Compression of Pending Interest Table with Bloom Filter in Content Centric Network

Zhaogeng Li*, Jun Bi†, Sen Wang*, Xiaoke Jiang*
{*Dept. of Computer Science, †Network Research Center}, Tsinghua University, Beijing, 100084, P.R.C
*{li-zg07, wangsen09, jiangxk10}@mails.tsinghua.edu.cn, †junbi@tsinghua.edu.cn

ABSTRACT
We propose compressed PIT structure based on Bloom Filter. And we argue that United Bloom Filter is better than Counting Bloom Filter in the PIT compression.

1. INTRODUCTION
Content Centric Network (CCN) is an emerging information centric network architecture. There are a lot of studies focusing on caching, applications and so on to make CCN getting perfect, while the scalability of Pending Interest Table (PIT), which keeps track of Interests forwarded upstream, is still one of the primary concerns. As the number of entries would be extremely large in some routers (only one 10Gbps line card may contribute 2.5×10^6 entries at most), it is really a heavy burden for CCN routers to complete fast forwarding since the PIT has to be stored in large and slow memories. As a result, the compression of PIT is necessary. We decide to use bloom filter to compress PIT with increasing little computation overhead, at the cost of few errors.

2. COMPRESSED PIT
Bloom Filter (BF) is a hash-based structure to test whether an element is a member of a set. To compress PIT with BF, we should transpose the logic structure of PIT. Before the transposition, the key of PIT is name and the value is face set. After that, the key is face and the value is name set. We associate one bloom filter with each face which represents the name set in the transposed PIT entries. And PIT is composed of all these BFs. The basic form of bloom filter may not suitable for this compression because element deletion is not supported. But for simplicity, we use BF here to represent all the possible extensions of BF.

For each Interest packet received, the router will check if its name matches in any of the BF. If yes, the Interest packet won't be forwarded. For each Data packet received from a face, the router will do the query in all the other BFs. The packet will be forwarded to the faces with BFs containing the name.

The main drawback of BF is false positive. As a result, the compressed PIT has to suffer from packet drop (for Interest packet) and redundant traffic (for Data packet). We call these side-effect *errors*. And the more storage we save with the compression, the more errors will be introduced. Retransmission can cope with the former *error* but it does hurt network performance. So we have to limit the *errors* in a reasonable scope.Intuitively, Counting Bloom Filter (CBF) should be employed as basic bloom filter cannot support the deletion. But for the reason of incorrect deletion, CBF's performance is not satisfactory. So we should try to find another extension of BF to replace CBF in this compression structure.

CFI'12, September 11–12, 2012, Seoul, Korea.

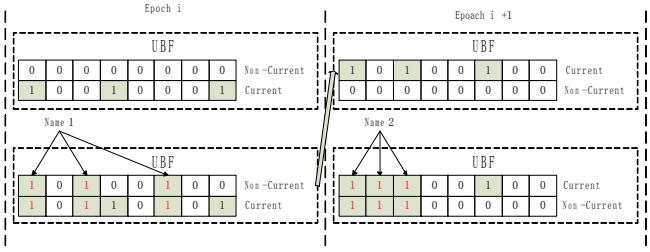

Figure.1. United Bloom Filter

3. UNITED BLOOM FILTER
We propose United Bloom Filter (UBF) to eliminate the need of deletion to avoid unexpected anomalies introduced by incorrect deletion in CBF. We note that all names in PIT will exist only for a short time. UBF takes advantages of this feature.

A UBF is composed of two bloom filters (bit array is enough since no deletion is required, see Figure.1). We divide time into different epochs, too. The two bloom filters take turns to be the *current*. At the beginning of each epoch, the *non-current* bloom filter will be cleared (all bits are set to 0). During this epoch, the insertion will be operated on both bloom filters while the query will be operated only on the *current* one. After the end of this epoch, the two bloom filters exchange their identities. The *current* bloom filter in UBF represents the name set of all Interest packets received in the last epoch and the current epoch. Every name in one bloom filter will be completely cleared at the end of the next epoch after it is inserted.

Timeout acts as implicit deletion in UBF. The most important advantage of UBF is that the density can be controlled since there is no incorrect deletion. But UBF also has a disadvantage: Interest packets with the same name cannot be forwarded twice or more times in one *epoch couple* (two consecutive epochs, the current one and the last one). Fortunately, CCN cache will alleviate this drawback. If a router receives two or more Interest for the same Data in one *epoch couple*, the Data is likely to be stored in cache as it is so popular.

There are two reasons that UBF performs better than CBF. First, UBF is composed of two bit arrays while CBF is composed of a counter array and a timeout bit array. That means UBF could reduce more storage with the same name set size. Second, UBF avoids explicit deletion. Therefore, there is no false negative in UBF and long timeout epoch won't destroy the compression structure (due to the length of this article, we won't explain why long timeout epoch destroy CBF here). For the reason above, we believe UBF is better than CBF in our PIT compression structure. Our simple simulation experiment proves this conclusion.