ICN Research Group
Internet-Draft
Intended status: Experimental
Expires: September 23, 2020

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

March 22, 2020

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

draft-irtf-icnrg-ccninfo-04

Abstract

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

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

Traceroute [7] is a useful tool for discovering the routing conditions in IP networks because it provides intermediate router addresses along the path between the source and destination and the Round-Trip Time (RTT) for the path. However, this IP-based network tool cannot trace the name prefix paths used in CCN. Moreover, such IP-based network tools do not obtain the states of the in-network cache to be discovered.

This document describes the specifications of "CCNinfo", an active networking tool for discovering the path and content caching

information in CCN. CCNinfo is designed based on a previous work [6].

CCNinfo can be implemented with the user commands (such as ccninfo described in Appendix A) and forwarding function implementation on a content forwarder (e.g., router). The CCNinfo user (e.g., consumer) invokes the ccninfo command with the name prefix of the content. The ccninfo command initiates the "Request" message (described in Section 3.1). The Request message, for example, obtains routing path and cache information. When an appropriate adjacent neighbor router receives the Request message, it retrieves the cache information. If the router is not the content forwarder for the request, it inserts its "Report" block (described in Section 3.1.2) into the Request message and forwards it to its upstream neighbor router(s) decided by its FIB. These two message types, Request and Reply messages, are encoded in the CCNx TLV format [1].

Thus, the Request message is forwarded by routers toward the content publisher and the Report record is inserted by each intermediate router. When the Request message reaches the content forwarder (i.e., a router that can forward the specified cache or content), the content forwarder forms the "Reply" message (described in Section 3.2) and sends it to the downstream neighbor router. The Reply message is forwarded back toward the user in a hop-by-hop manner. This request-reply message flow, walking up the tree from a consumer toward a publisher, is similar to the behavior of the IP multicast traceroute facility [8].

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

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

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

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

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

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

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

Within a network with multipath condition, there is a case (Figure 3) wherein a single CCNinfo Request is split into multiple Requests (e.g., at Router A), which are injected into a single router (Router

D). In this case, multiple Replies with the same Request ID and Node Identifier including different Report blocks are received by the router (Router D). To recognize different CCNinfo Reply messages, the routers MUST distinguish the PIT entries by the Request ID and exploiting path labels, which could be a hash value of the concatenation information of the cumulate Node Identifiers in the hop-by-hop header and the specified content name. For example, when Router D in Figure 3 receives a CCNinfo Request from Router B, its PIT includes the Request ID and value such as H((Router_A|Router_B)|content_name), where "H" indicates some hash function and "|" indicates concatenation. When Router D receives a CCNinfo Request from Router C, its PIT includes the same Request ID and value of H((Router_A|Router_C)|content_name). Two different Replies are later received on Router D and each Reply is appropriately forwarded to Router B and Router C, respectively.

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

Figure 3

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

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

2. Terminology

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

2.1. Definitions

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

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

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

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

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

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

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

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

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

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

Upstream router
The router that connects to an Incoming face of a router, which is responsible for forwarding data for the specified name prefix to the router.

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

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

3. CCNinfo Message Formats

CCNinfo uses two message types: Request and Reply. Both messages are encoded in the CCNx TLV format ([1], Figure 4). The Request message consists of a fixed header, Request block TLV (Figure 8), and Report block TLV(s) (Figure 11). The Reply message consists of a fixed header, Request block TLV, Report block TLV(s), and Reply block/sub-block TLV(s) (Figure 14).

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

Figure 4: Packet format [1]

The Request and Reply Type values in the fixed header are PT_REQUEST and PT_REPLY, respectively (Figure 5). These messages are forwarded in a hop-by-hop manner. When the Request message reaches the content forwarder, the content forwarder turns it into a Reply message by

changing the Type field value in the fixed header from PT_REQUEST to PT_REPLY and forwards it back toward the node that initiated the Request message.

Code	Type name
%x00	PT_INTEREST [1]
%x01	PT_CONTENT [1]
%x02	PT_RETURN [1]
%x03	PT_REQUEST
%x04	PT_REPLY

Figure 5: Packet Type Namespace

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

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

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

Figure 6: Top-Level Type Namespace

3.1. Request Message

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

The Request message format is shown in Figure 7. It consists of a fixed header, Request block TLV (Figure 8), Report block TLV(s) (Figure 11), and Name TLV. The Type value of the Top-Level type

namespace is T_DISCOVERY (Figure 6). The Type value for the Report message is PT_REQUEST.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|    Version    |  PacketType   |          PacketLength         |
+---------------+---------------+---------------+---------------+
|   HopLimit    |   ReturnCode  | Reserved(MBZ) | HeaderLength  |
+===============+===============+===============+===============+
|                                                               |
+                       Request block TLV                       +
|                                                               |
+---------------+---------------+---------------+---------------+
/                       Report block TLV 1                      /
+---------------+---------------+---------------+---------------+
/                       Report block TLV 2                      /
+---------------+---------------+---------------+---------------+
/                               .                               /
/                               .                               /
+---------------+---------------+---------------+---------------+
/                       Report block TLV n                      /
+===============+===============+===============+===============+
|          T_DISCOVERY          |         MessageLength         |
+---------------+---------------+---------------+---------------+
|            T_NAME             |             Length            |
+---------------+---------------+---------------+---------------+
/ Name segment TLVs (name prefix specified by ccninfo command)  /
+---------------+---------------+---------------+---------------+
```

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

HopLimit: 8 bits

HopLimit is a counter that is decremented with each hop whenever a Request packet is forwarded. It limits the distance that a Request may travel on the network.

ReturnCode: 8 bits

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

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

Reserved (MBZ): 8 bits

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

3.1.1. Request Block

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

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|           T_DISC_REQ          |             Length            |
+---------------+---------------+-------+-------+---------+-+-+-+
|           Request ID          |SkipHop|      Flags      |F|O|C|
+---------------+---------------+-------+-------+---------+-+-+-+
|                     Request Arrival Time                      |
+---------------+---------------+---------------+---------------+
/                        Node Identifier                        /
+---------------+---------------+---------------+---------------+
```

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

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

Figure 9: Hop-by-Hop Type Namespace

Type: 16 bits

Format of the Value field. For the type value of the first Request block TLV MUST be T_DISC_REQ. For all the available types for hop-by-hop type namespace, please see Figure 9.

Length: 16 bits

Length of Value field in octets.

Request ID: 16 bits

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

SkipHop (Skip Hop Count): 4 bits

Number of skipped routers for a Request. The maximum value of this parameter is 15. This value MUST be lower than that of HopLimit at the fixed header.

Flags: 12 bits

The Flags field is used to indicate the types of the content or path discoveries. Currently, as shown in Figure 10, three bits, "C", "O", and "F", are assigned, and the other 9 bits are reserved (MBZ) for the future use. These flags are set by CCNinfo users when they initiate Requests (see Appendix A), and the routers that receive the Requests deal with the flags and change the behaviors (see Section 5 for details). The Flag values defined in this Flags field correspond to the Reply sub-blocks.

Flag	Value	Description
C	0	Path discovery (i.e., no cache information retrieved) (default)
C	1	Cache information retrieval
O	0	Request to any content forwarder (default)
O	1	Publisher reachability (i.e., only FHR can reply) Type of Reply sub-block will be T_DISC_CONTENT_OWNER
F	0	Request based on FIB's strategy (default)
F	1	Full discovery request. Request to possible multiple upstream routers specified in FIB simultaneously

Figure 10: Codes and types specified in Flags field

Request Arrival Time: 32 bits

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

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

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

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

Note that it is RECOMMENDED for all the routers on the path to have synchronized clocks; however, if they do not have synchronized clocks, CCNinfo measures one-way latency.

Node Identifier: variable length

This field specifies the node identifier (e.g., node name or hash-based self-certifying name [9]) or all-zeros if unknown. This document assumes that the Name TLV defined in the CCNx TLV format [1] can be used for this field and the node identifier is specified in it.

3.1.2. Report Block

A CCNinfo user and each upstream router along the path would insert their own Report block TLV without changing the Type field of the fixed header of the Request message until one of these routers is ready to send a Reply. In the Report block TLV (Figure 11), the Request Arrival Time and Node Identifier MUST be inserted.

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

Figure 11: Report block TLV (hop-by-hop header)

Type: 16 bits

Format of the Value field. For the Report block TLV, the type value(s) MUST be T_DISC_REPORT in the current specification.

Length: 16 bits

Length of Value field in octets.

Request Arrival Time: 32 bits

Same definition as given in Section 3.1.1.

Node Identifier: variable length

Same definition as given in Section 3.1.1.

3.2. Reply Message

When a content forwarder receives a CCNinfo Request message from an appropriate adjacent neighbor router, it inserts its own Reply block TLV and Reply sub-block TLV(s) to the Request message and turns the Request into the Reply by changing the Type field of the fixed header of the Request message from PT_REQUEST to PT_REPLY. The Reply message (see Figure 12) is then forwarded back toward the CCNinfo user in a hop-by-hop manner.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|    Version    |  PacketType   |         PacketLength          |
+---------------+---------------+-------------+-+---------------+
|    HopLimit   |   ReturnCode  | Reserved(MBZ) | HeaderLength  |
+===============+===============+=============+=+===============+
|                                                               |
+                       Request block TLV                       +
|                                                               |
+---------------+---------------+---------------+---------------+
/                               .                               /
/                               .                               /
/                      n Report block TLVs                      /
/                               .                               /
/                               .                               /
+===============+===============+===============+===============+
|          T_DISCOVERY          |         MessageLength         |
+---------------+---------------+---------------+---------------+
|            T_NAME             |             Length            |
+---------------+---------------+---------------+---------------+
/ Name segment TLVs (name prefix specified by ccninfo command)  /
+---------------+---------------+---------------+---------------+
/                        Reply block TLV                        /
+---------------+---------------+---------------+---------------+
/                     Reply sub-block TLV 1                     /
+---------------+---------------+---------------+---------------+
/                     Reply sub-block TLV 2                     /
+---------------+---------------+---------------+---------------+
/                               .                               /
/                               .                               /
+---------------+---------------+---------------+---------------+
/                     Reply sub-block TLV k                     /
+---------------+---------------+---------------+---------------+
```

Figure 12: Reply message consists of a fixed header, Request block TLV, Report block TLV(s), Name TLV, and Reply block/sub-block TLV(s)

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

Figure 13: CCNx Message Type Namespace

3.2.1. Reply Block

The Reply block TLV is an envelope for the Reply sub-block TLV(s) (explained in Section 3.2.1.1).

```
                     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_DISC_REPLY         |             Length            |
+---------------+---------------+---------------+---------------+
```

Figure 14: Reply block TLV (packet payload)

Type: 16 bits

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

Length: 16 bits

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

3.2.1.1. Reply Sub-Block

In addition to the Reply block, the router on the traced path will add one or multiple Reply sub-blocks followed by the Reply block before sending the Reply to its neighbor router.

The Reply sub-block is flexible for various purposes. For instance, operators or developers may want to obtain various characteristics of

content such as content's ownership and various cache states and conditions.

This document describes the Reply sub-block TLVs for T_DISC_CONTENT and T_DISC_CONTENT_OWNER (Figure 15) (other Reply sub-block TLVs will be discussed in separate document(s)). Note that some routers may not be capable of reporting the following values such as Object Size, Object Count, # Received Interest, First Seqnum, Last Seqnum, Elapsed Cache Time, and Remain Cache Lifetime, as shown in Figure 15, or do not report these values due to their policy. These values therefore MAY be returned with null.

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

Figure 15: Reply sub-block TLV for T_DISC_CONTENT and T_DISC_CONTENT_OWNER (packet payload)

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

Figure 16: CCNinfo Reply Type Namespace

Type: 16 bits

Format of the Value field. For the Reply sub-block TLV, the type value MUST be one of the type values defined in the CCNinfo Reply Type Namespace (Figure 16). T_DISC_CONTENT is specified when the cache information is replied from a caching router. T_DISC_CONTENT_OWNER is specified when the content information is replied from a FHR attached to a publisher.

Length: 16 bits

Length of the Value field in octets.

Object Size: 32 bits

The total size (KB) of the unexpired content objects. Note that the maximum size expressed by the 32-bit field is approximately 4.29 TB. If the cache is refreshed after reboot, the counter MUST be refreshed (i.e., MUST be set to 0). If the cache remains after reboot, the counter MUST NOT be refreshed (i.e., MUST be kept as it is).

Object Count: 32 bits

The number of unexpired content objects. Note that the maximum count expressed by the 32-bit field is approximately 4.29 billion. If the cache is refreshed after reboot, the counter MUST be refreshed (i.e., MUST be set to 0). If the cache remains after reboot, the counter MUST NOT be refreshed (i.e., MUST be kept as it is).

Received Interest: 32 bits

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

First Seqnum: 32 bits

The first sequential number of the unexpired content objects.

Last Seqnum: 32 bits

The last sequential number of the unexpired content objects. The First Seqnum and Last Seqnum do not guarantee the consecutiveness of the cached content objects; however, knowing these values may help in the analysis of consecutive or discontinuous chunks such as [10].

Elapsed Cache Time: 32 bits

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

Remain Cache Lifetime: 32 bits

The lifetime (seconds) of a content object, which is removed first from the cached content objects.

Specifications of the Name TLV (whose type value is T_NAME) and the Name Segment TLVs are described in [1], which are followed by CCNinfo. CCNinfo also enables to specification of the content name either with a prefix name (such as "ccn:/news/today") or an exact name (such as "ccn:/news/today/Chunk=10"). When a CCNinfo user specifies a prefix name, s/he will obtain the summary information of the matched content objects in the content forwarder. In contrast, when a CCNinfo user specifies an exact name, s/he will obtain only about the specified content object in the content forwarder. A CCNinfo Request message MUST NOT be sent only with a scheme name, ccn:/, because it will be rejected and discarded by routers.

4. CCNinfo User Behavior

4.1. Sending CCNinfo Request

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

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

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

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

4.1.1. Routing Path Information

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

4.1.2. In-Network Cache Information

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

4.2. Receiving CCNinfo Reply

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

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

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

5. Router Behavior

5.1. User and Neighbor Verification

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

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

5.2. Receiving CCNinfo Request

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

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

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

3. If the Request is either "cache information discovery" or "routing path information discovery", the router examines its FIB and CS. If the router caches the specified content, it inserts its own Report block in the hop-by-hop header and sends the Reply message with its own Reply block and sub-block(s) (in the case of cache information discovery) or the Reply message with its own Reply block without adding any Reply sub-blocks (in the case of routing path information discovery). If the router does not cache the specified content but knows the upstream neighbor router(s) for the specified name prefix, it inserts its own Report block and forwards the Request to the upstream neighbor(s). If the router does not cache the specified content and does not know the upstream neighbor router(s) for the specified name prefix, it terminates the Request, as defined in Section 6.5.

4. If the Request is the "publisher discovery", the router examines whether it is the FHR for the requested content. If yes, it sends the Reply message with its own Report block and sub-blocks. If the router is not the FHR but knows the upstream neighbor router(s) for the specified name prefix, it adds its own Report block and forwards the Request to the neighbor(s). If the node is not the FHR and does not know the upstream neighbor router(s) for the specified name prefix, it terminates the Request, as defined in Section 6.5.

5.3. Forwarding CCNinfo Request

5.3.1. Regular Request

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

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

CCNinfo supports multipath forwarding. The Request messages can be forwarded to multiple neighbor routers. Some routers may have a strategy for multipath forwarding; when a router sends Interest messages to multiple neighbor routers, it may delay or prioritize to send the message to the upstream routers. The CCNinfo Request, as the default, complies with such strategies; a CCNinfo user could

trace the actual forwarding path based on the forwarding strategy and will receive a single Reply message such as a content object.

However, CCNinfo can discover all possible content forwarders. See the next section for more information.

5.3.2. Full Discovery Request

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

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

Figure 17: Full discovery request. Reply messages forwarded by publisher and routers. Each router forwards the Reply message along its PIT entry and finally, the CCNinfo user receives two Reply messages: one from the FHR (Router C) and the other from the Caching router.

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

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

5.4. Sending CCNinfo Reply

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

When a router decides to send a Reply message to its downstream neighbor router or the CCNinfo user with NO_ERROR return code, it inserts a Report block with the Request Arrival Time and Node Identifier to the hop-by-hop TLV header of the Request message. Then, the router inserts the corresponding Reply block with an appropriate type value (Figure 15) and Reply sub-block(s) to the payload. The router does not insert any Reply block/sub-blocks if there is an error. The router finally changes the Type field in the fixed header from PT_REQUEST to PT_REPLY and forwards the message back as the Reply toward the CCNinfo user in a hop-by-hop manner.

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

5.5. Forwarding CCNinfo Reply

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

The Flags field in the Request block TLV is used to indicate whether the router keeps the PIT entry during the CCNinfo Reply Timeout even after one or more corresponding Reply messages are forwarded. When the CCNinfo user does not set the F flag (i.e., "0"), the intermediate routers immediately remove the PIT entry whenever they forward the corresponding Reply message. When the CCNinfo user sets

the F flag (i.e., "1"), which means the CCNinfo user chooses the "full discovery request" (see Section 5.3.2), the intermediate routers keep the PIT entry within the [CCNinfo Reply Timeout] period. After this timeout, the PIT entry is removed.

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

6. CCNinfo Termination

When performing an expanding hop-by-hop trace, it is necessary to determine when to stop expanding. There are several cases when an intermediate router might return a Reply before a Request reaches the caching router or the FHR, as described in Section 3.2.

6.1. Arriving at First-hop Router

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

6.2. Arriving at Router Having Cache

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

6.3. Arriving at Last Router

A CCNinfo Request can be determined to have arrived at the last router of the specified HopLimit. If the last router does not have the corresponding cache, it MUST send the Reply message with NO_INFO return code without inserting the Reply block TLV.

6.4. Invalid Request

If the router does not validate the Request, the router MUST note a ReturnCode of INVALID_REQUEST in the fixed header of the message and forward the message without appending any Reply (sub-)block TLVs as the Reply back to the CCNinfo user. The router MAY, however, randomly ignore the received invalid messages. (See Section 10.7.)

6.5. No Route

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

6.6. No Information

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

6.7. No Space

If appending the Report block would make the Request packet longer than the MTU of the Incoming face or longer than 1280 bytes (in the case of IPv6 as the payload [5]), the router MUST note a ReturnCode of NO_SPACE in the fixed header of the message and forward the message as the Reply back to the CCNinfo user.

6.8. Fatal Error

If a CCNinfo Request has encountered a fatal error, the router MUST note a ReturnCode of FATAL_ERROR in the fixed header of the message and forward the message as the Reply back to the CCNinfo user. This may happen, for example, when the router detects some routing loop in the Request blocks (see Section 1).

6.9. CCNinfo Reply Timeout

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

6.10. Non-Supported Node

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

6.11. Administratively Prohibited

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

7. Configurations

7.1. CCNinfo Reply Timeout

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

7.2. HopLimit in Fixed Header

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

7.3. Access Control

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

8. Diagnosis and Analysis

8.1. Number of Hops

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

8.2. Caching Router Identification

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

8.3. TTL or Hop Limit

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

8.4. Time Delay

If the routers have synchronized clocks, it is possible to estimate the propagation and queuing delays from the differences between the timestamps at the successive hops. However, this delay includes the

control processing overhead; therefore, it is not necessarily indicative of the delay that would be experienced by the data traffic.

8.5. Path Stretch

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

8.6. Cache Hit Probability

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

9. IANA Considerations

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

10. Security Considerations

This section addresses some of the security considerations.

10.1. Policy-Based Information Provisioning for Request

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

The access policy regarding "who is allowed to retrieve" and/or "what kind of information" can be defined for each router. For the former type of access policy, routers with the specified content MAY examine the signature enclosed in the Request message and decide whether they should notify the content information in the Reply. If the routers decide to not notify the content information, they MUST send the CCNinfo Reply with the ReturnCode of ADMIN_PROHIB without appending any Reply (sub-)block TLVs. For the latter type of policy, the

permission, whether (1) All (all cache information is disclosed), (2) Partial (cache information with a particular name prefix can (or cannot) be disclosed), or (3) Deny (no cache information is disclosed), is defined at the routers.

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

10.2. Filtering CCNinfo Users Located in Invalid Networks

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

10.3. Topology Discovery

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

10.4. Characteristics of Content

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

10.5. Computational Attacks

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

these values owing to privacy/security policies, but also skip the implementations of the complex functions to report these values.

10.6. Longer or Shorter CCNinfo Reply Timeout

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

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

10.7. Limiting Request Rates

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

10.8. Limiting Reply Rates

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

10.9. Adjacency Verification

It is assumed that the CCNinfo Request and Reply messages are forwarded by adjacent neighbor nodes or routers. The CCNx message format or semantics do not define a secure way to verify the node/router adjacency, while HopAuth [9] provides a possible method for an adjacency verification and defines the corresponding message format

for adjacency verification as well as the router behaviors. CCNinfo MAY use a similar method for node adjacency verification.

11. Acknowledgements

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

12. References

12.1. Normative References

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

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

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

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

[5] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 8200, July 2017.

12.2. Informative References

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

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

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

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

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

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

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

Appendix A. ccninfo Command and Options

CCNinfo is implemented in Cefore [11][12]. The ccninfo command in Cefore enables the CCNinfo user to investigate the routing path based on the name prefix of the content (e.g., ccn:/news/today). The name prefix is mandatory but has exclusive options; that is, only one of them should be used with the ccninfo command at once.

The usage of ccninfo command is as follows:

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

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

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

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

o option
This option enables to trace the path to the content publisher. Each router along the path to the publisher inserts each Report

block and forwards the Request message. It does not send Reply even if it caches the specified content. FHR that attaches the publisher (who has the complete set of content and is not a caching router) sends the Reply message.

r option
Number of traced routers. If the CCNinfo user specifies this option, only the specified number of hops from the CCNinfo user traces the Request. Each router inserts its own Report block and forwards the Request message to the upstream router(s). The last router stops the trace and sends the Reply message back to the CCNinfo user. This value is set in the "HopLimit" field located in the fixed header of the Request. For example, when the CCNinfo user invokes the CCNinfo command with this option, such as "-r 3", only three routers along the path examine their path and cache information.

s option
Number of skipped routers. If the CCNinfo user specifies this option, routers corresponding to the value specified in this option are skipped and the CCNinfo Request messages are forwarded to the next router without the addition of Report blocks; the next upstream router then starts the trace. This value is set in the "SkipHopCount" field located in the Request block TLV. For example, when the CCNinfo user invokes the CCNinfo command with this option, such as "-s 3", three upstream routers along the path only forward the Request message but do not append their Report blocks in the hop-by-hop header and do not send Reply messages despite having the corresponding cache.

Authors' Addresses

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

Email: asaeda@nict.go.jp

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

Email: a-ooka@nict.go.jp

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

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