CCNx Cache Control
draft-mosko-icnrg-cachecontrol-00

Abstract

This document proposed a modification of the CCNx 1.0 cache control directives to allow use of either absolute times or relative times by the publisher. Intermediate caches do not require synchronized clocks even if the publisher uses an absolute time. This document updates the definition of the RecommendedCacheTime but leaves the ExpiryTime as previously defined.

Status of this Memo

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

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

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

This Internet-Draft will expire on February 17, 2017.

Table of Contents

TOC

1. Introduction

The current CCNx 1.0 specification [CCNSemantics] (Mosko, M., Solis, I., and C. Wood, “CCNx Semantics (Internet draft),” 2016.) defines two cache control directives: RecommendedCacheTime and ExpiryTime. Both use absolute times and thus require caches have synchronized time. We propose an extension to CCNx that allows the use of relative times and the processing of ExpiryTime without synchronized time between caches.

This draft does not change the semantics of ExpiryTime and RecommendedCacheTime (RCT). ExpiryTime is the maximum time during which a cache can respond with a ContentObject. RCT is the maximum time during which a cache should keep a ContentObject, though any cache may discard early or may choose to keep longer such as if it is popular. We introduce new MaxAge and Age fields for use in relative time expiry calculations.

This proposal is somewhat based on the HTTP/1.1 age system, but not exactly the same due to the definitions of our existing ExpiryTime and RecommendedCacheTime (RCT) fields. It also has different semantics because HTTP allows for stale and fresh, while in CCNx we have either alive or dead (expired).

To use relative times for cache control requires and Age field in the unsigned packet headers. The Age field is incremented by a cache when it serves a ContentObject based on its residency time in the cache. Combining a MaxAge field from the signed portion of the ContentObject with the Age field allows one to determine if the ContentObject is expired or not. If Age >= MaxAge, then the ContentObject is expired.

We can use a single Age field for ContentObjects with either MaxAge or ExpiryTime (it would not have both). If using MaxAge, then Age is a relative time with a base of 0. If using ExpiryTime, then Age is initialized to the current synchronized time by the publisher and intermediate caches increment it from there.

Because the Age field is outside the signature envelope, one cannot rely on it for trusted protocol operation. Not having it, however, results in worse side effects than relying on the unauthenticated Age field. If all nodes are operating correctly, we can make a correct and live protocol using the rule that only the publisher is allowed to decrement the age field and intermediate caches cannot decrement it. This is not true of only having a MaxAge field without an Age field, as a ContentObject could travel in a cycle and live for longer than MaxAge.

We do not consider allowing an Interest to have a min-age directive.

TOC

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.) [RFC2119].

TOC

2. Protocol Description

All of these headers only control when a cache and respond with a ContentObject. They do not affect the data plane.

- ContentObject.ExpiryTime: absolute time
- ContentObject.MaxAge: relative seconds
- PerhopHeaders.RCT: absolute time
- PerhopHeaders.Age: relative milli-seconds

We consider time in flight to be negligible compared to time in storage or the second resolution of MaxAge and Age. If a cache has reason to believe there is a large time in flight before a ContentObject was received from the previous hop, it should take that time into consideration of its residency time at the cache. For example, a space system might know that a certain communication takes 4 minutes or a satellite system might know that one hop takes 300 msec.

The RCT is re-defined to now always be relative to the Age. Once the Age exceeds the RCT, a node SHOULD stop serving it from cache, but is not required to stop.

TOC

2.1. Publisher

If it does not have synchronized time, it MUST NOT use ExpiryTime. It MAY use MaxAge.

A publisher MUST NOT use both ExpiryTime and MaxAge.

When first creating a Content Object:

- If using ExpiryTime, the publisher MUST add an Age header and set it to the current synchronized time.
- If using MaxAge, the publisher MUST add an Age header and set it to 0.
- The RCT field, if used, is always taken as a delta from the Age field.
- If using RCT and neither of ExpiryTime or MaxAge, the publisher MUST add an Age field with the desired delta before the RCT. It SHOULD use a base 0 for the Age, but MAY use any desired base.

If answering an Interest for a ContentObject that was already created:

- It MAY use the above procedure, which essentially refreshes the ContentObject.
- It MAY account for the elapsed time by using the current synchronized time or relative Age, as appropriate.

The ContentStore on the publisher node operates as normal for any ContentStore.

TOC

2.2. Content Store

In these calculations, we will assume a missing field has the value of 0. For example, a ContentObject with an ExpiryTime has an implicit MaxAge of 0. Remember that Age is in milli-seconds and MaxAge is in seconds, so one should re-scale MaxAge when comparing it to Age.

The residency_time is the rounded-up time in milli-seconds that a ContentObject has resided in the cache. The minimum value is 1. This ensures that the Age is always increasing as it is stored and served from a set of caches.

```
age_value = Age + residency_time
remaining_age = max(MaxAge - age_value, 0)
remaining_expiry = max(ExpiryTime - now, 0)
```

At most one of remaining_age and remaining_expiry will be positive. Both could be 0.

```
remaining_life = max(remaining_age, remaining_expiry)
```

If remaining_life > 0, then the ContentObject may be used from cache. Otherwise, it should be considered a cache miss.

This rule ensures that the Age field never decreases at an intermediate cache and will try to catch up to synchronized time whenever possible. When responding with a ContentObject from cache, a node MUST:

- If it has synchronized time and ExpiryTime in packet, set the Age field to max(now, Age)
- Otherwise, set the Age field to (Age + residency_time)

If the ContentStore receives the same ContentObject but with a younger Age, it should reset its Age to the new value. This allows the publisher to re-issue the same ContentObject with a new validity period without re-signing and creating a new ContentObject. Because the protocol maintains the invariant that only the publisher can decrement the Age, the protocol should not induce permanent cycles of ContentObjects bouncing between caches.

TOC

2.3. Consumer

If the consumer has synchronized time and the ContentObject has an ExpiryTime, it MAY perform a validation on the ContentObject to determine if it is out of date. Otherwise, it MAY consider the Age field relative to the MaxAge or ExpiryTime. It is possible for a ContentObject to expire while in flight, so a Consumer usually should not penalize recently expired objects, though this behavior is application dependent.

TOC

2.4. Allowed Combinations

None of these combinations require synchronized clocks at any node and all allow a mixture of synchronized and unsynchronized clocks.

No cache is required to keep a ContentObject for any amount of time. No cache is required to discard a ContentObject after the RCT, though it should unless it is popular. All caches must cease to respond with a ContentObject that has expired, based either on Age to MaxAge or Age to ExpiryTime comparison or based on comparing a synchronized clock to the ExpiryTime. Comparing a synchronized clock to ExpiryTime always takes precedence over the Age field to determine expiry.

1. RCT and Age only: The RCT functions relative to Age. The ContentObject never expires (dies) and could live in cache forever. However, nodes would normally discard it once RCT <= Age.
2. Age and ExpiryTime: Initialize the Age to the current synchronized time. Intermediate caches may serve it so long as Age < ExpiryTime.
3. Age and MaxAge: Initialize Age to 0. Intermediate caches may serve it so long as Age < MaxAge.
4. Age and ExpiryTime and RCT: As #2, except may discard once RCT <= Age
5. Age and MaxAge and RCT: As #3, except may discard once RCT <= Age.

To prevent a ContentObject from being served from cache, a producer should set the ExpiryTime to 0.

TOC

3. IANA Considerations

This memo includes no request to IANA. TODO: If this document is submitted as an official draft, this section must be updated to reflect the IANA registries described in [CCNMessages] (Mosko, M., Solis, I., and C. Wood, “CCNx Messages in TLV Format (Internet draft),” 2016.)

TOC

4. Security Considerations

Because the Age field is outside the security envelope, a system relying on only relative times (MaxAge) could be susceptible to intermediate nodes decreasing the Age and a ContentObject living beyond the desired MaxAge. If this is highly sensitive, then a consumer and producer could use synchronized clocks and ExpiryTime.

TOC

5. References

TOC

5.1. Normative References

[RFC2119] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).

TOC

5.2. Informative References

[CCNMessages] Mosko, M., Solis, I., and C. Wood, “CCNx Messages in TLV Format (Internet draft),” 2016.

[CCNSemantics] Mosko, M., Solis, I., and C. Wood, “CCNx Semantics (Internet draft),” 2016.

[CCNx] PARC, Inc., “CCNx Open Source,” 2007.

[RFC3552] Rescorla, E. and B. Korver, “Guidelines for Writing RFC Text on Security Considerations,” BCP 72, RFC 3552, July 2003 (TXT).

[RFC5226] Narten, T. and H. Alvestrand, “Guidelines for Writing an IANA Considerations Section in RFCs,” BCP 26, RFC 5226, May 2008 (TXT).

TOC

Author's Address

Marc Mosko
PARC, Inc.
Palo Alto, California 94304
USA
Phone: +01 650-812-4405
Email: marc.mosko@parc.com