Corrections to "A Neighborhood Aware Caching and Interest Dissemination Scheme for Content Centric Networks"

Amitangshu Pal and Krishna Kant, *Life Fellow, IEEE*

IN [1], Fig. 7 was incorrectly labeled and cited in the text. The figure is provided here in its complete form with full caption, along with its discussion in the text as found in of Section IV-A, "Cache Engine".

"The main challenge in enabling the neighborhood aware caching is the advertisement of the cached content-chunks, while keeping the overhead small. To address this issue, we propose a two-level caching scheme, as shown in Fig. 7(a). We assume that the entire cache/Content store (CS) of a node is divided into two levels, the upper level is the long-term cache (LTC) where the most useful content-chunks are cached. The rest is used to reserve the less useful chunks, and is known as short-term cache (STC). The STC cache is updated at each arrival of a chunk, to check whether the chunk is going to be cached or not. Occasionally the existing cache is reshuffled, where more useful chunks are transferred to the LTC and others are placed in the STC. This reshuffling can be done either periodically or when the STC is changed significantly. After such an update, the information regarding the LTC chunks is broadcast up to a certain number of hops, which is defined as *broadcast range* $\mathscr{B}$. As the CCN content names are much complex and longer than IP addresses, we use Bloom filter (BF) to encode the presence of a content in a router's LTC.

A Bloom filter is a hash-coding method used to represent a large set and at the same time supports membership queries on the set. The key difference between Bloom filters and traditional hash based representations of a set of elements is that the space required for Bloom filters is considerably reduced at the cost of permitting a small fraction of errors. Each content (key) is hashed using k different hash functions and the resulting "hash positions" are updated to 1. When there is a membership query for a key (or content), if all k hash positions of the key are set to 1, then a positive membership query is returned. While the false negative probability is zero, the false positive probability is a tunable parameter, which depends on the size of the filter. BF offers an efficient way to represent the set of cached chunks and takes $O(1)$ time to check whether a given chunk is within the set. A typical example of Bloom filter is shown in Fig. 7(b) where two contents a_1 and a_2 are inserted in a bloom filter by using three hash functions (h_1, h_2 and h_3) by setting the corresponding bit positions to 1. An illustration of a false positive scenario is also shown in this figure where the presence of content b is wrongly inferred as the three hash functions map b to the bit positions that are set to represent the presence of a_1 and a_2. In typical bloom filters elements can be added to the bloom filter, but cannot be removed."

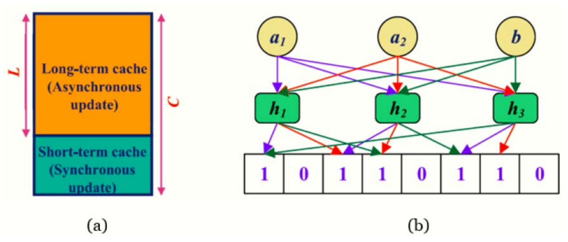
L
Long-term cache (Asynchronous update)
Short-term cache (Synchronous update)
C
(a)
a1
a2
b
h1
h2
h3
1 0 1 1 0 1 1 0
(b)

Fig. 7. (a) Two level caching and (b) a typical Bloom Filter.

REFERENCES

[1] A. Pal and K. Kant, "A neighborhood aware caching and interest dissemination scheme for content centric networks," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 3900–3917, Sep. 2021, doi: 10.1109/TNSM.2021.3079326.

Manuscript received September 22, 2021; accepted September 22, 2021. Date of current version December 9, 2021. The associate editor coordinating the review of this article and approving it for publication was G. Leduc. *(Corresponding author: Amitangshu Pal.)*

The authors are with the Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA (e-mail: amitangshu.pal@temple.edu; kkant@temple.edu).

Digital Object Identifier 10.1109/TNSM.2021.3115147