JOURNAL OF MECHANICS OF CONTINUA AND MATHEMATICAL SCIENCES
www.journalimcms.org

ISSN (Online) : 2454 -7190 Vol.-15, No.-7, July (2020) pp 271-281 ISSN (Print) 0973-8975

CUCKOO FILTER-BASED NAME LOOKUP IN NAME DATA NETWORKING

Ritika Kumari[1], R.L Ujjwal[2], Vishwa Pratap Singh[3]

[1,2,3]USIC&T, Guru Gobind Singh Indraprastha University, New Delhi, India

[1]ritikakumari.ggsipu@gmail.com, [2]ujjwal@ipu.ac.in,
[3]vishwapratap.phd@gmail.com

Corresponding Author: Ritika Kumari

https://doi.org/10.26782/jmcms.2020.07.00022

Abstract

Name Data Networking is a future Internet architecture and it depends on data. NDN takes advantage of the current Internet Architecture and aims to address the weaknesses. In NDN, interest messages are used to retrieve data. Each data has a name that is embedded inside each interest packet. Routers use these names to forward the messages as NDN does not use source or destination address. For each interest packet, a packet is issued that is called a Data packet or D-packet. D-pkt holds the name of the content and the data itself. In this paper, we propose a data structure which is the hybrid of Cuckoo filter and Trie for the name lookup process in NDN.

Keywords : NDN model, Routing and Forwarding in Name Data Networking, NDN forwarding Overview, Bloom Filter-Based Name Lookup, Cuckoo Filter based Name Lookup

I. Introduction

In NDN, the consumer is the one who starts the communication. NDN Interest Packet or I-packet and Data Packet or D-packet uses as shown in Fig.1. A consumer initiates by sending an interest packet, each interest packet holds the prefix/name of the desired data. A router by looking up the name from its Forwarding Information Base, or FIB, forwards the interest packet accordingly. A router also remembers the interface from which the request comes in. When the interest packet reaches the node that can provide the content, a corresponding data packet is issued. The data packet contains both the content and the content's name. The content is signed by using the producer's key [IV]. The D-packet traces back the path created by the I-packet. As NDN is a data-centric network, both the I-Packet and D-packet does not contain the host address. Routers in NDN keep the I-packet and D-packet for some timeframe. When a router receives more than one I-packet for the same content, the router follows a first come first serve policy. The I-packet that came first will be directed towards the data producers. The router keeps the record of remaining I-packets in the Pending Interest Table or PIT. Each entry of PIT contains the name of

the Interest and the set of interfaces in which the requests have been received. When the desired data arrives at the router, it checks the PIT entries and forwards the content to all the listed interfaces [IV]. Then the router removes the corresponding entry and also caches the data in the Content Store or CS. In this way, routers can satisfy future requests.

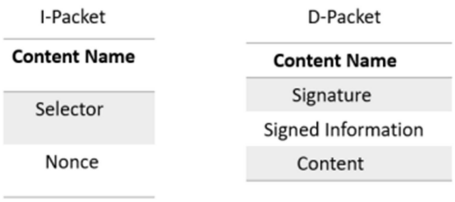


Fig. 1: Interest and Data packets

II. NDN Model

NDN has the hourglass-shaped model as of the Internet shown in Fig. 2 in which instead of using the IP addresses, it uses content names. The current Internet model has only Internet Protocol working in the network layer for communication [I]. Adding new functionalities and modifying the underlying existing technologies is very difficult.

NDN has two more important layers as it must support four functionalities:

- Efficiency
- Security
- Resiliency
- Scalability

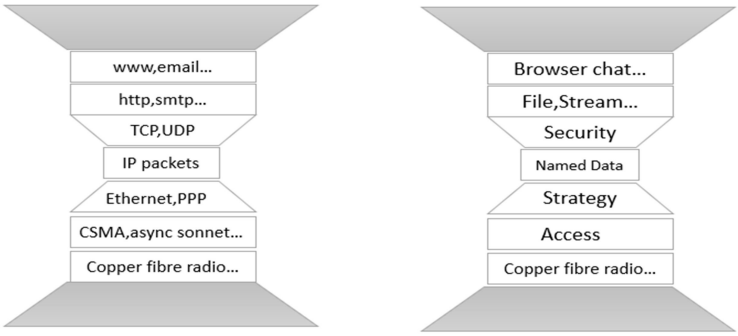


Fig. 2: Internet and NDN hourglass architecture

The Strategy Layer is responsible for the forwarding decisions and the security layer is responsible for securing each piece of data in the NDN network.

Names

In NDN, the most important thing is the name associated with each data. Each interest packet has the prefix/name that the consumer is looking for. The data packet also holds the name of the data for which this D-packet has been issued. The names used are opaque and the routers are unaware of the meaning of the name.

NDN uses the hierarchical structure for the names as shown in Fig. 3, e.g., an image produced by ggsipu may have the name as /ggsipu/images/umbrella.jpg. It enables each application to use the naming scheme as per their needs. A certain level of global uniqueness is required when retrieving the data globally.

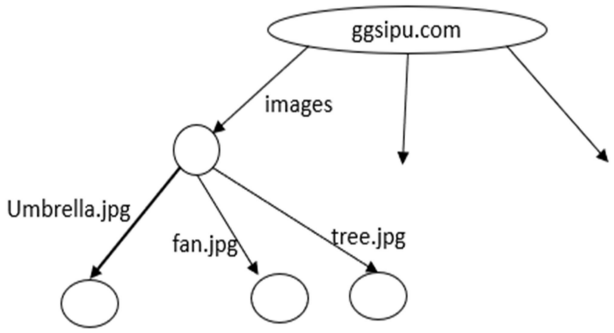


Fig. 3: Hierarchical representation of names in NDN

III. NDN Architecture Principles

The following principles give direction towards the design of the NDN architecture:

- NDN has the same hourglass-shaped architecture that mainly revolves around the network layer implementing the required functionality required for connectivity at the global level.
- NDN also follows the peer to peer principle which enables the applications to recover from the network failures.
- NDN has built security into the architecture as each data packet is being signed by the data producer to verify its authenticity.
- In NDN, we have separate routing and forwarding plane.

IV. Routing and Forwarding in Name Data Networking

Routers look into the route information, select the path for the forwarding of the I-packet and the D-packet based on the name that is embedded. This eliminates various problems that we faced in IP architecture. As NDN is a host-centric architecture, the producer does not need to expose its identity so there is no NAT traversal problem. There is a problem of changing addresses in IP when there is mobility. In NDN, as the name of the content remains the same so there is no

disturbance in communication [IV]. And at last, there is no need to assign the addresses and further, there is no need for address management.

Routers have the knowledge of the boundaries between the components present in a data's name. As the content names are totally opaque in NDN, the routers simply do the longest prefix matching. For example, /ggsipu/images/umbrella.jpg may match both /ggsipu and /ggsipu/images in the FIB and the latter is the longest prefix match.

NDN supports multipath routing. The interests cannot loop in the network as the content name and a random nonce value helps to identify the redundant so that they can be discarded. A router, without worrying about the loops can send out multiple I-packet on multiple interfaces. The first D-packet received can satisfy the interest and router caches the received data for future interests. The router discards all the other copies of D-packet [III].

This enables the NDN architecture to support load balancing. The routers can also measure the performance on the basis of returned data and can further choose the best performing path for future interests. In this way, NDN supports the forwarding strategy.

As each data packet is being signed by the producer, it provides great security to the data from being tampered or modified. By supporting multipath routing, NDN mitigates the prefix hijacking. In the case of prefix hijacking, the routers may try other paths to get the data. To be more effective, NDN must focus on denial of service attacks.

Name Lookup

Each node in NDN maintains 3 data structures:

(a) Forwarding Plane (FIB)

(b) Pending Interest Table (PIT)

(c) Content Store (CS).

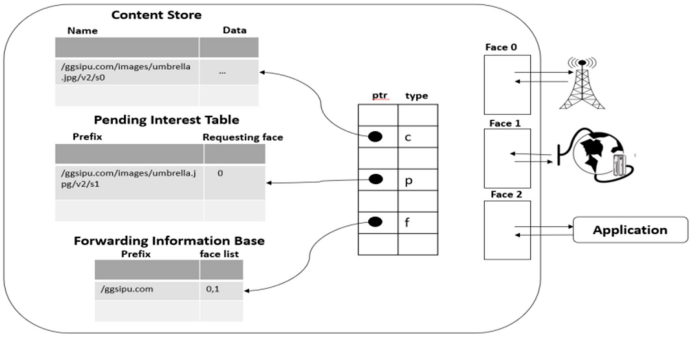


Fig. 4: Data Structures at NDN node

Forwarding Information Base: The purpose of FIB is to serve the forwarding purpose. A router can check the FIB entries to make a decision of forwarding a particular packet to a specific interface as per the current set up information.

Pending Interest Table: There is an entry for each interest packet in PIT sent by the consumer until the request is fulfilled. After the request has been satisfied, the entry will be removed from the PIT. The entry is maintained for a specific period of time. If the request is not fulfilled within the timeframe, I-packet has to be sent out again.

Content Store: As the D-packet is issued, the router keeps the cache of the data so that the router can fulfill the requests for the same data in the future. In this way, a router can speed-up the process of data retrieval.

NDN forwarding Overview

In NDN, as said earlier also, the communication process is started by the consumer as shown in fig. 5. The I-packet is released for data retrieval. On receiving the I-packet, the router checks the name in the content store to see if the router has a copy of the data. In this case, the router simply returns the corresponding D-packet [XII]. If there is no match, the router checks the PIT for an entry. If the router finds out that there is no entry for the name in its content store, it checks the PIT [XVI]. If there is an entry then the router adds an entry with the incoming interface as the Interest packet has already been forwarded but the D-packet has yet not issued. If there is no entry in the PIT for the name, it adds a new entry. The router uses the FIB information for the longest prefix match and for the forwarding purpose. If there is an entry in FIB, the I-packet is forwarded to the outgoing face. If no entry exists in FIB, the router may send a negative acknowledgment back to the incoming interface from which the interest packet has been received.

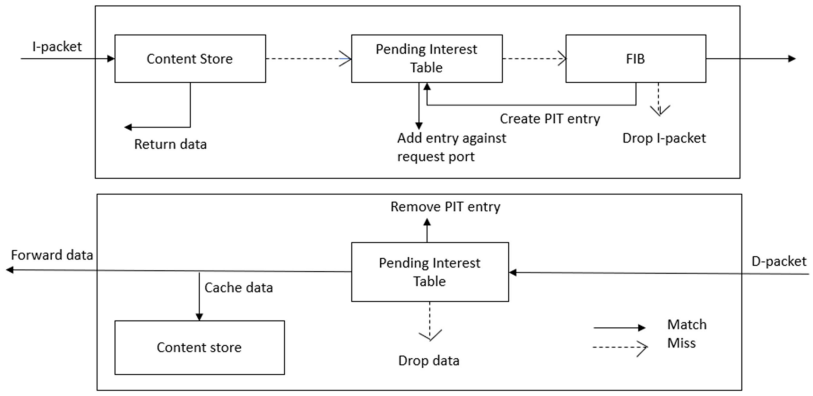


Fig. 5: Processing of packets at NDN node

On receiving a D-packet, the router looks for the PIT entry. If there is a match the data is cached to the content store and it will be forwarded to the corresponding interface [XVII]. If there is no entry in PIT, the data is dropped as it is either no

longer needed or is unrequested. NDN forwarding strategy decides whether and how to forward the interest packet in the network.

V. Bloom Filter-Based Name Lookup

In Name Data Networking, we have hierarchical names to fetch the content. The decision to forward a packet depends on the lookup operations [V].

Bloom Filters can be useful due to three factors:

- They give the solution in constant time.
- They take less amount of space.
- There is less probability of error.

As there is a possibility to have false positivity, bloom filters are not really optimal.

Problem with Bloom Filters

Let's say, we have a set N= $\{name_1, name_{2,\ldots,} name_n\}$

Bloom Filters can be helpful in the lookup process of a name in the given set P.

False Positives in Bloom Filter

The speed and size are very important as to concern with the name lookup process in Name Data Networking. This leads to the probability of error i.e. false positives. False Positives occurs when we report that the name is present in the hash table and it's actually not [II].

Let's say, we have an m bit array filled with zeros.

B

0	0	0	0	0	0	0	0	0	0

Hash each item $name_j$ in N *k* times. If $H_i(name_j)$=a, set B[a] =1.

B

0	1	0	0	1	0	0	1	0	0

Now, for the lookup of a name say *q* in N, check B at $H_i(q)$. All *k* values must be 1.

B

0	1	0	**0**	1	0	0	**1**	0	0

But with bloom filters, it is possible to have false positive i.e. all *k* values are 1 but the name says *s* is not in N.

B

0	1	0	0	**1**	0	0	**1**	0	0

Where items are n, m=cn bits and we have *k* has functions.

Pr (as pacific bit of filter is 0) is $p' \equiv (1-1/m)^{kn} \approx e^{-kn/m} \equiv p.(1)$

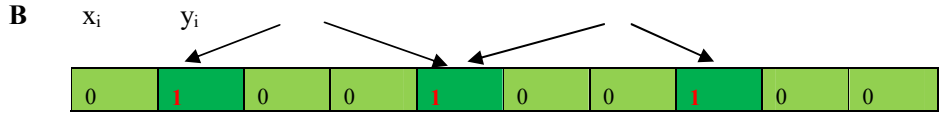
If p is the fraction of 0 bits in the filter then the probability of false-positive is

$(1-p)^k \approx (1-p')^k \approx (1-p)^k = (1-e^{-k/c})^k.(2)$

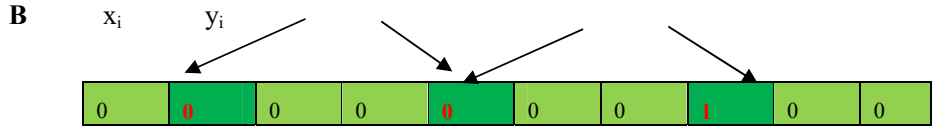
Find optimal at $k = (\ln 2)\, m/n$ by calculus.

Deletions in Bloom Filter

Not only there is a probability of false-positive in bloom filters but also, they cannot handle deletions. In Name data networking, when cache poisoning happens, there is a need to update the content and so that leads to deletions in the hash tables.



As shown in Fig. if we delete x_i means there is a need to replace the entry with 0. In this process, when we replace the entries of x_I with 0, it will affect the entry of y_i.



We can see clearly, deleting a name from the hash table in bloom filters interferes with the entries of the other names. If we search for y_i in the hash table, we will find that not all *k* values are 1. So as a result, it will report that the name is not present but actually, it does. It delays the forwarding decisions.

VI. Cuckoo Filter-Based Name Lookup

Cuckoo Filters are better than the bloom filters due to the following reasons:

- It supports deletions.
- It takes less time for lookup than Bloom filters.
- It requires less space than Bloom filters.
- It is simple to design and build.

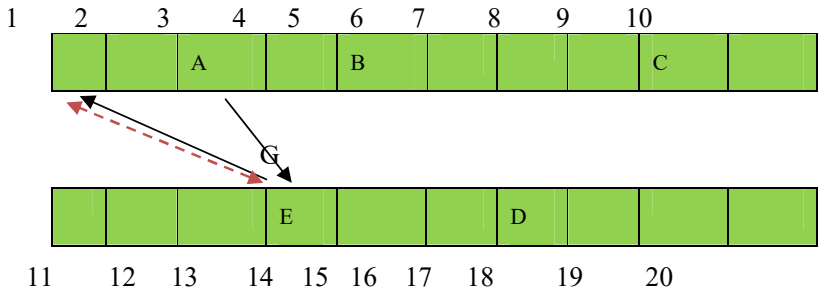
A cuckoo filter stores a bit string for each item generated by the hash function. It provides the efficient use of space as the filter is almost completely filled. If we need to find whether an item is present or not then we just need to find the fingerprint of the item and returns true if an identical fingerprint is found [X].

Cuckoo Hash Tables

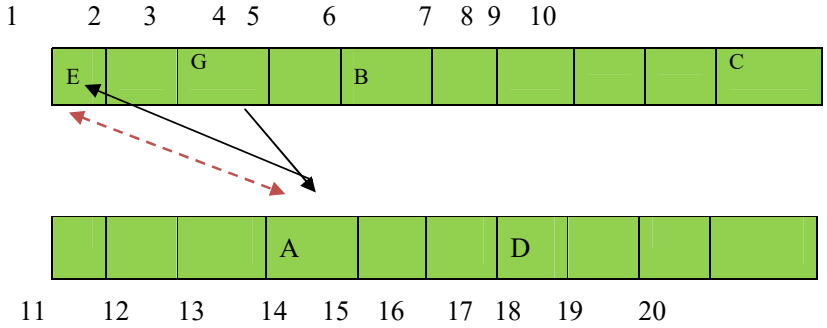
In cuckoo Hashing, each element x_i gets two possible locations.

- Let's say, we want to insert a name say n, we need to check both the locations for n. If there is a slot, insert.

- If both the positions are filled, n removes the previous item m. Then m moves to its other location.
- If m finds that the other location is already full, m removes p and this keeps on happening until the item finds an empty slot. As shown in Fig. item G needs to be inserted. Two possible locations are 3rd and17th.



At 3rd position, item A is present and at 17th position D is present.



G takes the 3rd position and removes A. A has two possible positions i.e. 3rd and 14th. So, A moves to 14th position and removes E. E moves to 1st position.

Therefore, Cuckoo hashing is useful in the name lookup process in Name Data Networking as it can give the solution in constant time, has high memory utilization and it is simple to build and design[IX].

Deletions in Cuckoo Filter

Cuckoo filters can handle deletions [VI]. When an item *p* is to be deleted from the hash table, then the cuckoo filter checks both candidate buckets for the item *p*. As we see that when two items share the positions in a bloom filter, it leads to false deletion. And it eventually leads to a false negative. But in case of Cuckoo filter, if two items p and q share one candidate bucket i_1 and collide on fingerprint f, then partial key cuckoo hashing ensures that p and q both also reside in bucket i_2. So, when the item *p* is deleted, q is still a member of the set[VII]. For an item to gets deleted, it must be present otherwise it may be possible that a different item that shares the same fingerprint is deleted.

VII. Proposed Solution for Name Lookup in NDN

We propose a data structure which is a hybrid of Cuckoo filter and Trie. Each name is divided into two sections: C-prefix and T-suffix. Cuckoo filter stores C-prefixes C_1, C_2…with a certain length. The Trie stores the T-Suffixes, T_1, T_2 … without any boundation on its length. We are constructing a set in which we have T-suffixes which are bind to the B-Prefixes with a hash table as shown in Fig. 6

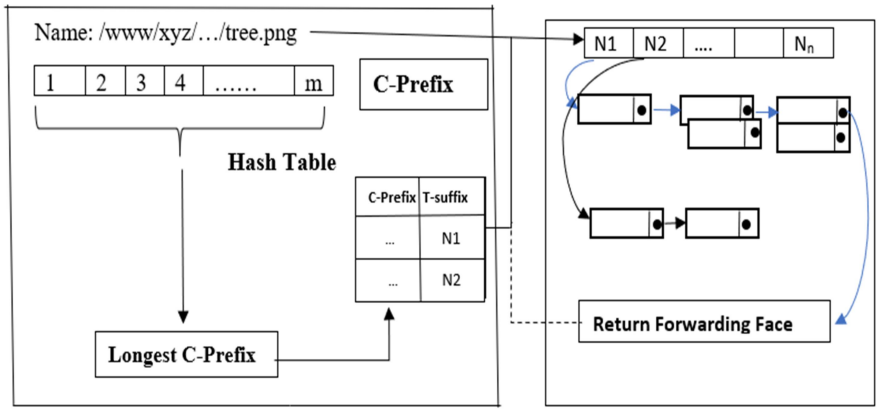


Fig. 6: Cuckoo Filter Based Name Lookup

This method is done in two stages:

- Cuckoo filter processes the C-prefix and using hash table we can get the location of the corresponding T-suffix tree. In case the length of the C-prefix is small then the outgoing face(s) will be returned directly.
- After getting the information of the root, the method will start the longest prefix matching. Then it will return the corresponding forwarding face(s).
- The hybrid data structure that we propose can improve the lookup operations as the Cuckoo filter take less time and space-efficient as compared to others.

Comparison between Bloom Filter and Cuckoo Filter

Parameter	Bloom Filter	Cuckoo filter
Space Efficiency	Less Space Efficient	More Space Efficient
Number of Memory Accesses	*k* bits with k hash functions	Fixed number of buckets
Value association	Do not provide such functionality	They can return an associated value for each matched item.
Maximum Capacity	Keep inserting new items.	They have a load threshold.
Limited Duplicates	It causes counter overflow.	It limits the insertion when the same item is inserted kb+1 times.

Fig. 7: Bloom Filter vs Cuckoo Filter

VIII. Conclusion

We propose a data structure which is a hybrid of Cuckoo filter and Trie. The hybrid data structure that we propose can improve the lookup operations as the Cuckoo filter take less time and space-efficient as compared to others. It supports deletions. It requires less space than Bloom filters. It is simple to design and build.

References

I. Amadeo M, Campolo C, Molinaro A. Forwarding strategies in named data wireless ad hoc networks: Design and evaluation. Journal of Network and Computer Applications. 2015 Apr 1; 50: 148- 58.

II. Bacanin N. An object-oriented software implementation of a novel cuckoo search algorithm. In Proc. of the 5th European Conference on European Computing Conference (ECC'11) 2011 Apr 28 (pp. 245-250).

III. Di Benedetto S, Papadopoulos C, Massey D. Routing policies in named data networking. In Proceedings of the ACM SIGCOMM workshop on Information-centric networking 2011 Aug 19 (pp. 38-43).

IV. Ding W, Yan Z, Deng RH. A survey on future Internet security architectures. IEEE Access. 2016 Jul 29; 4: 4374- 93.

V. Fan B, Andersen DG, Kaminsky M, Mitzenmacher MD. Cuckoo filter: Practically better than bloom. In Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies 2014 Dec 2 (pp. 75-88).

VI. Massawe EA, Du S, Zhu H. A scalable and privacy-preserving named data networking architecture based on Bloom filters. In2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops 2013 Jul 8 (pp. 22-26). IEEE.

VII. Mun JH, Lim H. Cache sharing using bloom filters in named data networking. Journal of Network and Computer Applications. 2017 Jul 15; 90: 74- 82.

VIII. Najafimehr M, Ahmadi M. SLCF: Single-hash lookup cuckoo filter. Journal of High Speed Networks. 2019(Preprint):1-2.

IX. Pan J, Paul S, Jain R. A survey of the research on future internet architectures. IEEE Communications Magazine. 2011 Jun 30; 49 (7): 26- 36.

X. Quan W, Xu C, Guan J, Zhang H, Grieco LA. Scalable name lookup with adaptive prefix bloom filter for named data networking. IEEE Communications Letters. 2013 Dec 6; 18(1):102-5.

XI. Saxena, D., Raychoudhury, V., Suri, N., Becker, C. and Cao, J., 2016. Named data networking: a survey. Computer Science Review, 19, pp.15-55.

XII. Wang L, Hoque AK, Yi C, Alyyan A, Zhang B. OSPFN: An OSPF based routing protocol for named data networking. Technical Report NDN-0003; 2012 Jul 25.

XIII. Wang L, Lehman V, Hoque AM, Zhang B, Yu Y, Zhang L. A secure link state routing protocol for NDN. IEEE Access. 2018 Jan 4; 6: 10470- 82.

XIV. Yi C, Afanasyev A, Moiseenko I, Wang L, Zhang B, Zhang L. A case for stateful forwarding plane. Computer Communications. 2013 Apr 1; 36 (7): 779- 91.

XV. Yi C, Afanasyev A, Wang L, Zhang B, Zhang L. Adaptive forwarding in named data networking. ACM SIGCOMM computer communication review. 2012 Jun 26; 42 (3): 62- 7.

XVI. Yi C, Abraham J, Afanasyev A, Wang L, Zhang B, Zhang L. On the role of routing in named data networking. In Proceedings of the 1st ACM Conference on Information-Centric Networking 2014 Sep 24 (pp. 27-36).

XVII. Yuan H, Song T, Crowley P. Scalable NDN forwarding: Concepts, issues and principles. In 2012 21st International Conference on computer communications and networks (ICCCN) 2012 Jul 30 (pp. 1-9). IEEE.