ccnd Content Store data structure

Junxiao Shi, 2013-07-03

This document attempts to find out how Content Store is organized and accessed in ccnd. The investigation is based on CCNx 0.7.2.

# Structure

ContentStore is stored in h->content\_tab hashtable, and indexed in h->content\_by\_accession, h->sparse\_straggler\_tab, and h->skiplinks.

## Accession number

Each unique ContentObject is assigned an accession number (ccn\_accession\_t). Numbers are assigned in order, and the last assigned number is recorded in h->accession.

## Content hashtable

h->content\_tab stores all Content entries.

Key type is portion of the ContentObject message in ccnb format up to end of SignedInfo, but excludes Content itself.

Mapped type is struct content\_entry.

### Content entry

struct content\_entry represents an entry in ContentStore.

A Content entry contains

* the accession number (accession)
* face from which the ContentObject is first arrived (arrival\_faceid)
* ContentObject (key, size), different from wire format in that the Name has an explicit digest component
* index of Name components (comps, ncomps)
* whether stale (CCN\_CONTENT\_ENTRY\_STALE flag)

## Content index by accession

h->content\_by\_accession and h->sparse\_straggler\_tab indexes Content entries by accession number.

h->content\_by\_accession is an array of struct content\_entry\* pointers. In this array, the range of accession is [h->accession\_base, h->accession\_base + h->content\_by\_accession\_window). h->content\_by\_accession[i], if not null, has accession number h->accession\_base + i.

h->sparse\_straggler\_tab is a hashtable that is responsible for Content entries with accession number [0, h->accession\_base). Key type is ccn\_accession\_t. Mapped type is struct sparse\_straggler\_entry which contains a struct content\_entry\* pointer.

## Content index by Name

A skip list indexes Content entries by Name in increasing order. The maximum depth of this skip list is CCN\_SKIPLIST\_MAX\_DEPTH (=30).

The head of this skip list is h->skiplinks. Each Content entry's skiplinks member points to next entries. A walk through skiplinks always visits Names in increasing order.

## Unsolicited Content set

Accession numbers of unsolicited ContentObjects are recorded in h->unsol, and CCN\_CONTENT\_ENTRY\_SLOWSEND flag is set.

# Operations

## Add new Content

Part of process\_incoming\_content adds new Content to ContentStore. There are 3 steps:

1. construct the key for h->content\_tab, and initialize struct content\_entry
2. assign accession number, and append to h->content\_by\_accession (enroll\_content)
	1. if h->content\_by\_accession array is full (new accession number is out of its range):
		1. if the array is too sparse (less than 1/8 of its elements are being used), earliest Content entries are moved to h->sparse\_stragglers\_tab until the remainder of this array is no longer sparse (cleanout\_stragglers)
		2. if there're empty elements in the front, the range is shifted (cleanout\_empties)
	2. if the array is still full, h->content\_by\_accession\_window is enlarged
3. insert to appropriate position of the skip list (content\_skiplist\_insert)

## Mark Content as stale

Whenever a Content entry is added or refreshed, set\_content\_timer schedules to mark the Content as stale. The timer is based on <FreshnessSeconds> tag, but the maximum allowed value is h->tts\_limit which comes from CCND\_MAX\_TIME\_TO\_STALE environment variable. To mark a Content as stale (mark\_stale):

1. set CCN\_CONTENT\_ENTRY\_STALE flag, and update h->n\_stale count
2. update h->min\_stale and h->max\_stale so that this Content's accession number is within [h->min\_stale,h->max\_stale]

## Refresh stale Content

When a ContentObject that is identical to a stale Content is received, the Content entry is refreshed: unset CCN\_CONTENT\_ENTRY\_STALE flag, and update h->n\_stale count.

## Maintain capacity

ContentStore has a weak capacity limit of h->capacity entries. If this capacity is exceeded, clean\_daemon is scheduled as h->clean event.

Unsolicited Contents are removed first. Stale Contents are removed next. If ContentStore is still over capacity, some oldest Contents are marked as stale even if FreshnessSeconds is not exceeded.

At most 500 Content entries can be removed at a time. The event would be rescheduled 5 seconds later if 500 Content entries have been removed. The process completes when ContentStore goes under capacity.

## Find Content(s) matching Interest

1. Interest is matched against ContentStore only if <AnswerOriginKind> permits CCN\_AOK\_CS
2. locate the first candidate with the Interest Name (find\_first\_match\_candidate)
	1. if the Interest has <Exclude><Any/><Component>..., the first component is appended to the Name used for locating
3. if <ChildSelector> prefers rightmost child, advance to the sibling Name (next\_child\_at\_level) until the Interest Name is no longer a prefix of Content Name; return the last matching Content, if a match was made
4. if <ChildSelector> prefers leftmost child, advance to the next Content (in Name order) until a match is made or the Interest Name is no longer a prefix of Content Name; return the first matching Content, if a match was made