XIA., "<https://www.cs.cmu.edu/~xia/>", 2010.
- [5] Observing Resources in CoAp, observe., "<https://tools.ietf.org/html/draft-ietf-core-observe-16>", 2015.
- [6] Amadeo, M., Campolo, C., and A. Molinaro, "Internet of Things via Named Data Networking: The Support of Push Traffic", Network of the Future (NOF), 2014 International Conference and Workshop on the , 2014.
- [7] Shang, W., Bannis, A., Liang, T., and Z. Wang, "Named Data Networking of Things.", IEEE IoTDI 2016, 2016.
- [8] Zhu, Z. and A. Afanasyev, "Let's chronosync: Decentralized dataset state synchronization in named data networking", The 21st IEEE International Conference on Network Protocols ICNP, 2013.
- [9] Moiseenko, I. and O. Oran, "TCP/ICN: Carrying TCP over Content Centric and Named Data Networks", Proceedings of the 3rd ACM Conference on Information-Centric Networking ICN, 2016.
- [10] Mosko, M., "Nameless Objects.", IETF/ICNRG, Paris Interim 2016, 2016.

- [11] Floyd, S. and F. Kevin, "Promoting The Use of End-to-End Congestion Control in The Internet.", IEEE ToN vol. 7(4), pp. 458-472, 1999.
- [12] Widmer, J. and M. Handley, "TCP-Friendly Multicast Congestion Control (TFMCC): Protocol Specification.", IETF RFC 4654, 2006.
- [13] Rizzo, L., "pgmcc: A TCP-Friendly Single-Rate Multicast Congestion Control Scheme.", SIGCOMM CCR vol. 30.4, pp. 17-28, 2000, 2000.
- [14] McCanne, S., Jacobson, V., and M. Vetterli, "Receiver-driven Layered Multicast.", SIGCOMM CCR pp. 117-130, 1996.
- [15] Chen, J., Arumaithurai, M., Fu, X., and KK. Ramakrishnan, "SAID: A Control Protocol for Scalable and Adaptive Information Dissemination in ICN.", arXiv vol. 1510.08530, 2015.
- [16] Floyd, S., Jacobson, V., Liu, C., McCanne, S., and L. Zhang, "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing.", IEEE TON vol. 5(6), pp. 784-803, 1997.
- [17] Floyd, N., Grossglauser, M., and KK. Ramakrishnan, "Distrust and Privacy: Axioms for Multicast Congestion Control.", Distrust and Privacy: Axioms for Multicast Congestion Control NOSSDAV, 1999.
- [18] Francois et al, J., "CCN Traffic Optimization for IoT", Proc. of NoF , 2013.
- [19] Chen, J., Arumaithurai, M., Jiao, L., Fu, X., and K. Ramakrishnan, "COPSS: An Efficient Content Oriented Publish/Subscribe System.", ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS 2011) , 2011.
- [20] DNS Security Introduction and Requirements, DNS-SEC., "<http://www.ietf.org/rfc/rfc4033.txt>", 2005.
- [21] Cisco System Inc., CISCO., "Cisco visual networking index: Global mobile data traffic forecast update.", 2009-2014.
- [22] CCNx Label Forwarding, CCNLF., "<http://www.ccnx.org/pubs/ccnx-mosko-labelforwarding-01.txt>", 2013.

Authors' Addresses

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

Email: ravi.ravindran@huawei.com

Asit Chakraborti
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: asit.chakraborti@huawei.com

Syed Obaid Amin
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: obaid.amin@huawei.com

Jiacheng Chen
Winlab, Rutgers University
671, U.S 1
North Brunswick, NJ 08902
USA

Email: jiachen@winlab.rutgers.edu

ICN Research Group
Internet-Draft
Intended status: Informational

Prakash Suthar
Milan Stolic
Anil Jangam
Cisco Systems
Dirk Trossen
InterDigital Inc.
November 12, 2017

Expires: May 16, 2018

Native Deployment of ICN in LTE, 4G Mobile Networks
draft-suthar-icnrg-icn-lte-4g-04

Abstract

LTE, 4G mobile networks use IP based transport for control plane to establish the data session and user plane for actual data delivery. In existing architecture, IP transport used in user plane is not optimized for data transport, which leads to an inefficient data delivery. IP unicast routing from server to clients is used for delivery of multimedia content to User Equipment (UE), where each user gets a separate stream. From bandwidth and routing perspective this approach is inefficient. Multicast and broadcast technologies have emerged recently for mobile networks, but their deployments are very limited or at an experimental stage due to complex architecture and radio spectrum issues. ICN is a rapidly emerging technology with built-in features for efficient multimedia data delivery, however majority of the work is focused on fixed networks. The main focus of this draft is on native deployment of ICN in cellular mobile networks by using ICN in 3GPP protocol stack. ICN has an inherent capability for multicast, anchorless mobility, security and it is optimized for data delivery using local caching at the edge. The native ICN (it runs directly on cellular layer 2 protocols like PDCP/RLC/MAC/L1) or dual stack (along with IP) deployment will bring all inherent benefits and help in optimizing mobile networks.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference

material or to cite them other than as "work in progress".

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Introduction	4
1.1	Terminology	4
1.2	3GPP Terminology and Concepts	4
2.	LTE, 4G Mobile Network	8
2.1	Network Overview	8
2.2	QoS Challenges	9
2.3	Data Transport Using IP	10
2.4	Virtualizing Mobile Networks	11
3.	Data Transport Using ICN	11
4.	ICN Deployment in 4G and LTE Networks	14
4.1	General ICN Deployment Considerations	14
4.2	ICN Deployment Scenarios	14
4.3	ICN Deployment in LTE Control Plane	17
4.4	ICN Deployment in LTE User Plane	18
4.4.1	Dual stack ICN Deployments in UE	19
4.4.2	Native ICN Deployments in UE	22
4.5	ICN Deployment in eNodeB	23
4.6	ICN Deployment in Packet Core (SGW, PGW) Gateways	24
5.	Security Considerations	26
6.	Summary	27

7	References	29
7.1	Normative References	29
7.2	Informative References	30
	Authors' Addresses	31

1 Introduction

LTE mobile technology is built as all-IP network. It uses IP routing protocols such as OSPF, ISIS, BGP etc. to establish network routes to route the data traffic to end user's device. Stickiness of IP address to a device is the key to get connected to a mobile network and the same IP address is maintained through the session until the device gets detached or moves to another network.

One of the key protocols used in 4G and LTE networks is GPRS Tunneling protocol (GTP). GTP, DIAMETER and other protocols are built on top of IP. One of the biggest challenges with IP based routing is that it is not optimized for data transport although it is the most efficient communication protocol. By native implementation of Information Centric Networking (ICN) in 3GPP, we can re-architect mobile network and optimize its design for efficient data transport by leveraging the caching feature of ICN. ICN also offers an opportunity to leverage inherent capabilities of multicast, anchorless mobility management, and authentication. This draft provides insight into different options for deploying ICN in mobile networks and how they impact mobile providers and end-users.

1.1 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].

1.2 3GPP Terminology and Concepts

Access Point Name

The Access Point Name (APN) is a Fully Qualified Domain Name (FQDN) and resolves to a set of gateways in an operator's network. APN identifies the packet data network (PDN) that a mobile data user wants to communicate with. In addition to identifying a PDN, an APN may also be used to define the type of service, QoS and other logical entities inside GGSN, PGW.

Control Plane

The control plane carries signaling traffic and is responsible for routing between eNodeB and MME, MME and HSS, MME and SGW, SGW and PGW etc. Control plane signaling is required to authenticate and authorize UE and establish mobility session with mobile gateways (SGW/PGW). Functions of the control plane also include system configuration and management.

Dual Address PDN/PDP Type

The dual address Packet Data Network/Packet Data Protocol (PDN/PDP) Type (IPv4v6) is used in 3GPP context in many cases as a synonym for dual-stack, i.e. a connection type capable of serving both IPv4 and IPv6 simultaneously.

eNodeB

The eNodeB is a base station entity that supports the Long-Term Evolution (LTE) air interface.

Evolved Packet Core

The Evolved Packet Core (EPC) is an evolution of the 3GPP GPRS system characterized by a higher-data-rate, lower-latency, packet-optimized system. The EPC comprises some of the sub components of the EPS core such as Mobility Management Entity (MME), Serving Gateway (SGW), Packet Data Network Gateway (PDN-GW), and Home Subscriber Server (HSS).

Evolved Packet System

The Evolved Packet System (EPS) is an evolution of the 3GPP GPRS system characterized by a higher-data-rate, lower-latency, packet-optimized system that supports multiple Radio Access Technologies (RATs). The EPS comprises the EPC together with the Evolved Universal Terrestrial Radio Access (E-UTRA) and the Evolved Universal Terrestrial Radio Access Network (E-UTRAN).

Evolved UTRAN The Evolved UTRAN (E-UTRAN) is a communications network, sometimes referred to as 4G, and consists of eNodeBs (4G base stations). The E-UTRAN allows connectivity between the User Equipment and the core network.

GPRS Tunnelling Protocol

The GPRS Tunnelling Protocol (GTP) [TS.29060] [TS.29274] [TS.29281] is a tunnelling protocol defined by 3GPP. It is a network-based mobility protocol and is similar to Proxy Mobile IPv6 (PMIPv6). However, GTP also provides functionality beyond mobility, such as in-band signaling related to Quality of Service (QoS) and charging, among others.

Gateway GPRS Support Node

The Gateway GPRS Support Node (GGSN) is a gateway function in the

GPRS and 3G network that provides connectivity to the Internet or other PDNs. The host attaches to a GGSN identified by an APN assigned to it by an operator. The GGSN also serves as the topological anchor for addresses/prefixes assigned to the User Equipment.

General Packet Radio Service

The General Packet Radio Service (GPRS) is a packet-oriented mobile data service available to users of the 2G and 3G cellular communication systems -- the GSM -- specified by 3GPP.

Home Subscriber Server

The Home Subscriber Server (HSS) is a database for a given subscriber and was introduced in 3GPP Release-5. It is the entity containing the subscription-related information to support the network entities actually handling calls/sessions.

Mobility Management Entity

The Mobility Management Entity (MME) is a network element that is responsible for control-plane functionalities, including authentication, authorization, bearer management, layer-2 mobility, etc. The MME is essentially the control-plane part of the SGSN in the GPRS. The user-plane traffic bypasses the MME.

Public Land Mobile Network

The Public Land Mobile Network (PLMN) is a network that is operated by a single administration. A PLMN (and therefore also an operator) is identified by the Mobile Country Code (MCC) and the Mobile Network Code (MNC). Each (telecommunications) operator providing mobile services has its own PLMN.

Policy and Charging Control

The Policy and Charging Control (PCC) framework is used for QoS policy and charging control. It has two main functions: flow-based charging, including online credit control and policy control (e.g., gating control, QoS control, and QoS signaling). It is optional to 3GPP EPS but needed if dynamic policy and charging control by means of PCC rules based on user and services are desired.

Packet Data Network

The Packet Data Network (PDN) is a packet-based network that

either belongs to the operator or is an external network such as the Internet or a corporate intranet. The user eventually accesses services in one or more PDNs. The operator's packet core networks are separated from packet data networks either by GGSNs or PDN Gateways (PGWs).

Serving Gateway

The Serving Gateway (SGW) is a gateway function in the EPS, which terminates the interface towards the E-UTRAN. The SGW is the Mobility Anchor point for layer-2 mobility (inter-eNodeB handovers). For each UE connected with the EPS, at any given point in time, there is only one SGW. The SGW is essentially the user-plane part of the GPRS's SGSN.

Packet Data Network Gateway

The Packet Data Network Gateway (PGW) is a gateway function in the Evolved Packet System (EPS), which provides connectivity to the Internet or other PDNs. The host attaches to a PGW identified by an APN assigned to it by an operator. The PGW also serves as the topological anchor for addresses/prefixes assigned to the User Equipment.

Packet Data Protocol Context

A Packet Data Protocol (PDP) context is the equivalent of a virtual connection between the User Equipment (UE) and a PDN using a specific gateway.

Packet Data Protocol Type

A Packet Data Protocol Type (PDP Type) identifies the used/allowed protocols within the PDP context. Examples are IPv4, IPv6, and IPv4v6 (dual-stack).

Serving GPRS Support Node

The Serving GPRS Support Node (SGSN) is a network element that is located between the radio access network (RAN) and the gateway (GGSN). A per-UE point-to-point (p2p) tunnel between the GGSN and SGSN transports the packets between the UE and the gateway.

Terminal Equipment

The Terminal Equipment (TE) is any device/host connected to the Mobile Terminal (MT) offering services to the user. A TE may communicate to an MT, for example, over the Point to Point

Protocol (PPP).

UE, MS, MN, and Mobile

The terms UE (User Equipment), MS (Mobile Station), MN (Mobile Node), and mobile refer to the devices that are hosts with the ability to obtain Internet connectivity via a 3GPP network. A MS is comprised of the Terminal Equipment (TE) and a Mobile Terminal (MT). The terms UE, MS, MN, and mobile are used interchangeably within this document.

User Plane

The user plane refers to data traffic and the required bearers for the data traffic. In practice, IP is the only data traffic protocol used in the user plane.

2. LTE, 4G Mobile Network

2.1 Network Overview

With the introduction of LTE, mobile networks moved to all-IP transport for all elements such as eNodeB, MME, SGW/PGW, HSS, PCRF, routing and switching etc. Although LTE network is data-centric, it has support for legacy Circuit Switch features like voice and SMS through transitional CS fallback and flexible IMS deployment [GRAYSON]. For each mobile device attached to the radio (eNodeB) there is a separate overlay tunnel (GPRS Tunneling Protocol, GTP) between eNodeB and Mobile gateways (i.e. SGW, PGW).

The GTP tunnel is used to carry user traffic between gateways and mobile devices so the data can only be distributed using unicast mechanism. It is also important to understand the overhead of a GTP and IPsec protocols because it has impact on the carried user data traffic. All mobile backhaul traffic is encapsulated using GTP tunnel, which has overhead of 8 bytes on top of IP and UDP [NGMN]. Additionally, if IPsec is used for security (which is often required if the Service provider is using a shared backhaul), it adds overhead based upon IPsec tunneling model (tunnel or transport), and encryption and authentication header algorithm used. If we factor Advanced Encryption Standard (AES) encryption with packet size the overhead can vary significantly [IPSEC2].

When any UE is powered up, it attaches to a mobile network based on its configuration and subscription. After successful attach procedure, UE registers with the mobile core network and IPv4 and/or IPv6 address is assigned. A default bearer is created for each UE and it is assigned to default Access Point Name (APN).

minimal. Additional dedicated bearer(s) with enhanced QoS parameters is established depending on the specific application needs.

While all traffic within a certain bearer gets the same treatment, QoS parameters supporting these requirements can be very granular in different bearers. These values vary for the control, management and user traffic, and depending on the application key parameters, such as latency, jitter (important for voice and other real-time applications), packet loss and queuing mechanism (strict priority, low-latency, fair etc.) can be very different.

Implementation of QoS for mobile networks is done at two stages: at content prioritization/marketing and transport marking, and congestion management. From the transport perspective, QoS is defined at layer 2 as class of service (CoS) and at layer 3 either as DiffServ code point (DSCP) or type of service (ToS). The mapping of CoS to DSCP takes place at layer 2/3 switching and routing elements. 3GPP has specified QoS Class Identifier (QCI) which represents different types of content and equivalent mapping to DSCP at transport layer [TS23.203] [TS23.401]; however, this again requires manual configuration at different elements and if there is misconfiguration at any place in the path it will not work properly.

In summary QoS configuration for mobile network for user plane (for user traffic) and transport in IP based mobile network is complex and it requires synchronization of parameters among different platforms. Normally QoS in IP is implemented using DiffServ, which uses hop-by-hop QoS configuration at each router. Any inconsistency in IP QoS configuration at routers in the forwarding path can result in poor subscriber experience (e.g. packet classified as high-priority can go to lower priority queue). By deploying ICN, we intend to enhance the subscriber experience using policy based configuration, which can be associated with the named contents [ICNQoS] at ICN forwarder. Further investigation is needed to understand how QoS in ICN can be implemented to meet the IP QoS requirements [RFC4594].

2.3 Data Transport Using IP

The data delivered to mobile devices is unicast inside GTP tunnel from a eNodeB to a PDN gateway (PGW), as described in 3GPP specifications [TS23.401]. While the technology exists to address the issue of possible multicast delivery, there are many difficulties related to multicast protocol implementation on the RAN side of the network. Transport networks in the backhaul and core have addressed the multicast delivery long time ago and have implemented it in most cases in their multi-purpose integrated transport, but the RAN part of the network is still lagging behind due to complexities related to mobility of the clients, handovers, and the fact that the potential

gain to the Service Providers may not justify the investment. With that said, the data delivery in the mobility remains greatly unicast.

To ease the burden on the bandwidth of the SGi interface, caching is introduced in a similar manner as with many Enterprises. In the mobile networks, whenever possible, a cached data is delivered. Caching servers are placed at a centralized location, typically in the Service Provider's Data Center, or in some cases lightly distributed in the Packet Core locations with the PGW nodes close to the Internet and IP services access (SGi interface). This is a very inefficient concept because traffic has to traverse the entire backhaul path for the data to be delivered to the end-user. Other issues, such as out-of-order delivery contribute to this complexity and inefficiency but could be addressed at the IP transport level.

The data delivered to mobile devices is unicast inside a GTP tunnel. If we consider combined impact of GTP, IPSec and unicast traffic, the data delivery is not efficient. By deploying ICN, we intend to either terminate GTP tunnel at the edge by leveraging control and user plane separation or replace it with the native ICN protocols.

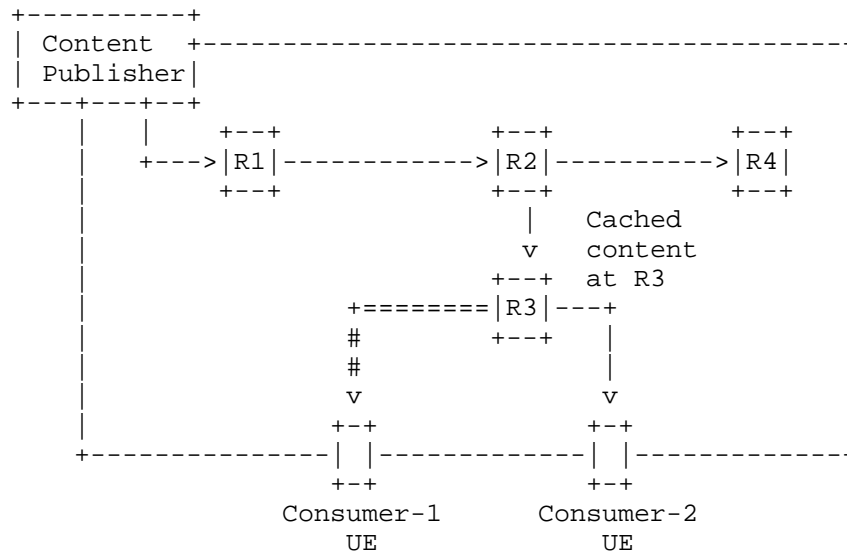
2.4 Virtualizing Mobile Networks

The Mobile packet core deployed in a major service provider network is either based on dedicated hardware or large capacity x86 platforms in some cases. With adoption of Mobile Virtual Network Operators (MVNO), public safety network, and enterprise mobility network, we need elastic mobile core architecture. By deploying mobile packet core on a commercially off the shelf (COTS) platform using virtualized infrastructure (NFVI) framework and end-to-end orchestration, we can simplify new deployments and provide optimized total cost of ownership (TCO).

While virtualization is growing and many mobile providers use hybrid architecture consisting of dedicated and virtualized infrastructures, the control and data delivery planes are still the same. There is also work underway to separate control plane and user plane so that the network can scale better. Virtualized mobile networks and network slicing with control and user plane separation provide mechanism to evolve GTP-based architecture to open-flow SDN-based signaling for LTE and proposed 5G. Some of early architecture work for 5G mobile technologies provides mechanism for control and user plane separation and simplifies mobility call flow by introduction of open-flow based signaling. This has been considered by 3GPP [EPCCUPS] and is also described in [SDN5G].

3. Data Transport Using ICN

For mobile devices, the edge connectivity to the network is between radio and a router or mobile edge computing (MEC) [MECSPEC] element. MEC has the capability of processing client requests and segregating control and user traffic at the edge of radio rather than sending all requests to the mobile gateway.



==> Content flow from cache
 ---> Content flow from publisher

Fig. 2. ICN Architecture

MEC transforms radio into an intelligent service edge that is capable of delivering services directly from the edge of the network, while providing the best possible performance to the client. MEC can be an ideal candidate for ICN forwarder in addition to its usual function of managing mobile termination. In addition to MEC, other transport elements, such as routers, can work as ICN forwarders.

Data transport using ICN is different compared to IP-based transport. It evolves the Internet infrastructure by introducing uniquely named data as a core Internet principle. Communication in ICN takes place between content provider (producer) and end user (consumer) as described in figure 2.

Every node in a physical path between a client and a content provider is called ICN forwarder or router, and it has the ability to route the request intelligently and also cache the content so that it can be delivered locally for subsequent request from any other client.

For mobile network, transport between a client and a content provider consists of radio network + mobile backhaul and IP core transport + Mobile Gateways + Internet + content data network (CDN).

In order to understand suitability of ICN for mobile networks, we will discuss the ICN framework describing protocols architecture and different types of messages, and then consider how we can use this in a mobile network for delivering content more efficiently. ICN uses two types of packets called "interest packet" and "data packet" as described in figure 3.

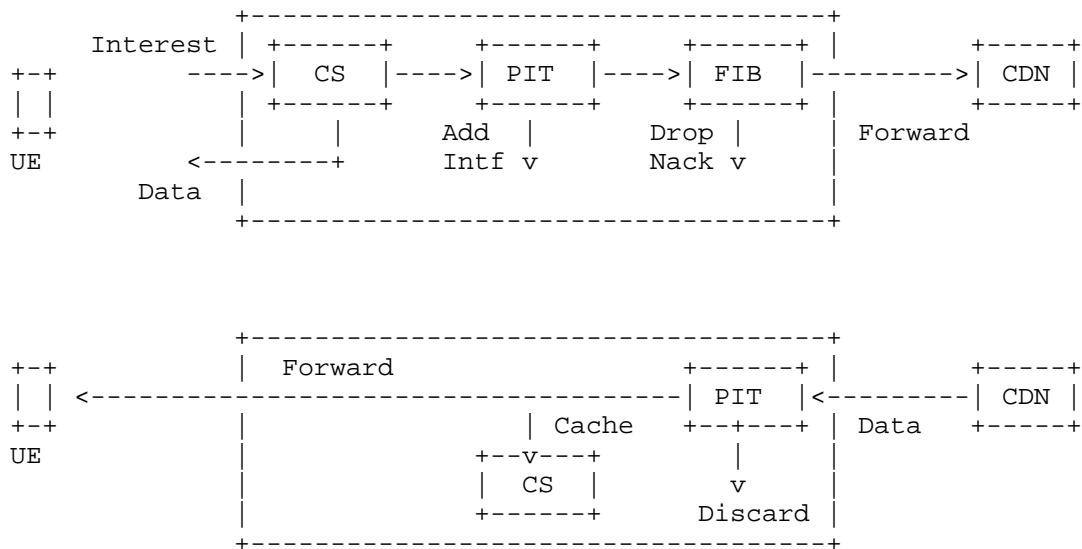


Fig. 3. ICN Interest, Data Packet and Forwarder

For LTE network, when a mobile device wants to get certain content, it will send an Interest message to the closest eNodeB. Interest packet follows the TLV format [CCNxTLV] and contains mandatory fields such as name of the content and nonce. Name and nonce together uniquely identify an Interest packet. Nonce is also used to detect looping Interest messages. Interest packet also contains optional fields such as selector and guider fields. Selectors provides a specific filtering action during matching and selection of the name prefixes. Guiders provides specific set of rules on how the Interest packet can be processed at the forwarder.

First ICN router will receive Interest packet and perform lookup if request for such content has come earlier from any other client. If yes, it is served from the local cache, otherwise request is forwarded to the next-hop ICN router. Each ICN router maintains three

data structures, namely Pending Interest Table (PIT), Forwarding Information Base (FIB), and Content Store (CS). The Interest packet travels hop-by-hop towards content provider. Once the Interest reaches the content provider it will return a Data packet containing information such as content name, signature, signed key and data.

Data packet travels in reverse direction following the same path taken by the interest packet so routing symmetry is maintained. Details about algorithms used in PIT, FIB, CS and security trust models are described in various resources [CCN], here we explained the concept and its applicability to the LTE network.

4. ICN Deployment in 4G and LTE Networks

4.1 General ICN Deployment Considerations

In LTE/4G mobile networks, both user and control plane traffic have to be transported from the edge to the mobile packet core via IP transport. The evolution of existing mobile packet core using CUPS [TS23.714] enables flexible network deployment and operation, by distributed deployment and the independent scaling between control plane and user plane functions - while not affecting the functionality of the existing nodes subject to this split.

In the CUPS architecture, there is an opportunity to shorten the path for user plane traffic by deploying offload nodes closer to the edge. This optimization allows for the introduction of ICN and amplifies its advantages. This section analyzes the potential impact of ICN on control and user plane traffic for centralized and disaggregate CUPS based mobile network architecture.

4.2 ICN Deployment Scenarios

Deployment of ICN provides an opportunity to further optimize the existing data transport in LTE/4G mobile networks. The various deployment options that ICN and IP provide are somewhat analogous to the deployment scenarios when IPv6 was introduced to inter operate with IPv4, except with ICN the whole IP stack is being replaced. We have reviewed [RFC6459] and analyzed the impact of ICN on control plane signaling and user plane data delivery. In general ICN can be deployed natively replacing IP transport (IPv4 and IPv6) or as an overlay protocol. Figure 4 describes a modified protocol stack to support ICN deployment scenarios.

As shown in figure 4, for applications running either in UE or in content provider system to use the new transport option, we propose a new transport convergence layer (TCL). This transport convergence layer helps determine what type of transport (e.g. ICN or IP), as

well as type of radio interface (e.g. LTE or WiFi or both), is used to send and receive the traffic based on preference e.g. content location, content type, content publisher, congestion, cost, quality of service etc. It helps to configure and decide the type of connection as well as the overlay mode (ICNoIP or IPoICN) between application and the protocol stack (IP or ICN) to be used.

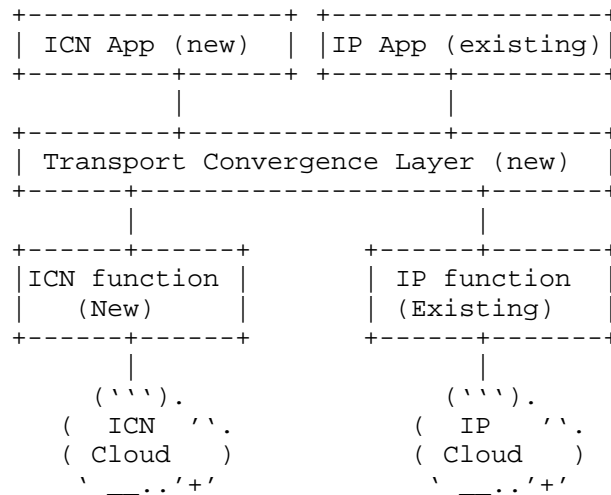


Fig. 4. IP/ICN Convergence and Deployment Scenarios

TCL can use a number of mechanisms for the selection of transport. It can use a per application configuration through a management interface, possibly even a user-facing setting realized through a user interface, similar to those used today that select cellular over WiFi being used for selected applications. In another option, it might use a software API, which an adapted IP application could use to specify e.g. an ICN transport for obtaining its benefits.

Another potential application of TCL is in implementation of network slicing, where it can have a slice management capability locally or it can interface to an external slice manager through an API [GALIS]. This solution can enable network slicing for IP and ICN transport selection from the UE itself. The TCL could apply slice settings to direct certain traffic (or applications) over one slice and others over another slice, determined by some form of 'slicing policy'. Slicing policy can be obtained externally from slice manager or configured locally on UE.

From the perspective of the applications either on UE or content provider, four different options are possible for the deployment of

ICN natively and/or with IP.

1. IP over IP

In this scenario UE uses applications tightly integrated with the existing IP transport infrastructure. In this option, the TCL has no additional function since the packets are directly forwarded using IP protocol stack, which in turn sends the packets over the IP transport.

2. ICN over ICN

Similar to case 1 above, ICN applications tightly integrate with the ICN transport infrastructure. The TCL has no additional responsibility since the packets are directly forwarded using ICN protocol stack, which in turn sends the packets over the ICN transport.

3. ICN over IP (ICNoIP)

In ICN over IP scenario, the underlying IP transport infrastructure is not impacted (i.e. ICN is implemented, as an IP overlay, between user equipment (UE) and content provider). IP routing is used from Radio Access Network (eNodeB) to mobile backhaul, IP core and Mobile Gateway (SGW/PGW). UE attaches to Mobile Gateway (SGW/PGW) using IP address. Also, the data transport between Mobile Gateway (SGW/PGW) and content publisher uses IP. Content provider is capable of serving content either using IP or ICN, based on UE request.

An alternative approach to implement ICN over IP is provided in Hybrid ICN [HICN], which implements ICN over IP by mapping of ICN names to the IPv4/IPv6 addresses.

Detailed deployment of use cases is described in section 4.4. Application conveys the preference to the TCL, which in turn sends the ICN data packets using the IP transport.

4. IP over ICN (IPoICN)

H2020 project [H2020] provides an architectural framework for deployment of IP as an overlay over ICN protocol [IPICN]. Implementing IP services over ICN provides an opportunity leveraging benefit of ICN in the transport infrastructure and there is no impact on end devices (UE and access network) as they continue to use IP. IPoICN however, will require an inter-working function (IWF/Border Gateway) to translate various transport primitives such as transport of tunnel mode. IWF

function will provide a mechanism for protocol translation between IPoICN and native IP deployment for mobile network. After reviewing [IPICN], we understand and interpret that ICN is implemented in the transport natively; however, IP is implemented in UE, eNodeB, and Mobile gateway (SGW/PGW), which is also called as network attach point (NAP).

4.3 ICN Deployment in LTE Control Plane

In this section we analyze signaling messages which are required for different procedures, such as attach, handover, tracking area update etc. The goal of analysis is to see if there is any benefit to replace IP-based protocols with ICN for LTE signaling in the current architecture. It is important to understand the concept of point of attachment (POA). When UE connects to a network it has at least three POAs:

1. eNodeB managing location or physical POA
2. Authentication and Authorization (MME, HSS) managing identity or authentication POA
3. Mobile Gateways (SGW, PGW) managing logical or session management POA.

In current architecture IP transport is used for all the messages associated with Control Plane for mobility and session management. IP is embedded very deeply into these messages and TLV carrying additional attributes as a layer 3 transport. Physical POA in eNodeB handles both mobility and session management for any UE attached to 4G, LTE network. The number of mobility management messages between different nodes in an LTE network per signaling procedure are given below in figure 5.

Normally two types of UE devices attach to LTE network: SIM based (need 3GPP mobility protocol for authentication) or non-SIM based (which connect to WiFi network), and authentication is required for both of these device types. For non-SIM based devices, AAA is used for authentication. We do not propose to change UE authentication procedures for user data transport using ICN, or any other mobility management messaging. A separate study would be required to analyze impact of ICN on mobility management messages structures and flows. We are merely analyzing the viability of implementing ICN as a transport for Control plane messages.

It is important to note that even if MME and HSS do not support ICN transport, they still need to support UE capable of dual stack or native ICN. When UE initiates attach request using the identity as

ICN, MME must be able to parse that message and create a session. MME forwards UE authentication to HSS so HSS must be able to authenticate an ICN capable UE and authorize create session [TS23.401].

LTE Signaling Procedures	MME	HSS	SGW	PGW	PCRF
Attach	10	2	3	2	1
Additional default bearer	4	0	3	2	1
Dedicated bearer	2	0	2	2	1
Idle-to-connect	3	0	1	0	0
Connect-to-Idle	3	0	1	0	0
X2 handover	2	0	1	0	0
S1 handover	8	0	3	0	0
Tracking area update	2	0	0	0	0
Total	34	2	14	6	3

Fig. 5. Signaling Messages in LTE Gateways

Anchorless mobility [ALM] has made some important comments on how mobility management is done in ICN. Author comments about handling mobility without having a dependency on the core network function e.g. MME. However, location update to the core network would still be required to support some of the legal compliance requirements such as lawful intercept and emergency services.

The main advantage of ICN is in caching and reusing the content, which does not apply to the transactional signaling exchange. After analyzing LTE signaling call flows [TS23.401] and messages inter-dependencies [Fig 4], our recommendation is that it is not beneficial to deploy ICN for control plane and mobility management functions.

4.4 ICN Deployment in LTE User Plane

We will consider figure 1 to discuss different mechanisms to deploy ICN in mobile networks. In section 4.2 we discussed generic deployment scenarios of ICN. In this section, we shall see the specific use cases of native ICN deployment in LTE user plane. We consider the following options:

1. Dual stack ICN deployment in UE
2. Native ICN Deployments in UE
3. ICN Deployment in eNodeB
4. ICN Deployment in mobile gateways (SGW/PGW)

4.4.1 Dual stack ICN Deployments in UE

The control and user plane communications in LTE, 4G mobile networks are specified in 3GPP documents [TS23.323] [TS23.203] [TS23.401]. It is important to understand that UE can be either consumer (receiving content) or publisher (pushing content for other clients). The protocol stack inside mobile device (UE) is complex as it has to support multiple radio connectivity access to eNodeB(s).

Figure 6 provides high level description of a protocol stack, where IP is defined at two layers: (1) at user plane communication, (2) Transport layer. User plane communication takes place between Packet Data Convergence Protocol (PDCP) and Application layer, whereas transport layer is at GTP protocol stack.

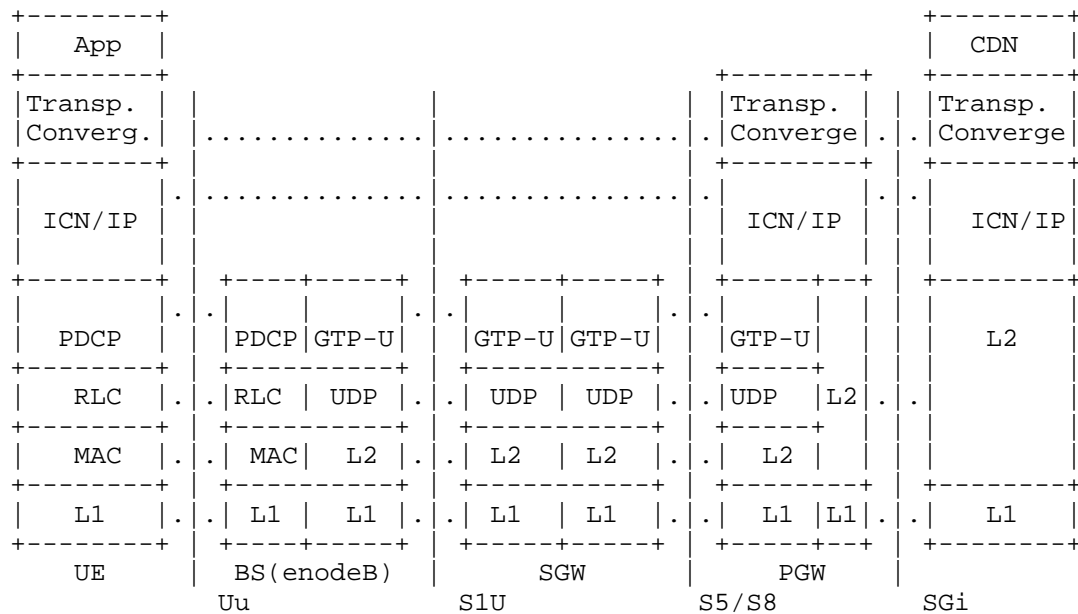


Fig. 6. Dual stack ICN Deployment in UE

The protocol interactions and impact of supporting tunneling of ICN packet into IP or to support ICN natively are described in figure 6 and figure 7 respectively.

The protocols and software stack used inside LTE capable UE support both 3G and LTE software interworking and handover. Latest 3GPP Rel.13 onward specification describes the use of IP and non-IP protocols to establish logical/session connectivity. We intend to leverage the non-IP protocol based mechanism to deploy ICN protocol

stack in UE as well as in eNodeB and mobile gateways (SGW, PGW).

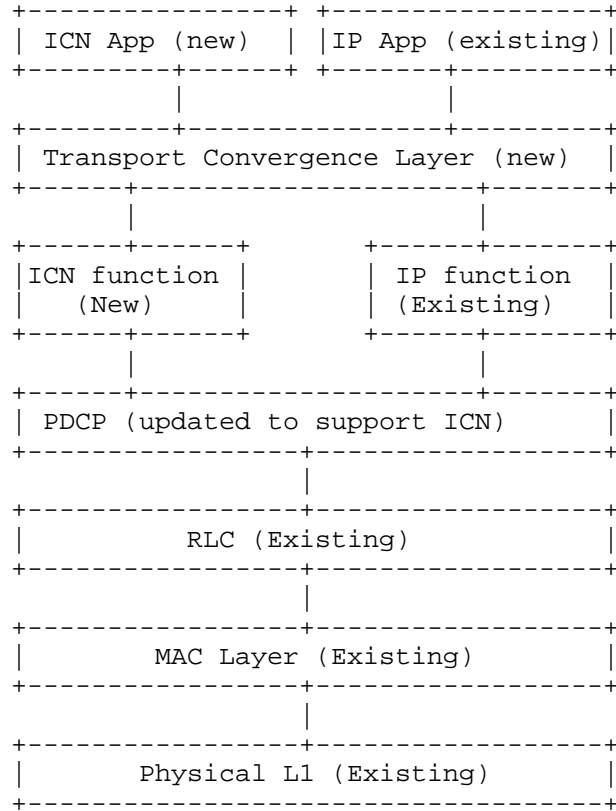


Fig. 7. Dual stack ICN protocol interactions

1. Existing application layer can be modified to provide options for new ICN based application and existing IP based applications. UE can continue to support existing IP based applications or host new applications developed either to support native ICN as transport, ICNoIP or IPoICN based transport. Application layer has the option of selecting either ICN or IP transport layer as well as radio interface to send and receive data traffic.

Our proposal is to provide a common Application Programming Interface (API) to the application developers such that there is no impact on the application development when they choose either ICN or IP transport for exchanging the traffic with the network. As mentioned in section 4.2, the transport convergence layer (TCL) function handles the interaction of application

with the multiple transport options.

2. The transport convergence layer helps determine what type of transport (e.g. ICN or IP) as well as type of radio interface (e.g. LTE or WiFi or both), is used to send and receive the traffic. Application layer can make the decision to select a specific transport based on preference e.g. content location, content type, content publisher, congestion, cost, quality of service etc. There can be an Application Programming Interface (API) to exchange parameters required for transport selection. The southbound interactions of Transport Convergence Layer (TCL) will be either to IP or ICN at the network layer.
3. ICN function (forwarder) is introduced in parallel to the existing IP layer. ICN forwarder contains functional capabilities to forward ICN packets, e.g. Interest packet to eNodeB or response "data packet" from eNodeB to the application.
4. For dual stack scenario, when UE is not supporting ICN at transport layer, we use IP underlay to transport ICN packets. ICN function will use IP interface to send Interest and Data packets for fetching or sending data using ICN protocol function. This interface will use ICN overlay over IP using any overlay tunneling mechanism.
5. To support ICN at network layer in UE, PDCP layer has to be aware of ICN capabilities and parameters. PDCP is located in the Radio Protocol Stack in the LTE Air interface, between IP (Network layer) and Radio Link Control Layer (RLC). PDCP performs following functions [TS36.323]:
 - a) Data transport by listening to upper layer, formatting and pushing down to Radio Link Layer (RLC)
 - b) Header compression and decompression using ROHC (Robust Header Compression)
 - c) Security protections such as ciphering, deciphering and integrity protection
 - d) Radio layer messages associated with sequencing, packet drop detection and re-transmission etc.
6. No changes are required for lower layer such as RLC, MAC and Physical (L1) because they are not IP aware.

One key point to understand in this scenario is that ICN is deployed

as an overlay on top of IP.

4.4.2 Native ICN Deployments in UE

We propose to implement ICN natively in UE by modifying PDCP layer in 3GPP protocols. Figure 8 provides a high-level protocol stack description where ICN is used at two different layers:

1. at user plane communication
2. at transport layer

User plane communication takes place between PDCP and application layer, whereas transport layer is a substitute of GTP protocol. Removal of GTP protocol stack is significant change in mobile architecture because GTP is used not just for routing but for mobility management functions such as billing, mediation, policy enforcement etc.

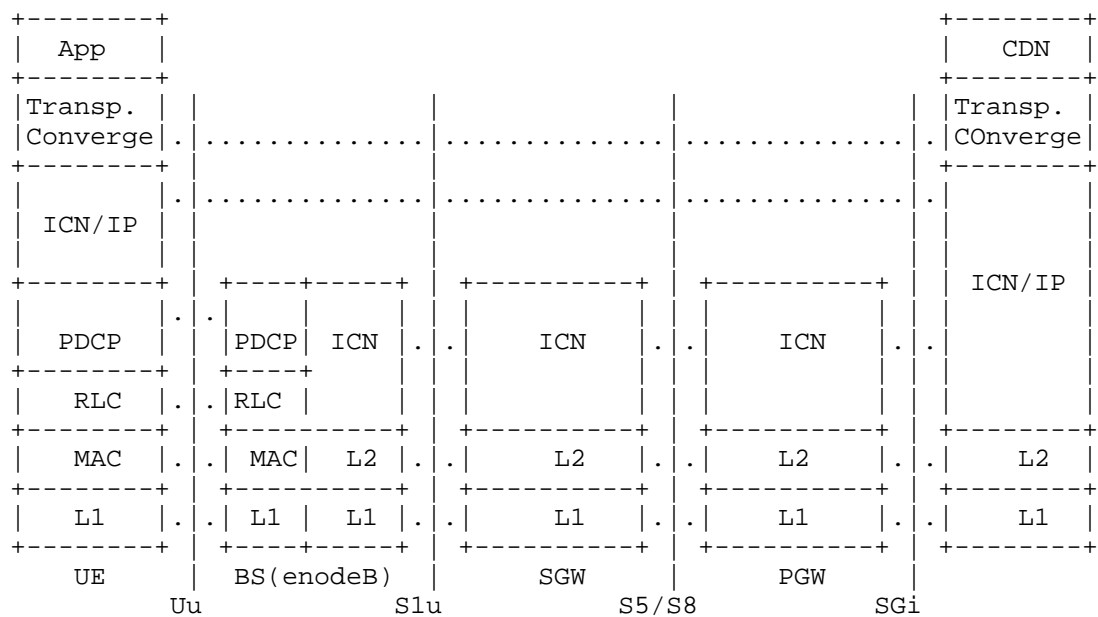


Fig. 8. Native ICN Deployment in UE

If we implement ICN natively in UE, communication between UE and eNodeB will change and also we will not need to tunnel user plane traffic from eNodeB to mobile packet core (SGW, PGW) using GTP tunnel.

For native ICN deployment, Application is configured to use ICN forwarder so there is no need for Transport Convergence. Also to support ICN at network layer in UE, we need to modify existing PDCP layer. PDCP layer has to be aware of ICN capabilities and parameters.

Native implementation will also provide opportunities to develop new use cases leveraging ICN capabilities such as seamless mobility, UE to UE content delivery using radio network without interactions with mobile gateways, etc.

4.5 ICN Deployment in eNodeB

eNodeB is physical point of attachment for UE, where radio protocols are converted into IP transport protocol as depicted in figure 7 and figure 8 for dual stack/overlay and native ICN respectively. When UE performs attach procedures, it is assigned an identity either as IP, dual stack (IP and ICN), or ICN. UE can initiate data traffic using any of three different options:

1. Native IP (IPv4 or IPv6)
2. Native ICN
3. Dual stack IP (IPv4/IPv6) or ICN

UE encapsulates user data transport request into PDCP layer and sends the information on air interface to eNodeB. eNodeB receives the information and using PDCP [TS 36.323], de-encapsulates air-interface messages and converts them to forward to core mobile gateways (SGW, PGW). In order to support ICN natively in eNodeB, it is proposed to provide transport convergence layer (TCL) capabilities in eNodeB (similar as provided in UE), which provides following functions:

1. It decides the forwarding strategy for user data request coming from UE. The strategy can make decision based on preference indicated by the application such as congestion, cost, quality of service, etc.
2. eNodeB to provide open Application Programming Interface (API) to external management systems, to provide capability to eNodeB to program the forwarding strategies.
3. eNodeB shall be upgraded to support three different types of transport: IP, ICN, and dual stack IP+ICN towards mobile gateways, as depicted in figure 9. It is also recommended to deploy IP and/or ICN forwarding capabilities into eNodeB for efficient transfer of data between eNodeB and mobile gateways. There can be four choices for forwarding data request towards

mobile gateways:

- a) Assuming eNodeB is IP-enabled and UE requests IP transfer, eNodeB forwards data over IP.
- b) Assuming eNodeB is ICN-enabled and UE requests ICN transfer, eNodeB forwards data over ICN.
- c) Assuming eNodeB is IP-enabled and UE requests ICN, eNodeB overlays ICN on IP and forwards the user plane traffic over IP.
- d) Assuming eNodeB is ICN-enabled and UE requests IP, eNodeB overlays IP on ICN and forwards the user plane traffic over ICN [IPoICN].

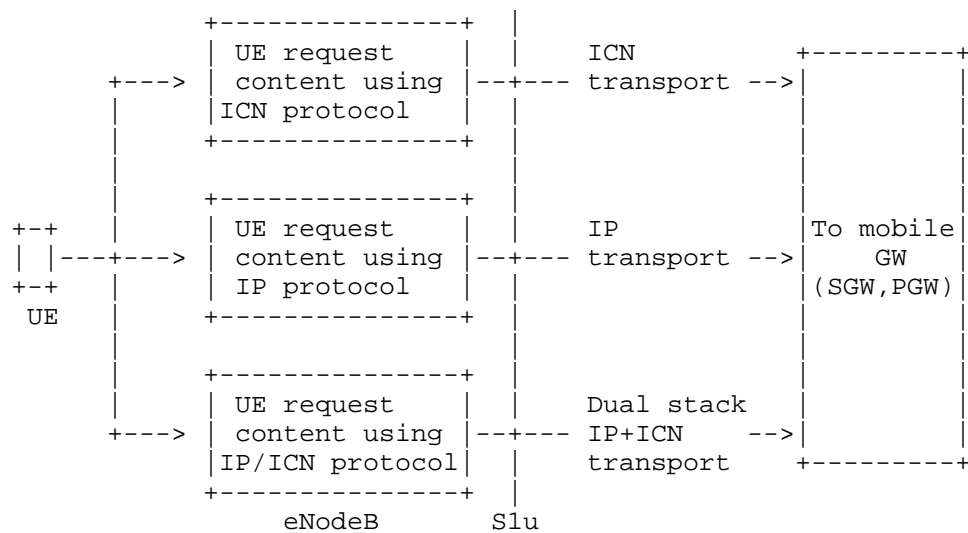


Fig. 9. Native ICN Deployment in eNodeB

4.6 ICN Deployment in Packet Core (SGW, PGW) Gateways

Mobile gateways a.k.a. Evolved Packet Core (EPC) include SGW, PGW, which perform session management for UE from the initial attach to disconnection. When UE is powered on, it performs NAS signaling and after successful authentication it attaches to PGW. PGW is an anchoring point for UE and responsible for service creations, authorization, maintenance etc. Entire functionality is managed using IP address(es) for UE.

In order to implement ICN in EPC, the following functions are needed.

1. Insert ICN function at session management layer as additional functionality with IP stack. Session management layer is used for performing attach procedures and assigning logical identity to user. After successful authentication by HSS, MME sends create session request (CSR) to SGW and SGW to PGW.
2. When MME sends Create Session Request message (step 12 in [TS23.401]) to SGW or PGW, it contains Protocol Configuration Option Information Element (PCO IE) containing UE capabilities. We can use PCO IE to carry ICN related capabilities information from UE to PGW. This information is received from UE during the initial attach request in MME. Details of available TLV, which can be used for ICN are given in subsequent sections. UE can support either native IP, or ICN+IP, or native ICN. IP is referred to as both IPv4 and IPv6 protocols.
3. For ICN+IP capable UE, PGW assigns the UE both IP address and ICN identity. UE selects either of the identities during the initial attach procedures and registers with network for session management. For ICN-capable UE it will provide only ICN attachment. For native IP-capable UE there is no change.
4. In order to support ICN-capable attach procedures and use ICN for user plane traffic, PGW needs to have full ICN protocol stack functionalities. Typical ICN capabilities include functions such as content store (CS), Pending Interest Table (PIT), Forwarding Information Base (FIB) capabilities etc. If UE requests ICN in PCO IE, then PGW registers UE with ICN names. For ICN forwarding, PGW caches content locally using CS functionality.
5. Protocol configuration options information elements described in [TS24.008] (see Figure 10.5.136 on page 598) and [TS24.008] (see Table 10.5.154 on page 599) provide details for different fields.
 - Octet 3 (configuration protocols defines PDN types) which contains details about IPv4, IPv6, both or ICN.
 - Any combination of Octet 4 to Z can be used to provide additional information related to ICN capability. It is most important that PCO IE parameters are matched between UE and mobile gateways (SGW, PGW) so that they can be interpreted properly and UE can attach successfully.
6. Deployment of ICN functionalities in SGW and PGW should be matched with UE and eNodeB because they will exchange ICN protocols and parameters.

7. Mobile gateways SGW, PGW will also need ICN forwarding and caching capability.
8. The transport between PGW and CDN provider can be either IP or ICN. When UE is attached to PGW with ICN identity and communicates with an ICN-enabled CDN provider, it will use ICN primitives to fetch the data. On other hand, for an UE attached with an ICN identity, if PGW has to communicate with an IP-enabled CDN provider, it will have to use an ICN-IP interworking gateway to perform conversion between ICN and IP primitives for data retrieval. Further study is required to understand how this ICN to IP (and vice versa) interworking gateway would function.

5. Security Considerations

To ensure only authenticated UEs are connected to the network, LTE mobile network implements various security mechanisms. From perspective of ICN deployment in user plane, it needs to take care of the following security aspects:

1. UE authentication and authorization
2. Radio or air interface security
3. Denial of service attacks on mobile gateway, services
4. Content positioning either in transport or servers
5. Content cache pollution attacks
6. Secure naming, routing, and forwarding
7. Application security

Security over the LTE air interface is provided through cryptographic technique. When UE is powered up, it performs key exchange between UE's USIM and HSS/Authentication Center using NAS messages including ciphering and integrity protections between UE and MME. Details of secure UE authentication, key exchange, ciphering and integrity protections messages are given in 3GPP call flow [TS23.401].

LTE is an all-IP network and uses IP transport in its mobile backhaul (e.g. between eNodeB and core network). In case of provider owned backhaul, it may not implement security mechanisms; however, they are necessary in case it uses shared or a leased network. The native IP transport continues to leverage security mechanism such as Internet key exchange (IKE) and the IP security protocol (IPsec). More details

of mobile backhaul security are provided in 3GPP network security [TS33.310] and [TS33.320]. When mobile backhaul is upgraded to support dual stack (IP+ICN) or native ICN, it is required to implement security techniques which are deployed in mobile backhaul. When ICN forwarding is enabled on mobile transport routers, we need to deploy security practices based on RFC7476 and RFC7927.

Some of the key functions supported by LTE mobile gateway (SGW, PGW) are content based billing, deep packet inspection (DPI), and lawful intercept (LI). For ICN-based user plane traffic, it is required to integrate ICN security for session between UE and gateway; however, in ICN network, since only consumers who are in possession of decryption keys can access the content, some of the services provided by mobile gateways mentioned above may not work. Further research in this area is needed.

6. Summary

In this draft, we have discussed complexities of LTE network and key dependencies for deploying ICN in user plane data transport. Different deployment options described cover aspects such as interoperability and multi-technology, which is a reality for any service provider. We are currently evaluating the ICN deployment options, described in section 4, using LTE gateway software and ICN simulator. One can deploy ICN for data transport in user plane either as an overlay, dual stack (IP + ICN) or natively (by integrating ICN with CDN, eNodeB, SGW, PGW and transport network etc.). It is important to understand that for above discussed deployment scenarios, additional study is required for lawful interception, billing/mediation, network slicing, and provisioning APIs.

Based on our study of control plane signaling it is not beneficial to deploy ICN with existing protocols unless further changes are introduced in the control protocol stack itself. As mentioned in [TS23.501], 5G network architecture proposes simplification of control plane messages and can be a candidate for use of ICN.

As a starting step towards ICN user plane deployment, it is recommended to incorporate protocol changes in UE, eNodeB, SGW/PGW for data transport. ICN has inherent capabilities for mobility and content caching, which can improve the efficiency of data transport for unicast and multicast delivery.

Mobile Edge Computing (MEC) [CHENG] provides capabilities to deploy functionalities such as Content Delivery Network (CDN) caching and mobile user plane functions (UPF) [TS23.501]. Recent research for delivering real-time video content using ICN has also been proven to be efficient [NDNRTC] and can be used towards realizing the benefits

of ICN deployment in eNodeB, MEC, mobile gateways (SGW, PGW) and CDN. The key aspect for ICN is in its seamless integration in LTE and 5G networks with tangible benefits so that we can optimize content delivery using simple and scalable architecture. Authors will continue to explore how ICN forwarding in MEC could be used in efficient data delivery from mobile edge.

7 References

7.1 Normative References

- [GRAYSON] Grayson M, Shatzkamer K, Wainner S.; Cisco Press book "IP Design for Mobile Networks" by. page 108-112.
- [IPSEC1] Cisco IPsec overhead calculator tool
<<https://cway.cisco.com/tools/ipsec-overhead-calc/ipsec-overhead-calc.html>>.
- [IPSEC2] IPsec Bandwidth Overhead Using AES
<<http://packetpushers.net/ipsec-bandwidth-overhead-using-aes/>>.
- [BROWER] Brower, E.; Jeffress, L.; Pezeshki, J.; Jasani, R.; Ertekin, E. "Integrating Header Compression with IPsec", Military Communications Conference, 2006. MILCOM 2006. IEEE, On page(s): 1 - 6.
- [TS25.323] 3GPP TS25.323 Rel. 14 (2017-03) Packet Data Convergence Protocol (PDCP) specification.
- [TS23.501] 3GPP TS23.501 Rel. 15 (2017-06) System Architecture for the 5G System.
- [TS23.203] 3GPP TS23.203 Rel. 14 (2017-03) Policy and charging control and QoS architecture
- [TS23.401] 3GPP TS23.401 Rel. 14 (2017-03) E-UTRAN Access procedures architecture
- [TS33.310] 3GPP TS33.310 Rel. 14 (2016-12) LTE Network Domain Security (NDS); Authentication Framework (AF)
- [TS33.320] 3GPP TS33.320 Rel. 14 (2016-12) Security of Home Node B (HNB) / Home evolved Node B (HeNB)
- [TS24.008] 3GPP TS24.008 Rel. 14 (2017-06) Mobile radio interface Layer 3 specification.
- [TS23.501] 3GPP TS23.501 Rel. 14 (2017-06) System Architecture for the 5G System
- [TS23.214] 3GPP TS23.214 Rel. 14 (2017-06) Architecture enhancements for control and user plane separation of EPC nodes
- [TS36.323] 3GPP TS36.323 Rel. 14 (2017-06) Evolved Universal

Terrestrial Radio Access (E-UTRA); Packet Data Convergence Protocol (PDCP) specification

[TS23.714] 3GPP TS23.714 Rel. 14 (2016-06) Technical Specification Group Services and System Aspects: Study on control and user plane separation of EPC nodes

[RFC7476] Information-Centric Networking: Baseline Scenarios

[RFC7927] Information-Centric Networking (ICN) Research Challenges

[RFC6459] IPv6 in 3GPP Evolved Packet System (EPS)

7.2 Informative References

[RFC2119] Key words for use in RFCs to Indicate Requirement Levels
<<https://www.ietf.org/rfc/rfc2119.txt>>

[MECSPEC] European Telecommunication Standards Institute (ETSI) MEC specification ETSI-GS-MEC-IEG-001 V1.1.1 (2015-11).

[NDNTLV] NDN Interest Packet Format Specification 0.2-2.
<<https://named-data.net/doc/ndn-tlv/interest.html>>

[CCNxTLV] CCNx Messages in TLV Format
<<https://datatracker.ietf.org/doc/draft-irtf-icnrg-ccnxmessages/>>

[NDNPUB] Named Data Networking <<http://named-data.net/publications/>>.

[CCN] Content Centric Networking <<http://www.ccnx.org> and <http://blogs.parc.com/ccnx/documentation-guide/>>.

[NDN] Lixia Z., Lan W. et al. SIGCOMM Named Data Networking

[ALM] J. Aug'e, G. Carofiglio et al. "Anchor-less producer mobility in icn," in Proceedings of the 2Nd ACM Conference on Information-Centric Networking, ACM-ICN '15, pp. 189-190, ACM, 2015.

[VNIIDX] Cisco Visual Networking Index (VNI) dated 16 Feb 2016, <<http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html>>.

[NDNRTC] Peter Gusev,Zhehao Wang, Jeff Burke, Lixia Zhang et. All,

IEICE Trans Communication, RealtimeStreaming Data Delivery over Named Data Networking, Vol E99-B, No.5 May 2016.

- [CHENG] Chengchao L., F. Richard Yu, Information-centric network fucntion virtualization over 5G mobile wireless networks, IEEE network (Volume:29, Issue:3), page 68-74, 01 June 2015.
- [NGMN] Backhaul Provisioning for LTE-Advanced & Small Cells <https://www.ngmn.org/uploads/media/150929_NGMN_P-SmallCells_Backhaul_for_LTE-Advanced_and_Small_Cells.pdf>
- [IPoICN] IP Over ICN - The Better IP? <<http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7194109>>
- [HICN] Cisco Hybrid ICN <<http://blogs.cisco.com/sp/cisco-announces-important-steps-toward-adoption-of-information-centric-networking>>
- [GALIS] Autonomic Slice Networking-Requirements and Reference Model <<https://www.ietf.org/id/draft-galis-anima-autonomic-slice-networking-02.txt>>
- [EPCCUPS] Control and User Plane Separation of EPC nodes (CUPS). <<http://www.3gpp.org/news-events/3gpp-news/1882-cups>>
- [SDN5G] Software-defined networking for low-latency 5G core network. <<http://ieeexplore.ieee.org/document/7496561/>>
- [ICNQoS] Quality of Service in an Information-Centric Network. <<http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7037079>>
- [RFC4594] Configuration Guidelines for DiffServ Service Classes

Authors' Addresses

Prakash Suthar
9501 Technology Blvd.
Rosemont, Illinois 60018

EMail: psuthar@cisco.com

Milan Stolic
9501 Technology Blvd.

Rosemont, Illinois 60018

Email: mistolic@cisco.com

Anil Jangam
3625 Cisco Way
San Jose, CA 95134
USA

Email: anjangam@cisco.com

Dirk Trossen
InterDigital Inc.
64 Great Eastern Street, 1st Floor
London EC2A 3QR
United Kingdom

Email: Dirk.Trossen@InterDigital.com

ICN Working Group
INTERNET-DRAFT
Intended Status: Informational
Expires: Apr 30, 2018

Xia Yong
China SARFT
S. Duan
China CAICT
Shu Liu
China CAICT
R.Huang
Huawei
Oct 30, 2017

the Consideration for the Application of Multi-Service Tag
draft-xia-icn-multiservtag-04

Abstract

According to the significant concepts and research challenges described in RFC7927, we think that the multi-service tag technology is a effective name mechanism for video contents in ICN. Because the video traffic is the primary traffic transferred in the Internet, it will tremendously promote the current Internet architecture to the ICN architecture that the name mechanism for the video contents is established. This document discusses the consideration for the design of multi-service tag in ICN and how to use the multi-service tag technology to establish the name mechanism for the video contents. This document also gives the typical cases which use the above name mechanism to improve the content distribution efficiency and cache system efficiency.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2013 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal
Provisions Relating to IETF Documents
(<http://trustee.ietf.org/license-info>) in effect on the date of
publication of this document. Please review these documents
carefully, as they describe your rights and restrictions with respect
to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1	Introduction	3
2	Brief background	3
3	Analysis of the limitation of current network	4
4	Name mechanism for the video contents in ICN	5
5	Design of multi-service tag	5
5.1	the design rules of multi-service tag	5
5.2	the preliminary design of multi-service tag	6
6	System of multi-service tag	7
7	Design of routing and route resolution for the multi-service tag	7
8	Some application cases	8
8.1	content resource sharing across ISP network	8
8.2	cache according to the content naming information	8
9	Security Considerations	9
10	IANA Considerations	9
11	Acknowledgements	9
12	References	9
12.1	Normative References	9
	Authors' Addresses	10

1 Introduction

Now the network traffic presents a rapid increase trend, the popularization of network video and the diversified viewing model modes support watch video in anytime and anywhere, which also results in the increase of network traffic. The network video Apps must provide terrific Quality of experience(QoE). These trends represent a developing direction of future networks. Recognition and handling of the application traffic is a key factor for network operation. Each network application uses different protocol and is deployed by different ISP, which incompletely depends on the network operators. The method of the recognition of traffic and applications uses the fuzzy heuristic modes which are based on the port scope and key information of the traffic and are similar with the DPI technology, but this series of technologies have some limitations. The heuristic methods can't effectively solve the problem of traffic recognition because they can't keep up with the synchronization update of application characteristics. The traffic recognition schemes based on the port scope detection face the great challenge because of enormous amount of ports which are discontinuous, especially for http traffic, the http traffic usually use 80 or 8080 port, so the content in http traffic is difficult to be identified accurately. Due to the encryption transmission of more and more traffic, these lead to the great increase of DFI/DPI calculated amount and make these two technologies be faced with invalidation. IP tunneling technology makes the operator's network more complex. So we need a new technology which can rapidly and uniquely recognize the traffic based on its characteristics without resolve the whole package.

The purpose of this document is to devise a mechanism allowing ICN forwarders, consumers, producers and other ICN nodes to name content identify content find content and share content.

2 Brief background

Now vast Internet video resources are concentrated in minority large operators. If other operator hope to operate the Internet video, they must buy the Internet traffics from the minority large operators. therefore inter-network settlement is a big cost for the Internet video operation and maybe over a half for the small operators. In order to reduce the cost, the small operators spend lots of expenditure to establish cache or CDN and thus each small operator has many caches and CDN. These caches and CDN distribute in the network just like some islands which stored data is repetitive for more than 90 percent, thus the operation effect isn't desired and the

maintain cost is high. From the view of resource scheduling mechanism, the content schedule relies on IP address under current technique system, but many small operators don't have enough public network IP address which leads to scheduling failure for OTT video services though they have own caches and CDN.

So the operators imminently hope a technical solution to help integrating the decentralized and repetitive content in the network and get rid of the limiting condition of IP address. We think that the name mechanism of Multi-Service Tag and ICN can effectively resolve this problem. We plan to establish an ICN network in ICN-as-an-overlay mode over different operators' network, get through the content switching channels between different small operators and realize unified schedule and sharing for the video resources through the name mechanism of Multi-Service Tag. For the Implementation, different small operators' networks distribute around ICN network like islands and have sole interface to ICN. This interface collects the information of content resource URL and its attributes and then generate only name for the content according to some rules and establish the mapping table between multi-service tag and URL for content addressing. We broadcast the content addressing based mapping tables to all ISPs through the ICN network and then all the operators connecting the ICN will know the content distribution all over the network, they can integrate the storage fragmentation and reduce the content repetitive storage, the large amount of content can be obtained by ICN sharing and decrease the cost spent on the Internet traffic purchase from the large operators.

3 Analysis of the limitation of current network

The traffic recognition ways based on IP address pool face difficulties. Because IP address is of large amount, dynamic, proprietary or private. According to the CDN protocol (RFC 6770), the content can be transferred to different CDN and this makes it impossible to track the content among different CDN in terms of its IP address. Though the traffic recognition based on IP address is possible in some scenes, it's impossible to exactly identify every flow. Because the same port is maybe repetitively used by different application, the traffic recognition based on port may lead the wrong results. DFI/DPI may lose efficacy or become very complicated with the more and more encrypted traffic in order to analyze the content contained by the traffic. A traffic flow of an application will end at user terminal through different network routes and this will affect the analysis of the traffic flow. There are no unified standards for traffic recognition and analysis and it will lead to different analysis results for the same traffic flow due to the

analysis ability and implementation ways. The traffic analysis will parse the payload of the packages, thus it will affect the package processing efficiency which need extra process, and the ever-increasing new protocols also affect the DFI/DPI devices efficiency. The flow tag is defined in RFC6437 and it only applies in IPv6 protocols. The flow tag changes along with the specific traffic flow and just like port. The flow tag can't identify the traffic flow independently and it must be used with source/destination IP addresses together. Because the flow tag is fixed in IPv6 header, it can identified easily, but it lacks of protect mechanism and there is no mechanism verifying its integrity.

In general, the current traffic recognition ways is limited in the analysis of traffic flows, they can't provide effective feedback data, so they can't support the self-adaptive network processing capability established by the operators.

4 Name mechanism for the video contents in ICN

The ICN includes many Named Data Object (NDO) and it turns the current "end host" network framework into "named information" framework. In ICN, NDO is the core concept and independent of IP address, which is the base of ICN network communication. The video traffic is the highest percentage traffic in the Internet traffics. As the network video gradually changes from standard definition video to high definition and ultra HD video. Some new video applications are rapidly popularized, such as short video application, video social contact application and some related video applications, and the video traffic is constantly growing. So it's necessary that the video content must be regarded as a special NDO class to have specialized design and consideration. The network video transmission mostly uses slice transmission mechanism such as HLS and DASH protocol. Based on the NDO granularity and transmission efficiency, we suggest that the NDO design will use a whole video file or video stream as a data unit and not take a slice as a NDO.

5 Design of multi-service tag

5.1 the design rules of multi-service tag

The design scheme of multi-service tag is a scheme just like URI hierarchy naming scheme and its design follows the following principles:

- a) no relationship with IP address or port number;
- b) one-to-one correspondence to the transferred content;
- c) stable in a traffic flow lifecycle;
- d) easily obtained and handled by the network operator;
- e) the tag can be recognized by the network, the network can draw up a strategy and adaptively transfer the content according to the tag information;
- f) confidence mechanism against tamper-proofing;
- g) decrease the complexity of network management.

5.2 the preliminary design of multi-service tag

Here we give a simple and preliminary design of multi-service tag, the scheme is not mature and may be changed along with the development of the new technologies.

The format of multi-service tag is as following:

xlables = base64(CID + content summary + type + random number + signature)

xlables: fixed string which identifies tags and encrypts the following information using base64.

CID: the identity of CP which is distributed by the tag service system in a unified way.

content summary: the summary is extracted according to the file content and corresponding to the file and actually is file hash. This field can be used to identify the same cached content.

type: the kinds of the transferred file, such as video, picture, document.

random number: it provides the signature identity

signature: it's produced according to the CID+content summary+type+file size+[code rate]+timestamp+random number and used to verify the validity of the tag.

6 System of multi-service tag

The system of multi-service tag is mainly made up by two function modulars:

1)generating modular: this modular is deployed at the network edge and interfaces with the operators. Its function is to generate multi-service tags aimed at the specific content and establish binding relationship between the video content and multi-service tag. This modular will establish matchup relation between the multi-service tag and video content actual store address in the operator network, and send this matchup information to the assembly modular.

2)assembly modular: this modular is deployed at the network center and responsible for collecting the matchup information between the multi-service tag and video content storage location which is sent by the generating modular. This modular will establish the whole network NDO routing table by collecting the multi-service tag and content storage address information all over the whole network and realize the content inquiry service and routing resolution service according naming information.

7 Design of routing and route resolution for the multi-service tag

The design of routing and route resolution for the multi-service tag will adopt RFC7927 Look-By-Name Routing LBNR scheme which can fully use the current network infrastructure.

1) The multi-service tag system will interface with the operator's CDN or cache system firstly, then the generating modular will scan the operator's content resource, establish the binding relationship between content resource and multi-service tag, generate the mapping relation for the network address and multi-service tag, send this information to the assembly modular and form the mapping relation table for the network address and multi-service tag in the assembly modular.

2) When the operator receives the user inquiry, it will extract the inquired naming information-multi-service tag through filter mechanism and send it to the assembly modular.

3) The assembly modular find the final address information related with the naming information through the mapping relation table of the network address and multi-service tag, and send this information to the user.

4) The user acquires the content through the network address.

8 Some application cases

8.1 content resource sharing across ISP network

The Internet video transmission usually uses the CDN technology and cache technology to provide service for users and the CP will deploy the CDN or cache nodes according to the user distribution in the operator network. In order to guarantee QoE, the CP will deploy CDN nodes with full resource in the network center and CDN nodes with hot resource at the network edge which usually locate in the operator's premises network. Each premises network operator has its own IP address field and the user's IP address is allocated by the premises network operator. In the current IP network, the CP can find the nearest resource only according to the IP address in the inquiry and then schedule the corresponding CDN node to serve the user, if the edge CDN node has no the resource asked by the user, the CP will haul the user inquiry back to the center CDN nodes with full resource and schedule the corresponding resource to serve the user, and this can easily form the network congestion of ISP haul-back route and increase the network delay. Though the different ISP premises networks have routing reachability, the content resource can't be sharing among different IPS.

Under the video scheduling mechanism based on the IP address, IP address will fragment the network resource and the same content will have many IP address or URL, thus CP or ISP have to use large storage resource to deploy the same hot content. IP address and URL are all the network address information independent of the content and the operator can't share the content through the address information.

In ICN, we can use the multi-service tag naming scheme to realize the content resource sharing among ISPs and form larger content resource sharing pool, thus all user can acquire the content in the pool and it breaks the IP-ISP resource closed mechanism. The multi-service tag assembly modular can acquire all ISP network resource information and the user can use this information to find the relevant content.

8.2 cache according to the content naming information

The cache technology is always one of the main technological means for decreasing inter-network settlement charge and enhancing QoE. The maximal challenge which the traditional cache technology faces is

that the repetitive contents waste the cache resource. The core technology of the traditional cache is to obtain URL contents and store them locally by monitoring the hot program's URLs through DPI. But the URL is not stable and the same contents may have different URLs. Though we can use DPI to decode the content and acquire partial content characteristics to compare, it has major limitations at decreasing the repetitive contents and greatly increases the computation complexity, what is more, the begin of the content is often advertisement or station caption and this makes content comparison different to work well. The multi-service tag contains the attribute information of carried content which is one-to-one correspondence to the content, then the cache system can use the tag as the base of comparison so as to quickly discover the repetitive contents and raise cache efficiency.

9 Security Considerations

TBD.

10 IANA Considerations

There is no IANA action in this document.

11 Acknowledgements

TBD.

12 References

12.1 Normative References

[RFC]7927, D. Kutscher, S., "Information-Centric Networking (ICN) Research Challenges", [RFC]7927, July 2016, <<https://www.rfc-editor.org/rfc/rfc7927.txt>>

Authors' Addresses

Yong Xia
China SARFT
Email: xiayong@abs.ac.cn

Shihui Duan
China Academy of Telecommunication Research of MIIT
Email: duanshihui@catr.cn

Shu Liu
China Academy of Telecommunication Research of MIIT
Email: liushu@catr.cn

Rachel Huang
Huawei
Email: rachel.huang@huawei.com

ICN Research Group
Internet-Draft
Intended status: Informational
Expires: December 28, 2017

Y. Zhang
D. Raychadhuri
WINLAB, Rutgers University
L. Grieco
Politecnico di Bari (DEI)
E. Baccelli
INRIA
J. Burke
UCLA REMAP
R. Ravindran
G. Wang
Huawei Technologies
A. Lindgren
B. Ahlgren
RISE SICS
O. Schelen
Lulea University of Technology
June 26, 2017

Design Considerations for Applying ICN to IoT
draft-zhang-icnrg-icniot-01

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 over a horizontal unified, defragmented IoT architecture. Such an architecture will make objects 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 mostly information-centric nature of the IoT system. To address this mismatch, an information-centric network (ICN) architecture can provide a common set of protocols and services, called 'ICN-IoT', which can be used to build IoT platforms. ICN-IoT leverages the salient features of ICN, and thus provides naming, security, mobility support, scalability, and efficient content and service delivery.

This draft summarizes general IoT demands, and covers the challenges and design considerations ICN faces to realize a ICN-IoT framework based on ICN architecture.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 28, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. IoT Motivation	3
2. Motivating ICN for IoT	4
3. IoT Architectural Requirements	9
3.1. Naming	9
3.2. Security and Privacy	10
3.3. Scalability	10
3.4. Resource Constraints	10
3.5. Traffic Characteristics	11
3.6. Contextual Communication	12
3.7. Handling Mobility	12
3.8. Storage and Caching	13
3.9. Communication Reliability	13
3.10. Self-Organization	14
3.11. Ad hoc and Infrastructure Mode	14

3.12. IoT Platform Management	15
4. State of the Art	15
4.1. Silo IoT Architecture	15
4.2. Application-Layer Unified IoT Solutions	16
4.2.1. Weaknesses of the Application-Layer Approach	17
4.2.2. Suitability of Delay Tolerant Networking(DTN)	19
5. Advantages of using ICN for IoT	19
6. ICN Design Considerations for IoT	21
6.1. Naming Devices, Data, and Services	21
6.2. Name Resolution	25
6.3. Security and Privacy	26
6.4. Caching	28
6.5. Storage	30
6.6. Routing and Forwarding	31
6.7. Mobility Management	32
6.8. Contextual Communication	33
6.9. In-network Computing	33
6.10. Self-Organization	34
6.11. Communications Reliability	35
6.12. Resource Constraints and Heterogeneity	35
7. Differences from T2TRG	36
8. Security Considerations	36
9. Conclusions	36
10. Acknowledgements	36
11. Informative References	37
Authors' Addresses	48

1. IoT Motivation

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 towards a more unified IoT architecture, in which a large number of objects connect to the Internet, available for interactions among themselves, as well as interactions with many different applications across boundaries of administration and domains. General IoT applications involve sensing, processing, and secure content distribution occurring at various timescales and at multiple levels of hierarchy depending on the application requirements. This requires the system to adopt a unified architecture providing pull, push and publish/subscribe mechanisms using application abstractions, common naming, payload, encryption and signature schemes. This requires open APIs to be generic enough to support commonly used interactions between consumers, content producer, and IoT services, as opposed to proprietary APIs that are common in today's systems. Building a unified IoT architecture, however, poses great challenges on the underlying network and systems. To name a few, it needs to support 50-100 Billion networked objects [1], many of which are mobile. The objects will have

extremely heterogeneous means of connecting to the Internet, often with severe resource constraints. Interactions between the applications and objects are often real-time and dynamic, requiring strong security and privacy protections. In addition, many IoT applications are inherently information centric (e.g., data consumers usually need data sensed from the environment without any reference to the sub-set of sensors that will provide the asked information).

Taking a general IoT perspective, we first motivate the discussion of ICN for IoT using well known scenarios. Then we discuss the IoT requirements generally applicable to many well known IoT scenarios. We then discuss how the current application-layer unified IoT architectures fail to meet these requirements. We follow this by key ICN features that makes it a better candidate to realize an unified IoT framework. We then discuss IoT design challenges from an ICN perspective and requirements posed towards its design.

2. Motivating ICN for IoT

ICN offers many features including name-based networking, content object security, caching, computing and storage, mobility, context-aware networking (see Section 3.6) and support for ad hoc networking features, all of which have to be realized in an application-specific means in the context of IP-IoT. These compelling features enable a distributed and intelligent data distribution platform to support heterogeneous IoT services with features like device bootstrapping with minimal configuration, simpler protocols to aid self-organizing among the IoT elements, natural support for compute and caching logic at strategic points in the network. We discuss these features through the following scenarios that are difficult to realize over IP today, and whose characteristics we argue match the features offered by ICN.

- o Smart Mobility: Smarter end-user devices and Machine-to-Machine (M2M) connection are undergoing a significant growth. By 2021, there will be more than 10 billion mobile devices and connection, including smartphones, tablets, wearables, vehicles [1]. Involved fields range from medical and healthcare, fitness, clothing, to environmental monitoring [40]. In particular, one of the most affected domain is transportation and the so-called Intelligent Transport Systems (ITS) [42]. It aims at providing multi-modal transportation, embracing public and private municipal, regional, national, trans-national vehicles and fleets. This extremely heterogeneous eco-system of means of transportation is made available to users and citizens through advanced services. These services are able to fulfill usability requirements while pursuing system level objectives, thus including: (i) the reduction of the 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 [117]. In this context, IoT technologies can play a pivotal, in particular, Traffic Management Systems (TMS) aided by IoT technologies are creating novel approaches to traffic modeling [47]. Moreover, such features enable advanced design paradigms (e.g., Mobility as a Service (MaaS) [39]) with huge implications in systems architectures [48]. As a consequence, smart mobility support can be a significant use case of ICN-IoT. The important ICN features that corroborate mobility support are:

- * The location independence of content allows one to manage consumer mobility in a simpler way than IP. Different from Mobile IP, that needs 'triangular routing' to locate moving hosts, ICN envisions that the consumers just needs to re-issue content requests after changing the attachment point [43];
- * Since content is not bound to a specific location, it can be cached anywhere in the network. This caching mechanism adds redundancy to the system. Therefore, if the producer loses connectivity while it is moving, a content request can be resolved to an intermediate node en-route or routed towards a caching node [43];
- * The content request-response communication paradigm decouples publications and subscriptions in time and space. Therefore, entities involved are not aware of each other and do not need to be connected at the same time [44];
- * The use of in-network Name Resolution Service design allows to identify content name's current location in the network, thanks to its network function of updating named entity location information [56].

From a technological perspective, open challenges are: (i) interoperability across different IoT technologies; (ii) namespace design able to harmonize ITS standards; (iii) scalable data-sharing model across real-time (and non real-time) traffic sources; (iv) definition of travel-centric services based on ICN-IoT; (v) seamless support to mobility; (vi) content authentication and cryptography.

- o Smart Building: Buildings are gaining smart capabilities that allow to enhance comfort, provide safety and security, manage efficiently energy [101]. In particular, smart buildings are no longer simple energy consumer, but can also be prosumers with on-site energy generation systems. These systems can improve

building's usability towards: (i) Smart heating, ventilation, and air conditioning (HVAC), (ii) Smart lightings, (iii) Plug loads, (iv) Smart windows. The main requirements of those sub-systems are [101]: (i) context awareness; (ii) resource-constrained devices; (iii) interoperability across heterogeneous technologies; (iv) security and privacy protection. The ICN paradigm could ease the fulfillment of those requirements because, usually, smart building services are information centric by design: this means that every time an autonomic management loop is established within the smart building to control some physical variables of interest, the information exchanged between users, sensors, actuators, and controllers do not immediately translate to specific nodes within the building but could be provided by multiple sensors / gateways. The relevance of ICN in Smart Building is recognized in literature with reference to the several frameworks deployed in various environments. For instance, in [61], nodes are distributed in different rooms, floors, and buildings of a campus university and their energy consumption and individual behavior are monitored. Smart home application is investigated in [103], by evaluating data retrieval delay and data packet loss. Moreover, [104] designed and tested lighting control over NDN in a theater. In this context, specific ICN challenges are: (i) design of a scalable namespace for uniquely identifying the information of interest, (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.

- o Smart Grid: Smart Grids are increasingly transforming into cyber-physical systems [18] with the goal of maximum automation towards efficiency and minimal human intervention. The system is very complex comprising of power distribution grids, end user applications (e.g. EV charging systems, appliances etc), smart monitoring systems (spanning end user and the power grids), heterogeneous energy producing sources (including prosumers), and load distribution and balancing systems. Current smart grid systems are managed using Supervisory Control and Data Acquisition (SCADA) frameworks that are centralized and highly restrictive unidirectional communication support [19]. Hence the requirement is towards : 1) greater flexibility to distribute the energy from the feeder through real-time reconfiguration of multiple monitoring devices (e.g. phasor measurement units (PMUs)), and management operations which require efficient data delivery infrastructure; 2) large scale data delivery infrastructure, which also include latency sensitive applications, inter-connecting heterogeneous smart grid producing, monitoring and consuming end user devices; 3) Resiliency, which is critical to

the operation and protection of the grid; 4)Security, to protect mission critical grid applications from network intrusions ; 5) understanding machine-to-machine traffic patterns for optimal placement of storage and computing for maximum efficiency. Smart grids can benefit from ICN in the following ways [20] :

- * Smart grid will benefit from naming content than hosts, as it is more likely that data generated by one subsystem will be useful for multiple entities. Further, naming content allows to enable many-to-many model of communication, which is very in-efficient in host-centric architectures.
 - * ICN features of in-network computing, storage and caching will enable better use of network resources and benefit diverse application needs varying from applications that has low bitrate and is latency tolerant (e.g. smart grid and energy pricing) to higher data rate ones with stringent delay/disruption requirements (e.g. synchrophasor measurements). Also it is typical in smart grid systems to have applications consuming the same data at different rates in which case in-network caching and computing could help.
 - * Host-centric networking exposes a mission critical infrastructure like smart grid infrastructure to intrusion and DOS attacks, this is directly related to exposing the IP addresses of critical applications and subsystems. Naming content, service or device de-couples it from the location, reducing the exposure to target a specific smart grid subsystem based on a geographical context.
 - * ICN's name based networking offers the potential for self-configuration both during bootstrapping and during the regular operation of the grid allowing scalable operation and self-recovery during faults or maintenance tasks in the system.
- o Smart Industrial Automation : In a smart and connected industry environment, there is a multitude of equipment with sensors that generate large volumes of data during normal operation. This range from highly time-critical data for real-time control of production processes, to less time-critical data that is collected to central cloud environment for control room monitoring, to pure log data without latency requirements that is mainly kept for a posteriori analysis. Industrial wireless networks are harsh environments with lots of potential interference at the same time as hard reliability and real-time requirements are placed by many applications. This means that available network capacity is not always high, so congestion is likely to be experienced by traffic with less stringent timing requirements. One such example is when

errors occur in the production process, a mobile workforce will need to investigate the problem on-site and will need high resolution data from the faulty machine as well as other process data from other parts of the plant. The mobile workforce will locally perform diagnostics or maintenance and they rely on the information from the production system both for safety and to solve any issues in the plant. They rely on both historical data in order to pinpoint the root cause of the problems, as well as the current data flows in order to assess the present state of the equipment under control. High resolution measurements are generated close to the mobile workforce while the historic data has to be retrieved from the historian servers. Multiple workers involved in the process will access the same data, possibly with a slight time-shift. The network thus need to support a mobile users to get access to data flows in a way suitable for their physical location and task requirements. Introducing ICN functionality into the system can introduce several benefits that will 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 OPC Foundation [10] maps well to the CCN/NDN name space.
- * ICN provides the possibility to get newest data without knowing the location of the caches or whether a particular piece of data is available locally or in a central repository. Also gives the possibility to get either local high-resolution data or remote low-resolution data (no need to store all data centrally, which is maybe not even possible due to large data volumes). May require known naming conventions or routing policies that can route interests to the right location.
- * Reduces network usage 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 is inherently supported without the need to maintain connection state.
- * Removing tedious configurations in clients since that is provided by the infrastructure.
- * Allow sharing of large data volumes between users that are in physical proximity without introducing additional traffic on the backbone.

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

3. IoT Architectural Requirements

A unified IoT platform has to support interactions among a large number of mobile devices across the boundaries of organizations and domains. As a result, it naturally poses stringent requirements in every aspect of the system design. Below, we outline a few important requirements that a unified 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 each device, data items generated by these devices, or a group of devices towards a common objective. Naming has the following requirements. Firstly, names need to be persistent against dynamic features that are common in IoT systems, such as lifetime, mobility or migration. Secondly, names that are derived from the keys need to be self-certifying, for both device-centric communication and content-centric communication. For device-centric communication, the binding between device names and the device must be secure. For content-centric communication, the binding between the names and the content has to be secure. Thirdly, names usually serve multiple purposes: routing, security (self-certifying) or human-readability. For IoT applications, the choice of flat versus human readable names needs to be made considering application and network requirements such as privacy and network level scalability, and the name space explosion that may occur because of complex relationship between name hierarchies [120] which might confound application logic. In order to ensure the trustworthiness of the names, a name certificate service (NCS) needs to be considered. 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's end. In short, the NCS must provide services analogous to those provided by a Public Key Infrastructure (PKI). In ICN, users may either generate their own public keys and submit them to the NCS for registration, or may contact the NCS to acquire public keys. Consequently, the NCS publishes approved cryptographic suites, object categories and object description formats, as well as allows users to self-certify themselves.

3.2. Security and Privacy

A variety of security and privacy concerns exist in IoT. For example the unified IoT architecture makes physical objects accessible to applications across organizations and domains. Further, it often integrates with critical infrastructure and industrial systems with life safety implications, bringing with it significant security challenges and regulatory requirements [13], as will be discussed in Section 6.3. 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: (1) privacy of data producer/consumer that is directly related to each individual vertical domain such as health, electricity, etc., (2) privacy of data content, and (3) privacy of contextual information such as time and location of data transmission [65].

3.3. Scalability

Cisco predicts there will be around 50 Billion IoT devices such as sensors, RFID tags, and actuators, on the Internet by 2020 [1]. As mentioned above, a unified IoT platform needs to name every entity such as data, device, service etc. Scalability has to be addressed at multiple levels of the IoT architecture including naming, security, name resolution, routing and forwarding level. Mobility adds further challenge 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. In addition scalability is also affected because of IoT system specific features such as IoT resource object count, state and rate of information updates 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 the most resource-constrained and type 3 the most resource-rich [45]. In general, there are the following types of resources: 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 [46]. Flexible techniques to collect the relevant information are required, and uploading every single produced data to a central server is undesirable. Computing constraints limit the type and amount of processing these devices can perform. As a result, more complex

processing needs to be conducted in cloud servers or at opportunistic points, example at the network edge, hence it is important to balance local computation versus communication cost.

Storage constraints of the IoT devices limit the amount of data that can be stored on the devices. This constraint means that unused sensor data may need to be discarded or stored in aggregated compact form time to time. Bandwidth constraints of the IoT devices limit the amount of communication. Such devices will have the same implication on the system architecture as with the power constraints; namely, we cannot afford to collect single sensor data generated by the device and/or use complex signaling protocols. It is also worth mentioning that idle chatter in the background is strongly discouraged to maintain connectivity or other volatile state.

User interface constraints refer to whether the device is itself capable of directly interacting with a user should the need arise (e.g., via a display and keypad or LED indicators) or requires the network connectivity, either global or local, to interact with humans.

The above discussed device constraints also affect application performance with respect to latency.

3.5. Traffic Characteristics

IoT traffic can be broadly classified into local area traffic and wide area traffic. Local area traffic is among nearby devices. For example, neighboring cars may work together to detect potential hazards on the highway, sensors deployed in the same room may collaborate to determine how to adjust the heating level in the room. These local area communications often involve data aggregation and filtering, have real time constraints, and require fast device/data/service discovery and association. At the same time, the IoT platform has to also support wide area communications. For example, in Intelligent Transportation Systems, re-routing operations may require a broad knowledge of the status of the system, traffic load, availability of freights, whether forecasts and so on. Wide area communications require efficient data/service discovery and resolution 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 volume in some applications such as [2], which means that we have to optimize the bandwidth/energy consumption in both directions. Further, IoT traffic demonstrates certain periodicity and burstiness [2]. As a

result, when provisioning the system, the shape of the traffic volume has to be properly accounted for.

3.6. Contextual Communication

Many IoT applications rely on dynamic contexts in the IoT system to initiate, maintain and terminate communication among 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 in the same location. There are two types of contexts: long-term quasi static contexts and short-term dynamic contexts. In this draft, we focus on the latter, which are more challenging to support, requiring fast formation, update, lookup and association. For example, cars traveling on the highway may form a "cluster" based upon their temporal physical proximity as well as the detection of the same event. These temporary groups are referred to as contexts. IoT applications need to support interactions among the members of a context, as well as interactions across contexts.

Temporal context can be broadly categorized into two classes, long-term contexts such as those that are based upon social contacts as well as stationary physical locations (e.g., sensors in a car/building), and short-term contexts such as those that are based upon temporary proximity (e.g., all taxicabs within half a mile of the Time Square at noon on Oct 1, 2013). Between these two classes, short-term contexts are more challenging to support, requiring fast formation, update, lookup and association.

3.7. Handling Mobility

There are several degrees of mobility in a unified IoT architecture, ranging from static as in fixed assets to highly dynamic in vehicle-to-vehicle environments.

Mobility in the IoT architecture can mean 1) the data producer mobility (i.e., location change), 2) the data consumer mobility, 3) IoT Network mobility (e.g., a body-area network in motion as a person is walking); and 4) disconnection between the data source and destination pair (e.g., due to unreliable wireless links). The requirement on mobility support is to be able to deliver IoT data below an application's acceptable delay constraint in all of the above cases, and if necessary to negotiate different connectivity or security constraints specific to each mobile context. More detailed discussions are presented in Section 6.7.

3.8. Storage and Caching

Storage and caching plays a very significant role depending on the type of IoT ecosystem, also a function subjected to privacy and security guidelines. Caching is usually done for increasing data availability in the network and reliability purposes, especially in wireless scenarios in the network access. Storage is more important for IoT, storing data for long term analysis. Data is stored in strategic locations in the network to reduce control and computation overhead. In a unified IoT architecture, depending on application requirements, content caching will be strictly driven by application level policies considering privacy requirements. If for certain kind of IoT data pervasive caching is allowed, intermediate nodes don't need to always forward a content request to its original creator; rather, receiving a cached copy is sufficient for IoT applications. This optimization may greatly reduce the content access latencies.

Furthermore considering hierarchical nature of IoT systems, ICN architectures enable flexible heterogeneous and potentially fault-tolerant approach to storage providing persistence at multiple levels.

Hence in the context of IoT while ICN allows resolution to replicated stored copies, it should also strive for the balance between content security/privacy and regulations considering application requirements.

3.9. Communication Reliability

IoT applications can be broadly categorized into mission critical and non-mission critical. 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 probability of error during information transfer. To summarize, reliable communication desires the following capabilities for the underlying system: (1) seamless mobility support under normal operating conditions, (2) efficient routing in the presence of intermittent disconnection, (3) QoS aware routing, (4) support for redundancy at all levels of a system (device, service, network, storage etc.), and (5) 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

The unified IoT architecture should be able to self-organize to meet various application requirements, especially the capability to quickly discover heterogeneous and relevant (local or global) devices/data/services based on the context. This discovery can be achieved through an efficient publish-subscribe service, or through private community grouping/clustering based upon trust and other security requirements. In the former case, the publish-subscribe service must be efficiently implemented, able to support seamless mobility, in- network caching, name-based routing, etc. In the latter case, the IoT architecture needs to discover the private community groups/clusters efficiently.

Another aspect of self-organization is decoupling the sensing Infrastructure from applications. In a unified IoT architecture, various applications run on top of a vast number of IoT devices. Upgrading the firmware of the IoT devices is a difficult work. It is also not practical to reprogram the IoT devices to accommodate every change of the applications. The infrastructure and the application specific logics need to be decoupled. A common interface is required to dynamically configure the interactions between the IoT devices and easily modify the application logics on top of the sensing/actuating infrastructure [30] [31].

3.11. Ad hoc and Infrastructure Mode

Depending upon whether there is communication infrastructure, an IoT system can operate either in ad-hoc or infrastructure mode.

For example, a vehicle may determine to report its location and status information to a server periodically through cellular connection, or, a group of vehicles may form an ad-hoc network that collectively detect road conditions around them. In the cases where infrastructure is unavailable, one of the participating nodes may choose to become the temporary gateway.

The unified IoT architecture needs to design a common protocol that serves both modes. Such a protocol should address the challenges that arise in these two modes: (1) scalability and low latency for the infrastructure mode and (2) 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

An IoT platforms' service, control and data plane will be governed by its own management infrastructure which includes distributed and centralized middleware, discovery, naming, self-configuring, analytic functions, and information dissemination to achieve specific IoT system objectives [25][26][27]. Towards this, new IoT management mechanisms and service metrics need to be developed to measure the success of an IoT deployment. Considering an IoT systems' defining characteristics such as, its potential large number of IoT devices, objective to save power, mobility, and ad hoc communication, autonomic self-management mechanisms become very critical. Further considering its hierarchical information processing deployment model, the platform needs to orchestrate computational tasks according to the involved sensors and the available computation resources which may change over time. An efficient computation 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, towards a unified IoT architecture in which the existing silo IoT systems, as well as new systems that are rapidly deployed. By unified, we mean all the application and network components that use common APIs to interact with each other. This will make their data and services accessible to general Internet applications (as in ETSI- M2M and oneM2M standards). In such a unified architecture, resources can be accessed over Internet and shared across the physical boundaries of the enterprise. However, current approaches to achieve this objective are mostly based upon service overlays over the Internet, whose inherent inefficiencies due to IP protocol [56] hinders the architecture from satisfying the IoT requirements outlined earlier, particularly in terms of scalability, security, mobility, and self-organization, discussed more in Section 4.2.

4.1. Silo IoT Architecture

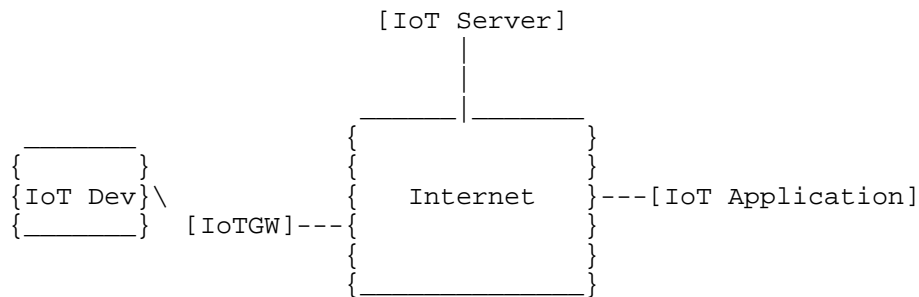


Figure 1: Silo architecture of standalone IoT systems

A typical standalone IoT system is illustrated in Figure 1, which includes devices, a gateway, a server and applications. Many IoT devices have limited power and computing resources, unable to directly run normal IP access network (Ethernet, WIFI, 3G/LTE etc.) protocols. Therefore they use the IoT gateway to the server. Through the IoT server, applications can subscribe to data collected by devices, or interact with devices.

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 the device-level abstraction, instead of information driven, which may sometimes lead to a fragmented protocol space that requires a higher-level solution for better interoperability.

4.2. Application-Layer Unified IoT Solutions

The current approach to a unified IoT architecture is to make IoT gateways and servers adopt standard APIs. IoT devices connect to the Internet through the standard APIs and IoT applications subscribe and receive data through standard control and data APIs. Building 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], oneM2M[4]. Network operators can use frameworks to build common IOT gateways and servers for their customers. In addition, IETF's CORE working group [5] is developing a set of protocols like CoAP (Constrained Application Protocol) [78], that is a lightweight protocol modeled after HTTP [79] and adapted specifically for the Internet of Things (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 Machine-to-Machine communications, as well as IoT. For example, oneM2M (which is one of leading standards for unified M2M architecture) has both the protocol bindings to HTTP and CoAP for its primitives. Figure 2 shows the architecture adopted in this approach.

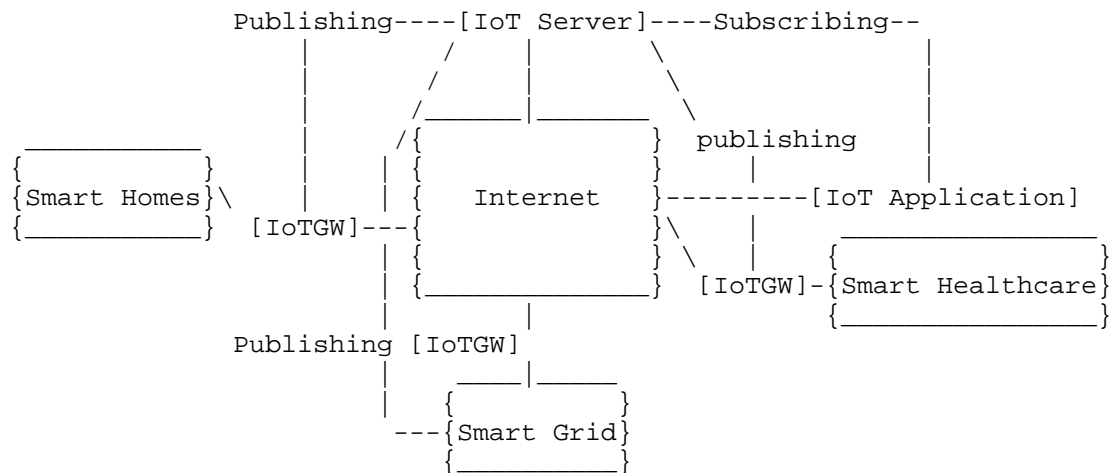


Figure 2: Implementing an open IoT architecture through standardized APIs on the IoT gateways and the server

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 we outlined in Section 3:

- o Naming. In current application-layer IoT systems the naming scheme is host centric, i.e., the name of a given resource/service is linked to the device that can provide it. In turn, device names are coupled to IP addresses, which are not persistent in mobile scenarios. On the other side, in IoT systems the same service/ resource could be offered by different devices.
- o Security and Trust. In IP, the security and trust model is based on session established between two hosts. Session-based protocols rely on the exchange of several messages before a secure session is established. Use of such protocols in constrained IoT devices

can have serious consequences in terms of energy efficiency because transmission and reception of messages is often more costly than the cryptographic operations. The problem may be amplified with the number of nodes the constrained device has to interact with because of both the computation cost and per session key state required to be managed by the constrained device. Also the trust management schemes are still relatively weak, focusing on securing communication channels rather than managing the data that needs to be secured directly. Though key management in ICN is no less complex than in host based interactions, the benefits is associated with the security credentials in the content instead of the host. Trust is via keys that are bound to names through certificates whose private keys are held by the principals of the system, with IP focusing on the channel model of security while ICN focusing on the object model.

- o 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. Further the Layer 2/3 management, and application-layer addressing and forwarding required to deploy current IoT solutions limit the scalability and management of these systems.
- o Resource constraints. The application-layer approach requires every device to send data to an aggregator, gateway or to the IoT server. Resource constraints of the IoT devices, especially in power and bandwidth, could seriously limit the performance of this approach. On the other hand, ICN supports in-network computing/caching/storage, which can alleviate this problem.
- o Traffic Characteristics. In this approach, applications are written in a host-centric manner suitable for point-to-point communication. IoT requires multicast support that is challenging the application-layer based IoT systems today, which has only limited deployment in current Internet.
- o Contextual Communications. This application-layer based IoT approach may not react to dynamic contextual changes in a timely fashion. The main reason is that context lists are usually kept at the IoT server in this approach, and they cannot help efficiently route requests information at the network layer.
- o 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 enabled via name-based routing or name-based lookup.

- o Self-Organization. The application-layer approach is topology-based as it is bound to IP semantics, and thus does not sufficiently satisfy the self-organization requirement. In addition to topological self-organization, IoT also requires data- and service-level self-organization [97], which is not supported by this approach.
- o Ad-hoc and infrastructure mode. As mentioned above, the overlay-based approach lacks self-organization, and thus does not provide efficient support for the ad-hoc mode of communication.

4.2.2. Suitability of Delay Tolerant Networking(DTN)

In [21][22], delay-tolerant networking (DTN) has been considered to support future IoT architecture. DTN was created to support information delivery in the presence of network disruptions and disconnections, which has 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 (endpoint IDs), which usually correspond to a host or a service, and is mainly a way to transport data, while ICN provides a different paradigm centered around named data that addresses additional issues for IoT applications [23] through features such as information naming, information discovery, information request and dissemination. Also, the endpoint IDs could be used to also 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 transport would however still require other components from an ICN architecture such as naming conventions, so since the exact transport is not a major focus of the issues in this draft, most of the discussions are applicable to a generic ICN architecture in general.

5. Advantages of using ICN for IoT

A key concept of ICN is the ability to name data independently from the current location at which it is stored, which simplifies caching and enables decoupling of sender and receiver. 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 general benefits that ICN could provide to IoT networks.

- o Naming of Devices, Data and Services. The heterogeneity of both network equipment deployed and services offered by IoT networks 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 to the application layer the task to name data and services. This causes different applications to use different naming schemes, and no consistent mapping from application layer names to network names exist. In many common applications of IoT networks, data and services are the main goal, and ICN provides an intuitive way to name those in a way that can be utilized on the network layer as well. Communication with a specific device is often secondary, but when needed, the same ICN naming mechanisms can be used. The network distributes content and provides a service, instead of only sending data between two named devices. In this context, data content and services can be provided by several devices, or 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.

- o Security and privacy. ICN advocates the model of object security to secure data in the network. This concept is based on the idea of securing information objects unlike 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 keys established at the application level. All of this means that the actual transmission of data does not have to be secured as the same security mechanisms protect the data after generation until consumed by a client, regardless of 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 could potentially be retrieved from a cache in a different location. Having a trust relationship with each of these different caches is not realistic. Through Name-Data Integrity, ICN automatically guarantees data integrity to the requester 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 in the device constraints are severe enough that it is not able to perform the required cryptographic operations for Object Security, 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; cryptographic hash of a public key also enables them to be self-certifying, i.e., authenticating the resource object does not require an external authority [25][26].

- o 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 systems, because of their resource constraints. Wireless bandwidth and power supply can be limited for multiple devices sharing a communication channel, and for small mobile devices powered by batteries. In this case, avoiding unnecessary transmissions with IoT devices to retrieve and distribute IoT data to multiple places is important, hence processing and storing such content in the network can save wireless bandwidth and battery power. Moreover, as for other types of networks, applications for IoT networks requiring shorter delays can benefit from local caches and services to reduce delays between content request and delivery.
- o Decoupling between Sender and Receiver. IoT devices may be mobile and face intermittent network connectivity. When 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.

6. 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 that will be involved in order to support large scale IoT deployment with diverse application requirements.

Though ICN integrates content/service/host abstraction, name-based routing, compute, caching/storage as part of the network infrastructure, IoT requires special considerations given heterogeneity of devices and interfaces such as for constrained networking [61][119][121], data processing, and content distribution models to meet specific application requirements which we identify as challenges in this section.

6.1. Naming Devices, Data, and Services

The ICN approach of named data and services (i.e., device independent naming) is typically desirable when retrieving IoT data. However, data centric naming may also pose challenges.

- o Naming of devices: Naming devices can be useful in an IoT system. For example, actuators require clients to act on a specific node of the deployed network, e.g. to switch it on or off; or it could be necessary to access to a particular device for administrator

purposes. This can be achieved through the specific name that uniquely identify the network entity of interest. Moreover, a persistent name allows a device to change attachment point without losing its identity. A friendly way to address devices is using contextual hierarchical names, where the same types of names as for data objects can be used. To ensure that the device is always reached, it is important that it is possible to disable caching and request aggregation, if used, for such names.

- o Size of data/service name: Content name can have variable length. 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), its length can reach high values. In particular, according to the specific application, content name size can exceed Data size. This can be the case of IoT sensed values that usually consist in few bytes: data could be as small as a short integer in case of temperature values, or one-byte in case of control messages of an actuator state (on/off). Moreover, a too long name would probably incur in fragmentation at the link layer, and related problems such as, several transmissions, delay and security issues. Viable solutions to handle ICN packets fragmentation and reassembly have been investigated in literature. For instance, the work in [105] proposes to perform the operations hop-by-hop: 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, thus 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 content is the one requested. This is only possible in contexts where the requested object is already existing, and where there is a directory service to look up names or learned through a manifest service. This approach is suitable for systems with large data objects where it is important to verify the content.
- o Hierarchical names: The use of hierarchical names, such as in the CCN and NDN architectures make it easier to create names a priori and also provides a convenient way to use the same naming scheme for node names. Since the names are not self-certifying, this will require other mechanisms for verification of object integrity. If routing is also done on the hierarchical names, the system will lose some of its location independence and caching will mostly only be done on the path to the publisher.
- o Semantic and Metadata based content name: A semantic-based naming approach can allow a successful name retrieving through keywords

(for example, 'noise level at position X'), even if a perfect matching of name is not available [62]. Moreover, enriching contents with metadata allows to better describe them and to establish association between similar ones. However this mechanism require more advanced functionality for matching of such metadata in data objects to the semantics of the name (such as comparing the position information of an object with the position information of the requested name). The need for such potentially computationally heavy tasks in intermediate nodes in the network may be considered understanding the trade-offs in terms of application and network performance.

- o Naming of services: Similar to naming of devices or data, services can be referred to with a unique identifier, provided by a specific device or by someone assigned by a central authority as the service provider. It can however also be a service provided by anyone meeting some certain metadata conditions. Example of services include content retrieval, that takes a content name/description as input and returns the value of that content, and actuation, that takes an actuation command as input and possibly returns a status code afterwards.
- o Trust: Names can be used to verify the authenticity and integrity of the data. To provide security functionalities through names, it is possible to use different approaches. On one hand, hierarchical, schematized, Web-of-Trust models allow the public key verification. On the other hand, self-certifying names allow in-network integrity check of the name-key or name-content binding without the need of a Public Key Infrastructure (PKI) or other third party to establish whether the key is trustworthy or not. This can be realized (i) directly: the hash of the content is bound to the name; or (ii) indirectly: first, the hash of the content is signed with the secret key of the publisher, then the public key of the publisher and the signed hash are bound to the name [44]. The hash algorithm can be applied to already existing contents and where there is a directory service or manifests to look up names. In case of contents not yet published, but generated on demand, the hash cannot be known a priori. Thus, different trust mechanisms should be investigated. Moreover, self-certified names approach can hide content semantics, thus making names less human friendly. Since trends show that users prefer to find contents through search engine using keywords, non-human-friendly names could be a barrier unless the content is enriched with keywords. But, this problem does not concern M2M applications. In fact, human-readable names may not be useful in a context of just communicating machines.

- o Flexibility: Further challenges arise for hierarchical naming schema: referring to requirements on "constructible names" and "on-demand publishing" [35][36]. 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 also using partially specified names. The latter refers the possibility to request a content that has not yet been published in the past, thus triggering its creation.
- o Scoping : From an application's point of view, scopes are used to gather related data. From the network's perspective, instead, scopes are used to mark where the content is available[65]. For instance, nodes involved in caching coordination can vary according to scope[66]. As a consequence, scoping allows to limit packet request propagation, improving bandwidth and energy resources usage, and control content dissemination thanks to access control rules, different for each scope[64]. However, 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 to limit interest propagation, provide a simple means to attain context-sensitive communication, etc. Finally, perimeter- and channel-based access control is often violated in 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 or more importantly violate privacy, 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 ICN naming scheme such as Attribute-Based Encryption [63] and access control delegation scheme [64]. In the first solution, a Trusted Third Party assigns a set of attributes to each network entity. Then, a publisher (i) encrypts the data with a random key; (ii) generates the metadata for the decryption phase; (iii) creates an access policy used to encrypt the random key; (iv) appended 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 Provider). 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.

6.2. 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 [57] [69] [70] [91]. 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 [50] [52]:

- o 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 [57]. 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 [86] 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 [95]. In addition, fast name lookup are necessary to ensure soft/hard real time services [106][107][108]. This challenge is especially important

for applications with stringent latency requirements, such as health monitoring, emergency handling and smart transportation [109].

- 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 [97].
- 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.

6.3. 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 [74], while at the same time the 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 TLS/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 [83], 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 data and

services are the main goal and specific communication between two devices is secondary. 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 [80] 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.

- o In the resource-constrained nodes, energy limitation is the biggest challenge. Moreover, 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 [75] [55].
- o In the infrastructure part, we have several new threats introduced by ICN-IoT [85] particularly in architectures employing name resolution service [119]. 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 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.
- o As far as the IoT application server is concerned, data privacy is one of the biggest concerns. 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.

6.4. Caching

In-network caching helps bring data closer to consumers, but its usage differs in constrained and infrastructure part of the IoT network.

Caching in ICN-IoT faces several challenges:

- o An important challenge is to determine which nodes on the routing path should cache the data. According to [52], caching the data on a subset of nodes can achieve a better gain than caching 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 [53], selecting a random router to cache data is as good as caching the content everywhere. In [88], the authors suggest that edge caching provides most of the benefits of in-network caching typically discussed in NDN, with simpler deployment. However, it and other

papers consider workloads that are analogous to today's CDNs, not the IoT applications considered here. Further work is likely required to understand the appropriate caching approach for IoT applications.

- o Another challenge in ICN-IoT caching is what to cache for IoT applications. In many IoT applications, customers often access a stream of sensor data, and as a result, caching a particular sensor data item for longer time may not be beneficial. In [90], proposed a caching scheme that ensures that older instances of the same sensor stream were first to be evicted from the cache when needed. In [55], the authors suggest to cache IoT services on intermediate routers, and in [57], the authors suggest to cache control information such as pub/sub lists on intermediate nodes. In addition, it is yet unclear 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 repeated actuation requests within a given time period), or have little meaning at all if actuation uses authenticated requests as in [89].
- o Another challenge is that the efficiency of distributed caching may be application dependent. When content popularity is heterogeneous, some content is often requested repeatedly. In that case, the network can benefit from caching. Another case where caching would be beneficial is when devices with low duty cycle are present in the network and when access to the cloud infrastructure is limited. In [90], it is also shown that there are benefits to caching in the network when edge links are lossy, in particular if losses occur close to the content producer, as is common in wireless IoT networks. 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 request and later. It may also be less beneficial to have caches distributed throughout ICN nodes 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 variable content/state. For example, when the last value for a sensor reading is requested or desired, the returned data should change every time the sensor reading is updated. In that case, ICN caching may increase the risk that cache inconsistencies result in old data being returned.

6.5. Storage

Storage is useful for IoT systems both at longer and small time scales.

Long terms storage can be distributed at vantage points including both the edge and the main IoT service aggregation points such as in the data centers, the difference being in the size of data, processing intelligence and heterogeneity of information that has to be dealt at the two points. The purpose of long terms storage at the edge is to analyze, filter, aggregate and re-publish data for consumption by either by the parent service components or directly by the consumers. The aggregation service points, republish data to be presented as part of the global pub/sub service to interested consuming parties. Long term storage for IoT data also serves the purpose of data backup and replication. Specifically, we face several issues here. Firstly, we need to decide how many replicas we should have for each stream of IoT data, and where we should store these replicas. Given that many IoT applications consume data locally, storage locations should be kept near to data sources as well. Since IoT data are mostly appended to the end of a stream, instead of being updated, managing multiple replicas becomes easier. Secondly, we need to adopt a mechanism that can efficiently route traffic to the nearest data replica. ICN provides several solutions to this problem. For example, global name resolution service (GNRS) can keep track of each replica's location [56].

Short-term in-network storage (here storage refers to temporary buffer when an outgoing link is not available) helps improve communication reliability, especially when network links are unreliable, such as wireless links. ICN-IoT could adopt a generalized storage-aware routing algorithm to support delay and disruption tolerance in the routing layer. Each router employs in-network storage that facilitates store vs. forward decisions in response to varying link quality and disconnections [111]. These decisions are based on both short-term and long-term path quality metrics. In addition, packets along paths that become disconnected are handled by a disruption tolerant networking (DTN) mode of the protocol with delayed delivery and replication features. In particular, each router maintains two types of topology information: (i) An intra-partition graph is formed by collecting flooded link state advertisements which carry fine-grained, time-sensitive information about the intra-network links; (ii) A DTN graph is maintained via epidemically disseminated link-state advertisements which carry connection probabilities between all nodes in the network. In-network storage faces the following challenges: (1) when to store and how long to store the data, and (2) the next step after the short-term storage. In [90] the authors also shows 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.

6.6. Routing and Forwarding

ICN-IoT supports both device-to-device (D2D) communication and device-to-infrastructure (D2I) communication. Some D2D communications are within a single IoT domain, while others might cross IoT domains involving 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 sensor network popular routing protocols such as RPL [81], AODV[82], etc. The main challenge it faces is the resource constraint of the IoT nodes. In order to address this challenge, we could adopt a light-weight, much shorter ICN name for each communicating party within an IoT domain (see Section 6.12 for details). Before we leave the IoT domain, the gateway node will translate the party's short ICN name to its original ICN name. Data forwarding in the ICN infrastructure part can adopt either direct name-based routing or indirect routing using a name resolution service (NRS).

- o In direct name-based routing, packets are forwarded by the name of the data [91][61][71] or the name of the destination node [72]. Here, the main challenge is to keep the ICN router state required to route/forward data low. This challenge becomes more serious when a flat naming scheme is used due to the lack of aggregation capabilities.
- o In indirect routing, packets are forwarded based upon the locator of the destination node, and the locator is obtained through the name resolution service. In particular, the name-locator binding can be done either before routing (i.e., static binding) or during routing (i.e., dynamic binding). For static binding, the router state is the same as that in traditional routers, and the main challenge is the need to have fast name resolution, especially when the IoT nodes are mobile. For dynamic binding, ICN routers need to main a name-based routing table, hence the challenge of keeping the state information low. At the same time, the need of fast name resolution is also critical.

6.7. Mobility Management

Considering the diversity of IoT applications mobility ranges from tracking sensor data from mobile human beings to large fleets of diverse mobile elements such as drones, vehicles, trucks, trains associated with a transport infrastructure. These mobility could be over heterogeneous access infrastructure ranging from short range 802.15.4 to cellular radios. Further, handling information delivery in ad hoc setting involving vehicles, road side units (RSU) and the corresponding infrastructure based services offers more challenges. ICN architectures has generally been shown to handle consumer and producer mobility [59], and even suitability to V2V scenarios [60]. Networking tools to handle mobility varies with application requirements, which varies from being tolerant to packet losses and latency to those that are mission critical with stringent requirement on both these QoS metrics.

Related to this, the challenge is to quantify the cost associated with mobility management both in the control and forwarding plane, to handle both static binding versus dynamic binding (dynamic binding here refers to enabling seamless mobility) of named resources to its location when either or both consumer and producer is mobile.

During a network transaction, either the data producer or the consumer may move away and thus we need to handle the mobility to avoid information loss. ICN may differentiate mobility of a data consumer from that of a producer:

- o When a consumer moves to a new location after sending out the request for Data, the Data may traverse to the previous point of attachment (PoA) but leaving copies of it through its previous path, which can be retrieved by the consumer by retransmitting its request, a technique used by direct routing approach. Indirect routing approach doesn't differentiate between consumer and producer mobility [91], as it only requires an update to the name resolution system, which can update the routers to rebind the named resource to its new location, while using late-binding to route the packet from the previous PoA to the new one.
- o If the data producer itself has moved, the challenge is to control the control overhead while searching for a new data producer (or for the same data producer in its new position) [58]. To this end, flooding techniques could be used rediscover the producer, or the direct routing techniques can be enhanced with late-binding feature to enable seamless mobility [59].

6.8. Contextual Communication

Contextualization through metadata in ICN control or application payload allows IoT applications to adapt to different environments. This enables intelligent networks which are self-configurable and enable intelligent networking among consumers and producers [55]. For example, let us look at the following smart transportation scenario: "James walks on NYC streets and wants to find an empty cab closest to his location." In this example, the context is the relative locations of James and 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, naming conventions could be used to 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 in the 100m grid area 4QFJ 123 678 of 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 is very challenging:

- o We need to have a fast context resolution service through which the involved IoT devices can continuously update its contextual information to the application (e.g., each taxi's location and James's information in the above example). Or, in the namespace driven approach, mechanisms for continuous nearest neighbor queries in the namespace need to be developed.
- o The difficulty of this challenge grows rapidly when the number of devices involved in a context as well as the number of contexts increases.

6.9. 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, in which each sensor node or ICN router implements context reasoning [55]. Another major purpose of in-network computing is to filter and cleanse sensed data in IoT applications, that is critical as the data is noisy as is [73].

Named Function Networking [113] describes an extension of the ICN concept to named functions processed in the network, which could be used to generate data flow processing applications well-suited to, for example, time series data processing in IoT sensing applications.

Related to this, is the need to support efficient function naming. Functions, input parameters, and the output result could be encapsulated in the packet header, the packet body, or mixture of the two (e.g. [31]). If functions are encapsulated in packet headers, the naming scheme affects how a computation task is routed in the network, which IoT devices are involved in the computation task (e.g. [54]), and how a name is decomposed into smaller computation tasks and deployed in the network for a better performance.

Another challenge is related to support computing-aware routing. Normal routing is for forwarding requests to the nearest source or cache and return the data to the requester, whereas the routing for in-network computation has a different purpose. If the computation task is for aggregating sensed data, the routing strategy is to route the data to achieve a better aggregation performance [51].

In-network computing also includes synchronization challenges. Some computation tasks may need synchronizations between sub-tasks or IoT devices, e.g. a device may not send data as soon as it is available because waiting for data from the neighbours may lead to a better aggregation result; some devices may choose to sleep to save energy while waiting for the results from the neighbours; while aggregating the computation results along the path, the intermediate IoT devices may need to choose the results generated within a certain time window.

6.10. Self-Organization

General IoT deployments involve heterogeneous IoT systems consisting of embedded systems, aggregators and service gateways in a IoT domain. To scale IoT deployments to large scale, scope-based self-organization is required. This relates to IoT system middleware functions [118] which include device bootstrapping and discovery, assigning local/global names to device and/or content, security and trust management functions towards device authentication and data privacy. ICN based on-boarding protocols have been studied [96] and has shown to offer significant savings compared to existing approaches. These challenges span both the constrained devices as well as interaction with the aggregators and the service gateways which may have to contact external services like authentication servers to on-board devices. A critical performance optimization metric of these functions while operating at scale is to have low control and data overhead in order to maximize energy efficiency. Further, in the infrastructure part 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.

6.11. 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 leveraging state-based forwarding in protocols like CCN/NDN. However these features have not been analyzed from the QoS perspective when heterogeneous traffic patterns are mixed in a router, in general QoS for ICN is an open area of research [121]. In-network reliability comes at the cost of a complex network layer; hence the research challenges here is to build redundancy and reliability in the network layer to handle a wide range of disruption scenarios such as congestion, short or long term disconnection, or last mile wireless impairments. Also an ICN network should allow features such as opportunistic store and forward mechanism to be enabled only at certain points in the network, as these mechanisms also entail overheads in the control and forwarding plane overhead which will adversely affect application throughput, Please see the discussion on in-network storage (Section 6.5) for more details .

6.12. Resource Constraints and Heterogeneity

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

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

7. Differences from T2TRG

T2TRG [9] is a IoT research group under IRTF focusing on research challenges of realizing IoT solutions considering IP as the narrow waist. IP-IoT has been a research topic over a decade and with active industry solutions, hence this group provides an venue to study advanced issues related to IP-IoT security, provisioning, configuration and inter-operability considering various heterogeneous application environments. ICN-IoT is a recent research effort, where the objective to exploit ICN feature of name based routing and security, caching, multicasting, mobility etc in an end-to-end manner to enable IoT services spanning both ad hoc, infrastructure and hybrid scenarios. More detailed comparison of IP-IoT versus ICN-IoT is given in Section 4.

8. Security Considerations

ICN puts security in the forefront of its design which ICN-IoT can leverage to build applications with varying security requirements, which has been discussed quite elaborately in this draft. This is an informational draft and doesn't create new considerations beyond what has been discussed.

9. 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 towards 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 application and networking perspective. We then discuss why current approach of application layer unified IoT solutions based on IP falls short of meeting these requirements, and how ICN architecture is a more suitable towards this. 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 varying IoT service needs.

10. Acknowledgements

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

11. Informative References

- [1] Cisco System Inc., CISCO., "Cisco visual networking index: Global mobile data traffic forecast update.", 2016-2021.
- [2] Shafiq, 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 Initiative 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, "Mobiiscape: 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] Chai, W., Katsaros, K., Strobbe, M., and P. Romano, "Enabling Smart Grid Applications with ICN", ICN Sigcomm, 2015.
- [20] 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.
- [21] 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.
- [22] Patrice, R. and H. Rivano, "Tests Scenario on DTN for IOT III Urbanet collaboration", Dissertation, INRIA, 2015.
- [23] Kevin, F., "Comparing Information-Centric and Delay-Tolerant Networking", Local Computer Networks (LCN), 2012 IEEE 37th Conference on. IEEE, 2012..
- [24] Miao, Y. and Y. Bu, "Research on the Architecture and Key Technology of Internet of Things (IoT) Applied on Smart Grid", IEEE, ICAEE, 2010.

- [25] Castro, M. and A. Jara, "An analysis of M2M platforms: challenges and opportunities for the Internet of Things", IMIS, 2012.
- [26] Gubbi, J., Buyya, R., and S. Marusic, "Internet of Things (IoT): A vision, architectural elements, and future directions", Future Generation Computer Systems, 2013.
- [27] 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.
- [28] 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.
- [29] Zhou, H., Liu, B., and D. Wang, "Design and Research of Urban Intelligent Transportation System Based on the Internet of Things", Springer Link, 2012.
- [30] 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.
- [31] 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.
- [32] Zhang, M., Yu, T., and G. Zhai, "Smart Transport System Based on the Internet of Things", Applied Mechanics and Materials, 2012.
- [33] Zhang, A., Yu, R., Nekovee, M., and S. Xie, "The Internet of Things for Ambient Assisted Living", IEEE, ITNG, 2010.
- [34] Savola, R., Abie, H., and M. Sihvonen, "Towards metrics-driven adaptive security management in E-health IoT applications.", ACM, BodyNets, 2012.
- [35] Jacobson, V., Smetters, D., Plass, M., Stewart, P., Thornton, J., and R. Braynard, "VoCCN: Voice-over Content-Centric Networks", ACM, ReArch, 2009.

- [36] Piro, G., Cianci, I., Grieco, L., Boggia, G., and P. Camarda, "Information Centric Services in Smart Cities", ACM, Journal of Systems and Software, 2014.
- [37] 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.
- [38] 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.
- [39] 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.
- [40] 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.
- [41] 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.
- [42] 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.
- [43] 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.

- [44] 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 .
- [45] Mavromoustakis, C., Mastorakis, G., and J. Batalla, "Internet of Things (IoT) in 5G Mobile Technologies", ISBN,3319309137, Springer.
- [46] Firner, S., Medhekar, S., and Y. Zhang, "PIP Tags: Hardware Design and Power Optimization", in Proceedings of HotEmNets, 2008.
- [47] Masek, P., Masek, J., Frantik, P., and R. Fajdiak, "A Harmonized Perspective on Transportation Management in Smart Cities: The Novel IoT-Driven Environment for Road Traffic Modeling", Sensors, Volume 16, Issue 11, 2016.
- [48] 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.
- [49] Ravindran, R., Biswas, T., Zhang, X., Chakrabort, A., and G. Wang, "Information-centric Networking based Homenet", IEEE/IFIP, 2013.
- [50] Dannewitz, C., D' Ambrosio, M., and V. Vercellone, "Hierarchical DHT-based name resolution for information-centric networks", 2013.
- [51] Fasoloy, E., Rossey, M., and M. Zorziy, "In-network Aggregation Techniques for Wireless Sensor Networks: A Survey", IEEE Wireless Communications, 2007.
- [52] Chai, W., He, D., and I. Psaras, "Cache "less for more" in information-centric networks", ACM, IFIP, 2012.
- [53] Eum, S., Nakauchi, K., Murata, M., Shoji, Yozo., and N. Nishinaga, "Catt: potential based routing with content caching for icn", IEEE Communication Magazine, 2012.
- [54] 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.

- [55] 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.
- [56] 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.
- [57] Sun, Y., Qiao, X., Cheng, B., and J. Chen, "A low-delay, lightweight publish/subscribe architecture for delay-sensitive IOT services", IEEE, ICWS, 2013.
- [58] Azgin, A., Ravindran, R., and GQ. Wang, "Mobility study for Named Data Networking in wireless access networks", IEEE, ICC, 2014.
- [59] 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.
- [60] Wang, L., Wakikawa, R., Kuntz, R., and R. Vuyyuru, "Data Naming in Vehicle-to-Vehicle Communications", IEEE, Infocm, Nomen Workshop, 2012.
- [61] 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.
- [62] Simona, C. and M. Mongiello, "Pushing the role of information in ICN", Telecommunications (ICT), 2016 23rd International Conference on. IEEE, 2016..
- [63] 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..
- [64] 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..

- [65] 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..
- [66] Trossen, D., Sarela, M., and K. Sollins, "Arguments for an information-centric internetworking architecture.", ACM SIGCOMM Computer Communication Review 40.2 (2010): 26-33.
- [67] Zhang, G., Li, Y., and T. Lin, "Caching in information centric networking: A survey.", Computer Networks 57.16 (2013): 3128-3141.
- [68] Gronbaek, I., "Architecture for the Internet of Things (IoT): API and interconnect", IEEE, SENSORCOMM, 2008.
- [69] 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.
- [70] Roussos, G. and P. Chartier, "Scalable id/locator resolution for the iot", IEEE, iThings, CPSCOM, 2011.
- [71] Amadeo, M. and C. Campolo, "Potential of information-centric wireless sensor and actuator networking", IEEE, ComManTel, 2013.
- [72] Nelson, S., Bhanage, G., and D. Raychaudhuri, "GSTAR: generalized storage-aware routing for mobilityfirst in the future mobile internet", ACM, MobiArch, 2011.
- [73] Trappe, W., Zhang, Y., and B. Nath, "MIAMI: methods and infrastructure for the assurance of measurement information", ACM, DMSN, 2005.
- [74] 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.
- [75] Liu, R. and W. Trappe, "Securing Wireless Communications at the Physical Layer", Springer, 2010.
- [76] 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.

- [77] Sun, S., Lannom, L., and B. Boesch, "Handle system overview", IETF, RFC3650, 2003.
- [78] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<http://www.rfc-editor.org/info/rfc7252>>.
- [79] 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, <<http://www.rfc-editor.org/info/rfc7230>>.
- [80] Barnes, R., "Use Cases and Requirements for JSON Object Signing and Encryption (JOSE)", RFC 7165, DOI 10.17487/RFC7165, April 2014, <<http://www.rfc-editor.org/info/rfc7165>>.
- [81] 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, <<http://www.rfc-editor.org/info/rfc6550>>.
- [82] Perkins, C., Belding-Royer, E., and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, DOI 10.17487/RFC3561, July 2003, <<http://www.rfc-editor.org/info/rfc3561>>.
- [83] marc.mosko@parc.com, m., Uzun, E., and C. Wood, "CCNx Key Exchange Protocol Version 1.0", draft-wood-icnrg-ccnxkeyexchange-01 (work in progress), October 2016.
- [84] Sun, S., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", 2014.
- [85] Liu, X., Trappe, W., and Y. Zhang, "Secure Name Resolution for Identifier-to-Locator Mappings in the Global Internet", IEEE, ICCCN, 2013.
- [86] Boguna, M., Fragkiskos, P., and K. Dmitri, "Sustaining the internet with hyperbolic mapping", Nature Communications, 2010.
- [87] Shang, W., "Securing building management systems using named data networking", IEEE Network 2014.

- [88] Fayazbakhsh, S. and et. et al, "Less pain, most of the gain: Incrementally deployable icn", ACM, Siggcomm, 2013.
- [89] Burke, J. and et. et al, "Securing instrumented environments over Content-Centric Networking: the case of lighting control", INFOCOM, Computer Communications Workshop, 2013.
- [90] 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.
- [91] Li, S., Zhang, Y., Dipankar, R., and R. Ravindran, "A comparative study of MobilityFirst and NDN based ICN-IoT architectures", IEEE, QShine, 2014.
- [92] Chen, J., Li, S., Yu, H., Zhang, Y., and R. Ravindran, "Exploiting icn for realizing service-oriented communication in iot", IEEE, Communication Magazine, 2016.
- [93] 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.
- [94] 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.
- [95] Grieco, L., Alaya, M., and K. Drira, "Architecting Information Centric ETSI-M2M systems", IEEE, Pervasive and Computer Communications Workshop (PERCOM), 2014.
- [96] Compagno, A., Conti, M., and R. Dorms, "OnboardICNg: a Secure Protocol for On-boarding IoT Devices in ICN", ICN, Sigcomm, 2016.
- [97] 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.
- [98] InterDigital, WhitePaper., "Standardized M2M Software Development Platform", 2011.

- [99] Boswarthick, D., "M2M Communications: A Systems Approach", 2012.
- [100] 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.
- [101] 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..
- [102] 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..
- [103] Hassan, A. and D. Kim, "Named data networking-based smart home", ICT Express 2.3 (2016): 130-134..
- [104] Burke, J., Horn, A., and A. Marianantoni, "Authenticated lighting control using named data networking", UCLA, NDN Technical Report NDN-0011 (2012)..
- [105] Afanasyev, A., "Packet fragmentation in ndn: Why ndn uses hop-by-hop fragmentation.", UCLA, NDN Technical Report NDN-0032 (2015)..
- [106] 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.
- [107] 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.
- [108] So, W., Narayanan, A., Oran, D., and Y. Wang, "Toward fast NDN software forwarding lookup engine based on Hash tables", ACM, ANCS, 2012.
- [109] Amadeo, M., Campolo, C., Iera, A., and A. Molinaro, "Named data networking for IoT: An architectural perspective", IEEE, EuCNC, 2014.

- [110] 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.
- [111] 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.
- [112] Blefari Melazzi, N., Detti, A., Arumaithurai, M., and K. Ramakrishnan, "Internames: A name-to-name principle for the future internet", QShine, August 2014.
- [113] Sifalakis, M., Kohler, B., Christopher, C., and C. Tschudin, "An information centric network for computing the distribution of computations", ACM, ICN Sigcomm, 2014.
- [114] Lu, R., Lin, X., Zhu, H., and X. Shen, "SPARK: a new VANET-based smart parking scheme for large parking lots", INFOCOM, 2009.
- [115] Wang, H. and W. He, "A reservation-based smart parking system", The First International Workshop on Cyber-Physical Networking Systems, 2011.
- [116] Qian, L., "Constructing Smart Campus Based on the Cloud Computing and the Internet of Things", Computer Science 2011.
- [117] Project, BonVoyage., "European Unions - Horizon 2020, <http://bonvoyage2020.eu/>", 2016.
- [118] 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.
- [119] 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.
- [120] Adhatarao, S., Chen, J., Arumaithurai, M., and X. Fu, "Comparison of naming schema in ICN", IEEE LANMAN, June , 2016.

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

Authors' Addresses

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

Email: yyzhang@winlab.rutgers.edu

Prof. Dipankar Raychadhuri
WINLAB, Rutgers University
671, U.S 1
North Brunswick, NJ 08902
USA

Email: ray@winlab.rutgers.edu

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

Email: alfredo.grieco@poliba.it

Prof. Emmanuel Baccelli
INRIA
Room 148, Takustrasse 9
Berlin 14195
France

Email: Emmanuel.Baccelli@inria.fr

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

Email: jburke@ucla.edu

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

Email: ravi.ravindran@huawei.com

Guoqiang Wang
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: gq.wang@huawei.com

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

Email: anders.lindgren@ri.se

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

Email: bentg.ahlgren@ri.se

Olov Schelen
Lulea University of Technology
Lulea SE-971 87
SE

Email: lov.schelen@ltu.se

ICN Research Group
Internet-Draft
Intended status: Informational
Expires: January 17, 2018

Y. Zhang
D. Raychadhuri
WINLAB, Rutgers University
L. Grieco
Politecnico di Bari (DEI)
S. Sabrina
Universita degli studi dell Insubria
H. Liu
The Catholic University of America
S. Misra
New Mexico State University
R. Ravindran
G. Wang
Huawei Technologies
July 16, 2017

ICN based Architecture for IoT
draft-zhang-icnrg-icniot-architecture-01

Abstract

The Internet of Things (IoT) technology promises to connect billions of objects to Internet. Today, the IoT landscape is very fragmented from both the technology and service perspectives. As a consequence, it is difficult to integrate and cross-correlate data coming from the heterogeneous contexts and build value-added services on top. This reason has motivated the current trend to develop a unified de-fragmented IoT architecture so that objects can be made accessible to applications across organizations and domains. Several proposals have been made to build a unified IoT architecture as an overlay on top of today's Internet. Such overlay solutions, however, inherit the existing limitations of the IP protocol: mobility, security, scalability, and communication reliability. To address this problem, A unified IoT architecture based on the Information Centric Networking (ICN) architecture, which we call ICN-IoT [1] has been proposed. ICN-IoT leverages the salient features of ICN, and thus provides seamless device-to-device (D2D) communication, mobility support, scalability, and efficient content and service delivery. Furthermore, in order to guarantee the real diffusion of ICN-IoT architecture it is fundamental to deal with self-configuring features such as device onboarding and discovery, service discovery, scalability, security and privacy requirements, content dissemination since the IoT system will comprise of diverse applications with heterogenous requirements including connectivity, resource constraints and mobility. Towards this, the draft elaborates on a set of middleware functions to enable large operation of an ICN-IoT deployment.

This draft begins by motivating the need for an unified ICN-IoT architecture to connect heterogeneous IoT systems. We then propose an ICN-IoT system architecture and middleware components which includes device/network-service discovery, naming service, IoT service discovery, data discovery, user registration, and content delivery. For all of these components, discussions for secure solutions are offered. We also provide discussions on inter-connecting heterogeneous ICN-IoT domains.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 17, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. ICN-Centric Unified IoT Architecture	3
1.1. Strengths of ICN-IoT	4
2. ICN-IoT System Architecture	6
3. ICN-IoT Middleware Architecture	7
4. ICN-IoT Middleware Functions	9
4.1. Device Onboarding and Discovery	10
4.2. Detailed Discovery Process	11
4.3. Naming Service	14
4.4. Service Discovery	16
4.5. Context Processing and Storage	17
4.6. Publish-Subscribe Management	19
4.7. Security	22
5. Support to heterogeneous core networks	23
5.1. Interoperability with IP legacy network	23
5.2. Named protocol bridge	23
5.3. Inter-domain Management	23
6. Informative References	24
Authors' Addresses	26

1. ICN-Centric Unified IoT Architecture

In recent years, the current Internet has become inefficient in supporting rapidly emerging Internet use cases, e.g., mobility, content retrieval, IoT, contextual communication, etc. As a result, Information Centric Networking (ICN) has been proposed as a future Internet design to address these inefficiencies. ICN has the following main features: (1) it identifies a network object (including a mobile device, a content, a service, or a context) by its name instead of its IP address, (2) a hybrid name/address routing, and (3) a delay-tolerant transport. These features make it easy to realize many in-network functions, such as mobility support, multicasting, content caching, cloud/edge computing and security.

Considering these salient features of ICN, we propose to build a unified IoT architecture, in which the middleware IoT services are proposed for administrative purposes such as towards onboarding devices, device/service discovery and naming. Other functions such as name publishing, discovery, and delivery of the IoT data/services is directly implemented in the ICN network. Figure 1 shows the proposed ICN-IoT approach, which is centered around the ICN network instead of today's Internet.

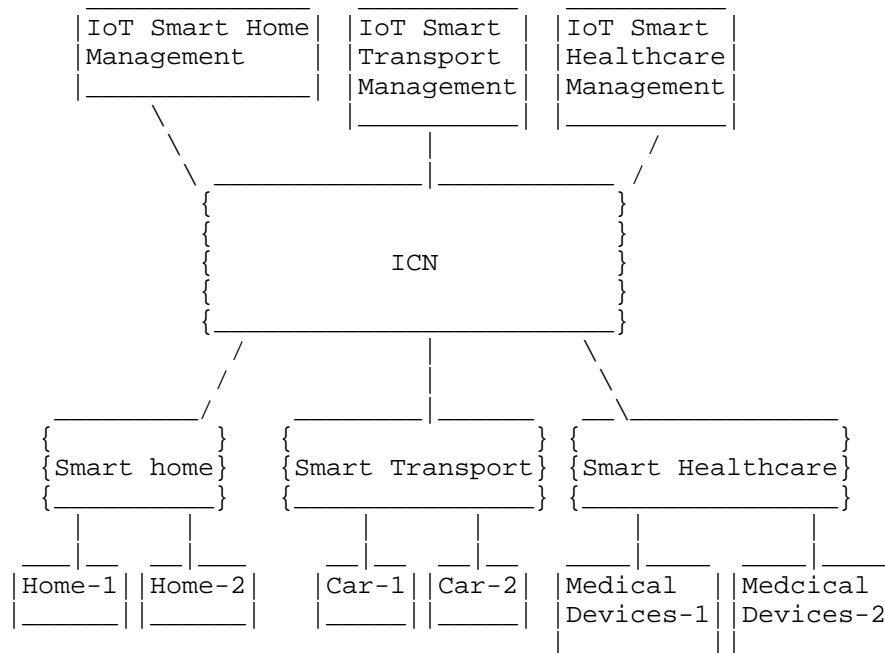


Figure 1: The proposed ICN-IoT unified architecture.

1.1. Strengths of ICN-IoT

Our proposed ICN-IoT architecture delivers IoT services over the ICN network, which aims to satisfy the requirements of the open IoT networking system (discussed in the ICN-IoT design considerations draft [1]):

- o Naming. In ICN-IoT, we assign a unique name to an IoT object, an IoT service, or even a context. These names are persistent throughout their scopes.
- o Security and privacy. In ICN-IoT, secure binding between names and data objects instead of IP addresses to identify devices/data/services, is inherently more secure than the traditional IP paradigm [16].
- o Scalability. In ICN-IoT, the name resolution is performed in the network layer in an online or using a mapping system with name state distributed within the entire network. Thus, it can achieve

high degree of scalability exploiting features like content locality, local computing, and multicasting.

- o Resource efficiency. In ICN-IoT, in both the constrained and non-constrained parts of the network, only those data that are subscribed by applications or requested by clients in the specified context will be delivered. Thus, it offers a resource-efficient solution.
- o Local traffic Pattern. In ICN-IoT, we can easily cache data or deploy computation services in the network, hence making communications more localized and reducing the overall traffic volume.
- o Context-aware communications. ICN-IoT supports contexts at different layers, including device layer, networking layer and application layer. Contexts at the device layer include information such as battery level or location; contexts at the network layer include information such as network status and statistics; contexts at the application layer are usually defined by individual applications. In ICN-IoT, device context may be available to the network layer, while network elements (i.e., routers) are able to resolve application-layer contexts to lower-layer contexts. As a result of adopting contexts and context-aware communications, communications only occur under situations that are specified by applications, which can significantly reduce the network traffic volume.
- o Seamless mobility handling. In ICN-IoT, ICN's name resolution layer allows multiple levels of mobility relying on receiver-oriented nature for self-recovery for consumers, and using multicasting and late-binding techniques to realize seamless mobility support of the producing nodes.
- o Self-Organization: Name based networking allows service level self configuration and bootstrapping of devices with minimal manufacturer or administrator provisioned configuration. This configuration involves naming, key distribution and trust association to enable data exchange between applications and services.
- o Data storage and Caching. In ICN-IoT, data are stored locally, either by the mobile device or by the gateway nodes or at service points. In-network caching [4] also speeds up data delivery and also offer local repair over unreliable links such as over wireless mediums.

- o Communication reliability. ICN-IoT supports delay tolerant communications [23], which in turn provides reliable communications over unreliable links as mentioned earlier. Also opportunistic caching allows to increase the content copies in the network to help consumers with diverse application requirements (such as operating at different request rates) or situations dealing with mobility scenarios.
- o Ad hoc and infrastructure mode. ICN-IoT supports both applications operating in ad-hoc [2] and infrastructure modes.
- o In-network processing. ICN offers in-network processing that supports various network services, such as context resolution, data aggregation and compression.

2. ICN-IoT System Architecture

The ICN-IoT system architecture, which is also a general abstraction of an IoT system, is shown in Figure 2, has the following six main system components:

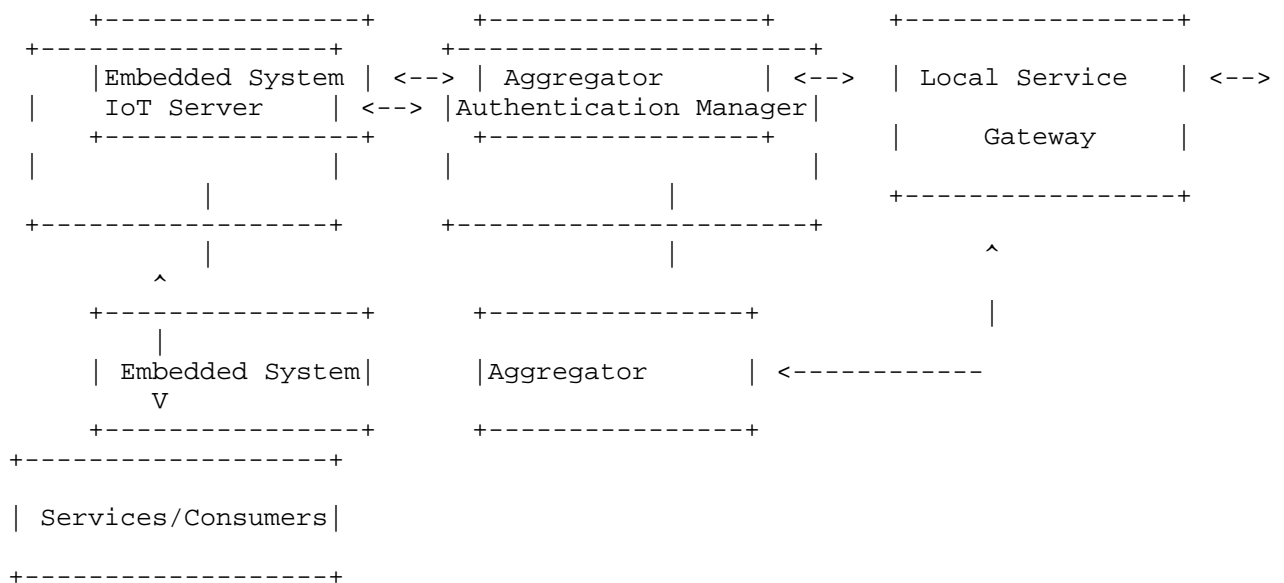


Figure 2: ICN-IoT System Architecture

- o Embedded Systems (ES): The embedded sensor has sensing and actuating functions and may also be able to relay data for other sensors to the Aggregator, through wireless or wired links.
- o Aggregator: It interconnects various entities in a local IoT network. Aggregators serve the following functionalities: device discovery, service discovery, and name assignment. Aggregators can communication with each other directly or through the local service gateway.

- o Local Service Gateway (LSG): A LSG serves the following functionalities: (1) it is at the administrative boundary, such as, the home or an enterprise, connecting the local IoT system to the rest of the global IoT system, (2) it serves to assign ICN names to local sensors, (3) it enforces data access policies for local IoT devices, and (4) it runs context processing services to generate information specified by application-specific contexts (instead of raw data) to the IoT server.
- o IoT Server: Within a given IoT service context, the IoT server is a centralized server that maintains subscription memberships and provides the lookup service for subscribers. Unlike legacy IoT servers that are involved in the data path from publishers to subscribers -- raising the concern of its interfaces being a bottleneck -- the IoT server in our architecture is only involved in the control path where publishers and subscribers exchange their names, certificates, and impose other security functions such as access control.
- o Authentication Manager (AM): The authentication manager serves to enable authentication of the embedded devices when they are onboarded in the network and also if their identities need to be validated at the overall system level. The authentication manager may be co-resident with the LSG or the IoT server, but it can also be standalone inside the administrative boundary or outside it.
- o Services/Consumer: These are other application instances interacting with the IoT server to fetch or be notified of anything of interest within the scope of the IoT service.

The logical topology of the IoT system can be mesh-like, with every ES attached to one or more aggregators or to another ES, while every aggregator is attached to one or more LSG and all the LSGs connected to the IoT server. Thus, each ES has its aggregators, and each of which in turn has its LSG. All ES belonging to the same IoT service share the same IoT server. All the aggregators that are attached to the same LSG management are reachable to one another hence capable of requesting services or content from them. While such richer connectivity improves reliability, it also requires control plane sophistication to manage the inter-connection. Though our discussion can be generalized to such mesh topologies, in the rest of the draft, we will focus on the tree topology for the sake of simplicity.

3. ICN-IoT Middleware Architecture

The proposed ICN-IoT middleware aims to bridge the gap between underlying ICN functions, IoT applications and devices to achieve self-organizing capability.

The middleware functions are shown in Fig. 3 and it includes six core functions: device discovery, naming service, service discovery, context processing and storage, pub/sub management, and security which spans all these functions.

In contrast to centralized or overlay-based implementation in the legacy IP-based IoT platform, ICN-IoT architecture pushes a large portion of the functionalities to the network layer, such as name resolution, mobility management, in-network processing/caching, context processing, which greatly simplifies the middleware design.

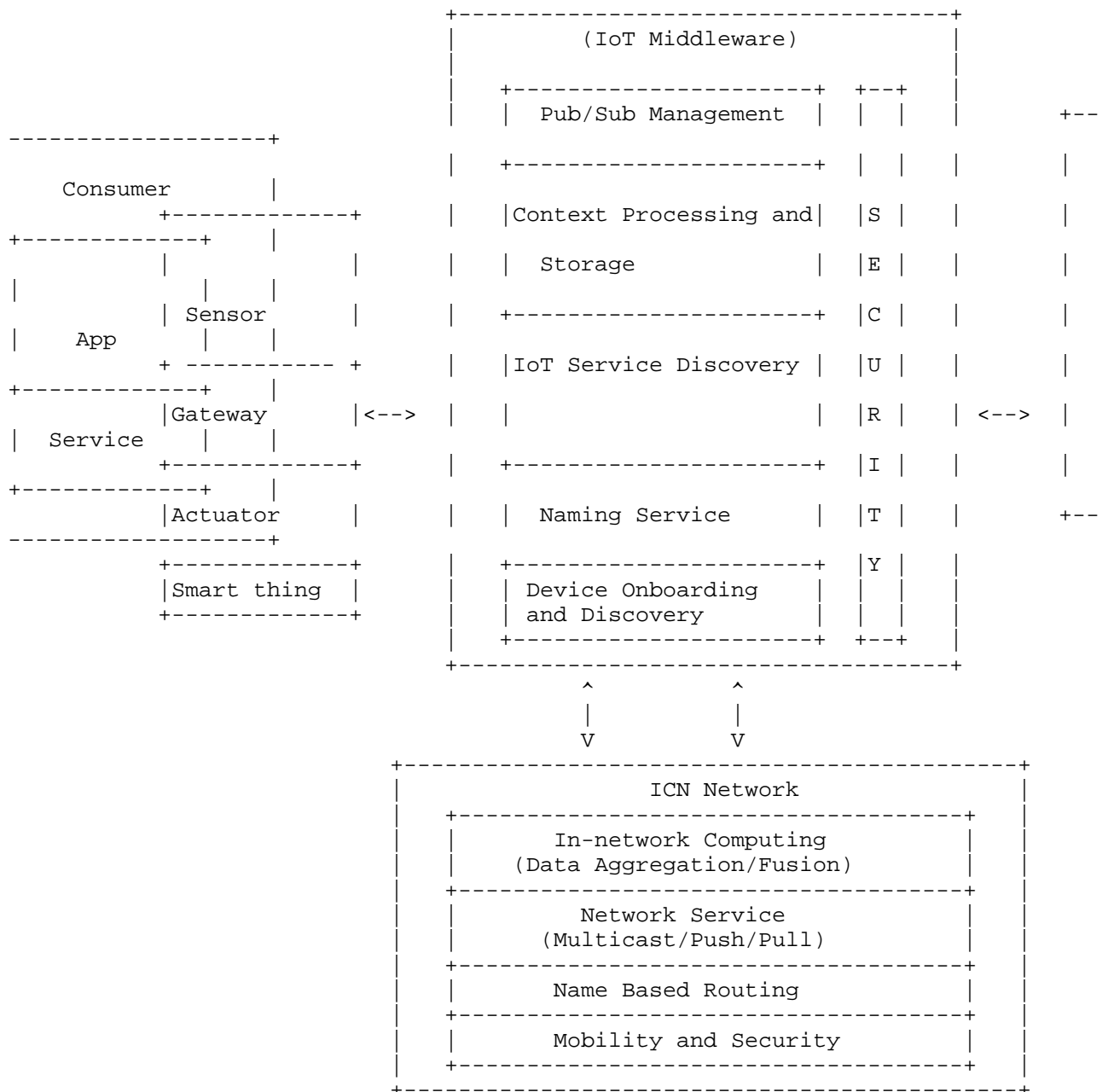


Figure 3: The ICN-IoT Middleware Functions

4. ICN-IoT Middleware Functions

For each of these middleware functions we highlight what these function achieve, advantages an ICN architecture enables in realizing each function, and provide discussion of how the function can be realized considering two ICN protocols i.e. NDN [6] and MobilityFirst (MF) [5].

Please note most of these middleware functions are implemented on unconstrained aggregators, LSGs and the IoT servers, only very limited functions (mainly for device discovery and naming service) are implemented on resource-constrained ES, while unconstrained devices within an aggregator can execute more functions such as service discovery.

4.1. Device Onboarding and Discovery

In the literature several works do not differentiate between device onboarding and device discovery. In this draft, we make a distinction. The objective of onboarding is to connect new devices to the rest and enable them to operate in the ecosystem. Every entity should be exposed to its direct upstream neighbor and may be another embedded system or aggregator. Specifically, it includes the following three aspects: (1) a newly added ES should be exposed to its neighbor (ES or aggregator) and possibly to its LSG, AM, and the IoT server; (2) a newly added aggregator is exposed to its LSG, and possibly to its neighbor aggregators; (3) a newly added AM should be exposed to the IoT server and the LSG; and (4) a newly added LSG should be exposed to the IoT server. Device discovery serves two functions: 1) it is used in the context of discovering neighboring ESs to form routing paths, where existing mechanisms can be used; 2) for device onboarding, on which we focus here. During onboarding, the ES passes its device-level information (such as manufacturer-ID and model number) and application-level information (such as service type and data type) to the upstream devices. In the NDN architecture (and other name-based approaches), there is no need to identify the devices with IDs. But, if the device is required to have a globally unique ICN ID, it can be provided one by the naming service (described in Section 4.3), and recorded by the LSG (and possibly the aggregator and the IoT server). As part of the device discovery process, each ES will also be assigned a local network ID (LID) that is unique in its local domain. Then the device will use its LID for routing within the local domain (i.e. between ESs and the aggregator) because the globally unique ICN ID associated with a device is quite long and not energy efficient for constrained IoT devices. One approach to generate a short LID is to hash its persistent ID. In the name-based ICN approaches where device identity is not needed, a device can publish within the name scope of the aggregator (e.g., /iot/aggl/dev10 for Device numbered 10 under aggregator numbered 1). In most IoT systems, devices interact with the aggregator for data or information processing or aggregation, hence there is no direct communication between devices under an aggregator. If in some set-up devices under the aggregator need to communicate with each other a scalable mechanism is to allow direct neighbors to communicate with each other while others communicate through the aggregator.

ICN enables flexible and context-centric device discovery which is important in IoT ecosystem where heterogeneous IoT systems belonging to different IoT services may co-exist sharing the same wireless resources. Contextualization is a result of name-based networking where different IoT services can agree on unique multicast names that can be pre-provisioned in end devices and the network infrastructure using the routing control plane. This also has an advantage of localizing device discovery to regions of network relevant to an ICN service, also enabling certain level of IoT asset security by isolation. In contrast IP offers no such natural IoT service mapping; any forced mapping of this manner will entail high configuration cost both in terms of device configuration, and network control and forwarding overhead.

4.2. Detailed Discovery Process

A device can be an embedded device, a virtual device, a process, or a service instance such as a sensing service. We assume that the device has pre-loaded secure keys. Specifically, we consider both resource-constrained devices and resource-rich devices, and assume that the pre-loaded secure keys are symmetric keys or passwords for the former, while the asymmetric key pair (public key certificate and the corresponding private key) for the latter.

Below we discuss the detailed device discovery process considering both resource-constrained devices and resource-rich devices. As assumed for the former there is a mechanism for either securely pre-loading symmetric keys or passwords, while for the latter asymmetric key-pair using the public key infrastructure and certificate are used to exchange/generate the symmetric key (denoted as $SMK_{\{device\}}$). We note that the use of asymmetric keys follows the standard PKI procedures with the use of either self-certificates, certificates generated by a local (or domain specific) or global authority. The bootstrapping of the constrained devices with symmetric keys can be performed in several ways. As mentioned, pre-loading the device with keys before shipping is one approach, or the symmetric keys can be loaded by the administrator (or home owner or site-manager) at the time of bootstrapping of the device on-site. The approach is based on the level of trust and the threat model in which the system operates. We also note that with ongoing research constrained devices are becoming increasingly powerful and new low-cost and computation based PKI approaches are being proposed. In the future, constrained devices may be able to also use PKI mechanisms for creating symmetric keys. In addition, we assume that there is a local authentication service (AS) that performs authentication, authorization and transient action key distribution. The local authentication service is a logical entity that can be co-hosted at the LSG or IoT server. The location of the AS may be informed by

efficiency, security, and trust considerations. The design offloads the complexity to the local AS and simplifies the operations at the devices. Mechanisms can be devised for authenticating and onboarding a device onto the IoT network even if the device does not trust its neighbors and the aggregator using the AS. This can be done by having the key `SMK_{device}` shared with an AS, which can be communicated this information during device bootstrapping. The AS is used to authenticate the device in the network. The mechanism can be extended for the device to authenticate the network it is connecting to as well [10].

The general steps for device discovery assuming pre-shared symmetric keys are as follows :

- o New device polling: The configuration service periodically sends out messages pulling information from new devices. Then the device can communicate its manufacturer ID to the aggregator who forwards the message to the AS. The AS will send back a discover-reply, with a nonce encrypted with `SMK_{device}` and another nonce to be used by the device in the reply; this message is forwarded back to the device. The device decrypts the message obtains the nonce, and encrypts a concatenation of the two nonces with `SMK_{device}`, which it sends back to the AS. The AS decrypts the content again and authenticates the device. Then it creates a symmetric key to be shared between the aggregator and the device and encrypts a copy of the key with a symmetric key shared by the aggregator and the other copy with `SMK_{device}` shared with the device and replies back to the device. The reply message reaches the device via the aggregator and they both receive the key to perform secure communication. In NDN, this process can be initiated by the configuration service running on the LSG, which periodically broadcasts discovery Interests (using the name `/iot/aggl/discover` message) or the discovery can be initiated by the new device in the network which can send a discover message using its globally unique ID (e.g., manufacturer ID). If no authentication of the device's identity is required, then the discover message can be forwarded to the aggregator, which can reply with its namespace and a locally unique ID for the device, say `dev10`, so the namespace for the device is `/iot/aggl/dev10`. Or the globally unique ID of the device can also be used as the locally unique ID. This mechanism does not preclude the communication between the device and the aggregator going over a multi-hop path consisting of other IoT devices that are already on-boarded. If device authentication is required, In MF, we can set a group-GUID as the destination address, and the configuration service issues a polling via multicasting. Once the new device enters the IoT domain and receives the polling message, it sends a association request (AssoReq) message, including its manufacture

ID, or ICN ID (if it has been assigned one before), its network and security capabilities, and a message ID which is used for the further message exchange between the new device and aggregator. If the device is already assigned a symmetric key, it can use the symmetric key to communicate with the LSG (the ID can be transmitted in clear or by using a pseudonym [17]) to facilitate quick identification by the LSG. If the device can use the PKI, a symmetric key can also be generated. After the aggregator receives the AssoReq from the new device, it will request the LSG naming service to issue a local LID for this new device. The aggregator shall send an Association Reply (AssoRep) message to the new device, which includes the message ID copied from the previous AssoReq message from the new device to indicate this association reply is for this new device, the selected authentication method according to the new device security capabilities, the assigned LID. The AssoRep is sent to the new device via a specific multicast group (as the new device does not have a routable ID yet at this moment). The LID is a short ID unique in the local IoT domain, and is used for the routing purpose in the local domain. This specification will not limit the format of the LID and the method to generate a LID. If authentication is not required, the device discovery is completed and the device can communicate with the aggregator using the LID. If the Authentication is required, this LID is blocked by the aggregator from passing general data traffic between two devices until the authentication transaction completes successfully with the local authentication service. The unauthenticated LID can only send traffic to the authentication service. The aggregator forwards the traffic between the device and the local AS. The aggregator may also implement the policy to regulate the amount of traffic to be sent by an unauthenticated LID to mitigate the DoS attack. If the authentication is required, the following steps shall be performed.

- o Mutual Authentication: The mutual authentication allows only authorized device to register and use the network, and to provide the device with assurance that it is communicating with a legitimate network. If the authentication is required in the AssoRep, the device shall send a Authentication Request (AuReq) message to the aggregator using the selected authentication method. The AuReq is signed with the pre-loaded SMK{device} for authentication. The aggregator forwards the AuReq to the local AS. The local AS performs authentication locally or contacts a third-party AS according to the authentication method. If the authentication is successful, the local AS generates a master symmetric key SMK{device, aggregator} for the communications between the device and the aggregator. It sends Authentication Reply (AuRep) with master SMK{device, aggregator} to the device

via the aggregator. The master $SMK\{device, aggregator\}$ is protected with the pre-loaded $SMK\{device\}$. The local AS also sends a copy of master $SMK\{device, aggregator\}$ to the aggregator through the secure connection between the local AS and the aggregator. This same approach will work equally well in the NDN architecture as well.

- o Key generation and distribution: Once the master $SMK\{device, aggregator\}$ is placed on the device and aggregator, the session keys (AKs) and group keys (GTKs) are generated and placed on the device and the aggregator for unicast and multicast communications, respectively, using the master $SMK\{device, aggregator\}$.
- o Protected Data Transfer: The session keys (AKa and AKe) are used for message integrity and data confidentiality, respectively, which can be renewed periodically. The renewal can happen using key generation functions with the shared secrets and some nonces used for generating the new session keys [10].

The other case is when devices have sufficient resources to run asymmetric keys. That is, the device is pre-loaded with a manufacture ID, a pair of public/private keys ($PK_{\{device\}}$, $SK_{\{device\}}$) and a certificate which binds the device identity and public key. In this case, we also go through the above three steps, with the only difference being in the second step which is Mutual Authentication. To illustrate it using MF as the architecture, in this case, the AuReq message shall include the device certificate and a message authentication code signed by the device private key $SK_{\{device\}}$. The local AS will authenticate the device once receiving the AuReq. If the authentication succeeds, then the local AS will send the master $SMK\{device, aggregator\}$ along with its certificate in AuRep. AuRep contains a MAC signed by the local AS private key. The master $SMK\{device, aggregator\}$ is encrypted using the device public key $PK_{\{device\}}$. $SMK\{device, aggregator\}$ will be used for generation of AKs to ensure the integrity and confidentiality of future data exchange between the device and the aggregator.

4.3. Naming Service

The objective of the naming service is to assure that either the device or the service itself is authenticated, attempting to prevent sybil (or spoofing) attack [18] and that the assigned name closely binds to the device (or service). Naming service assigns and authenticates ES and device names. An effective naming service should be secure, persistent, and able to support a large number of application agnostic names.

Traditional IoT systems use IP addresses as names, which are insecure and non-persistent. IP addresses also have relatively poor scalability, due to their fixed structure. Instead, ICN separates names from locators, and assigns unique and persistent names to each ES, which satisfies the above requirements.

If a device needs a global unique name/ID, but does not have one, it may request the naming service to obtain one after it is authenticated. Alternatively, the IoT domain (LSG or aggregator) may determine ID (name) for an authenticated device is required based on the policy. The proposed naming process works as follows. After a device has been authenticated, it may request an ID from the naming service (or the aggregator, if it can give the device a locally unique name). It sends a ID request (IDReq) to the naming service or aggregator. If the aggregator can accept request to give a unique name to the device, it will do that. For instance, in NDN the device can create content within the aggregator's namespace. If the aggregator cannot then it can serve as the devices' proxy and sends the IDReq to the naming service at the LSG. The naming service assigns a ID to the device, which can be self-certified or a URI. . The naming service also generates a certificate, binding the ID/public key with the devices' manufacture ID or human-readable name. The LSG sends the ID reply (IDRep) message to the aggregator that sends the IDRep to the device. The IDRep includes the ID certificate and the corresponding private key. The private key is encrypted and the entire message is integrity-protected with $AK_{\{device\}}$ when the message is delivered to the device. Alternatively, if the LSG determines that an authenticated device requires an ID when the aggregator registers this device, it will contact the naming service to generate the ID, certificate, and corresponding private key for the device. It sends the ID information to the device. If the device already has a pre-loaded public key, the naming service may use this pre-loaded public key as the device's ID.

The LSG maintains the mapping between every devices' LID and the ID. When the LSG receives a message from the external network that is intended for a device within the domain, the LSG will translate the destination devices' ID (which is included in the message) to its LID and then route the message to the device using its LID. Similarly, when the LSG receives a message from within the domain, it will replace the source devices' LID with its ID and then forward the message to the next-hop router. Such a mechanism can ensure global reachability of any IoT device as well as energy efficiency for resource-constrained devices.

Finally, we note that the same naming mechanism can be used to name higher-level IoT devices such as aggregators and LSGs.

4.4. Service Discovery

Service discovery intends to learn IoT services that are hosted by one aggregator by its neighbor aggregators. The aggregators themselves learn service capability of the devices during the device discovery process or separately after authenticating (or during or after naming) them. The requirements for any discovery mechanism includes low protocol overhead (including low latency and low control message count), and discovery accuracy.

In today's IoT platforms, ESs, aggregators and LSGs are connected via IP multicast, which involves complicated group management and multicast name to IP translation service. Multicast, however, is greatly simplified in ICN as most ICN architectures have natural support for multicast.

Service discovery is widely accepted as an essential element in pervasive computing environments. Many research activities on service discovery has been conducted, but privacy has often been ignored. While it is essential that legitimate users can discover the services for which they have the proper credentials, it is also necessary that services were hidden from illegitimate users. Since service information, service provider's information, service requests, and credentials to access services via service discovery protocols could be sensitive, it is important to keep them private. In [8], the authors present a user-centric model, called Prudent Exposure, as an approach designed for exposing minimal information privately, securely, and automatically for both service providers and users of service discovery protocols.

Below, we explain how service discovery is implemented. The key to service discovery is to expose aggregator's services to its neighbor aggregators. How this is implemented differs in NDN and MF.

In NDN, the following procedures are performed: 1) The source aggregator broadcasts an interest using the well-known name /area/servicename/certificate, which will eventually reach the destination aggregator. NDN's Interest/Data mechanisms allows only one response for each Interest sent while discovery may require to learn multiple entities. Efficient discovery can be realized using exclusion via Selectors in the protocol or as an overlay protocol [7]; 2) The destination aggregator that hosts the service checks the certificate and registers the source Aggregator if there is a matched service. It replies with an acknowledgment containing certificate to the source aggregator.

As an example of an NDN smart home, a thermostat expresses a request to discover a AC service using well-known name /home/ac/certificate

via broadcast channel. In MF case, a multicast group GUID 1234 can be assigned to all home appliance IoT service. The thermostat sends the request containing the service name and certificate to 1234. In both cases, the AC hosting this service replies with acknowledgment if all conditions match.

As regards to secure multicast service request, it is possible to use the following solution in MF. In fact, especially in MF IoT, secured group GUID can be utilized, which may be owned by multiple hosts, hence conventional public/private key scheme may not be suitable for this case. For secure service discovery, a secured name needs to be assigned to the service host. As an alternative, group key management protocol (GKMP) [31] can be adopted to resolve the issue above -- A naming service residing at LSG or IoT server (depending on application scope) generates a group public key that is used as group GUID for a service, then this group public/private keys pair is assigned to each Aggregator that hosts this service. The service hosting Aggregator in the group then listen on this group GUID, and use the group private key to decrypt the incoming discovery message. Finally, we note that this form of secure service discovery is difficult for NDN because of the use of self-certified names by MF.

4.5. Context Processing and Storage

In order to facilitate context-aware communication and data retrieval, we need to support context processing in the IoT system. The objective of context processing is to expose the ES's low-level context information to upstream aggregators and LSGs, as well as to resolve the application's high-level context requirements using lower-level ES contexts. The context processing service usually runs on both aggregators and LSGs.

Context processing requires the underlying network to be able to support in-network computing at both application and network levels. ICN supports in-networking computing (e.g. using named functions [14] and computing layer in MF) and caching, which thus offers unique advantages compared to traditional IP network where the support for in-network computing and caching is poor.

Application level contexts differ from application to application, and therefore, we need to provide a set of basic mechanisms to support efficient context processing. Firstly, the network needs to define a basic set of contextual attributes for devices (including ESs, aggregators, and LSGs), including device-level attributes (such as location, data type, battery level, multiple interfaces, available cache size, etc), network-level attributes (such as ICN names, latency per interface, packet loss rates, etc.), and service-level attributes (such as max, min, average, etc.).

Secondly, we need to have means to expose ES/aggregator/LSG contextual attributes to the rest of the system, through centralized naming resolution service or distributed routing protocols.

Thirdly, the IoT server needs to allow applications (either producers or consumers) to specify their contextual requirements. Fourthly, the unconstrained part of ICN-IoT needs to be able to map the higher-level application-specific contextual requirements to lower-level device-level, network-level, and service-level contextual information.

Once the contextual requirements and the corresponding mappings are in place, then in-network caching can be leveraged to optimize overall network performance and system QoS. In an IoT network, cached data can be used to perform data aggregation, in-network processing, and quick turnaround for answering queries. In-network caching can also be used to store data that may be relevant to many nodes and has temporal popularity in a region, and the processed data to serve the users. The contextual requirements can help define in-network processing. This goes beyond the traditional way of aggregators doing the data gathering, processing, and reduction, but also moving computation to the data (also termed network functions). Network functions in essence serves to move the computation into the network for it to happen where the context and the information is available, with the results returned to the requester. In an ICN-IoT these functionalities can be easily incorporated at scale. A good use case for both in-network caching and processing is that of an IoT network of cameras working together to gather a complete field-of-view (FoV) of an area and transmit it for aggregation at the aggregator (may be implemented in the pub/sub model). A user could fire a query that involves processing on the gathered field-of-views at the aggregator (or some other node storing the FOV-data) to answer the query.

In-network processing can be implemented at the network level and application level. For example, a user may request the information regarding the maximum car speed in an area. With the network-level implementation, the user issues a request with a function expression `/max(/area/carspeed)`[14]. The network returns the user the computed results. This result can be obtained in two steps (a)name manipulation and orchestration of computation, an aggregator or LSG will check whether it already has the cached result for `/max(/area/carspeed)`. If not, it will analyze the function expression, retrieve the function `/max` and the data `/area/carspeed`, and compute the result, or forward the request to another node for execution (b) Actual function execution for computation and data processing, that is, calculate `/max(/area/carspeed)`. Alternatively, a user may issue a request `/carspeed_service{max_car_speed in the area}`[22]. The

request is sent to a `car_speed` service application. The car speed service processes application-level request, i.e. `max_car_speed` in the area. With the former approach, the data processing and result retrieval may be more efficient. However, the network, that is, the aggregators and LSGs should have a runtime execution environment at the network layer to understand and process the function expression logic. The later approach is simple and robust because the more complex function execution can be performed at the application layer using dedicated virtual machines.

4.6. Publish-Subscribe Management

Data Publish/Subscribe (Pub/Sub) is an important function for ICN-IoT, and is responsible for IoT information resource sharing and management. The objective of pub/sub system is to provide centralized membership management service. Efficient pub/sub management poses two main requirements to the underlying system: high data availability and low network bandwidth consumption.

In conventional IP network, most of the IoT platforms provide a centralized server to aggregate all IoT service and data. While this centralized architecture ensures high availability, it scales poorly and has high bandwidth consumption due to high volume of control/data exchange, and poor support of multicast.

Next we consider two decentralized pub/sub models. The first one is the Rendezvous mode that is commonly used for today's pub/sub servers, and the second one involves Data-Control separation that is unique to ICN networks where the control messages are handled by the centralized IoT server and the data messages are handled by the underlying ICN network. Compared to the popular Rendezvous mode where both control and data messages both meet at the centralized server, separating data and control messages can greatly improve the scalability of the entire system, which is enabled by the ICN network.

In today's IP network, Rendezvous mode is the classic pub/sub scheme in which data and requests meet at an intermediate node. In this case the role of the IoT server is only required to authenticate the consumers and providing it Rendezvous service ID.

While NDN is a Pull-based architecture that inherently does not support the Pub/Sub mode naturally, there are a couple of approaches proposed to create a pub/sub model on top of NDN: namely COPSS [19] and persistent interests. COPSS integrates a push based multicast feature with the pull based NDN architecture at the network layer by introducing Rendezvous Node(RN). RN is a network layer logical entity that resides in a subset of NDN nodes. The publisher first

forwards a Content Descriptor (CD) as a snapshot to the RN. RN maintains a subscription table, and receives the subscription message from subscriber. The data publisher just sends the content using Publish packet by looking up FIB instead of PIT. If the same content prefix is requested by multiple subscribers, RN will deliver one copy of content downstream, which reduces the bandwidth consumption substantially.

Compared with the Rendezvous mode in which data plane and control plane both reside on the same ICN network layer, we consider an architecture where the control message is handled by the centralized server while data is handled by ICN network layer. Following the naming process mentioned above, the LSG has the ICN name for the local resource which is available for publishing on IoT server. IoT server maintains the subscription membership, and receives subscription requests from subscribers. Since the subscribers has no knowledge about the number of resource providers and their identities in a dynamic scenario, IoT server has to take responsibility of grouping and assigning group name for the resource.

MF takes advantage of Group-GUID to identify a service provided by multiple resources. This Group-GUID will be distributed to the subscriber as well as the publisher. In an example of NDN, it uses the common prefix /home/monitoring/ to identify a group of resource that provides multiple monitoring services such as /home/monitoring/temperature and /home/monitoring/light. The subscriber retrieves the prefix from the IoT server, and sends Interest toward the resource. In a MF example, GUID-x identifies the "home monitoring" service that combines with "light status" and "temperature". The resource producers, i.e. the host of "temperature" and the host of "light status" are notified that their services belong to GUID-x, then listen on GUID-x. The subscriber sends the request containing GUID-x through multicasting which ultimately reach the producers at the last common node. Once receiving the request, the resource producer unicasts the data to the subscriber. In addition, if multiple resource consumers subscribe to the same resource, the idea of Group-GUID can be reused to group the consumers to further save bandwidth using multicast.

Another approach to extend the NDN architecture to enable pub/sub is for the subscriber(s) to send Interests that are identified as long-term Interests [11]. The Interests do not expire from the PIT of the intermediate forwarding routers on the path from the publisher to the subscribers. Each time the publisher creates a new content, it publishes it into the network, the content reaches each subscriber along the reverse-path from the producer based on the faces stored in the PIT entry. However, the Interest is not removed from the PIT entry. This allows the creation of a multicast tree rooted at the

publisher, reaching every subscriber and enables the pushing of content from the publisher to the subscribers as needed.

With a pub/sub framework, important considerations should be given towards user registration and content distribution.

User Registration: A user, who wants to access/subscribe to a service, has to perform the registration operation by sending information that depends on the specific application domain to the IoT server. The information can be secured with the help of the PKI infrastructure. Upon successful registration the IoT server securely transmits an identifier, a user signature key $SK_{\{user\}}$ (to be used to sign messages), a user encryption key $EK_{\{user\}}$ (to communicate data confidentially), and an access password to the user in an encrypted message. Upon reception of the message, the user accesses the system to modify his/her password (function `changePassword`). With respect to existing secure application-layer solutions, a further benefit of the presented approach is the introduction of a second level of security, represented by the use of a temporary password (immediately replaced) and a couple of keys (signature $SK_{\{user\}}$ and encryption $EK_{\{user\}}$), which is well suited for the heterogeneous and distributed IoT environment.

Content Distribution: In literature, there are some solutions able to guarantee content security [9] [15][12]. In fact, the work presented in [9] [12] aims to ensure a high availability of the cached data only to legitimate users. The authors design a security framework for ICN able to deliver trusted content securely and efficiently to legitimate users/subscribers. They assume that the content is encrypted by the content provider, either at the servers or in the content distribution network (if it is trusted), by means of a popular symmetric key encryption algorithm. A group of contents may be encrypted using the broadcast encryption key, which only legitimate users can decrypt. The goal is to ensure that the encrypted content cannot be used by an entity that is not a legitimate user/customer. The authors achieve this goal by guaranteeing that only a legitimate user can obtain the symmetric key to decrypt the content, whereas a fake or a revoked user cannot. In this way, the framework does not require any user authentication, for example by an online server, each time a content is requested. Instead, Zhang et al. in [15] consider trust and security as built-in properties for future Internet architecture. Leveraging the concept of named content in recently proposed information centric network, the authors proposed a name-based trust and security protection mechanism. Their scheme is built with identity-based cryptography (IBC), where the identity of a user or device can act as a public key string. Uniquely, in named content network such as content-centric network (CCN), a content name or its prefixes can be used as a public

identity, with which content integrity and authenticity can be achieved by means of IBC algorithms. The trust of a content is seamlessly integrated with the verification of the content's integrity and authenticity with its name or prefix, instead of the public key certificate of its publisher. In addition, flexible confidentiality protection is enabled between content publishers and consumers. For scalable deployment purpose, they further propose to use a hybrid scheme combined with traditional public-key infrastructure (PKI) and IBC. Keeping in mind the available solutions, in our proposed method, the device sends to the aggregator its ICN name, its ID encrypted with its signature key $SK_{\{device\}}$ and the data encrypted with its own action key $AK_{\{device\}}$, in order to guarantee confidentiality and integrity. The action key $AK_{\{device\}}$ has been distributed during the device discovery (see Section Device discovery). The aggregator is able to decrypt the data using the corresponding action key $AK_{\{device\}}$, stored with the device ID, the signature key $SK_{\{device\}}$ and the device ICN name obtained during the name service (see Section Name service), in particular the aggregator uses the device name for identifying the related action key $AK_{\{device\}}$ (function `contentDecryption`). Note that the data are encrypted only if it is required by the application domain (i.e., some contexts may not have any security requirements - in this case the function `contentDecryption` is not applied). As regards the content delivery towards a user who subscribes to a service, the ICN IoT server transmits to the user the data encrypted with the user action key $AK_{\{user\}}$ in order to guarantee security and privacy, if it is a requirement of the application domain. The user decrypts the received data using his/her action key $AK_{\{user\}}$ (function `contentDecryption`). In such a situation, the services are treated as multiple-unicast ones, since the aggregator has to use different keys for different devices. In order to address a multicast approach, a group signature key system may be adopted, as in MF approach.

4.7. Security

This spans across all the middleware functions. Generally speaking, the security objective is to assure that the device that connects to the network should be authenticated, the provided services are authenticated and the data generated (through sensing or actuating) by both devices and services can be authenticated and kept privacy (if needed). To be specific, we consider the approach to secure device discovery, naming service and service discovery, because other services, such as pub/sub management and context processing and storage, can be properly secured according to application-specific demands.

5. Support to heterogeneous core networks

5.1. Interoperability with IP legacy network

Interoperability between the IP legacy network and ICN networks is an important property that the middleware must meet in order to ensure the co-existence and gradual migration from the today IP-based technologies and protocols. This could provide a market strength to the deployment of the ICN technologies. To this end, the Internames architecture [21][22] provides an embedded capability to manage different network domains (or realms), and to support legacy web applications and services. In this sense, a crucial role is played by the Name Resolution Service (NRS), whose functionalities can decouple names from network locators as function of time/location/context/service, and provide ICN functionalities in IP networks. By integrating these functionalities on appropriated nodes a distributed database is created to ease internet-working among heterogeneous protocol stacks in the core network.

5.2. Named protocol bridge

In an heterogeneous network, composed of different ICN networks and legacy IP-based networks, interoperability can be pursued, thanks to the name-to-name primitives. To this end, a name-based protocol bridge could be deployed at different points of the heterogeneous core network so as to provide bridging functionalities at the border of different administered network domains. In order to correctly forward the message through the network, the NRS node could aid the name-based protocol bridge providing inter-domain routing functionalities.

5.3. Inter-domain Management

In heterogeneous networks the IoT server has to strictly cooperate with the NRS nodes in the core network in order to build a virtual network topology to efficiently support Req/Res and Pub/Sub functionalities. The IoT Server could provide the names of the internal resources to the NRS, so that when the internal network changes, hence the connectivity to the resources. This ensures that the NRS database is always synchronized and updated with every IoT subsystems. In order to support Req/Res and Pub/Sub services management efficiently in an heterogeneous core network, the IoT Servers of the different administered domains have to strictly cooperate with the NRS nodes in the core network. This is to provide internal information of their own administered domain, such as the names and or the locators of the internal resources. When the internal network changes, the status of the resources are synced in

order to maintain an accurate database of the virtual network topology view comprising of various IoT subsystems.

6. Informative References

- [1] Zhang, Y., Raychadhuri, D., Grieco, L., Baccelli, E., Burke, J., Ravindran, R., Wang, G., Lindgren, A., Ahlgren, B., and O. Schelen, "Design Considerations for Applying ICN to IoT", draft-zhang-icnrg-icniot-01 (work in progress), June 2017.
- [2] Grassi, G., Pesavento, D., and Giovanni. Pau, "VANET via Named Data Networking.", IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) , 2014.
- [3] ICN based Architecture for IoT - Requirements and Challenges, ICN-IoT., "<https://tools.ietf.org/html/draft-zhang-icnrg-icniot-requirements-01>", IETF/ICNRG 2015.
- [4] 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.
- [5] NSF FIA project, MobilityFirst., "<http://www.nets-fia.net/>", 2010.
- [6] NSF FIA project, NDN., "<http://named-data.net/>", 2010.
- [7] Ravindran, R., Biswas, T., Zhang, X., Chakrabort, A., and G. Wang, "Information-centric Networking based Homenet", ACM, Sigcomm, 2013.
- [8] Feng, Z., Mutka, M., and L. Ni, "Prudent Exposure: A private and user-centric service discovery protocol", Pervasive Computing and Communications, 2004, Proceedings of the Second IEEE Annual Conference on. IEEE, 2004.
- [9] Misra, S., Tourani, R., and N. Majd, "Secure content delivery in information-centric networks: design, implementation, and analyses", Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking. ACM, 2013.
- [10] Mick, T., Misra, S., and R. Tourani, "LASER: Lightweight authentication and secured routing for NDN IoT in smart cities", arXiv:1703.08453v1 (in submission in IEEE IoT Journal).

- [11] Tourani, R., Misra, S., Mick, T., and S. Biswal, "iCenS: An information-centric smart grid network architecture", In IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 417-422, 2016.
- [12] Misra, S., Tourani, R., Mick, T., Brahma, T., Biswal, S., and D. Ameme, "iCenS: An information-centric smart grid network architecture", In IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 417-422, 2016..
- [13] Liu, H., Eldaratt, F., Alqahtani, H., Reznik, A., De Foy, X., and Y. Zhang, "Mobile Edge Cloud System: Architectures, Challenges, and Approaches", IEEE Systems Journal, pp. 1-14, DOI: 10.1109/JSYST.2017.2654119, Dec. 2016..
- [14] Sifalakis, M., Kohler, B., Scherb, C., and C. Tschudin, "An information centric network for computing the distribution of computations", In Proceedings of the 1st ACM International conference on Information-Centric Networking, pp. 137-146, 2014..
- [15] Zhang, X., "Towards name-based trust and security for content-centric network", Network Protocols (ICNP), 2011 19th IEEE International Conference on. IEEE, 2011.
- [16] Nikander, P., Gurtov, A., and T. Henderson, "Host identity protocol (HIP): Connectivity, mobility, multi-homing, security, and privacy over IPv4 and IPv6 networks", IEEE Communications Surveys and Tutorials, pp: 186-204, 2010.
- [17] Misra, S. and G. Xue, "Efficient anonymity schemes for clustered wireless sensor networks", International Journal of Sensor Networks, volume 1, number 1/2, pp: 50-63, 2006.
- [18] Newsome, J., Shi, E., Song, DX., and A. Perrig, "The sybil attack in sensor networks: analysis and defenses", IEEE, IPSN, 2004.
- [19] Jiachen, C., Mayutan, A., Lei, J., Xiaoming, Fu., and KK. Ramakrishnan, "COPSS: An efficient content oriented publish/subscribe system", ACM/IEEE ANCS, 2011.
- [20] Marica, A., Campolo, C., and A. Molinaro, "Multi-source data retrieval in IoT via named data networking", ACM ICN Siggcomm, 2014.

- [21] Blefari-Melazzi, A., Mayutan, A., Detti, A., and KK. Ramakrishnan, "Internames: a name-toname principle for the future Internet", Proc. of International Workshop on Quality, Reliability, and Security in Information-Centric Networking (Q-ICN), 2014.
- [22] Piro, G., Signorello, S., Palatella, M., Grieco, L., Boggia, G., and T. Engel, "Understanding the Social impact of ICN: between myth and reality", AI Society: Journal of Knowledge, Culture and Communication, Springer, pp. 1-9, 2016.
- [23] Nelson, S., Bhanage, G., and D. Raychaudhuri, "GSTAR: generalized storage-aware routing for mobilityfirst in the future mobile internet", Proceedings of the sixth international workshop on MobiArch, pp. 19--24, 2011.

Authors' Addresses

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

Email: yyzhang@winlab.rutgers.edu

Prof. Dipankar Raychadhuri
WINLAB, Rutgers University
671, U.S 1
North Brunswick, NJ 08902
USA

Email: ray@winlab.rutgers.edu

Prof. Luigi Alfredo Grieco
Politecnico di Bari (DEI)
671, U.S 1
Via Orabona 4, Bari 70125
Italy

Email: alfredo.grieco@poliba.it

Sicari Sabrina
Universita degli studi dell Insubria
Via Mazzini 5
Varese, VA 21100
Italy

Email: sabrina.sicari@uninsubria.it

Hang Liu
The Catholic University of America
620 Michigan Ave., N.E.
Washington, DC 20064
USA

Email: liuh@cua.edu

Satyajayant Misra
New Mexico State University
1780 E University Ave
Las Cruces, NM 88003
USA

Email: misra@cs.nmsu.edu

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

Email: ravi.ravindran@huawei.com

G.Q.Wang
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: gq.wang@huawei.com