Designing Cuckoo Based Pending Interest Table for CCN Networks

https://doi.org/10.3991/ijim.v15i07.21149

Mohammad Alhisnawi(✉), Aladdin Abdulhassan
University of Babylon, Babylon, Iraq
mohammad.alhisnawi@uobabylon.edu.iq

Abstract—Content Centric Networking (CCN) is a modern architecture that got wide attention in the current researches as a substitutional for the current IP-based architecture. Many studies have been investigated on this novel architecture but only little of them focused on Pending Interest Table (PIT) which is very important component in every CCN router. PIT has fundamental role in packet processing in both upstream process (Interest packets) and downstream process (Data packets). PIT must be fast enough in order to not become an obstruction in the packet processing and also it must be big enough to save a lot of incoming information. In this paper, we suggest a new PIT design and implementation named CF-PIT for CCN router. Our PIT design depends on modifying and utilizing an approximate data structure called Cuckoo filter (CF). Cuckoo filter has ideal characteristics like: high insertion/query/deletion performance, acceptable storage demands and false positive probability which make it with our modification convenient for PIT implementation. The experimental results showed that our CF-PIT design has high performance in different side of views which make it very suitable to be implemented on CCN routers.

Keywords—Content centric networking, Interest packet, Data packet, Pending interest table, Cuckoo filter

1 Introduction

Content Centric Networks (CCN) is a new paradigm that aims to dominate the restrictions that are exist in the present network paradigm [1]. The concept of communication in CCN is requester-driven [2,3]. A requester (i.e., consumer) transmits out an Interest packet, which holds a CCN name that identify, uniquely, the required data [4].

The role of PIT in any CCN router is to remember these CCN names and from which port the Interest packet comes in. When this Interest packet reach the content provider, the latter will respond by sending back the requested data in a Data packet [5]. Whenever Data packet reaches any CCN router, the latter will search its PIT for the CCN name in order to retrieve its associated port number (which already requested this data) and send this Data packet back through this port [6]. Later, CCN router will remove this CCN name from its PIT because it has been satisfied [7]. So, when

Interest packets reach the CCN router interests are recorded into PIT and removed from it when Data packets arrive. Because of the high packet arrival rate, PIT should has high insert/query/delete processing rate and it should be large enough to hold the arrived information [8].

This paper proposes a novel design and implementation for PIT table in Content Centric Networking (CCN) by modifying a previously suggested data structure: Cuckoo filter (CF) [9]. CF has several preferences compared with the previous ones, Bloom and Quotient filters. Among these preferences: higher insertion/query/deletion throughput, lower storage requirements, and lower false positive probability [10]. The proposed design for PIT by employing our modification for Cuckoo filter (we called CF-PIT) will result in quick search, and low memory demand which consider the essential significant characteristics for suggesting an effective PIT. Our suggested CF-PIT is autonomous from the composition of the CCN naming strategy. Moreover, it does not need a dedicated hardware to be implemented. The fundamental contributions of this study are:

- Suggesting a modified version of Cuckoo filter for PIT implementation (CF-PIT). CF-PIT has the following characteristics:
  - The number of slots in every bucket is set to be equal to the number of ports in CCN router. Thus, it will be easy to specify the port number for every incoming Interest packet.
  - Every slot holds a mini bucket rather than a single position in the standard Cuckoo filter. The purpose of this mini bucket is to hold more than one CCN name that come from every port.
- Utilizing our proposed modification for Cuckoo filter (CF-PIT) to design and implement pending interest table for CCN router.

The paper organization is as follows. Section 2 explains the former related studies. Section 3 depicts content centric networking. Section 4 presents a general description for Cuckoo filter. Section 5 explains the suggested PIT design. Section 6 gives an evaluation to the proposed PIT. In Section 7, we conclude our work.

2 Related Works

NPHT [11] is the early attempt for designing PIT for CCN routers. In this work, the authors suggested to share the hash table logic between FIB and PIT. They employed two types of data structures: Forwarding Information Entries (FIEs) which are used to store metadata for pending interest information and Propagation Entries (PEs) which are used to store metadata for forwarding information. NPHT suffers from several weak points: it utilizes extra storage because the element representation is very high, it cannot satisfy scalability issue, extra delay time, and finally it cannot deal in a good manner with Interest flooding which make it vulnerable to flooding attack. Later, ENPT [12] has been proposed which was aimed to increase the performance by enhancing insert/lookup/delete operations. This method depends on utiliz-

ing Name Component Encoding (NCE) technique to encode the CCN names in order to minimize PIT size. The main limitations of this technique are: the need to dedicated encoding algorithm, the need to utilize more sophisticated architecture to ensure PIT size minimization, extra utilization of memory space, and difficult to collect statistics that are necessary to overcome some kinds of attacks. DiPIT [13] has been suggested later to improve the achievement of PIT. This technique depends on utilizing several counting Bloom filters (CBFs) one filter for every port inside CCN router. Moreover, it employs a central/shared CBF that holds the information of all CCN names that arrive from all ports. The central filter has been utilized in order to minimize the false positive ratio of BFs and to ensure information consistency. DiPIT has several drawbacks: it utilizes five hash functions and, so, will result in doing extra calculations and extra memory accesses, the high false positive probability of Bloom filters, the utilization of counting Bloom filters requires extra memory usage to support delete operation, and also the additional overhead that result from the central/shared Bloom filter to ensure information consistency. The authors of [14] focus on compressing PIT by utilizing one of the variations of Bloom filter which called United Bloom Filter (UBF) to avoid the necessity of elimination to obviate unforeseen exceptions resulted by improper elimination in CBF. The main weaknesses of this method are the inability to handle, correctly, the Interest packet that holds the same CCN names, and the growing delay of CCN name routing. In [15] a novel PIT design technique has been suggested in which fixed-length fingerprints have been employed instead of complete CCN names. It aims to perform packet routing with minimum storage usage. The main obstacles of this technique are inability to overcome PIT overflow, and it cannot cope with the increasing link speed specially in the lookup process. Finally, [16] proposed a new variant of BF named Mapping Bloom Filter (MBF). They employed this new variant to build PIT for CCN router called MaPIT that has the ability to employs the existing faster memory chips. MBF consists of two modules: Index Table (IT) and Packet Store (PS). The main disadvantages of this technique are the need for proceeding algorithm which consumes more memory usage, there is no way to overcome false negative and PIT overflow when they are take place.

3 CCN background

In contrast to the current networks, the basic variation is that CCN forwards requested contents depending on their names instead of the IP addresses of their hosts [11]. These names are unique and hierarchically structured and have similar structures to the URLs. Interest and Data packets are the fundamental two kinds of packets that are employed in CCN [17,18]. Each of these packets holds a CCN name of the requested/retrieved data. The end host (i.e., the consumer) generates an Interest packet that is forwarded depending on the name of the requested data and, as a response to it, a Data packets will be forwarded back from the original container or from any router on the way between them. There exist three essential data structures in every CCN router: Pending Interest Table (PIT), Forwarding Information Base (FIB), and Content Store (CS) [19]. PIT is used as a reference to the ports that requested the data. FIB is

used to get the appropriate port to transmit the Interest packet. CS represents a cashe memory that holds, temporarily, some the retrieved data [20]. Whenever an Interest packet access the router, the latter will examine its content store first and if it found the desired data, it will generate and transmits back, immediately, a Data packet with the desired data. Otherwise, it will register the incoming input port inside its pending interest table without transmitting the Interest packet if the desired data is already registered and it will register the desired name of the data along with its incoming port and will transmit the Interest packet, if not. When a Data packet access the router, it will transmit back this packet to every port that already registered in its pending interest table if there is such registered ports or it will drop it if not. Also, the router may cashe a copy of the retrieved data to be used in satisfying the latter requests [21,22].

4 Proposed Pending Interest Table Design

Here, we will present our proposed PIT design for CCN routers in which we focus on making it fast, cost effective, high performance, and space efficient to meet CCN node requirements. Doing lookup operation in O(1) time regardless of PIT size can be considered as the main characteristic that all former PIT design attempts try to satisfy. Our fundamental design for PIT relies on employing a modified version of Cuckoo filter. CF [23] has been employed in many networking applications like: DPI [24], IP search operations [25], packet processing [22] and so on. Every position in the standard Cuckoo filter holds a fingerprint of the inserted element whereas in our modified version of Cuckoo filter for PIT design (we called CF-PIT) we made some modifications. The number of slots in every bucket of CF-PIT is like the number of ports in the CCN router to reflect the incoming port number for every inserted CCN name. Also, rather than storing only one fingerprint in every slot in CF, we will replace this single position by a mini bucket in order to hold more than one fingerprint of the CCN names that arrive from every port. Note that in CCN paradigm the terms interface and port are utilized interchangeably. Figure 1 illustrates the general structure of the proposed CF-PIT.

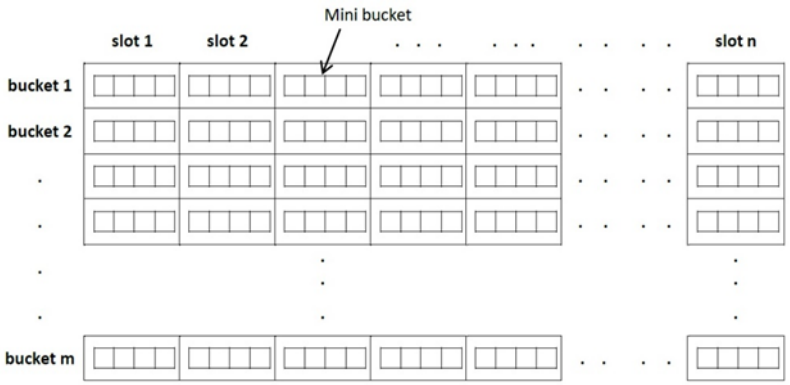

Fig. 1. General structure of CF-PIT

Figure 2 depicts an example for processing an incoming Interest packet with our proposed CF-PIT that holds 7 buckets and 4 slots (i.e.,4 ports). For every incoming CCN name, incoming Interest packet, the two candidate buckets (b1 and b2) will be calculated and then the specified slot (depending on the incoming port number) in b1 will be accessed. If the mini bucket inside this slot has a vacant entry, then the fingerprint for this CCN name will be inserted. Otherwise, the same slot number in bucket b2 will be accessed and also if its mini bucket has a vacant entry the fingerprint will be inserted and if not the relocation process will take place to find an empty place to insert this fingerprint. In our example in Fig. 3 the incoming port is (2) and the fingerprint (F) has been stored in b1 because it has a vacant entry.

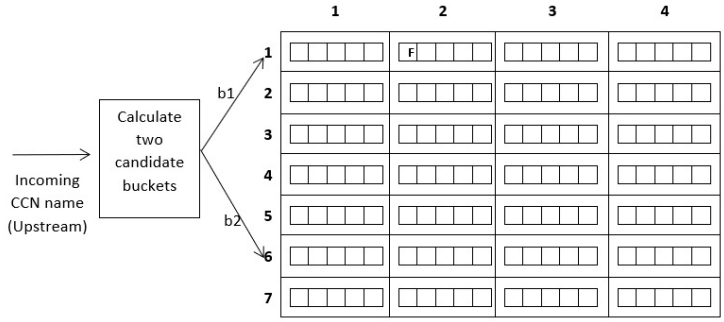

Fig. 2. Handling an incoming Interest packet

Figure 3 depicts an example for processing an incoming Data packet with our proposed CF-PIT with the same previous specifications. For every incoming CCN name, incoming Data packet, the two candidate buckets (b1 and b2) will be calculated and then all slots in b1 and b2 will be accessed looking for the fingerprint of this incoming CCN name. If there is a match with any slot, then the incoming Data packet will be transmitted back to the specified port depending on the number of slots that hold the fingerprint. In our example in figure 3 the fingerprint has been found in b1 in slot 1 and slot 4 and, thus, the Data packet will be sent back through both port 1 and port 4. Then, the fingerprint will be deleted from these two slots.

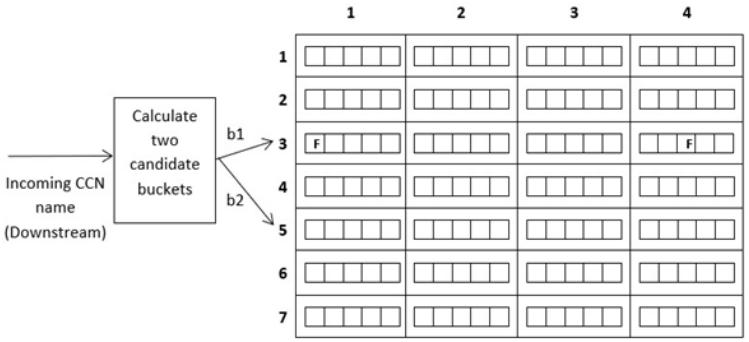

Fig. 3. Handling an incoming Data packet.

Here, in our discussion, we will concentrate on the main job of PIT in the CCN router and neglect the other operations that occur in both (CS) and (FIB). Remember that PIT deals with two kinds of streams, upstream (Interest packets) and downstream (Data packets). When an incoming Interest packet arrives at any port of CCN node, the CCN name will be matched against PIT and here there will be three scenarios. First, if this CCN name is already inserted into the PIT from the same interface, which means that the same interface demanded the same data twice, then the incoming Interest packet will be dropped. Second, if this CCN name is not already inserted in the

PIT then this name will be inserted into PIT with its incoming port number. Second, if this CCN name is already inserted into PIT (but with different interface) then only the incoming port number will be registered inside PIT. Algorithm 1 illustrates the main steps for handling an incoming Interest packet based on our PIT design (CF-PIT).

Algorithm 1: Handling of Interest Packet

```
1 Input: incoming CCN name (N), incoming port (p);
2 fin = getfingerprint(N);
3 b1 = gethash(N);
4 b2 = b1 ⊕ gethash(fin);
5 if (either bucket (b1 or b2) has an entry in its mini
bucket in slot
  number (p) that already holds fin) then
6   Drop Interest packet;
7 Exit;
8 if (either bucket (b1 or b2) has an empty entry in
its mini bucket
  in slots number (p)) then
9   insert fin to this mini bucket;
10  exit;
11 else
12  // Else, perform a relocation;
13  b = select randomly between (b1 and b2);
14  for i ← 0 to maximum number of attempts do
15    Select an entry (e) randomly inside the mini buck-
et from
  bucket (b) slot (p);
16    swap fin and the fingerprint saved in entry e;
17    b = b ⊕gethash(fin);
18    if bucket(b) has an empty entry in its mini bucket
in slot
  number (p) then
19      insert fin to this mini bucket;
20  //Otherwise PIT is full;
```

Algorithm 1 explained how to deal with an Interest packet that reach to the CCN router from port number (p). First, the fingerprint and the indices to the two indicated buckets of CCN name is calculated. Next, if either of these two indicated buckets have an entry that holds fin in the same incoming port (i.e., p) then the incoming Interest packet will be dropped because this port is already requested this content. Latter, if either of these two indicated buckets have an empty entry in their mini buckets of slot number (p) then the fingerprint of CCN name will be inserted. Otherwise, just like the standard Cuckoo filter does, a relocation must be performed to make an empty entry to insert this fingerprint.

Whenever a Data packet comes from any port, the CCN name is checked against CF-PIT to specify all ports that already requested this Data packet. Thus, satisfying these Interests by sending back the Data packet through these ports. The main steps for dealing with an incoming Data packet are depicted in Algorithm 2.

Algorithm 2: Handling of Data Packet

```
1 Input: incoming CCN name (N);
2 fin = getfingerprint(N);
3 b1 = gethash(N);
4 b2 = b1 ⊕ gethash(fin);
5 if (both buckets (b1 and b2) have no entry in their
mini buckets in all
  slots that holds fin) then
6 Drop Data packet;
7 Exit;
8 if (both buckets (b1 and b2) have an entry in their
mini buckets in all
  slots that holds fin) then
9   Specify port numbers depending on slot numbers;
10  Send Data packet back through all these ports;
11  Delete fin from these entries;
12  Exit;
```

Algorithm 2 presented the main steps that must be followed when a Data packet access the CCN router. First, the fingerprint and the indices to the two candidate buckets of CCN name is calculated. Next, if both buckets have no entry that holds the fingerprint of the incoming CCN name then the incoming Data packet will be dropped. The latter situation occurs when the fingerprint has been removed from PIT because it was stay in the PIT for a period that exceed the predefined threshold. Latter, if the previous situation does not occur, the fingerprint is examined against the mini buckets in all the slots in these two candidate buckets in order to specify the number of ports that already requested that Data packet. Then the incoming Data packet will be sent back through all these ports and, finally, the fingerprint of CCN name will be deleted from all these slots (because it has been satisfied).

5 Results and Discussions

Here, we will debate our evaluation for the proposed CF-PIT aiming to evince the competence of our suggested technique. We utilized ndnSIM [26], an open-source package, which performs CCN protocol stack for NS-3 simulator. In our evaluation, a standard dataset has been utilized. It has been gotten from an open-source store website [27] which holds datasets of CCN [28,29] names in different sizes and formats which are gathered from URLs or by tracing web pages. Four previously proposed PIT design techniques: DiPIT [13], CBFPIT [14], MaPIT [16], and NCE [30] have been utilized in our evaluation.

5.1 Memory usage

Figure 4 depicts the memory usage for all techniques in terms of the arriving CCN names. NCE utilizes trie structure which causes CCN names in the PIT strongly systematic and simple to administrate, despite that, it has no ability to participate much to decreasing PIT storage or noticeably increasing PIT access achievement. DiPIT utilizes several counting Bloom filters (CBF) in a distributed manner (one filter for every port) along with a main counting Bloom filter that is shared among all ports. Also, MaPIT consumes larger storage than CF-PIT because of its extra utilization of counting BF. CF-PIT consumes smaller amount of memory than the previous three techniques but little larger than of CBFPIT because it needs to store the fingerprints of CCN names in only one data structure. This little storage requirements of CBFPIT comes with extra degradation for the performance.

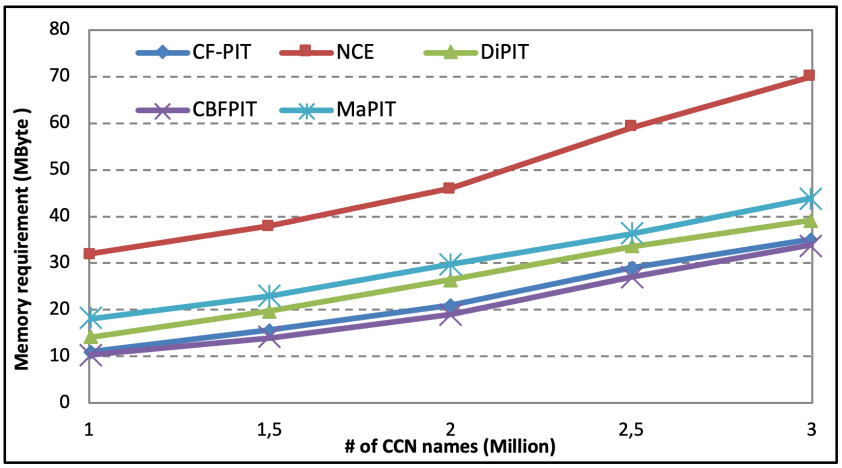

Fig. 4. Memory requirement for CF-PIT, NCE, DiPIT, CBFPIT, and MaPIT.

5.2 False positive probability

Figure 5 illustrates the false positive probability for all techniques with different number of CCN names. It can be noticed that the false positive average of CF-PIT is considerably less than the false positive average of both DiPIT and NCE. DiPIT utilizes Bloom filter as its main data structure and it has high false positive rate. NCE tries to specify code allocation for CCN names which results in false positive in determining some of these CCN names. Because of the compression operation that has been utilizes in CBFPIT, the false positive rate of it will be higher than all the other techniques.

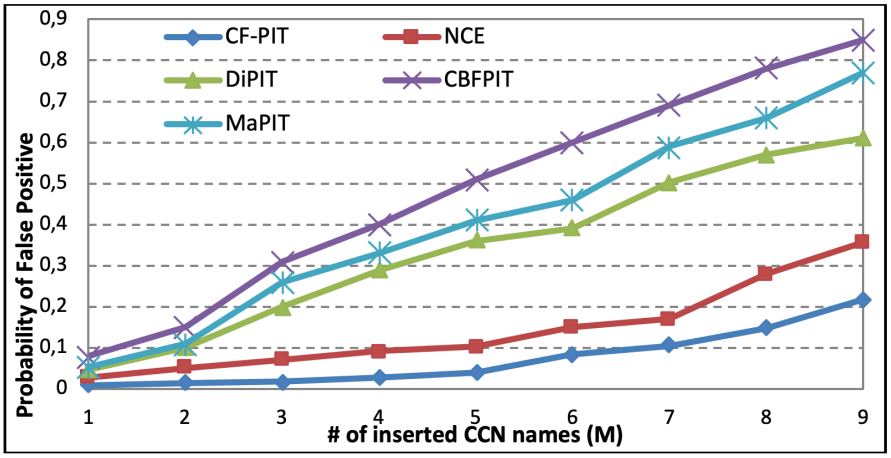

Fig. 5. False positive probability for CF-PIT, NCE, DiPIT, CBFPIT, and MaPIT.

5.3 Memory accesses

DiPIT, CBFPIT, and MaPIT techniques utilize several CBFs in a distributed manner and it employs five hash functions which results in five memory accesses to accomplish element insertion/query and deletion. Moreover, along with these CBFs, it utilizes a main CBF (shared among all ports) which also employs the same number of hashes which make it expends additional memory accesses. NCE is built on utilizing trie structure which requires traversing this trie beginning from root node down to each part of CCN name. CF-PIT can perform CCN name insertion/query/deletion and regain its port number in O(1) time and, thus, it consumes lower memory accesses comparing with the other two mechanisms.

5.4 Insertion performance

Figure 6 depicts the insertion performance for all techniques. DiPIT, CBFPIT, and MaPIT employ counting Bloom filter with five hash functions for every insertion which consume extra calculation. Also, it needs to insert the CCN name in both the counting Bloom filter on the incoming port and the central/shared counting BF. NCE

utilizes trie structure and, so, every insertion needs extra work to access the right node starting from the root node. CF-PIT has the ability to perform CCN name insertion faster than the other two techniques because of its simple structure.

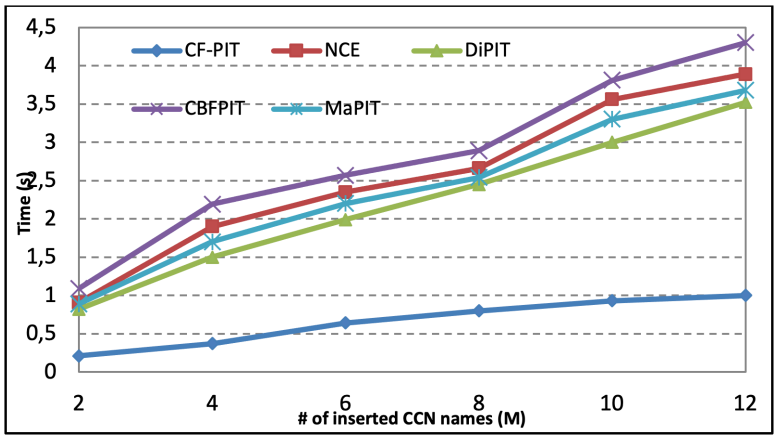

Fig. 6. Insertion performance for CF-PIT, NCE, DiPIT, CBFPIT, and MaPIT.

5.5 Lookup performance

The lookup performance for all techniques can be found in figure 7 CF-PIT has the ability to access the required CCN name directly with simple calculation. DiPIT and MaPIT needs to do extra calculation to check the existence of CCN name because it needs to examine five positions (with one hash function to access every position). The compression operation that has been utilizes in CBFPIT will add extra burden on the insertion achievement of this technique. NCE requires examining the trie structure starting from rote node moving down to reach the required leaf node.

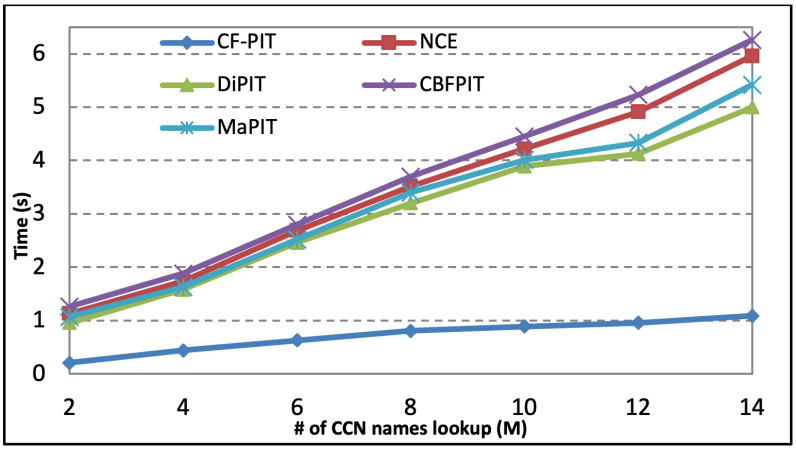

Fig. 7. Lookup throughput for CF-PIT, NCE, DiPIT, CBFPIT, and MaPIT

5.6 Delete performance

Figure 8 illustrates the delete performance for all techniques. Deleting CCN name from NCE involves visiting the trie nodes starting from root node to access the required leaf node. Then the delete operation is accomplished by backtracking the way to check if the parent node still requires to be eliminated, and so on. DiPIT requires subtracting one from every position from the five counting Bloom filter positions. Moreover, it also needs to make this update on the central/shared counting Bloom filter. CBFPIT and MaPIT need to do extra calculations to access the CCN to be deleted. Whereas, CF-PIT has the ability to access and delete the required CCN name directly with simple calculations.

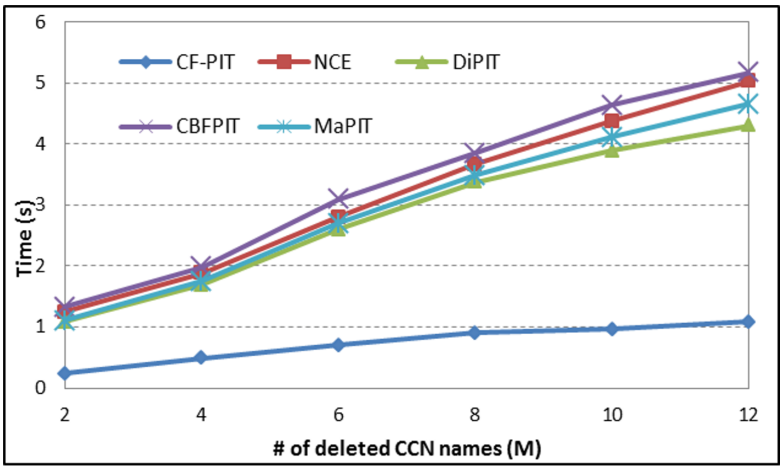

Fig. 8. Delete performance for CF-PIT, NCE, DiPIT, CBFPIT, and MaPIT.

5.7 PIT security

The proposed CF-PIT has been designed to confront the worst-case flow balance situation in order to dominate an Interest flooding attack. CF-PIT provides a straightdirect way for collecting the necessary statistics on the routers to detect any offensive and then enforce appropriate procedures. Moreover, CF-PIT has the ability to deal with the Interest flooding issue by its architecture that stores fingerprints only which make it sufficiently huge to store the entire flooded packets. Nevertheless, the storage demand may still be reasonable because just a fingerprint with a fixed-length is saved for every CCN name.

6 Conclusion

CCN is a new promising networking architecture, which has the ability to increase the process of content delivery. PIT is one of the main parts of the CCN routers that plays crucial rule in the content delivery process. Fast and scalable PIT design is a

challenge in CCN. In this paper, we proposed PIT design for CCN router that ensures fast packet transforming and substantially minimizes the storage usage by saving fingerprints instead of CCN names. Our design depends on utilizing our modified version of CF as a fundamental structure to save the fingerprint for the CCN names (called CF-PIT). CF-PIT architecture has reasonable memory specification and high insertion/query/deletion performance with minimum false positive probability. Our promising results discovered that the proposed PIT can meet the major demands of designing an effective PIT.

7 References

[1] Ahmed, M. Z., Abdalla Hashim, A. H., O. Khalifa, O., H. Alkali, A., Bt Midi, N. S., & Abd. Rahman, F. B. (2019). Evaluating Mobility Management Models for Content Forwarding in Named Data Networking Environments. International Journal of Interactive Mobile Technologies (iJIM), 13(04), 47. https://doi.org/10.3991%2Fijim.v13i04.10519.

[2] Dash, S., Sahu, B.J.R., Saxena, N., and Roy, A. (2017). Flooding Control in Named Data Networking. IETE Technical Review. 35(3). 266–274. https://doi.org/10.1080%2F02564602.2017.1281173.

[3] Kumar, S., and Tiwari, R.,(2020). Optimized content centric networking for future internet: Dynamic popularity window-based caching scheme. Computer Networks, vol. 179, p. 107434. https://doi.org/10.1016%2Fj.comnet.2020.107434.

[4] Kumar, S. and Tiwari, R., (2020). An efficient content placement scheme based on normalized node degree in content centric networking. Cluster Computing. https://doi.org/10.1007%2Fs10586-020-03185-0.

[5] Chi, K., Du, X., Yin, G., Wu, J., Guizani, M., Han, Q., and Yang, Y. (2020). Efficient and fair Wi-Fi and LTE-U coexistence via communications over content centric networking. Future Generation Computer Systems, vol. 112, pp. 297–306. https://doi.org/10.1016%2Fj.future.2020.05.026.

[6] Qiao, X., Wang, H., Ren, P., Tu, Y., Nan, G., Chen, J., and Blake, M. B. (2020). Interest packets scheduling and size-based flow control mechanism for content-centric networking web servers. Future Generation Computer Systems, vol. 107, pp. 564–577. https://doi.org/10.1016%2Fj.future.2020.02.004.

[7] Pakle, G., Manthalkar, R. (2019) Optimal Forwarding in Named Data Networking. In: Panda G., Satapathy S., Biswal B., Bansal R. (eds) Microelectronics, Electromagnetics and Telecommunications. Lecture Notes in Electrical Engineering, vol 521. Springer, Singapore. https://doi.org/10.1007/978-981-13-1906-8_4.

[8] Shigeyasu, T., and Sonoda, A. (2020). Detection and mitigation of collusive interest flooding attack on content centric networking. International Journal of Grid and Utility Computing, vol. 11, no. 1, p. 21. https://doi.org/10.1504%2Fijguc.2020.10025640.

[9] Fan, B., Andersen, D.G., Kaminsky, M., and Mitzenmacher, M.D. (2014). Cuckoo filter: Practically better than bloom. In Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies. 75–88. https://dl.acm.org/doi/10.1145/2674005.2674994.

[10] Alhisnawi, M., and Ahmadi, M. (2018). QCF for deep packet inspection. IET Networks. 7(5). 346–352. https://doi.org/10.1049%2Fiet-net.2017.0037.

[11] Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H. and Braynard, R.L. (2009) Networking named content. In ACM CoNEXT. Rome, Italy. 1-12. https://dl.acm.org/doi/10.1145/2063176.2063204.

[12] Dai, H., Liu, B., Chen, Y., and Wang, Y. (2012). On pending interest table in named data networking. In Proceedings of the eighth ACM/IEEE symposium on Architectures for networking and communications systems. 211-222. https://doi.org/10.1145%2F2396556.2396600.

[13] You, W., Mathieu, B., Truong, P., Peltier, J., and Simon, G. (2012). Dipit: A distributed bloom filter based pit table for ccn nodes. In Computer Communications and Networks (IC- CCN), 21st International Conference IEEE. 1-7. https://doi.org/10.1109%2Ficccn.2012.6289282.

[14] Li, Z., Bi, J., Wang, S., and Jiang, X. (2012). Compression of pending interest table with bloom filter in content centric network. In Proceedings of the 7th International Conference on Future Internet Technologies. ACM. 46-46. https://doi.org/10.1145%2F2377310.2377326.

[15] Yuan, H. and Crowley, P. (2014) Scalable pending interest table design: From principles to practice. In INFOCOM, 2014 Proceedings IEEE, 2049-2057. https://doi.org/10.1109/infocom.2014.6848146

[16] Li, Z., Liu, K., Zhao, Y., and Ma, Y. (2014). MaPIT: an enhanced pending interest table for NDN with mapping bloom filter. IEEE Communications Letters. 18(11). 1915-1918. https://doi.org/10.1109%2Finfocom.2014.6848146.

[17] Liu, T., Zhang, M., Zhu, J., Zheng, R., Liu, R., and Wu, Q. (2018). ACCP: adaptive congestion control protocol in named data networking based on deep learning. Neural Computing and Applications. 1-9. https://doi.org/10.1007%2Fs00521-018-3408-2.

[18] Byun, H., and Lim, H. (2019). A New Bloom Filter Architecture for FIB Lookup in Named Data Networking. Applied Sciences. 9(2). 329. https://doi.org/10.3390%2Fapp9020329.

[19] Wang, L.J., Lv, Y.Q., Moiseenko, I., and Wang, D.S. (2018). A Dataflow-Oriented Programming Interface for Named Data Networking. Journal of Computer Science and Technology. 33(1). 158-168. https://doi.org/10.1007%2Fs11390-018-1812-9.

[20] Shubbar, R., and Ahmadi, M. (2018). Efficient name matching based on a fast two-dimensional filter in named data networking. International Journal of Parallel, Emergent and Distributed Systems. 34(2). 203-221. https://doi.org/10.1080%2F17445760.2017.1363202.

[21] Alhisnawi, M., and Ahmadi, M. (2018). Detecting and Mitigating DDoS Attack in Named Data Networking. Journal of Network and Systems Management. vol. 28, no. 4, pp. 1343–1365. https://doi.org/10.1007%2Fs10922-020-09539-8.

[22] Pagh, R., and Rodler, F. (2001). Cuckoo hashing. In European Symposium on Algorithms, Springer Berlin Heidelberg. 121-133. https://doi.org/10.1007/springerreference_57616.

[23] Alhisnawi, M., and Ahmadi, M. (2017). Deep packet inspection using cuckoo filter. In Annual Conference on New Trends in Information and Communications Technology Applications (NTICT). 197-202. https://doi.org/10.1109%2Fntict.2017.7976111.

[24] Kwon, M., Reviriego, P., Pontarelli, S. (2016). A length-aware cuckoo filter for faster IP lookup. In IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). 1071-1072. https://doi.org/10.1109%2Finfcomw.2016.7562258.

[25] Abdulhassan, A.A., and Ahmadi, M. (2019). Cuckoo filter-based many-field packet classification using X-tree. The Journal of Supercomputing. 1-21. https://doi.org/10.1007%2Fs11227-019-02818-5.

[26] Afanasyev, A., Moiseenko, L., and Zhang, L. (2012). ndnSIM: NDN simulator for NS-3. University of California, Los Angeles, Technical Report .

[27] re3data.org: The Content Name Collection; editing status 2020-02-07; re3data.org - Registry of Research Data Repositories. http://doi.org/10.17616/R3M33J last accessed: 26-06-2019.

[28] Fethellah, N. E. H., Bouziane, H., & Chouarfia, A. (2019). New Efficient Caching Strategy based on Clustering in Named Data Networking. International Journal of Interactive Mobile Technologies (iJIM), 13(12), 104. https://doi.org/10.3991%2Fijim.v13i12.11403.

[29] Hussaini, M., Awang Nor, S., & Ahmad, A. (2019). Optimal Broadcast Strategy-Based Producer Mobility Support Scheme for Named Data Networking. International Journal of Interactive Mobile Technologies (iJIM), 13(04), 4. https://doi.org/10.3991%2Fijim.v13i04.10513.

[30] Wang, Y., He, K., Dai, H., Meng, W., Jiang, J., Liu, B., and Chen, Y. (2012). Scalable name lookup in NDN using effective name component encoding. In Distributed Computing Systems (ICDCS), IEEE 32nd International Conference. 688-697. https://doi.org/10.1109%2Ficdcs.2012.35.

8 Authors

Mohammad Alhisnawi received the B.S. degree in Computer science from University of Babylon, Babylon, Iraq in 2004. He received the M.Sc. degrees in Computer science from University of Babylon, Babylon, Iraq in 2010. He received his PhD from Razi university-Iran in 2018. He works as a lecturer in Faculty of Information Technology, Department of Information Networks, University of Babylon, Babylon, Iraq. His research interests include computer security, network processing, and content centric networks.

Aladdin Abdulhassan received the B.S. degree in Computer science from University of Babylon, Babylon, Iraq in 2005. He received the M.Sc. degrees in Computer science from University of Babylon, Babylon, Iraq in 2010. He received his PhD from Razi university-Iran in 2019. He works as a lecturer in Faculty of Information Technology, Department of Information Networks, University of Babylon, Babylon, Iraq. His research interests include software defined networking, computer networks security, computer networks management, computer networks switching, cryptography, data mining, wireless communications.

Article submitted 2021-01-14. Resubmitted 2021-02-24. Final acceptance 2021-02-24. Final version published as submitted by the authors.