ICNRG
Internet-Draft
Intended status: Experimental
Expires: 25 July 2022

C. Gundogan
TC. Schmidt
HAW Hamburg
D. Oran
Network Systems Research and Design
M. Waehlisch
link-lab & FU Berlin

21 January 2022

Alternative Delta Time Encoding for CCNx Using Compact Floating-Point Arithmetic
draft-gundogan-icnrg-ccnx-timetlv-05

Abstract

CCNx utilizes delta time for a number of functions. When using CCNx in environments with constrained nodes and/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 [IEEE.754.2019]. This document updates _CCNx messages in TLV Format_ (RFC8609) to specify this alternative encoding.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
2. Terminology 2
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 8
7. Security Considerations 8
8. References 8
 8.1. Normative References 8
 8.2. Informative References 9
Appendix A. Test Vectors 9
Acknowledgments 10
Authors' Addresses 10

1. Introduction

CCNx utilizes time values for a number of functions. Some of these are expressed as absolute time, others as delta time. When using CCNx in environments with constrained nodes and/or bandwidth constrained networks, it is valuable to have a compact representation of time values. For example [RFC9139] specifies a compression scheme useful over IEEE 802.15.4 networks. However, any compact time representation has to sacrifice either accuracy or dynamic range or both. 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 [RFC8609] and section

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.

```
 || |  |   |    |     |      |       |        |         |          |
 +-----------------------------------------------------------------+
 milliseconds                                                  years
```

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.

```
<---          one byte wide          --->
+----+----+----+----+----+----+----+----+
|      exponent (b)      | mantissa (a) |
+----+----+----+----+----+----+----+----+
  0    1    2    3    4    5    6    7
```

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 is the least intrusive integration, but adds a TLV overhead that negates the benefits of the compact time representation.

2. A new TLV type for T_INTLIFETIME with a compact time representation (T_INTLIFETIME_COMPACT) is defined. The packet header grammar from [RFC8609] is updated to allow for T_INTLIFETIME_COMPACT at the same level of the currently defined T_INTLIFETIME with an exclusive or.

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).

```
                     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_INTLIFE           |           Length > 1          |
+---------------+---------------+---------------+---------------+
/                                                               /
/                      Lifetime (Length octets)                 /
/                                                               /
+---------------+---------------+---------------+---------------+

                     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_INTLIFE           |           Length = 1          |
+---------------+---------------+---------------+---------------+
| COMPACT_TIME  |
+---------------+
```

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

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).

```
                     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_CACHETIME          |           Length = 8          |
+---------------+---------------+---------------+---------------+
/                                                               /
/                     Recommended Cache Time                    /
/                                                               /
+---------------+---------------+---------------+---------------+
                     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_CACHETIME          |           Length = 1          |
+---------------+---------------+---------------+---------------+
| COMPACT_TIME  |
+---------------+
```

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.

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

[IEEE.754.2019]
Institute of Electrical and Electronics Engineers, C/MSC - Microprocessor Standards Committee, "Standard for Floating-Point Arithmetic", June 2019, <https://standards.ieee.org/content/ieee-standards/en/standard/754-2019.html>.

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

[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>.

[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>.

8.2. Informative References

[RFC9139] Gündoğan, C., Schmidt, T., Wählisch, 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

Acknowledgments

We would like to thank the active members of the ICNRG research group for constructive thoughts. In particular, we thank Marc Mosko for his feedback on the encoding scheme, the provided pseudo code, and test vectors.

Authors' Addresses

Cenk Gundogan
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

Dave Oran
Network Systems Research and Design
4 Shady Hill Square
Cambridge, MA 02138
United States of America

Email: daveoran@orandom.net

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

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