ICN Research Group
Internet-Draft
Intended status: Experimental
Expires: January 1, 2019

H. Asaeda
NICT
X. Shao
Kitami Institute of Technology

June 30, 2018

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

draft-asaeda-icnrg-ccninfo-01

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, device name, and function/application, 2) the Round-Trip Time (RTT) between content forwarder and consumer, and 3) the states of in-network cache per name prefix. In addition, it discovers a gateway that supports different protocols such as CCN and Named-Data Networks (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 January 1, 2019.

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 6
2.1. Definitions 6
3. CCNinfo Message Formats 7
3.1. Request Message 8
3.1.1. Request Block 10
3.1.2. Report Block 12
3.2. Reply Message 14
3.2.1. Reply Block 16
3.2.1.1. Reply Sub-Block 16
4. CCNinfo User Behavior 19
4.1. Sending CCNinfo Request 19
4.1.1. Gateway Discovery 19
4.1.2. Routing Path Information 20
4.1.3. In-Network Cache Information 20
4.2. Receiving CCNinfo Reply 20
5. Router Behavior 21
5.1. Receiving CCNinfo Request 21
5.1.1. Request Packet Verification 21
5.1.2. Request Normal Processing 21
5.2. Forwarding CCNinfo Request 22
5.3. Sending CCNinfo Reply 23
5.4. Forwarding CCNinfo Reply 24
6. Publisher Behavior 24
7. CCNinfo Termination 25
7.1. Arriving at Publisher or Gateway 25
7.2. Arriving at Router Having Cache 25
7.3. No Route 25
7.4. No Information 25
7.5. No Space 25
7.6. Fatal Error 25
7.7. CCNinfo Reply Timeout 26
7.8. Non-Supported Node 26
7.9. Administratively Prohibited 26
8. Configurations 26
8.1. CCNinfo Reply Timeout 26
8.2. HopLimit in Fixed Header 26
8.3. Access Control 26
9. Diagnosis and Analysis 26
9.1. Number of Hops 26

9.2. Caching Router and Gateway Identification 27
9.3. TTL or Hop Limit 27
9.4. Time Delay 27
9.5. Path Stretch 27
9.6. Cache Hit Probability 27
10. Security Considerations 27
10.1. Policy-Based Information Provisioning for Request . . . 27
10.2. Filtering of CCNinfo Users Located in Invalid Networks . 28
10.3. Topology Discovery 28
10.4. Characteristics of Content 29
10.5. Longer or Shorter CCNinfo Reply Timeout 29
10.6. Limiting Request Rates 29
10.7. Limiting Reply Rates 29
10.8. Adjacency Verification 29
11. Acknowledgements 30
12. References 30
12.1. Normative References 30
12.2. Informative References 30
Appendix A. ccninfo Command and Options 30
Appendix B. Change History 33
Authors' Addresses 33

1. Introduction

In Content-Centric Networks (CCN), publishers provide content through the network, and receivers retrieve content by name. In this network architecture, routers forward content requests by means of 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 which content forwarder is transmitting the content to them, operators and developers may want to identify the content forwarder and observe the routing path information per name prefix for troubleshooting or investigating the network conditions.

Traceroute [5] is a useful tool for discovering the routing conditions in IP networks as it provides intermediate router addresses along the path between 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 tool does not obtain the states of the in-network cache to be discovered.

This document describes the specification of "CCNinfo", an active networking tool for discovering the path and content caching information in CCN. CCNinfo potentially discovers devices and

functions/applications in CCN. CCNinfo is designed based on the work previously published in [4].

CCNinfo can be implemented with the ccninfo user command and the forwarding function implementation on a content forwarder (e.g., router or publisher). The CCNinfo user (e.g., consumer) invokes the ccninfo command (described in Appendix A) with the name prefix of the content, the device name, or the function (or application) name. 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 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 the Request message 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].

In this way, 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 or the publisher who has 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 inspired by the design of the IP multicast traceroute facility [6].

CCNinfo supports multipath forwarding. The Request messages can be forwarded to multiple neighbor routers. When the Request messages forwarded to multiple routers, the different Reply messages will be forwarded from different routers or publisher. To support this case, PIT entries initiated by CCNinfo remain until the defined timeout value is expired.

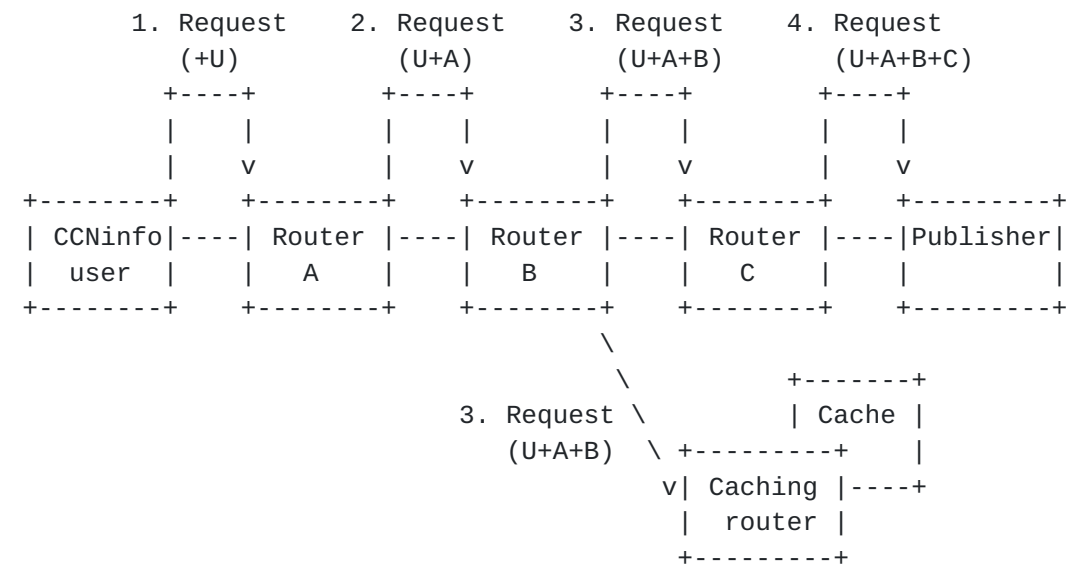


Figure 1: Request messages forwarded by consumer and routers. CCNinfo user and routers (i.e., Router A,B,C) insert their own Report blocks into the Request message and forward the message toward the content forwarder (i.e., caching router and publisher)

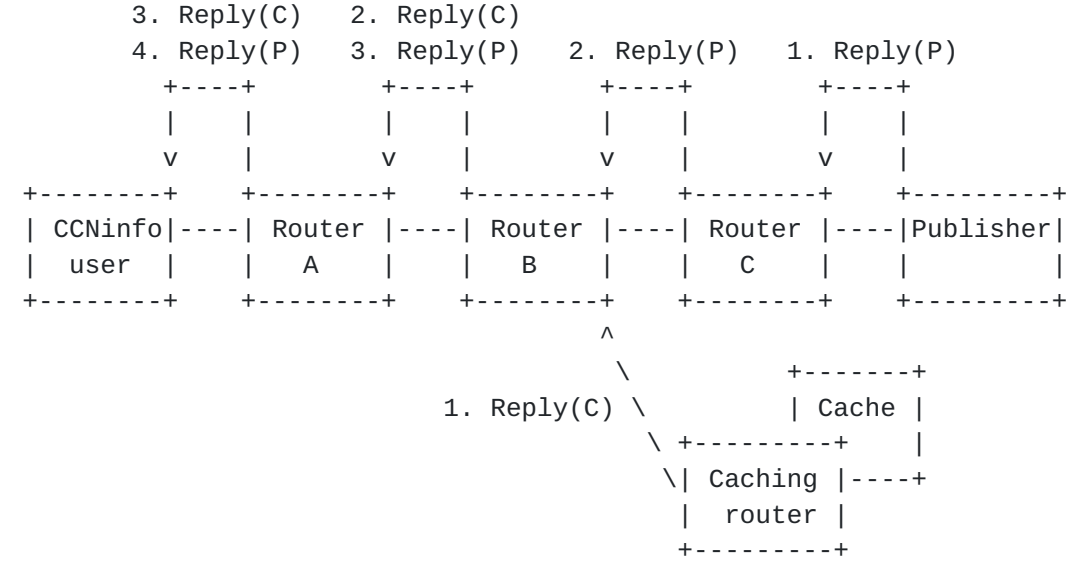


Figure 2: Reply messages forwarded by publisher and routers. Each router forwards the Reply message, and finally the CCNinfo user receives two Reply messages: one from the publisher and the other from the caching router.

CCNinfo facilitates the tracing of a routing path and provides: 1) the RTT between content forwarder (i.e., caching router or publisher) and consumer, 2) the states of 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 the cached content, 2) number of the cached chunks of the content, 3) number of the accesses (i.e., received Interests) per cache or chunk, and 4) lifetime and expiration time per cache or chunk. The number of received Interests per cache or chunk on the routers indicates the popularity of the content.

Furthermore, CCNinfo implements policy-based information provisioning that enables administrators to "hide" secure or private information, but does not disrupt the forwarding of messages. 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 [2], and indicate requirement levels for compliant CCNinfo implementations.

2.1. Definitions

Since 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 facilitating name-based content/device/function name or characteristic retrieval in the path between consumer and publisher.

Scheme name
: It indicates a URI and protocol such as "ccn:/" and "ndn:/". This document considers the protocol for name-based content/device/function name or characteristic retrieval.

Gateway
: It is a router supporting multiple scheme names in the path between consumer and publisher. The router has multiple FIBs for different protocols and establishes the connections with different neighbor routers for each protocol.

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

Content forwarder

It is either a caching router or a publisher that holds the cache (or content) and forwards it to consumers.

CCNinfo user
: It is a node that invokes the ccninfo command and initiates the CCNinfo Request.

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

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

3. CCNinfo Message Formats

CCNinfo uses two message types: Request and Reply. Both messages are encoded in the CCNx TLV format ([1], Figure 3). The Request message consists of a fixed header, Request block TLV Figure 7, 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.

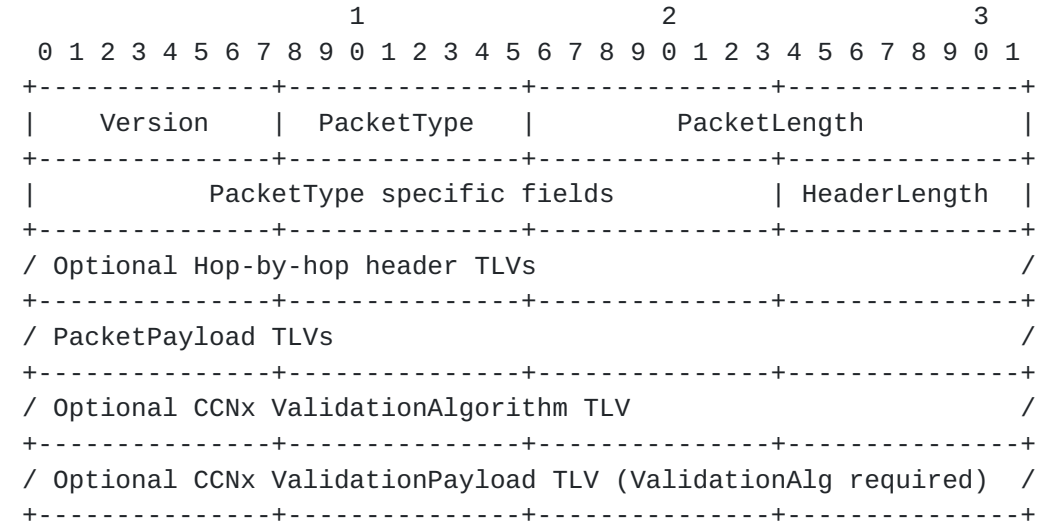


Figure 3: Packet format [1]

The Request and Reply Type values in the fixed header are PT_REQUEST and PT_REPLY, respectively (Figure 4). These messages are forwarded in a hop-by-hop manner. When the Request message reaches the content forwarder, the content forwarder turns the Request message into a Reply message by changing the Type field value in the fixed header

from PT_REQUEST to PT_REPLY and forwards back to the node that has initiated the Request message.

Code	Type name
1	PT_INTEREST [1]
2	PT_CONTENT [1]
3	PT_RETURN [1]
4	PT_REQUEST
5	PT_REPLY

Figure 4: 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, one top-level TLV type, T_DISCOVERY (Figure 5), 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
1	T_INTEREST [1]
2	T_OBJECT [1]
3	T_VALIDATION_ALG [1]
4	T_VALIDATION_PAYLOAD [1]
5	T_DISCOVERY

Figure 5: Top-Level Type Namespace

3.1. Request Message

When a CCNinfo user initiates a discovery request (e.g., by 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 as shown in Figure 6. It consists of a fixed header, Request block TLV (Figure 7), Report block TLV(s) (Figure 11), and Name TLV. The Type value of Top-Level type namespace is T_DISCOVERY (Figure 5). The Type value for the Report message is PT_REQUEST.

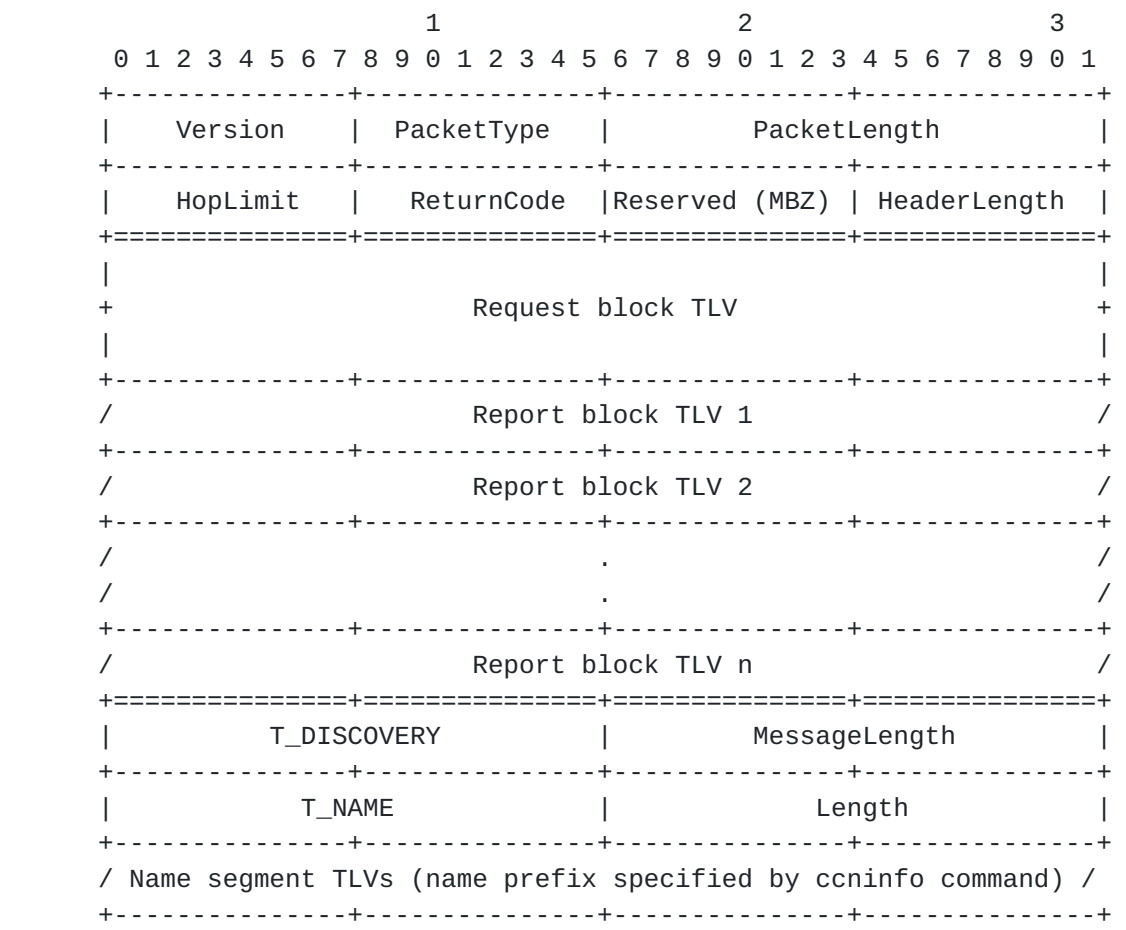


Figure 6: Request message consists of a fixed header, Request block TLV, Report block TLV(s), and Name TLV

HopLimit: 8 bits

HopLimit is a counter that is decremented with each hop. It limits the distance 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
0x00	NO_ERROR	No error
0x01	WRONG_IF	CCNinfo Request arrived on an interface to which this router would not forward for the specified name/function toward the publisher.
0x02	INVALID_REQUEST	Invalid CCNinfo Request is received.
0x03	NO_ROUTE	This router has no route for the name prefix and no way to determine a potential route.
0x04	NO_INFO	This router has no cache information for the specified name prefix, device information, or function.
0x05	NO_SPACE	There was not enough room to insert another Report block in the packet.
0x06	NO_GATAWAY	CCNinfo Request arrived on a non-gateway router.
0x07	INFO_HIDDEN	Information is hidden from this discovery because of some policy.
0x0E	ADMIN_PROHIB	CCNinfo Request is administratively prohibited.
0x0F	UNKNOWN_REQUEST	This router does not support/recognize the Request message.
0x80	FATAL_ERROR	A fatal error is one where 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, it MUST insert the Request block TLV (Figure 7) and the Report block TLV (Figure 11) of its own to the Request message before sending it through the Incoming face(s).

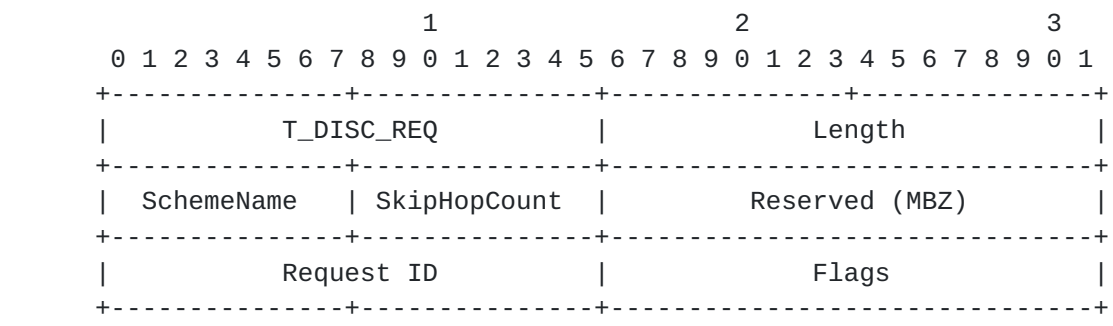


Figure 7: Request block TLV (hop-by-hop header)

Code	Type name
1	T_INTLIFE [1]
2	T_CACHETIME [1]
3	T_MSGHASH [1]
4 - 7	Reserved [1]
8	T_DISC_REQ
9	T_DISC_REPORT
%x0FFE	T_PAD [1]
%x0FFF	T_ORG [1]
%x1000-%x1FFF	Reserved [1]

Figure 8: Hop-by-Hop Type Namespace

Type: 16 bits

Format of the Value field. For the single Request block TLV, the type value MUST be T_DISC_REQ. For all the available types for hop-by-hop type namespace, please see Figure 8.

Length: 16 bits

Length of Value field in octets. For the Request block, it MUST be 4 in the current specification.

SchemeName: 8 bits

Currently, the following scheme names are defined.

Code	Scheme name
0	ccn:/
1	ndn:/
%x02-%FF	Not assigned

Figure 9: Scheme Names

SkipHopCount: 8 bits

Number of skipped routers. This value MUST be lower than the value of HopLimit at the fixed header.

Request ID: 16 bits

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

Flags: 16 bits

The discovery conditions specified as the ccninfo command options (described in Appendix A) are transferred in the Flags field. The discovery conditions depend on the specified name (i.e., name_prefix, device_name, or function_name) as shown in Figure 10. Note that code %x01 and %x02 are exclusive options; that is, only one of them should be turned on at once.

Code	Type name
%x01	Cache retrieval allowing partial match (name_prefix)
%x02	No cache information required (name_prefix)
%x04	Publisher reachability (name_prefix and device_name)
%x08	Parallel request. Request to multiple upstream routers simultaneously (name_prefix, device_name, and function_name)
%x16	Discovery of gateway supporting specified scheme name (name_prefix, device_name, and function_name)
%x32	Function's or application's version number retrieval (function_name)
%x64-%x32768	Not assigned

Figure 10: Codes and types specified in Flags field

3.1.2. Report Block

A CCNinfo user and each upstream router along the path would insert its 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 the Node Identifier MUST be inserted.

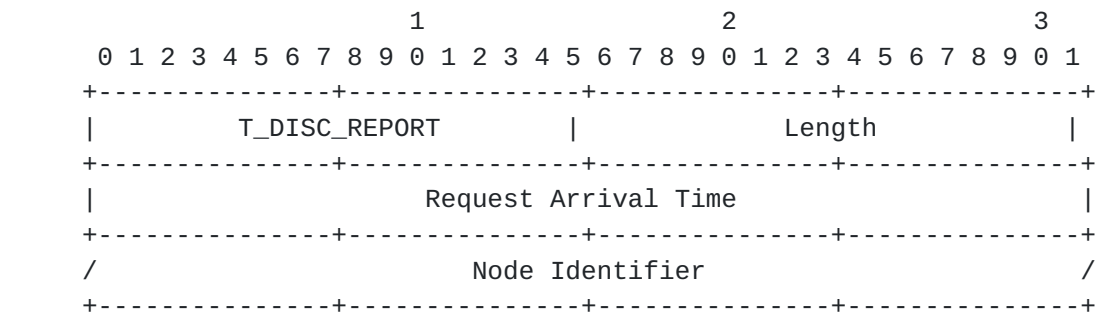


Figure 11: Report block TLV (hop-by-hop header)

Type: 16 bits

Format of the Value field. For the Request block TLV(s), the type value(s) MUST be T_DISC_REPORT.

Length: 16 bits

Length of Value field in octets.

Request Arrival Time: 32 bits

The Request Arrival Time is a 32-bit NTP timestamp specifying the arrival time of the CCNinfo Request packet at this 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 CCNinfo does not require all the routers on the path to have synchronized clocks in order to measure one-way latency.

Node Identifier: variable length

This field specifies the CCNinfo user or the router identifier (e.g., IPv4 address) of the Incoming face on which packets from the publisher are expected to arrive, or all-zeros if unknown or unnumbered. Since we may not always rely on the IP addressing

architecture, it would be necessary to define the identifier uniqueness (e.g., by specifying the protocol family) for this field. However, defining such uniqueness is out of scope of this document. Potentially, this field may be defined as a new TLV, which might be defined in the document for the CCNx TLV format[1].

3.2. Reply Message

When a content forwarder receives a CCNinfo Request message from the appropriate adjacent neighbor router, it would insert a Reply block TLV and Reply sub-block TLV(s) of its own to the Request message and turn 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) would then be forwarded back toward the CCNinfo user in a hop-by-hop manner.

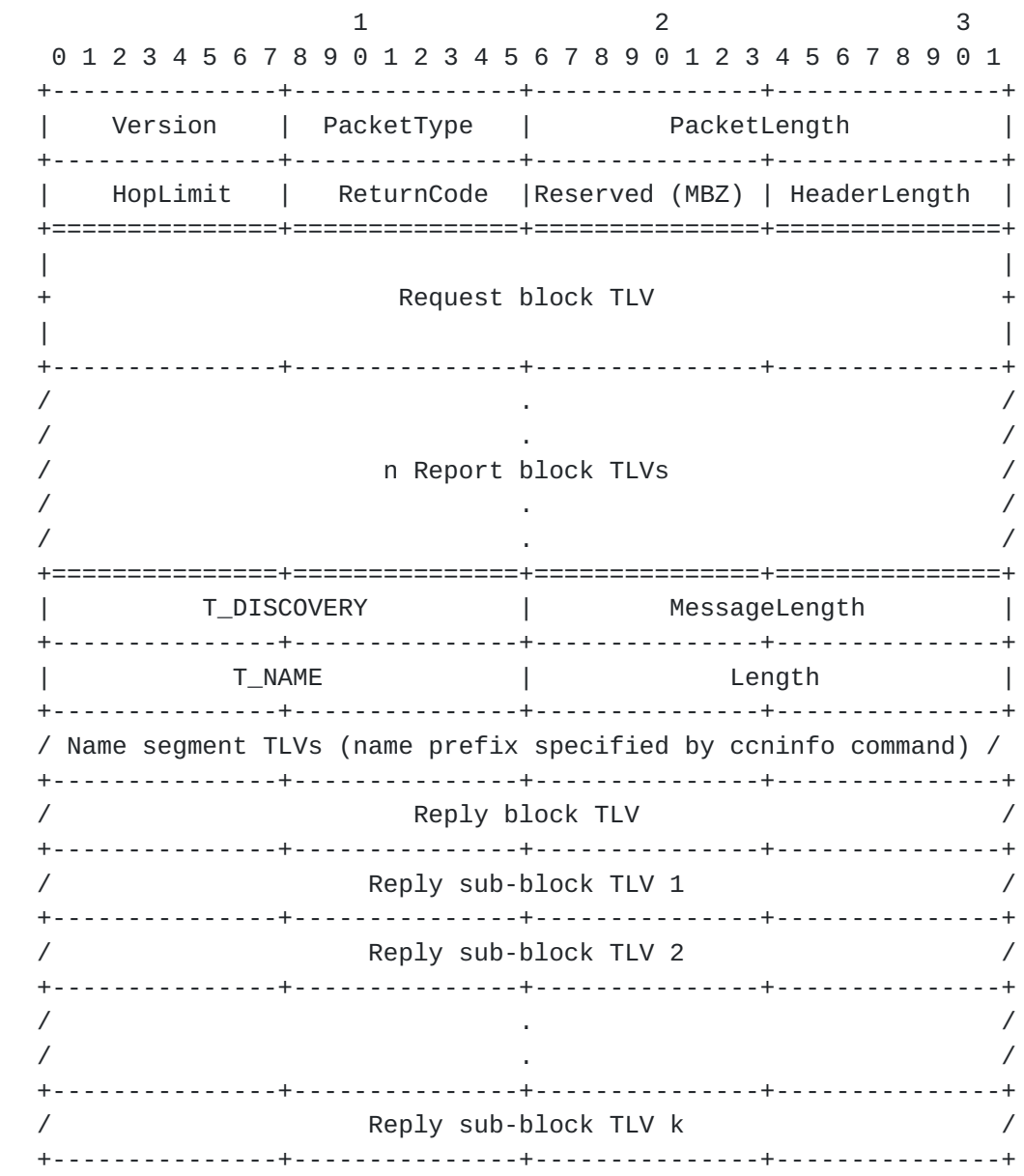


Figure 12: Reply message consists of a fixed header, Request block TLV, Report block TLV(s), Name TLV, and Reply block/sub-block TLV(s)

Code	Type name
0	T_NAME [1]
1	T_PAYLOAD [1]
2	T_KEYIDRESTR [1]
3	T_OBJHASHRESTR [1]
5	T_PAYLDTYPE [1]
6	T_EXPIRY [1]
7	T_DISC_REPLY
8 - 12	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 Reply sub-block TLV(s) (explained in Section 3.2.1.1).

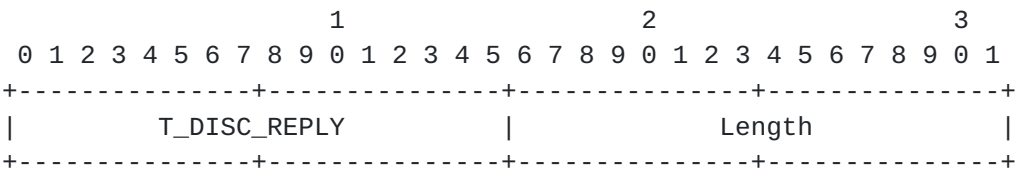


Figure 14: Reply block TLV (packet payload)

Type: 16 bits

Format of the Value field. For the Report block TLV, the type value MUST be T_DISC_REPLY.

Length: 16 bits

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

3.2.1.1. Reply Sub-Block

In addition to the Reply block, a 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 and developers may want to obtain various characteristics of content such as content's ownership and copyright, or other cache

states and conditions. Various information about device or function (or application) may be also retrieved by the variety of Reply sub-blocks. In this document, Reply sub-block TLVs for T_DISC_CONTENT and T_DISC_CONTENT_OWNER (Figure 15) and for T_DISC_GATEWAY (Figure 16) are defined; other Reply sub-block TLVs will be defined in separate document(s).

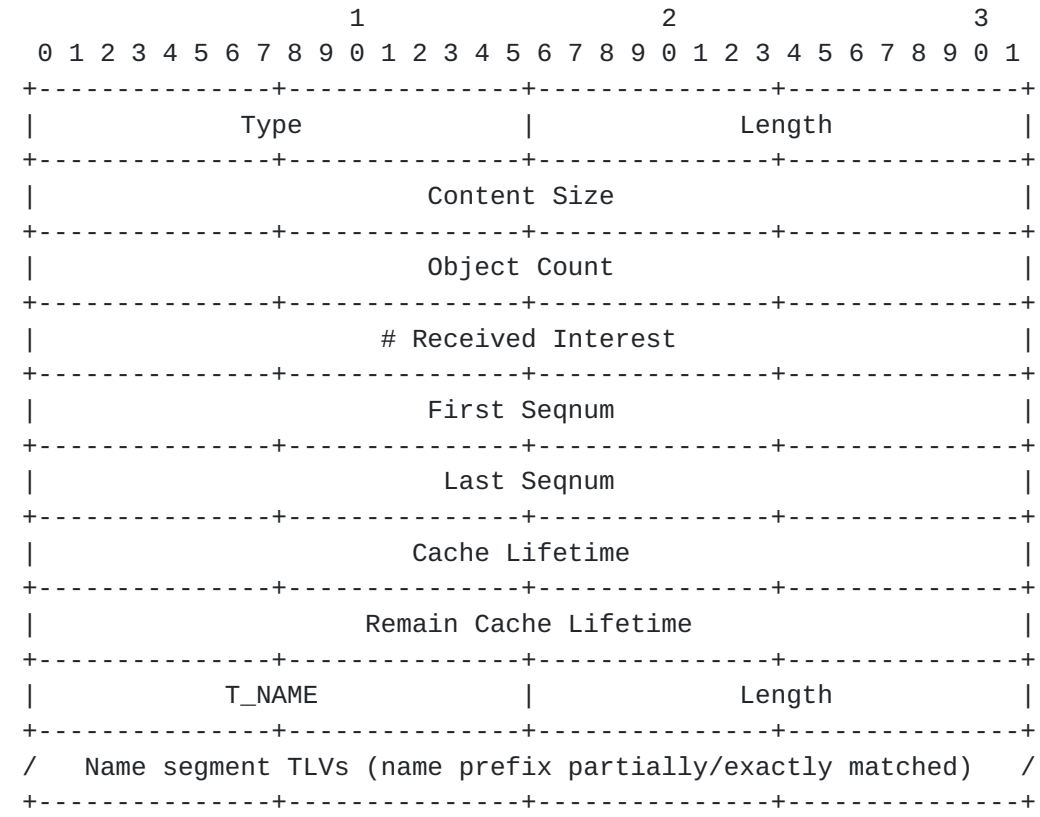


Figure 15: Reply sub-block TLV for T_DISC_CONTENT and T_DISC_CONTENT_OWNER (packet payload)

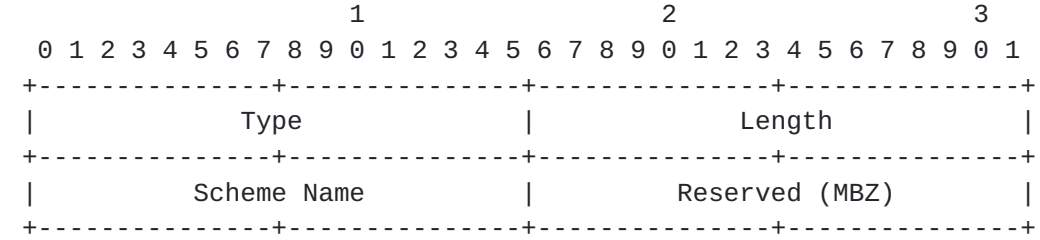


Figure 16: Reply sub-block TLV for T_DISC_GATEWAY (packet payload)

Code	Type name
0	T_DISC_CONTENT
1	T_DISC_CONTENT_OWNER
2	T_DISC_GATEWAY
3	T_DISC_DEVICE
4	T_DISC_FUNCTION
%x0FFF	T_ORG
%x1000-%x1FFF	Reserved (Experimental Use)

Figure 17: CCNinfo Reply Type Namespace

Type: 16 bits

Format of the Value field. For the Reply sub-block TLV, the type value MUST be one of the type value defined in the CCNinfo Reply Type Namespace (Figure 17). T_DISC_CONTENT is specified when the cache information is replied from a caching router. T_DISC_CONTENT_OWNER is specified when the content information is replied from a publisher. T_DISC_GATEWAY is used to discover a gateway that has a FIB for the specified scheme name.

Length: 16 bits

Length of Value field in octets.

Scheme Name: 8 bits

The code of the scheme name defined in Figure 9.

Content Size: 32 bits

The total size (MB) of the (cached) content objects. Note that the maximum size expressed by 32 bit field is 65 GB.

Object Count: 32 bits

The number of the (cached) content objects.

Received Interest: 32 bits

The number of the received Interest messages to retrieve the content.

First Seqnum: 32 bits

The first sequential number of the (cached) content objects.

Last Seqnum: 32 bits

The last sequential number of the (cached) content objects. Above First Seqnum and this Last Seqnum do not guarantee the consecutiveness of the cached content objects.

Cache Lifetime: 32 bits

The elapsed time after the oldest content object in the cache is stored. The Cache Lifetime is a 32-bit NTP timestamp, and the formula converts from a UNIX timeval to a 32-bit NTP timestamp is same as that of Section 3.1.2.

Remain Cache Lifetime: 32 bits

The lifetime of a content object, which is removed first among the cached content objects. The Remain Cache Lifetime is a 32-bit NTP timestamp.

4. CCNinfo User Behavior

4.1. Sending CCNinfo Request

A CCNinfo user initiates a CCNinfo Request by sending the Request message to the adjacent neighbor router(s) of interest. As a typical example, a CCNinfo user invokes the ccninfo command (detailed in Appendix A) that forms a Request message and sends it to the user's adjacent neighbor router(s).

When the CCNinfo user's program initiates a Request message, it MUST insert the necessary values, the "Request ID" (in the Request block) and the "Node Identifier" (in the Report block), in the Request and Report blocks. CCNinfo user's program MUST also record the Request ID at the corresponding PIT entry. The Request ID is a unique identifier for the CCNinfo Request.

After the CCNinfo user's program sends the Request message, until the Reply times out, the CCNinfo user's program MUST keep the following information; Request ID and Flags specified in the Request block, Node Identifier and Request Arrival Time specified in the Report block, and HopLimit specified in the fixed header.

4.1.1. Gateway Discovery

A CCNinfo Request can be used for gateway discovery; if a CCNinfo user invokes a CCNinfo Request with a scheme name (e.g., ccn:/ or ndn:/) and the "gateway discovery" flag value (i.e., "%x16" bit as seen in Figure 10), s/he could potentially discover a gateway that

supports different protocols such as CCN and NDN. The CCNinfo Request for gateway discovery only indicates the routing path information (see Section 4.1.2) and the scheme name whether the router is a gateway or not; it does not provide other information, e.g., cache information.

4.1.2. Routing Path Information

A CCNinfo user can send a CCNinfo Request for investigating routing path information for the specified named content. By the Request, the legitimate user can obtain; 1) identifiers (e.g., IP addresses) of intermediate routers, 2) identifier of content forwarder, 3) number of hops between content forwarder and consumer, and 4) RTT between content forwarder and consumer, per name prefix. This CCNinfo Request is terminated when it reaches the content forwarder. The ccninfo command enables user to obtain both the routing path information and in-network cache information (see below) in a same time.

4.1.3. In-Network Cache Information

A CCNinfo user can send a CCNinfo Request for investigating in-network cache information. By this Request, the legitimate user can obtain; 1) size of the cached content, 2) number of the cached chunks of the content, 3) number of the accesses (i.e., received Interests) per cache or chunk, and 4) lifetime and expiration time per cache or chunk, for Content Store (CS) in the content forwarder. 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 the Request ID noted in the Reply. If they do not match, the Reply message SHOULD be silently discarded.

If the number of the Report blocks in the received Reply is more than the initial HopLimit value (which was inserted in the original Request) + 1, the Reply SHOULD be silently ignored.

After the CCNinfo user has determined that s/he has traced the whole path or as much as s/he can expect to, s/he might collect statistics by waiting a timeout. Useful statistics provided by CCNinfo can be seen in Section 9.

5. Router Behavior

5.1. Receiving CCNinfo Request

5.1.1. Request Packet Verification

Upon receiving a CCNinfo Request message, a router MUST examine whether the message comes from a valid adjacent neighbor node. If it is invalid, the Request MUST be silently ignored. The router next examines the value of the "HopLimit" in the fixed header and the value of the "SkipHopCount" in the Request block (Figure 7). If SkipHopCount value is equal or more than the HopLimit value, the Request MUST be silently ignored.

5.1.2. Request Normal Processing

When a router receives a CCNinfo Request message, it performs the following steps.

1. HopLimit and SkipHopCount are counters that are decremented with each hop. The router terminates the CCNinfo Request when the HopLimit value becomes zero. Until the SkipHopCount value becomes zero, the router forwards the CCNinfo Request messages to the upstream router(s) (if it knows) without adding its own Report block and without replying the Request. If the router does not know the upstream router(s), without depending on the SkipHopCount value, it replies the CCNinfo Reply message with NO_ROUTE return code.

2. The router examines the Flags field of the Request block of received CCNinfo Request. If the flag value indicates "%x00" or "%x01" bit (as seen in Figure 10) for "cache information discovery", the router examines its FIB and CS. If the router caches the specified content, it inserts own Report block to the message and sends the Reply message with own Reply block and sub-block. If the router does not cache the specified content but knows the neighbor router(s) for the specified name prefix, it inserts 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 replies the CCNinfo Reply message with NO_ROUTE return code.

3. If the flag value indicates "%x02" bit for "routing path information discovery", the router examines its FIB and CS. If the router caches the specified content, it inserts own Report block to the message and sends the Reply message with own Reply block. The router does not insert any Reply sub-block here. If

the router does not cache the specified content but knows the neighbor router(s) for the specified name prefix, it inserts 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 replies the CCNinfo Reply message with NO_ROUTE return code.

4. If the flag value indicates "%x04" bit for "publisher discovery", the node receiving the Request message examines whether it owns the requested content as the publisher. If it is the publisher, it sends the Reply message with own Report block and sub-block. If the node is not the publisher but know the upstream neighbor router(s) for the specified name prefix, it adds the own Report block and forwards the Request to the neighbor(s). If the node is not the publisher and does not know the upstream neighbor router(s) for the specified name prefix, it replies the CCNinfo Reply message with NO_ROUTE return code.

5. When a router receives a CCNinfo Request in which the "gateway discovery" flag (i.e., "%x16") is set in the Flags field and a scheme name is specified, the router examines whether it has the FIB for the specified scheme name and the connections with the neighbor router(s) for the scheme protocol. If the router is the gateway, it sends the Reply message back toward the CCNinfo user. If the router does not have the FIB for the specified scheme name or does not connect to any neighbor router for the specified scheme name, the router returns the Reply with NO_GATEWAY return code.

5.2. Forwarding CCNinfo 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 at the corresponding PIT entry. The router can later decide the PIT entry to correctly forward back the Reply message even if it receives multiple Reply messages within the same timeout period. (See below.)

CCNinfo supports multipath forwarding. The Request messages can be forwarded to multiple neighbor routers. Some router may have strategy for multipath forwarding; when it 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 strategy; a CCNinfo user could trace the actual forwarding path based on the strategy. On the other hand, there may be the case that a CCNinfo user wants to discover all potential forwarding paths based on routers' FIBs. If a CCNinfo user invokes a CCNinfo Request with the parallel request flag (i.e., "%x08" bit as seen in Figure 10), the forwarding strategy will be ignored and the router sends Requests to multiple upstream routers simultaneously, and the CCNinfo user could trace the all potential forwarding paths. Note that this flag may be ignored according to the router's policy.

When the Request messages forwarded to multiple routers, the different Reply messages will be forwarded from different routers or publisher. To support this case, PIT entries initiated by CCNinfo remain until the configured CCNinfo Reply Timeout (Section 8.1) passes. In other words, unlike the ordinary Interest-Data communications in CCN, the router SHOULD NOT remove the PIT entry created by the CCNinfo Request until the timeout value expires.

CCNinfo Requests SHOULD NOT result in PIT aggregation in routers during the Request message transmission.

5.3. Sending CCNinfo Reply

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 having the Request Arrival Time and Node Identifier to the hop-by-hop TLV header of the Request message. And then the router inserts the corresponding Reply block and Reply sub-block to the payload. The router does not insert any Reply block/ sub-block 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.

When a router decides to send the Reply message for the Request for the cache or routing path information discovery, it forms the Reply message including a Reply block and a Reply sub-block with the T_DISC_CONTENT type value (Figure 15) and various cache information. After the router puts the NO_ERROR return code in the fixed header, it sends the Reply back toward the CCNinfo user.

When a router decides to send the Reply message for the Request for the publisher discovery, it forms the Reply message including a Reply block and a Reply sub-block with the T_DISC_CONTENT_OWNER type value (Figure 15) and various cache information. After the router puts the

NO_ERROR return code in the fixed header, it sends the Reply back toward the CCNinfo user.

When a router decides to send the Reply message for the Request for the gateway discovery, it forms the Reply message including a Reply block and a Reply sub-block with the T_DISC_GATEWAY type value (Figure 16) and the scheme name (Figure 9). After the router puts the NO_ERROR return code in the fixed header, it sends the Reply back toward the CCNinfo user.

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 7.

5.4. Forwarding CCNinfo Reply

When a router receives a CCNinfo Reply whose Request ID matches the one in the original CCNinfo Request block TLV from a valid adjacent neighbor node, it MUST relay the CCNinfo Reply back to 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.

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

6. Publisher Behavior

Upon receiving a CCNinfo Request message, a publisher MUST examine whether the message comes from a valid adjacent neighbor node. If it is invalid, the Request SHOULD be silently ignored.

If a publisher cannot accept the Request, it will note 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 message as the Reply back to the CCNinfo user. See Section 7 for details.

If a publisher accepts the Request forwarded by a valid adjacent neighbor node, it retrieves the local content information. The Reply message having a Reply block and Reply sub-block is transmitted back to the neighbor node that had forwarded the Request message.

7. CCNinfo Termination

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

7.1. Arriving at Publisher or Gateway

A CCNinfo Request can be determined to have arrived at the publisher or gateway.

7.2. 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.

7.3. No Route

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

7.4. No Information

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

7.5. No Space

If appending the Report block would make the CCNinfo Request packet longer than the MTU of the Incoming face, or longer than 1280 bytes (especially in the situation supporting IPv6 as the payload [3]), the router MUST note a ReturnCode of NO_SPACE in the fixed header of the message, and forwards the message as the Reply back to the CCNinfo user.

7.6. Fatal Error

A CCNinfo Request has encountered a fatal error if the last ReturnCode in the trace has the 0x80 bit set (see Section 3.1).

7.7. CCNinfo Reply Timeout

If a router receives the Request or Reply message that expires its own [CCNinfo Reply Timeout] value (Section 8.1), the router will discard the Request or Reply message.

7.8. Non-Supported Node

Cases will arise in which a router or a publisher 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.

7.9. Administratively Prohibited

If CCNinfo is administratively prohibited, a router or a publisher rejects the Request message, and the router or the publisher, or its downstream router will reply the CCNinfo Reply with the ReturnCode of ADMIN_PROHIB.

8. Configurations

8.1. 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 4 (seconds). The [CCNinfo Reply Timeout] value SHOULD NOT be larger than 5 (seconds) and SHOULD NOT be lower than 2 (seconds).

8.2. HopLimit in Fixed Header

If a CCNinfo user does not specify the HopLimit value in a fixed header for a Request message as the HopLimit, the HopLimit is set to 32. Note that 0 HopLimit is an invalid Request and hence discarded.

8.3. 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.

9. Diagnosis and Analysis

9.1. Number of Hops

A CCNinfo Request message is forwarded in a hop-by-hop manner and each forwarding router appended its own Report block. We can then

verify the number of hops to reach the content forwarder or the publisher.

9.2. Caching Router and Gateway Identification

It is possible to identify the caching routers or a gateway in the path from the CCNinfo user to the content forwarder, while some routers may hide their identifier (with all-zeros) in the Report blocks (Section 10.1).

9.3. TTL or Hop Limit

By taking the HopLimit from the content forwarder and forwarding 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.

9.4. Time Delay

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

9.5. Path Stretch

By getting 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 improvement in path stretch in various cases, such as different caching and routing algorithms. We can then facilitate investigation of the performance of the protocol.

9.6. Cache Hit Probability

CCNinfo can show the number of received interests per cache or chunk on a router. By this, CCNinfo measures the content popularity (i.e., the number of accesses for each content/cache), and you can investigate the routing/caching strategy in networks.

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 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 access policy, routers having the specified content can examine the signature enclosed in the Request message and decide whether they should notify the content information in the Reply or not. If the routers decide to not notify the content information, they reply the CCNinfo Reply with the ReturnCode of ADMIN_PROHIB without appending any Reply (sub-)block TLV. For the latter policy, the permission, whether (1) All (all cache information is disclosed), (2) Partial (cache information with the particular name prefix can (or cannot) be disclosed), or (3) Deny (no cache information is disclosed), is defined at routers.

On the other hand, we entail that each router does not disrupt forwarding CCNinfo Request and Reply messages. When a Request message is received, the router SHOULD insert Report block. 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. At last, the router SHOULD forward the Request message to the upstream router toward the content forwarder if no fatal error occurs.

10.2. Filtering of CCNinfo Users Located in Invalid Networks

A router MAY support an access control mechanism to filter out Requests from invalid CCNinfo users. For it, invalid networks (or domains) could, for example, be configured via a list of allowed/disallowed networks (as seen in Section 8.3). If a Request is received from the disallowed network (according to the Node Identifier in the Request block), the Request SHOULD NOT be processed and the Reply with the ReturnCode of INFO_HIDDEN may 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 a secret, CCNinfo Requests may 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 what publishers are sending to what kinds of contents. If this information is a secret, CCNinfo Requests may be restricted at the border of the domain, using the ADMIN_PROHIB return code.

10.5. Longer or Shorter CCNinfo Reply Timeout

Routers can configure the CCNinfo Reply Timeout (Section 8.1), which is the allowable timeout value to keep the PIT entry. If routers configure the longer timeout value, there may be an attractive attack vector against PIT memory. Moreover, especially when the parallel request option (Section 5.2) is specified for the CCNinfo Request, a number of Reply messages may come back and cause a response storm. (See Section 10.7 for rate limiting to avoid the storm). In order to avoid DoS attacks, routers may configure the shorter timeout value 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 the all Replies and only retrieves the partial path information (i.e., information about part of the tree).

There may be the way to allow for incremental exploration (i.e., to explore the part of the tree the previous operation did not explore), whereas discussing such mechanism is out of scope of this document.

10.6. Limiting Request Rates

A router may 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. No error is returned. The rate limit is left to the router's implementation.

10.7. Limiting Reply Rates

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

10.8. Adjacency Verification

CCNinfo Request and Reply messages MUST be forwarded by adjacent neighbor nodes or routers. Forwarding CCNinfo messages given from non-adjacent neighbor nodes/routers MUST be prohibited. Such invalid messages SHOULD be silently discarded. Note that defining the secure

way to verify the adjacency cannot rely on the way specified in CCNx message format or semantics. An adjacency verification mechanism and the corresponding TLV for adjacency verification using hop-by-hop TLV header will be defined in a separate document.

11. Acknowledgements

The authors would like to thank Spyridon Mastorakis, Ilya Moiseenko, and David Oran 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", draft-irtf-icnrg-ccnxmessages-07 (work in progress), March 2018.

[2] Bradner, S., "Key words for use in RFCs to indicate requirement levels", RFC 2119, March 1997.

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

12.2. Informative References

[4] 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.

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

[6] Asaeda, H., Mayer, K., and W. Lee, "Mtrace Version 2: Traceroute Facility for IP Multicast", draft-ietf-mboned-mtrace-v2-24 (work in progress), June 2018.

Appendix A. ccninfo Command and Options

The ccninfo command enables the CCNinfo user to investigate the routing path based on the name prefix of the content (e.g., ccn:/news/today), device name, and function (or application) name. The name prefix, device name, and function name (or application name) are mandatory but 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 [-P] [-g] [-f] [-n] [-o] [-r hop_count] [-s hop_count]
       name_prefix; or,

Usage: ccninfo [-r hop_count] [-s hop_count] device_name |
       function_name (or application_name)
```

name_prefix
: Name prefix of the content (e.g., ccn:/news/today) the CCNinfo user wants to trace. If the CCNinfo user specifies only a scheme name, e.g., "ccn:/", s/he must specify "-g" option (i.e., ccninfo -g ccn:/). In that case, the CCNinfo user discovers the router having the FIB of the specified scheme name and the RTT between CCNinfo user and the router. The "-P" option allows a partial match for the name prefix; otherwise, an exact match is required.

device_name
: Device name (e.g., ccn:/%device/server-A, ccn:/%device/sensor-123) the CCNinfo user wants to trace. Here, we assume the ccninfo command with the "%device" prefix indicates the trace request for specified device/server/node, but defining the syntax of device name specification is [TBD].

function_name (or application_name)
: Function name (e.g., ccn:/%function/firewall, ccn:/%function/transcoding/mpeg2-h.264) or application name (e.g., ccn:/%application/mplayer) the CCNinfo user wants to trace. Here, we assume the ccninfo command with the "%function" or "%application" prefix indicates the trace request for specified function or application, but defining the syntax of function or application name specification is [TBD].

g option
: This option enables to discover a gateway that supports specified scheme name and has multiple FIBs. When a CCNinfo user specifies only a scheme name, e.g., "ccn:/", this option must be specified and other content name prefix is ignored.

f option
: This option enables to ignore the forwarding strategy and send CCNinfo Requests to multiple upstream routers simultaneously. The CCNinfo user could then trace the all potential forwarding paths.

n option
: This option can be specified if a CCNinfo user only needs the routing path information to the specified content/cache and RTT

between CCNinfo user and content forwarder (i.e., cache information is not given).

o option

This option enables to trace the path to the content publisher. If this option is specified, each router along the path to the publisher only forwards the Request message; it inserts each Report block but does not send Reply even if it caches the specified content. The publisher (who has the complete set of content and is not a caching router) replies the Reply message. Specifying only a scheme name is not allowed with this option.

r option

Number of traced routers. If the CCNinfo user specifies this option, only the specified number of hops from the CCNinfo user trace the Request; each router inserts its own Report block and forwards the Request message to the upstream router(s), and 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. If there is a caching router within the hop count along the path, the caching router sends back the Reply message and terminates the trace request. If the last router does not have the corresponding cache, it replies the Reply message with NO_INFO return code (described in Section 3.1) with no Reply block TLV inserted. The Request messages are terminated at publishers; therefore, although the maximum value for this option a CCNinfo user can specify is 255, the Request messages should be in general reached at the publisher within significantly lower than 255 hops.

s option

Number of skipped routers. If the CCNinfo user specifies this option, the number of hops from the CCNinfo user simply forward the CCNinfo Request messages without adding its own Report block and without replying the Request, and the next upstream router 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", the three upstream routers along the path only forwards the Request message, but does not append their Report blocks in the hop-by-hop headers and does not send the Reply messages even though they have the corresponding cache. The Request messages are terminated at publishers; therefore, although the maximum value for this option a CCNinfo user can specify is 255, if the

Request messages reaches the publisher, the publisher silently discards the Request message and the request will be timed out.

Appendix B. Change History

This document was created based on the previous "Contrace" document whose initial version had been published on October 31, 2016.

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

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

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