

ICNRG
Internet-Draft
Intended status: Experimental
Expires: 29 October 2022

C. Gündoğan
TC. Schmidt
HAW Hamburg
D. Oran
Network Systems Research and Design
M. Waehlich
link-lab & FU Berlin
27 April 2022

Alternative Delta Time Encoding for CCNx Using Compact Floating-Point
Arithmetic
draft-irtf-icnrg-ccnx-timetlv-00

Abstract

CCNx utilizes delta time for a number of functions. When using CCNx in environments with constrained nodes or bandwidth constrained networks, it is valuable to have a compressed representation of delta time. In order to do so, either accuracy or dynamic range has to be sacrificed. Since the current uses of delta time do not require both simultaneously, one can consider a logarithmic encoding such as that specified in [RFC5497] and [IEEE.754.2019]. This document updates _CCNx messages in TLV Format_ [RFC8609] to specify this alternative encoding.

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 29 October 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Revised BSD License.

Table of Contents

1.	Introduction	2
2.	Terminology	3
3.	Usage of Time Values in CCNx	3
3.1.	Relative Time in CCNx	3
3.2.	Absolute Time in CCNx	3
4.	A Compact Time Representation with Logarithmic Range	4
5.	Protocol Integration of the Compact Time Representation	6
5.1.	Interest Lifetime	7
5.2.	Recommended Cache Time	8
6.	IANA Considerations	9
7.	Security Considerations	9
8.	References	9
8.1.	Normative References	9
8.2.	Informative References	9
Appendix A.	Test Vectors	10
Appendix B.	Efficient Time Value Approximation	11
	Acknowledgments	11
	Authors' Addresses	11

[1.](#) Introduction

CCNx utilizes time values for a number of functions. Some of these are expressed as absolute time, others as delta time. CCNx is well suited for Internet of Things (IoT) applications [[RFC7927](#)]. When using CCNx in environments with constrained nodes or bandwidth constrained networks, it is valuable to have a compact representation of time values. For example [[RFC9139](#)] and [[ICNLOWPAN](#)] specify a compression scheme useful over IEEE 802.15.4 networks. However, any compact time representation has to sacrifice accuracy or dynamic range. For some time uses this is relatively straightforward to achieve, for other uses, it is not. This document discusses the various cases, and proposes a compact encoding that is easily

accommodated for delta times.

[2.](#) Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

This document uses the terminology of [[RFC8569](#)] and [[RFC8609](#)] for CCNx entities.

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

byte: synonym for octet

time value: a time offset measured in seconds

time code: an 8-bit encoded time value

[3.](#) Usage of Time Values in CCNx

[3.1.](#) Relative Time in CCNx

CCNx, as currently specified in [[RFC8569](#)], utilizes delta time for only the lifetime of an Interest message (see sections [2.1](#), [2.2](#), [2.4.2](#), [10.3](#) of [[RFC8569](#)]). It is a hop-by-hop header value, and is currently encoded via the T_INTLIFE TLV as a 64-bit integer ([[RFC8609](#)] [section 3.4.1](#)). While formally an optional TLV, in all but some corner cases every Interest message is expected to carry the Interest Lifetime TLV, and hence having compact encoding is particularly valuable for keeping Interest messages short.

Since the current uses of delta time do not require both accuracy and dynamic range simultaneously, one can consider a logarithmic encoding such as that specified in [[IEEE.754.2019](#)] and outlined in [Section 4](#). This document updates _CCNx messages in TLV Format_ [[RFC8609](#)] to permit this alternative encoding for selected time values. See [Section 6](#) for the specific actions needed to register this

alternative compact representation of Interest Lifetime.

[3.2.](#) Absolute Time in CCNx

CCNx, as currently specified in [\[RFC8569\]](#), utilizes absolute time for various important functions. Each of these absolute time usages poses a different challenge for a compact representation. These are discussed in the following subsections.

[3.2.1.](#) Signature Time and Expiry Time

`_Signature Time_` is the time the signature of a content object was generated (sections [8.2–8.4](#) [\[RFC8569\]](#)). `_Expiry Time_` indicates the expiry time of a content object ([section 4](#) [\[RFC8569\]](#)). Both values are content message TLVs and represent absolute timestamps in milliseconds since the UTC epoch (i.e., an NTP timestamp). They are currently encoded via the `T_SIGTIME` and `T_EXPIRY` TLVs as 64-bit unsigned integers (see [section 3.6.4.1.4.5](#) and [3.6.2.2.2](#) [\[RFC8609\]](#)).

Both time values could be in the past, or in the future, potentially by a large delta. They are also included in the security envelope of the message. Therefore, it seems there is no practical way to define an alternative compact encoding that preserves its semantics and security properties; hence we don't consider it further as a candidate.

[3.2.2.](#) Recommended Cache Time

`_Recommended Cache Time_` (RCT) for a content object (see [section 4](#) [\[RFC8569\]](#)) is a hop-by-hop header stating the expiration time for a cached content object in milliseconds since the UTC epoch (i.e., an NTP timestamp). It is currently encoded via the `T_CACHETIME` TLV as a 64-bit unsigned integer (see [section 3.4.2](#) [\[RFC8609\]](#)).

A recommended cache time could be far in the future, but cannot be in the past and is likely to be a reasonably short offset from the current time. Therefore, this document allows the recommended cache time to be interpreted as a relative time value rather than an

absolute time, since the semantics associated with an absolute time value do not seem to be critical to the utility of this value. This document therefore updates the recommended cache time with the following rule set:

- * Use absolute time as per [\[RFC8609\]](#)
- * Use relative time, if the compact time representation is used (see [Section 4](#) and [Section 5](#))

4. A Compact Time Representation with Logarithmic Range

This document uses the compact time representation of ICNLoWPAN (see [section 7 of \[RFC9139\]](#)) that is inspired by [\[RFC5497\]](#) and [\[IEEE.754.2019\]](#). Its logarithmic encoding supports a representation ranging from milliseconds to years. Figure 1 depicts the logarithmic nature of this time representation.

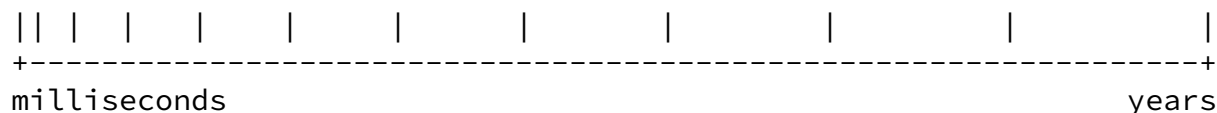


Figure 1: A logarithmic range representation allows for higher precision in the smaller time ranges and still supports large time deltas.

Time codes encode exponent and mantissa values in a single byte, but in contrast to the representation in [\[IEEE.754.2019\]](#), time codes only encode positive numbers and hence do not include an extra sign bit. Figure 2 shows the configuration of a time code: an exponent width of 5 bits, and a mantissa width of 3 bits.

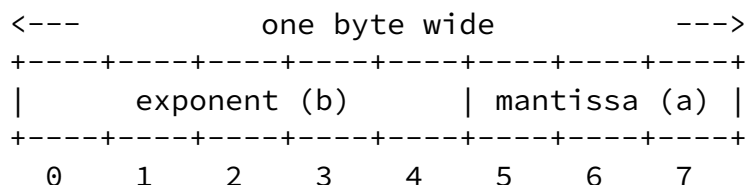


Figure 2: A time code with exponent and mantissa to encode a logarithmic range time representation.

The base unit for time values are seconds. A time value is calculated using the following formula (adopted from [[RFC5497](#)] and [[RFC9139](#)]), where (a) represents the mantissa, (b) the exponent, and (C) a constant factor set to $C := 1/32$.

Subnormal ($b == 0$): $(0 + a/8) * 2 * C$

Normalized ($b > 0$): $(1 + a/8) * 2^b * C$

The subnormal form provides a gradual underflow between zero and the smallest normalized number. Eight time values exist in the subnormal range [0s, ~0.054688s] with a step size of ~0.007812s between each time value. This configuration also encodes the following convenient numbers in seconds: [1, 2, 4, 8, 16, 32, 64, ...]. [Appendix A](#) further includes test vectors to illustrate the logarithmic range.

An example algorithm to encode a time value into the corresponding exponent and mantissa is given as pseudo code in Figure 3. Not all time values can be represented by a time code. For these instances, the closest time code is chosen that is smaller than the value to encode.

```
input: float v    // time value
output: int a, b  // mantissa, exponent of time code

(a, b) encode (v):

    if (v == 0)
        return (0, 0)

    if (v < 2 * C)                                // subnormal
        a = floor (v * 4 / C)                    // round down
        return (a, 0)
    else                                           // normalized
        if (v > (1 + 7/8) * 2^31 * C)            // check bounds
            return (7, 31)                        // return maximum
        else
```

```

b = floor (log2(v / C))           // round down
a = floor ((v / (2^b * C) - 1) * 8) // round down
return (a, b)

```

Figure 3: Algorithm in pseudo code.

As an example: No specific time code for 0.063 exists, but this algorithm maps to the closest valid time code that is smaller, i.e., exponent 1 and mantissa 0 (the same as for time value 0.0625).

5. Protocol Integration of the Compact Time Representation

A straightforward way to accommodate the compact time approach is to use a 1-byte length field to indicate this alternative encoding while retaining the existing TLV registry entries. This approach has backward compatibility problems, but may still be considered for the following reasons:

- * Both CCNx RFCs are experimental and not Standards Track, hence expectations for forward and backward compatibility are not as stringent. "Flag day" upgrades of deployed CCNx networks, while inconvenient, are still feasible.
- * The major use case for these compressed encodings are smaller-scale IoT and/or sensor networks where the population of consumers, producers, and forwarders is reasonably small.
- * Since the current TLVs have hop-by-hop semantics, they are not covered by any signed hash and hence may be freely re-encoded by any forwarder. That means a forwarder supporting the new encoding can translate freely between the two encodings.

- * The alternative of assigning new TLV registry values does not substantially mitigate the interoperability problems anyway.

The following lists alternative approaches of integrating the compact time representation for time offsets in CCNx messages. A further analysis, discussion, and decision on the best suited approach will be added as the document progresses.

1. Relative time TLVs (e.g., T_INTLIFETIME) include nested TLVs to hint at the used encoding. This approach allows for versatility in defining new time encodings, but adds a TLV overhead that negates the benefits of the compact time representation.
2. A new TLV type for the compact time representation is defined (T_INTLIFETIME_COMPACT). The packet header grammar from [RFC8609] is updated to allow for T_INTLIFETIME_COMPACT at the same level of the currently defined T_INTLIFETIME.

5.1. Interest Lifetime

The Interest Lifetime definition in [RFC8609] allows for a variable-length lifetime representation, where a length of 1 encodes the linear range [0,255] in milliseconds. This document changes the definition to always encode 1-byte Interest lifetime values in the compact time value representation (Figure 4).

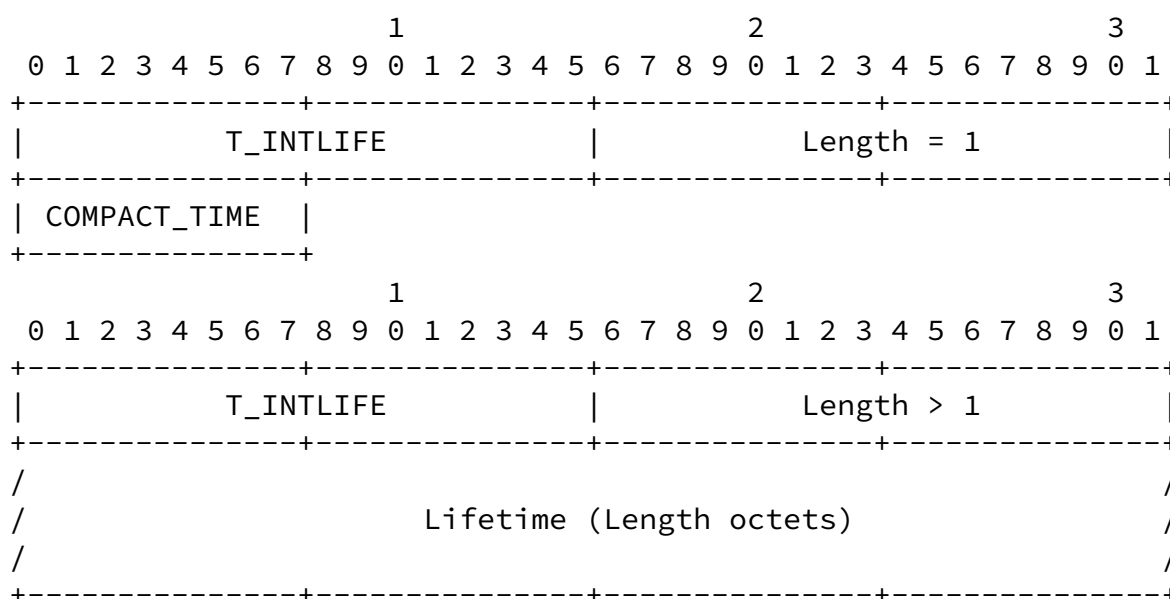


Figure 4: Changes to the definition of the Interest Lifetime TLV.

compact time representation will interpret the time value as an Interest lifetime between 0 and 255 milliseconds. This may lead to a degradation of network performance, since Pending Interest Table (PIT) entries timeout quicker than expected.

5.2. Recommended Cache Time

The Recommended Cache Time definition in [RFC8609] specifies an absolute time representation that is of a length fixed to 8 bytes. This document changes the definition to always encode 1-byte Recommended Cache Time values in the compact relative time value representation (Figure 5).

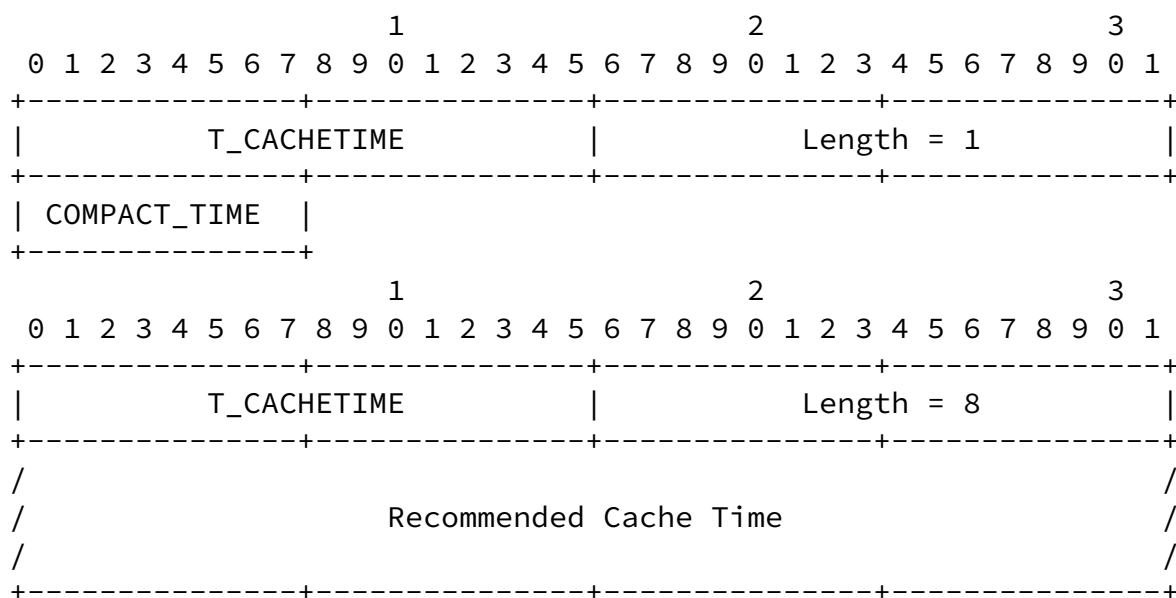


Figure 5: Changes to the definition of the Recommended Cache Time TLV.

The packet processing is adapted to calculate an absolute time from the relative time code based on the absolute reception time. On transmission, a new relative time code is calculated based on the current system time.

A note on legacy forwarders: A forwarder that does not support this compact time representation is expected to consider a Recommended Cache Time with length 1 as a structural or syntactical error and discard the packet. Otherwise, a forwarder interprets the compact 1-byte time value as an absolute time far in the past, which impacts cache utilization.

6. IANA Considerations

Based on the approach of integration, certain TLV registries from [RFC8609] need to be updated.

7. Security Considerations

This document makes no semantic changes to [RFC8569], nor to any of the security properties of the message encodings of [RFC8609], and hence has the same security considerations as those two existing documents.

8. References

8.1. Normative References

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

8.2. Informative References

[ICNLOWPAN]

Gündoğan, C., Kietzmann, P., Schmidt, T., and M. Wählisch, "Designing a LoWPAN convergence layer for the Information Centric Internet of Things", Computer Communications, Vol. 164, No. 1, p. 114–123, Elsevier, December 2020, <<https://doi.org/10.1016/j.comcom.2020.10.002>>.

[IEEE.754.2019]

Institute of Electrical and Electronics Engineers, C/MSD – Microprocessor Standards Committee, "Standard for Floating-Point Arithmetic", June 2019, <<https://standards.ieee.org/content/ieee-standards/en/standard/754-2019.html>>.

[RFC5497] Clausen, T. and C. Dearlove, "Representing Multi-Value Time in Mobile Ad Hoc Networks (MANETs)", [RFC 5497](#), DOI 10.17487/RFC5497, March 2009, <<https://www.rfc-editor.org/info/rfc5497>>.

[RFC7927] Kutscher, D., Ed., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlich, "Information-Centric Networking (ICN) Research Challenges", [RFC 7927](#), DOI 10.17487/RFC7927, July 2016,

- [RFC8569] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", [RFC 8569](#), DOI 10.17487/RFC8569, July 2019, <<https://www.rfc-editor.org/info/rfc8569>>.
- [RFC8609] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format", [RFC 8609](#), DOI 10.17487/RFC8609, July 2019, <<https://www.rfc-editor.org/info/rfc8609>>.
- [RFC9139] Gündoğan, C., Schmidt, T., Wählich, M., Scherb, C., Marxer, C., and C. Tschudin, "Information-Centric Networking (ICN) Adaptation to Low-Power Wireless Personal Area Networks (LoWPANs)", [RFC 9139](#), DOI 10.17487/RFC9139, November 2021, <<https://www.rfc-editor.org/info/rfc9139>>.

[Appendix A](#). Test Vectors

The test vectors in Table 1 show sample time codes and their corresponding time values according to the algorithm outlined in [Section 4](#).

+=====+	
Time Code	Time Value (seconds)
+=====+	
0x00	0.000000
+-----+	
0x01	0.007812
+-----+	
0x04	0.031250
+-----+	
0x08	0.062500
+-----+	
0x15	0.203125
+-----+	
0x28	1.000000
+-----+	
0x30	2.000000
+-----+	

0xF8 67108864.000000
+-----+
0xFF 125829120.000000
+-----+

Table 1: Test Vectors

[Appendix B](#). Efficient Time Value Approximation

A forwarder frequently converts compact time into milliseconds to compare Interest lifetimes and the duration of cache entries. On many architectures, multiplication and division perform slower than addition, subtraction, and bit shifts. The following equations approximate the formulas in [Section 4](#), and scale from seconds into the milliseconds range by applying a factor of 2^{10} instead of 10^3 . This results in an error of 2.4%.

$$\begin{aligned} b == 0: & \quad 2^{10} * a * 2^{-3} * 2^1 * 2^{-5} \\ & = a \ll 3 \end{aligned}$$

$$\begin{aligned} b > 0: & \quad (2^{10} + a * 2^{-3} * 2^{10}) * 2^b * 2^{-5} \\ & = (1 \ll 5 + a \ll 2) \ll b \end{aligned}$$

Acknowledgments

We would like to thank the active members of the ICNRG research group for constructive thoughts. In particular, we thank Marc Mosko and Ken Calvert for their valuable feedback on the encoding scheme and the protocol integration.

Authors' Addresses

Cenk Gündoğan
HAW Hamburg
Berliner Tor 7
D-20099 Hamburg
Germany
Phone: +4940428758067
Email: cenk.guendogan@haw-hamburg.de

URI: <http://inet.haw-hamburg.de/members/cenk-gundogan>

Thomas C. Schmidt

HAW Hamburg

Berliner Tor 7

D-20099 Hamburg

Germany

Email: t.schmidt@haw-hamburg.de

URI: <http://inet.haw-hamburg.de/members/schmidt>

Gündoğan, et al.

Expires 29 October 2022

[Page 11]

Internet-Draft

TimeTLV for CCNx

April 2022

Dave Oran

Network Systems Research and Design

4 Shady Hill Square

Cambridge, MA 02138

United States of America

Email: daveoran@orandom.net

Matthias Waehlich

link-lab & FU Berlin

Hoenower Str. 35

D-10318 Berlin

Germany

Email: mw@link-lab.net

URI: <http://www.inf.fu-berlin.de/~waehl>

