An Analysis on Memory and Look-up for Pending Interest Table in Named Data Networking

[1]SivaChakkravarthy.G, [2]DeviKala, [3]Arunekumar.N.B, [4]SureshJoseph.K, [5]MirothaliChand
Department of Computer Science and Engineering, Pondicherry University,

ABSTRACT—In IP, the data are transferred amid the source and destination using the IP address. But in NDN instead of using the IP address, the name prefix is used to access the data directly from the source. In general, NDN comprises of three vital components which are the content store (CS), pending interest table (PIT) and forwarding information base (FIB). The content store and pending interest table would compare, analyze and match the content name prefix exactly however forwarding interest base matches the longest prefix. The PIT possesses two issues which are the high memory usage and the name lookup. This survey has covered several papers related to PIT memory and lookup. The paper has been concluded by identifying the problems and challenges in the PIT.

KEYWORDS: ndn, pit, name look-up, name prefix, ip, data packets.

I.INTRODUCTION

Named Data Networking (NDN) is completely based on content-based networking (CCN) or information-centric networking, but NDN is the only open source framework. IP uses the source address and the destination address to communicate or access a node from the other node. NDN searches the content, using the prefix name of the data. NDN is similar to the Content-Centric Networking (CCN), which was introduced and developed by Van Jacobson.

In general, IP addresses are in the form of a datagram, which has only name communication endpoints (the IP destination and source addresses). The names in an NDN are in the form of router name/organization name/data name, but it has the data name as a key for searching the related prefix.

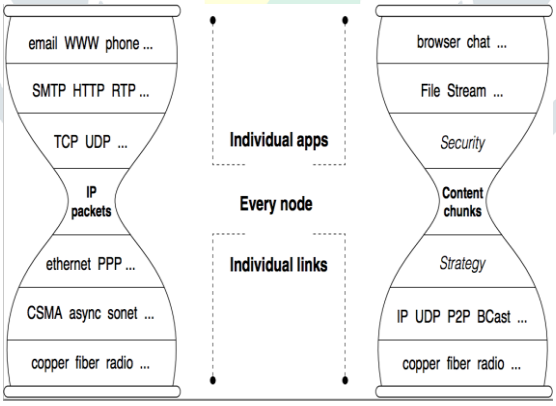

Fig 1: IP VS NDN

This simple modification in the hourglass model, allows the use of data names instead of IP addresses for data transfer. It makes data transfer as a new revolution in the Internet architecture.

II. NDN ARCHITECTURE

In NDN, the communication is provided from one node to another, by the transmission of two different types of packets they are given as:

1. Interest-packet
2. Data-packet.

The content name plays a predominant role in the transfer of data. If the requested content name which is present in the Interest packet matches with the needed data in the source, then the corresponding data packet is being sent to the requester.

Interest-packet- A requester sends the Interest-packet that contains the name in the form of content name/organization name/data name and sends it as a request to the network.

Data-packet- The data packet has a content name/signature/signed info/data. If the requestor interest packet reaches the router, it searches the related data packet using name prefix and returns the data if present. If the interest reaches the data producer, it checks the authentication of the data receiver and then it transfers the data packet, to the related data consumer.

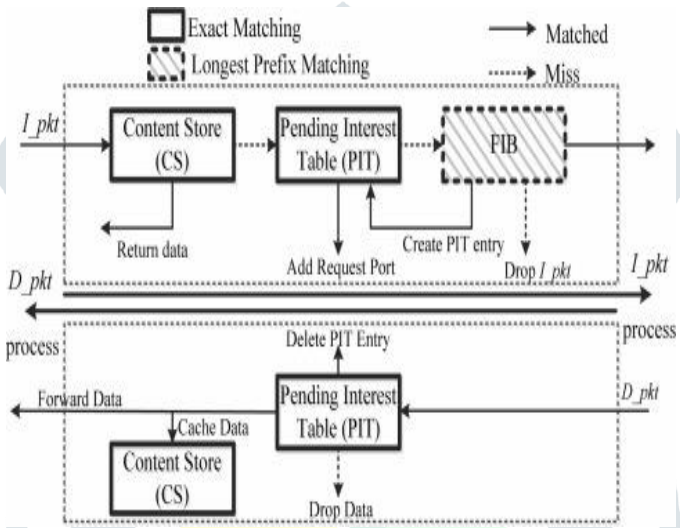

Fig 2: NDN architecture

Content store (CS): In the NDN router, CS is the network storage for the data packets arrived in the router. It caches the data packet to satisfy the future interest that arrived in the CS.

Pending interest table (PIT): It maintains each and every entry that is forwarded by the CS and also stores all the Interests that satisfies the related data packet. PIT is the core part of the NDN, which acts as a base for both incoming and outgoing interface.

Forwarding information base (FIB): In this component, the routing process takes place. When compared to CS and PIT, it does not match prefixes exactly but it matches with the longest prefix found. It also maintains the next hops and other information of content.

III. SERVICES IN NDN

The various services in the NDN are given as:

1. Routing
2. Caching
3. Forwarding
4. Security and privacy
5. Mobility

A. Routing

Routing is the method of finding the related path in the network traffic. The main advantage in NDN routing is that, it detects automatically the fault in network and recovers on its own. NDN routing follows the global routing process, with a minimal modification. The main function of routing in the NDN architecture is to traverse the interest packet to the data producer. There are several modalities such as the greedy and dijkstra's for choosing a path to convey the packets. Based upon the routing policy and network topology, routing is classified into:

1. Inter-domain routing
2. Intra-domain routing

In these routing protocol, inter-domain protocol makes the decision based on the feasible path, while intra-domain routing depends on the shortest path of the node in the network.

B. Caching

Content caching is also called as router storage or network storage that highly support for data delivery, in less time consumption. Through using caching in the NDN router, it reduces the overhead for data producer and maintains many duplicates data or information for further usage. Usually the performance of the cache calculated by using a few things they are:

1. Hit ratio
2. Content retrieval delay
3. Average number of hops travelled
4. Dissemination speed

C. Forwarding

The main purpose of forwarding in the NDN planes are forwarding packets from one component to another. It function like as a control plane over all the three components. In NDN forwarding is classified into two types they are:

1. Scalable forwarding
2. Stateful forwarding

On using NDN forwarding, the NDN routers calculates the throughput, packet loss during transmission, the alternative path in routing and packet time taken to match the related interest during the time of congestion and overhead.

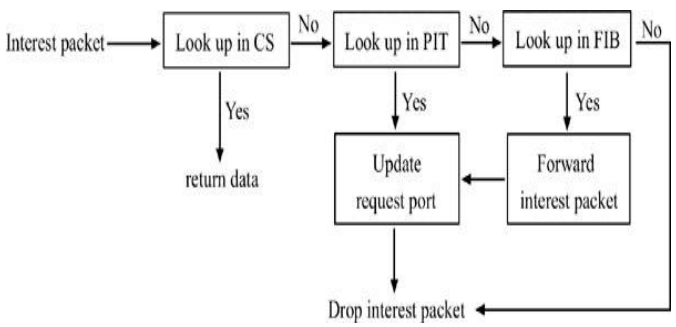

Fig 3: NDN Forwarding

D. Security and Privacy

NDN is based on the content transfer from one node to another node in the form of interest packet and data packet. But securing the content in the NDN is very essential.

Security

In NDN, each and every data packet are signed with a certain secured key to access the data. NDN has the digest signed key in both interest and data packet, to verify its own user. NDN nearly uses so many encryption and encoding technique, cryptographic digest, fingerprint and key locator, to access and verify the content in the NDN**.**

Privacy

NDN has very less router level privacy when compared to the IP. NDN stores the user request and satisfied data in the router itself, so NDN needed privacy in the router level.

E. Mobility

In general, the devices connect with other devices remotely and it may change on the every network architecture. NDN uses the content names, to access the data from producer instead of IP address, so it need not connect on the same IP, to establish the connection and enables the user a better method of accessing data without any loss or failure. The various problem in the IP, when compared to the NDN mobility are:

1. Host multi-tasking
2. Network address changing
3. Removal of connection in unwanted usage
4. Replication

IV. ISSUES ON PIT IN NDN

The issues in the pending interest table which affect the data transmission because of PIT is the core part of the NDN. The two issues that mainly reduces the PIT efficiency are:

1. High memory
2. Name lookup

The PIT follows exact matching policy. When the interest packet is not matched in the cache, it forwards the interest packet to PIT. If it has match in the previous interest, increases its token size otherwise adds a new token and forward to the forwarding information base. It fetches data from the network and reverse forwards to the PIT. Once PIT receives the data, it directly sends data to all the nodes and deletes certain tokens such that it saves cache for the future interest.

A. HIGH MEMORY

In the PIT, there are operations such as insert, delete, and update for each and every interest packets to be performed. The insert and delete operations in the PIT, depends on the track of the record. It is predictable because, the number of entries is increased along with PIT. Likewise, lifetime of the PIT is still alive without deleting the unsatisfied one. It may cause PIT overflow, otherwise PIT receives and removes the interest packet correspondingly.

B. NAME LOOKUP

This specific issue is related to the entire three components, for example, if the NDN interest packet has the nearly 1000 of components, it needs to match the entire components. This makes the PIT overhead. So this is the major issue in the PIT.

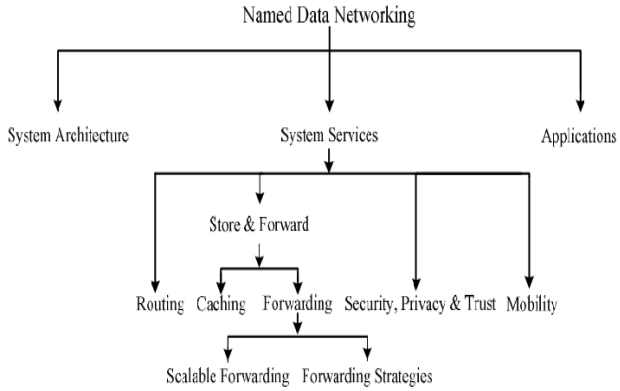

Fig 4: NDN Classification

IV. VARIOUS METHODS FOR PIT TO MAKE FAST NAME LOOKUP AND LESS MEMORY USAGE

In the NDN to design an efficient PIT, its mandatory to construct a well-defined and efficient data structure to insert, delete, and update the certain status of data. The name lookup depends on the components presented in the NDN interest packet which is stored in the PIT. The most efficient method is name-encoding, in which the name is encoded along with a related keyword for the fast lookup and less memory consumption. This avoids PIT overhead even when it matches over 1000 components of same interest packet.

A. Encoded Trie Methodology

In this method each and every component in the PIT are encoded, which makes the lengthy names into smaller names without change of information with relative encryption technique. In this methodology, radix trie technique is used for the fast lookup and memory efficient. In the radix trie, it consists of only one child node formed from the root node, where that node merges with the relative root node. The radient scheme consists of three modules, they are given as:

1. Memory decomposition,
2. Component radix trie (CRT),
3. Name prefix trie (NPT)

Further CRT consists of two modules they are given as: Available token stack and Token frequency map. When the request arrives in the PIT, its interest packet is decomposed into smaller one, by using delimiters (/) and the decomposed content inserted into CRT and character wise name-lookup takes place. In name-lookup, each and every component process until no matches founded

in it. Then it encodes the name into numeral tokens and pass this token, through NPT module. The NPT module checks, whether it has any repetition or not. If any repetition found, it increases its frequency and update the incoming and outgoing faces.

The N-PIT algorithm [2] is mainly composed of two operations, they are PIT aggregation and PIT lookup. The efficient N-PIT method, which supports the PIT aggregation and control PIT overflow. In this method, it uses Patricia trie for storing the names and it forwards the requestor interest to the certain data producer without any failure. Patricia trie minimizes the depth of the trie, increases the security and fast name-lookup. When the routing process takes place through using N-PIT algorithm, it minimizes the updates in the PIT. When comparing N-PIT algorithm with previous name consumption algorithm by using datasets such as blacklist and shallist, it reduces the unwanted updates and showed that they increases 68.18% efficiency with compared to another trie algorithm [2].

Title	Author	Ds-type	Algorithm	Gains	Issues
Scalable name lookup with adaptive prefix bloom filter for NDN	Wei quan, changqiao xu, jianfeng guan, hongke zhang, and luigi alfredo grieco(2013)	Trie and bloom filter	Adaptive prefix bloom filter mechanism	Decreased memory and time consumption	Longest prefix matching
On pending interest table in named data networking	Huichen dai, bin liu, yan chen, yi wang(2013)	Trie	Encoded name prefix trie	Memory consumption is less	Lookup
Name filter: achieving fast name lookup with low memory cost via applying two-stage bloom filters	Yi wang, tian pan, zhian mi, huichen dai, xiaoyu guo, ting zhang, bin liu(2014)	Bloom filter	Name filter	Memory consumption low and capability is high due to optimization	Collision probability
An efficient pending interest table for ndn with mapping bloom filter	Zhuo li,kaihua liu,yang zhao,yongtao ma(2014)	Bloom filter	Mapit	Minimized the chip memory	Probability of false positive is high
Radient: scalable memory efficient name lookup algorithm for ndn	Diviya saxena and vaskar raychoudry(2016)	Radix trie	Radix scheme	Memory consumption is less	Lookup time
N-fib: scalable, memory efficient name-based forwarding	Diviya saxena and vaskar raychoudry(2016)	Patricia trie	N-fib	Memory consumption is less	Aggregation
A new name prefix trie with path compression	Jungwon lee, hyesook lim(2016)	Trie	Path compression	Fast lookup	Name compression
BFAST: high speed memory efficient approach for NDN forwarding engine.	Huichen dai, member, jianyuan lu, student member, yi wang, member, tian pan, member, bin liu(2016)	Bloom filter with hash table	BFAST	Memory consumption is low	False positive rate
A fast and memory efficient approach to ndn name lookup	Dacheng he, dafang zhang, ke xu1, kun huang, yanbiao li (2017)	Bloom filter	Bbs (bloom filter assisted binary search)	Fast lookup	Overhead
Pending interest table control management in NDN	Raaid alubady, suhaidi hassan, adib habbal (2018)	Prefix trie	Pitcm	Overflow is decreased	No flexibility

Table 1: VARIOUS FORWARDING TECHNIQUES

B. Bloom Filter Based Searching Methodology

The previous research is based on searching using the hashing table, but it has a high collision. Binary searching method can reduce the hash function and it reduces the collision in the names, even in the worst case scenario. In order to make high efficient searching of names, either the backtrack searching or storing name of certain prefixes is required. This leads to name lookup or memory issues as a result. By using a binary search method along with bloom filter, it improves memory consumption and high speed of name-lookup data, every hash table is combined with the bloom filter, to perform fast searching of the name prefixes.

The other research such as the designing and evaluating a memory efficient PIT without overflow is a very difficult task in the overall NDN research area. The proposed PIT algorithm is developed by using bloom filter probabilistic data structure i.e. Mapping Bloom Filter (MBF). By using this bloom filter, this method is known as MaPIT. In these MaPIT [4], three design advantage of a PIT with the relative model MBF are,

1. Fast name lookup
2. Reduces the entries of packet
3. Low memory consumption

Through using MaPIT algorithm, when compared to the MBF algorithm, it decreases the memory cost and also it reduces the PIT overflow in the form of fast name lookup.

C. Name Component Encoding Methodology

In general, NDN names have hierarchical names and non-definite lengths of components, which are very higher than IP. But making the fast lookup and to minimize router memory depletion in NDN is difficult task, so the effective name encoding component (NCE) [7] method is framed and other technique called code allocation mechanism is used in name component encoding algorithm, which makes NCE as a memory efficient and also improves as a longest name prefix matching algorithm. To implement the high memory efficient and fast name-lookup design, it further needs to satisfy the structure of NDN. The process in the NDN components such as insert, modify and delete uses content name for both incoming and outgoing interfaces. The Code Allocation method optimizes the codes and reuses it till the desired PIT entry lifespan expires. Likewise, the other scheme such as radient and N-PIT reduces the depth of the trie in their desired path.

D. Bloom Filter Aided Hash Table Methodology

BFAST (Bloom Filter Aided hash Table) is used for managing entries in the PIT table [3]. This hash table is designed mainly for the lookup, insertion, and deletion. In BFAST algorithm, the PIT table indexing by using a bloom filter aids hash table and its functions. But the collision may occur on using the BFAST algorithm. Bloom filter is a modified probabilistic data structure that induces high speed name-lookup. The linked list along with hash table, which decreases the hash collisions. The hash table is suitable for both LPM (Longest Prefix Match) and EM (Exact Match) [3], along with longer or smaller prefixes.

It totally depends on the content names. The hash table also has the limitation of unbounded namespace and hash collision in the PIT table, i.e., the variable lengths of the bloom filter along with hash table, will results the longer searching time. These may delay in forwarding process, which takes a long time in execution or delay in the related applications.

E. Path Compressed Trie (Pc-Trie) Methodology

The path compression trie is associated with binary tree, because it reduces or compresses the empty nodes, while searching the name prefixes. This methodology is based on the compression, through each and every empty child node are compressed based on their path. According to the relative algorithm, the simulation results show that the proposed approach compresses nearly 97% of unwanted child nodes. When compared to the existing NPT algorithm, path-compressed trie increases the querying speed of a binary tree. The only null values are presented in the empty child node of path-compressed trie, in which the empty child node always merge with its own root node. The PC trie increases the performances, in the form of eliminating unused empty child nodes. When implementing the path compression trie in the NDN name-lookup will improve, when compared to the IP architecture.

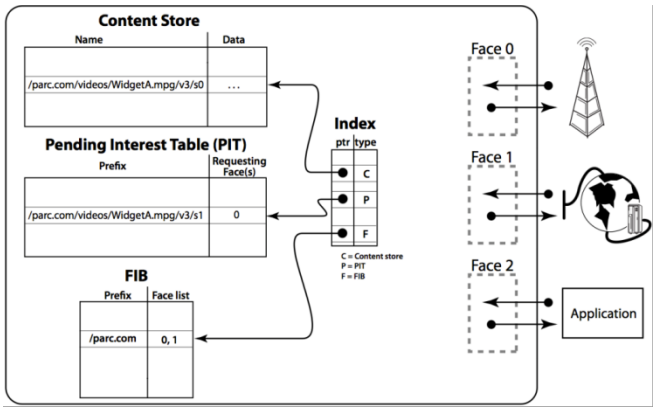

Fig 5: NDN PACKET DELIVERY

F. Adaptive Prefix Bloom Filter Methodology

In NDN, the router looks for popular name and the highly used name prefixes. In this adaptive prefix bloom filter method has two modules in name-lookup process, they are B prefix and T suffix [6], and this allows memory consumption and high speed name-lookup. In the B prefix module, the hash table is used to validate the size and location for T-suffix. If it matches the name prefix, the entry directly returns to the PIT. In the second module, it revives the information about the content name and forwards by using a trie. Similarly, the updates also follow this two-stage filter process. The matching allows much more repetitive memory, to access the router memory in the second stage. The B-prefix is always different, when comparing to its name prefix it reduces the excess entries in the PIT. The related framework like radient, N-PIT, BFAST, methods also analyzed by using encoded prefix name. Similarly, this method also analyzed by using trie, so it decreases the memory cost and it limits the frequent forwarding and updating. Through this method, it decreases the number of entries in the PIT and also, it reduces the size of the NDN names, as much as smaller. Instead of a trie, using the filter it decreases the chip based forwarding (CBF) and enhances the high-speed name-lookup. The experimental result shows that bloom filters reduce the false negative rate and it increases the unwanted insertion and name-lookup in the PIT.

V. CHALLENGES IN NDN

NDN uses the data names for data delivery, which makes a new architectural design for the network. But in this architecture, they are many challenges in the design, because NDN neglects the change of IP addresses to the data name. This may be the major issue that how an application in the NDN select its data names to makes the application development and network data delivery.

The IP has name-spaces ranges from 0 to 255, but NDN name-spaces are unbounded. Nowadays the routing takes place by using IP prefix aggregation, but NDN uses the same routing principles rather simple changes when compared to the IP.

VI. CONCLUSION

The main aim of this survey paper explains about different PIT management mechanisms, which makes fast lookup and less memory consumption. In this paper we concluded various algorithm and different data structure that makes less memory consumption and increases the fast insertion, updating the PIT table. This makes efficient forwarding strategy. The PIT follows the exact match policy, through this policy it needs to match all the component in the naming URL. For this problem, the component is encoded into a smaller basis, which makes less memory consumption and fast name lookup.

REFERENCES

[1] Saxena, D., & Raychoudhury, V. (2016). Radient: Scalable, memory efficient name lookup algorithm for named data networking. *Journal of Network and Computer Applications*, *63*, 1–13. https://doi.org/10.1016/j.jnca.2015.12.009

[2] Saxena, D., & Raychoudhury, V. (2016). N-FIB: Scalable, memory efficient name-based forwarding. *Journal of Network and Computer Applications*, *76*(September), 101–109. https://doi.org/10.1016/j.jnca.2016.09.007

[3] Dai, H., Lu, J., Wang, Y., Pan, T., & Liu, B. (2017). BFAST: High-Speed and Memory-Efficient Approach for NDN Forwarding Engine. *IEEE/ACM Transactions on Networking*, *25*(2), 1235–1248. https://doi.org/10.1109/TNET.2016.2623379

[4] Li, Z., Liu, K., Zhao, Y., & 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/LCOMM.2014.2359191

[5] Luo, J., Wu, C., Jiang, Y., & Tong, J. (2015). Name Label Switching Paradigm for Named Data Networking. *IEEE Communications Letters*, *19*(3),335–338. https://doi.org/10.1109/LCOMM.2014.2387344

[6] Quan, W., Xu, C., Guan, J., Zhang, H., & Grieco, L. A. (2014). Scalable name lookup with adaptive prefix bloom filter for named data networking. *IEEE Communications Letters*, *18*(1), 102–105. https://doi.org/10.1109/LCOMM.2013.112413.132231

[7] Saxena, D., Raychoudhury, V., Suri, N., Becker, C., & Cao, J. (2016). Named Data Networking: A survey. *Computer Science Review*, *19*, 15–55. https://doi.org/10.1016/j.cosrev.2016.01.001

[8] Alubady, R., Hassan, S., & Habbal, A. (2018). Pending interest table control management in Named Data Network. *Journal of Network and Computer Applications*, *111*(September2017),99–116. https://doi.org/10.1016/j.jnca.2017.11.002

[9] Lee, J., & Lim, H. (2017). A new name prefix trie with path compression. *2016 IEEE International Conference on Consumer Electronics-Asia, ICCE-Asia 2016*, 0–3. https://doi.org/10.1109/ICCE-Asia.2016.7804781

[10] Wang, Y., Pan, T., Mi, Z., Dai, H., Guo, X., Zhang, T Dong, Q. (2013). NameFilter: Achieving fast name lookup with low memory cost via applying two-stage Bloom filters. *Proceedings - IEEE INFOCOM*, 95–99. https://doi.org/10.1109/INFCOM.2013.6566742

[11] Wang, Y., He, K., Dai, H., Meng, W., Jiang, J., Liu, B., & Chen, Y. (2012). Scalable name lookup in NDN using effective name component encoding. *Proceedings - International Conference on Distributed Computing Systems*, (20100002110051),688–697. https://doi.org/10.1109/ICDCS.2012.35

[12] Duarte, J. M., Braun, T., & Villas, L. A. (2019). MobiVNDN: A distributed framework to support mobility in vehicular named-data networking. *Ad Hoc Networks*, *82*, 77–90. https://doi.org/10.1016/J.ADHOC.2018.08.008

[13] Kirkpatrick, S., Boniface, M., Bouckaert, S., Grace, P., Jimenez, J., Lahnalampi, T., … Schaffers, H. (2013). F Uture I Nternet R Esearch and E Xperimentation : V Ision and S Cenarios 2020, 61–69.

[14] W. So, A. Narayanan, and D. Oran, “Named data networking on a router: Fast and dos-resistant forwarding with hash tables,” in Proc. ACM/IEEE Symp. Architectures Network and Commun. Syst. (ANCS), 2013, pp. 215–226.