FQ-BF: An Efficient Forwarding and Query Optimization Framework based on Bloom Filters in NDN

Arshdeep Singh, Gurjit Singh Bhathal

***Abstract:** With the increase in internet users and data generation resources; Named Data Networking (NDN), a content-driven communication model that has come into the limelight. Most of the applications in today's world are data-intensive, prefer a data-based network over the address-based network for communication. Continuous efforts by researchers try to optimize the NDN performance in various areas like routing, forwarding, caching, security, etc. NDN basically works on searching for the required data packet, so the content search is one of the promising areas in NDN that still needs improvement. This paper proposes a novel content search algorithm based on Bloom Filters (BF) named FQ-BF; which focuses on efficient and smart forwarding of I_pkt and reduces the number of queries to search a packet in the network. The important metrics where the proposed approach has shown significant improvement as compared to standard NDN approach; include the average delay of packets, improving the packet serving capacity of the network.*

***Keywords:** Bloom Filters (BF), Bloom Matrix (BM), Compressed Bloom Filters (CPBF), Counting Bloom Filters (CBF), Named Data Networking(NDN).*

I. Introduction

The basic design of the internet was to provide end to end connectivity between two clients. Further, TCP/IP made it even simpler to share pictures, videos, etc. in packets [1]. But with time and technology advancement, many information-centric applications come into the picture like Facebook, Amazon, Twitter etc. which generates lots of data and no content support from the internet leads to its failure. Also, the internet does not have any embedded security and mobility.

A. Named Data Networking(NDN)

As a result of above, researches were encouraged to work towards a new technology that can work based on content along with providing better support for Security and Mobility of data. Henceforth, NDN came up as a suitable candidate for the above needs that works with user-requested data irrespective of its location.

Manuscript published on November 30, 2019.
* Correspondence Author
Arshdeep Singh*, Computer Engineering, Punjabi University, Patiala, India. Email: arshdeep.singh209@gmail.com
Gurjit Singh Bhathal*, Computer Engineering, Punjabi University, Patiala, India. Email: gurjit.bhathal@gmail.com

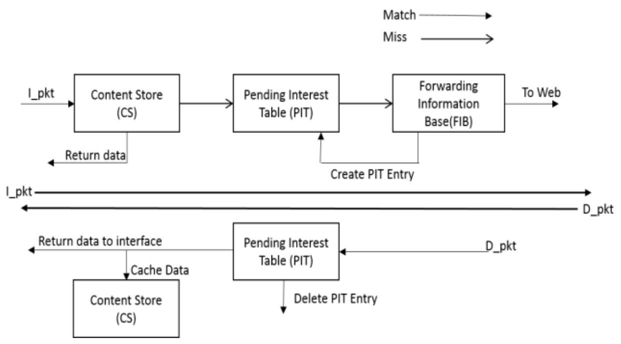

Fig. 1. Communication process at NDN node. [2]

The basic design principle of NDN is similar to the Internet. NDN uses self-contained packets with unique content names associated with it. NDN had resolved many issues with IP packets like Address space exhaustion, Address management, Congestion Control and Security [3, 4].

NDN Communicates in form of Interest packets I_pkt (request packet) and Data Packets D_pkt (Actual data packets in response to a request). As mentioned earlier, both I_pkt and D_pkt have a unique content name associated with them. Each NDN node maintains 3 data structures [5]:

- **Content Store(CS)** – CS maintains a copy of each D_pkt passed through it to fulfill any future requests for the same I_pkt.
- **Pending Interest Table(PIT)** – an I_pkt is searched in PIT, if not found then PIT creates a new entry and sends it further to FIB. If found in PIT i.e. request regarding the same I_pkt has been observed and no insertion in PIT. PIT associates a timeout with every packet and deletes entry if D_pkt is not received in that timeframe.
- **Forwarding Information Base(FIB)** – FIB sends the packet upstream to search it on another NDN nodes and keeps track of all the packets w.r.t. nodes on which it forwards.

As shown in Fig. 1, when NDN node receives an I_pkt form consumer, it searches in its own CS first and if found, returns the D_pkt corresponding to request. If not found, I_pkt is sent to PIT, where an entry with incoming interface and timeout is created and I_pkt is sent to FIB. FIB then sends it upstream to the web to search the D_pkt on respective nodes. When a D_pkt is found on one node, it will be sent on the same way back deleting all the PIT entries from intermediate nodes and caching the D_pkt in each node.

Main functional characteristics for NDN are Routing (to check available nodes and decide the topologies), Caching (to cache the content in every CS of node), Forwarding (to route the I_pkt and D_pkt to feasible routes to ensure performance), Security (to ensure the security of each packet traveling in and out of NDN) and Mobility (to enable the content availability without acquiring an IP address).

When a new I_pkt arrives at each node three queries are fired i.e. CS. PIT and FIB which increases the search time and routing delay for a packet, in turn, leads to the packet timeout and packet is lost before reaching the destination. So there arises a need to optimize and reduce the query time for specific content in NDN to provide better throughput, packet serving, lesser routing delay etc. This paper proposes a new framework based on BFs in NDN that will reduce the search at each node significantly.

The rest of the paper is organized as follows: Section 2 provides a literature study on BF's and existing applications in NDN. Section 3 provides the problem statement. In Section 4, the proposed approach is discussed in detail. Section 5 provides a comparative analysis of existing approaches with the proposed approach. Finally, Section 6 concludes the paper with possible future work in this domain.

II. LITERATURE SURVEY

A. Bloom Filters

A Bloom filter is space-efficient PDS that is used to check the presence of an item in a data set as shown in figure 2. BF is a fixed-length linear array in which we set a particular bit to 1 or 0 based on the hashing values of particular data [6].

To check whether the queried element is part of the set or not, similar hashing is used, if respective bits are found 1 in BF i.e. present otherwise not. False positives could also occur in BF i.e. it may result in true for a content that is not actually present because the respective bits might be set by some other content. There are various kinds of BF's proposed to date and research is going on. *Singh A et al* in

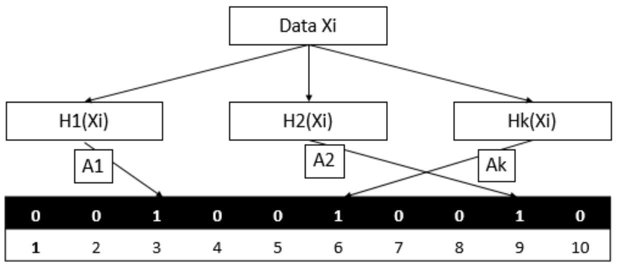

Fig. 2. Insertion of an element in BF.

[7] categorized various BF's according to their functions and applications. Some of them are given as:

• **Counting BF (CBF)** – CBF was introduced by *Fan et al in [8]* is an enhanced version of standard bloom filter having counters instead of bits at every position. It supports insertion and deletion both. Whenever a new element comes to the BF, counters on respective bits are incremented by one. While deleting a content respective counters are decremented by one. CBF can increase the memory overhead by consuming a lot of space to store the counters instead of bits in the array. Various versions of CBF are variable increment CBF[12], L-CBF[13] etc.

• **Compressed BF (CPBF)** – CPBF was introduced by *Michael Mitzenmacher in* [9] is a compressed version of counting bloom filters that are used for the transmission purpose in order to save the network bandwidth and transmission time. CPBF replaces all the counters with bits being if the counter value is 0 then CPBF inserts a 0 at that position whereas if the counter value is greater than 0 then CPBF inserts 1. By doing this we can save the lot memory compared to transmitting the original CBF. The main application of CPBF are Web Cache, distributed routing table, etc.

• **Bloom Matrix (BM)** – BM was introduced by *Francesco Concas et al*. in [10]. This data structure is an extension of standard BF that stores multiple same sized BF's in a 2D array structure. BM is used for membership query in cases where we have to search the data from multiple sets. BM provides efficient search results by finding out the desired sets which can be further queried to get data. This is a space-efficient data structure. The main application of BM could be document search, web caching, etc.

B. Application of BF in NDN

BF has been used at various functional areas of NDN like routing, caching, forwarding and security. *A Singh & G. S. Bhathal* in [11] discussed the various bloom filters used in NDN. Some are given as:

1. Routing –

- *BF based Routing* uses simple BF's for routing the packets. It reduces the communication cost, roundtrip delay and also overcomes the challenges of topology robustness [14].
- *COBRA* is a Content-driven BF based Intra domain routing algorithm that uses stable BF's. it reduced the network overhead and also provides the same hit distance as Dijkstra's algorithm [15].
- *Pull Based BFR* is a modified version of BFR using simple BF's. It reduced the bandwidth consumption, memory requirements and provided better average round trip delay [16].

2. Caching –

- *Cache Sharing using BFs* uses the Stable BF's and Shifting BF's to store the cache information and share it with the nearby nodes, also avoiding the caching of neighbor's data. It provides better content diversity and cache hit ratio [21].
- *Line-speed and accurate on-line popularity monitoring using BF* is a BF based algorithm to capture the content popularity and relace it as part of the content replacement methods. It used simple BF [22].

3. Forwarding –

- *MaPIT* is an enhanced PIT implementation using the mapping BFs. it works under scalable forwarding. It mainly targets to reduce the on-chip memory consumption and false positive rate of the network [17].

- *NameFilter* is a BF based name lookup method that uses one memory access BF and merged BF. It mainly targets to provide fast querying and high memory efficiency [18]. It had outperformed all exiting name lookup data structures like character trie [25], bloom hash[26] etc.
- *Name Lookup Adaptive Prefix BF* uses adaptive prefix BF to enhance the name lookup in FIB. It works on the longest prefix matching using 2 data structures BFs and Trie to get the best of both the data structures. By using both, it reduces the false positives of BF's and memory complexity of a Trie [19].
- *A new BF based architecture for FIB Lookup using 2-phase BF* is new techniques used for name lookup using 2-Phase BF. It works on the principle of Longet Prefix Trie (LPT). It stores the prefix information in on-chip BF's. It was able to address 99.99% queries from on-chip BF's, in turn, reducing the delay significantly [20].

4. Security –

- *Security and Privacy-preserving NDN architecture* uses simple BF's to store the secret keys in compressed form and transmit securely over the network. It uses multiple BF's to reduce false positivity. Content Dependent Key Management Tree is used for key management [23].

As given above multiple BFs have been used at various stages in NDN like CS, PIT, FIB, etc. They have improved the respective areas like routing, forwarding, caching of the content respectively. But still, there is a major concern in content search, because to search for any content in NDN we generally perform three queries at every NDN node irrespective of the availability of packet on that node, which increases the routing delay and in turn packet loss due to the timeout. In this paper, we have proposed an approach to address the above problems in NDN, using a hybrid BF based approach.

III. Problem Statement

NDN works on content names instead of the address. All the packets contain unique content names with them and all the NDN functions like search, routing, caching, forwarding works based upon the content names only. Many researchers have proposed efficient techniques using BF's in various domain of NDN like Routing, caching, forwarding and security. All these techniques have contributed to the improvement of NDN significantly. But, NDN's main functionality lies in content search on NDN nodes. When an I_pkt is received, we perform three queries at every node i.e. checking whether the content is present in CS, if not whether there is an entry in PIT (Add if not present in PIT, else modify existing), at last we go to FIB to route the I_pkt to next available nodes. Every process takes its own time and as the content and number of NDN nodes increases, it will lead to the following problems:

• **Timeout** – When an entry for an I_pkt is made in PIT, a specific timeout is associated along with that i.e. if D_pkt is not received before timeout PIT deletes the entry. With the increase in content chances of packet-timeout occurrences also increase.

• **Packet Loss** – As earlier said, if the timeout expires then PIT deletes the entry leading to packet loss. So, as packet-timeout occurrence will increase, packet loss will also increase significantly.

• **Routing Delay** – Routing delay is the time taken by a packet to reach its destination, as content and number of nodes will increase, then it will lead to increase in routing delay in turn leading to timeout and packet loss.

• **Number of queries** – As the content increases in the network, the number of nodes will also increase in turn leads to an increase in the number of queries to search a particular content in the network.

So, to overcome the above-stated problems, we need to come up with a solution that focuses on reducing the unnecessary queries at NDN nodes.

IV. Proposed Framework

To overcome the issues addressed in the problem statement section, this paper proposes a hybrid framework using BF's and BM named FQ-BF: that will reduce the number of queries to be performed at each node and in turn, will provide us the better packet serving and lesser routing delay. The following subsections will give the details on the proposed setup and the approach that is followed.

A. Proposed framework

The FQ-BF makes one node from the network as a master node which is

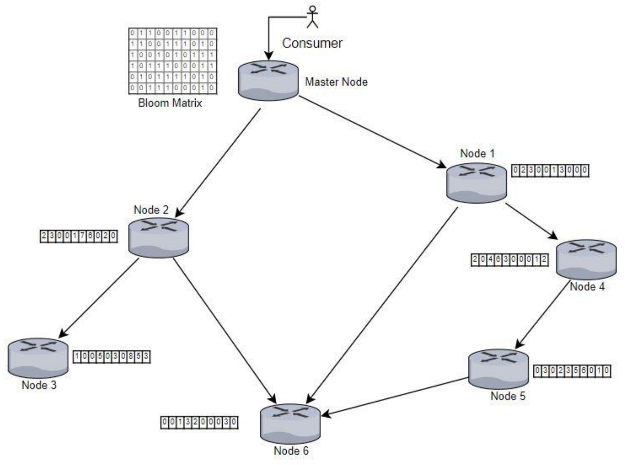

Figure 3. Setup of FQ-BF

connected directly or indirectly to all the nodes in the given network. All the nodes in the network will maintain one Counting BF (CBF) for CS, to show which all data is present. CBF is used here to support the insertion and deletion of the dynamic content available in CS. The length of the CBF stored at all the nodes will be fixed. Master node will maintain a BM that will store the data from all the nodes in compressed form i.e. all the nodes will send its CBF in compressed form (Compressed BF) to master node periodically.

CPBF is used for reducing the memory consumption at the master node and reducing the transmission bandwidth to send data from nodes to the master node. All the nodes will periodically send updated CPBF's to the master node to be updated in BM. Structure of the FQ-BF is shown in figure 3.

In the FQ-BF, every node will use a CBF to present the content stored in CS and the same is been transmitted to the master node in compressed form i.e. CPBF. All the data from different nodes will be stored at the master node in compressed form. So master node will contain the content information of all the nodes in the network at a given time. Whenever a query comes to the network for a specified D_pkt will always enter from the master node. Figure 4 shows the flow of control in the FQ-BF. When a consumer requests a D_pkt in NDN, corresponding I_pkt will go to the master node and search whether D_pkt is present at any of the nodes. If TRUE, I_pkt will be cached at the master node with a specified timeout. As BM is updated periodically with data from all nodes, cached I_pkt will be checked for existence at the master node before the timeout expires. If the corresponding D_pkt exists in the BM at the master node, find out the destination node that contains the D_pkt. Once the destination node is confirmed, the FQ-BF uses the NDN sim shortest path algorithm to find out the shortest path to the destination node. I_pkt is routed on the shortest path in the network. While I_pkt is moving towards the destination node, it will not be queried at any intermediate node in CS and PIT. I_pkt will directly go to FIB and routes the packet to the next node on the same path. Once I_pkt reaches the destination node, CS will be queried against given I_pkt,

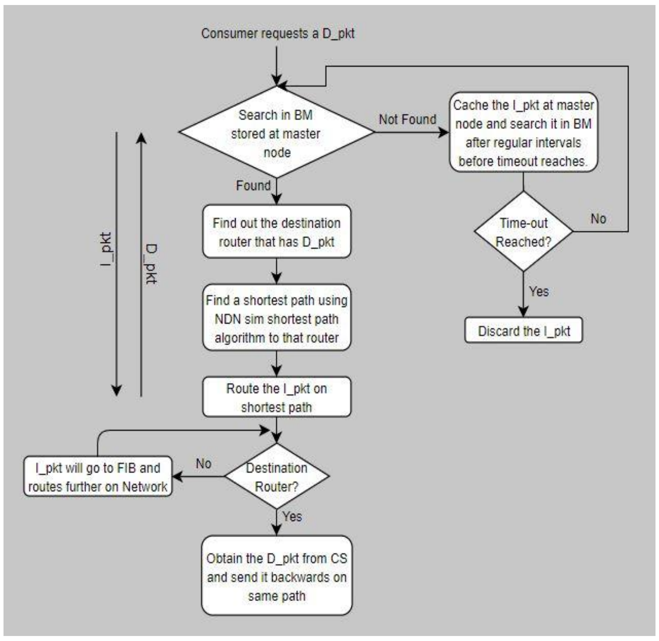
Consumer requests a D_pkt
Search in BM stored at master node
Not Found
Cache the I_pkt at master node and search it in BM after regular intervals before timeout reaches.
Found
Find out the destination router that has D_pkt
Time-out Reached?
No
Yes
Discard the I_pkt
Find a shortest path using NDN sim shortest path algorithm to that router
Route the I_pkt on shortest path
I_pkt
D_pkt
I_pkt will go to FIB and routes further on Network
No
Destination Router?
Yes
Obtain the D_pkt from CS and send it backwards on same path

Fig. 4. Flow chart of FQ-BF

obtain the D_pkt and sends it back on the same way to the customer. While traversing back it will not cache any content on intermediate nodes.

So, the FQ-BF is able to reduce the number of queries on each node while searching a D_pkt for an I_pkt. This leads to improvements in packet serving capacity and average routing delay of the network.

V. Results

To evaluate the performance of the FQ-BF, we performed a comparative analysis between the proposed technique and standard NDN where BF's are not implemented at the CS level. The comparative analysis is plotted based on the average results of 100 runs of script per every scenario.

A. Experimental Setup

The various mechanisms for PIT are implemented in R language on a system with i5-7200U CPU @2.50 GHz, 8GB of RAM and 64-bit Windows 10 OS. Taking into account the implementation of S-PIT, there are three major functions defined in RStudio for insertion, deletion and decrement operations. Considering the essentials of PIT, a standard dataset from 'Content Name Collection' (CNC) [24] is used which consists of open datasets of Information-Centric Networking (ICN) names having different formats and size. The experimental dataset used is unibas-\-icn-\-names-\-2016-\-08 which consists of 141 files each of which consists of 10,000,000 content names except the last one.

B. Packets served in a specified time

Packet loss is a very big concern in NDN as in due to delay and number of queries, the packet reaches a timeout and it is lost. The number of satisfied interest packets in a given time is the base of the performance measure for any technique in NDN. The FQ-BF

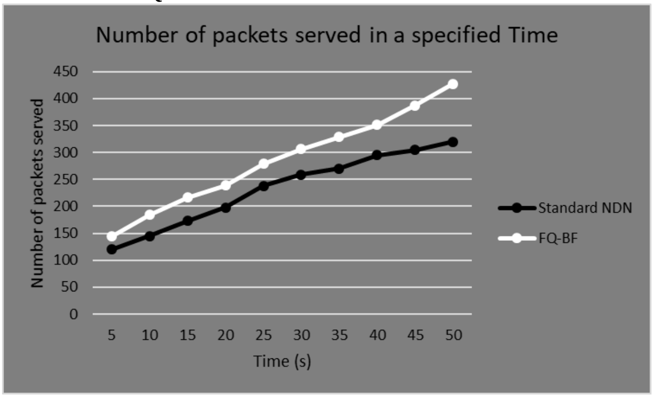

Figure 5. Comparative analysis of packets served in a given time.

provides a better packet serving capacity by reducing the number of queries and routing delay for the packets. The FQ-BF provides an average increase of 20% in the number of satisfied interest packets as compared to standard NDN. A comparative analysis is shown in Figure 5.

C. Number of queries based on internal nodes

The number of queries per I_pkt to search for the desired D_pkt plays an important role in the performance of the NDN. As the number of queries increases, routing delay will increase, packet-timeout will increase in turn reducing the number of satisfied interest packets. The FQ-BF aims to reduce the number of queries by performing the query only when needed as compared to standard NDN. Table-I shows the average number of queries for Standard NDN and FQ-BF after 100 runs of the script on the simulator.

The FQ-BF provides an average decrease of 60% in the number of queries to be performed for an I_pkt in NDN. A comparative analysis is shown in Figure 6.

Table- I: Average number queries based on internal nodes (avg. of 100 runs of the script)

Internal Nodes	Standard NDN	FQ-BF
50	145	65
100	270	108

150	415	168
200	610	225
250	723	259
300	920	309
350	1027	372
400	1194	437
450	1338	486
500	1523	538

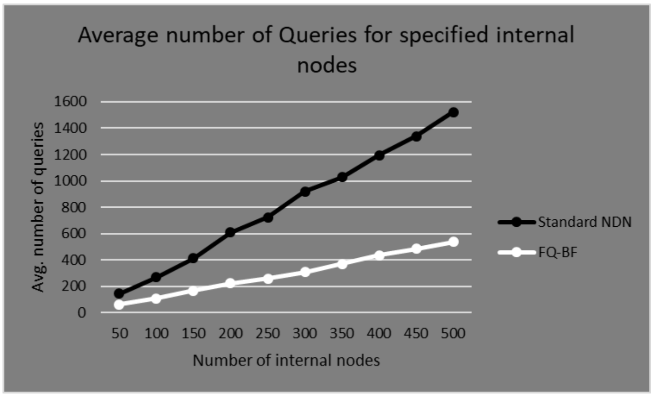

Figure 6. Comparative analysis of the average number of queries.

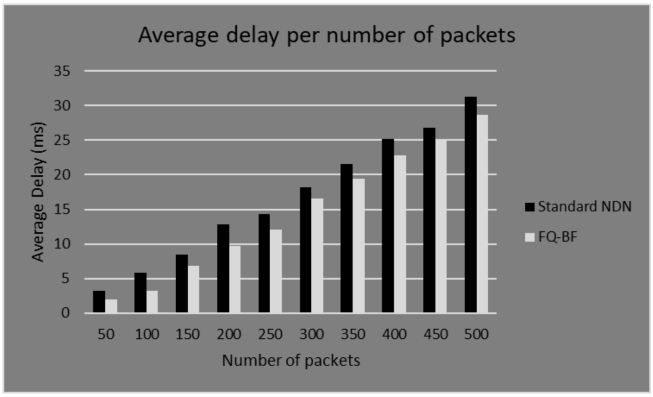

Figure 7. Comparative analysis of average delay based on the number of packets (number of internal nodes = 100).

D. Average Delay per number of packets/internal nodes

Routing delay is the time taken by the network to return a desired D_pkt for an incoming I_pkt. More the routing delay in the network, more will be the packet timeouts and resulting in lesser number of satisfied I_pkt's. We did the comparative analysis of average routing delay for packets based on two parameters, i.e. number of packets and number of internal nodes:

- **Based on the number of packets**: We kept the number of nodes as constant i.e. 100 in the network and varied the number of packets by fixed numbers. The FQ-BF provides the average decrease by 2-3 milliseconds in the overall average delay of the NDN as compared to standard NDN without BF. A comparative analysis is shown in Figure 7.

- **Based on the number of internal nodes**: We kept the number of packets as constant in the network and varied the number of internal nodes by fixed numbers. The FQ-BF provides the average decrease by 3-5 milliseconds in the overall average delay of the NDN as compared to standard NDN without BF. A comparative analysis is shown in Figure 8.

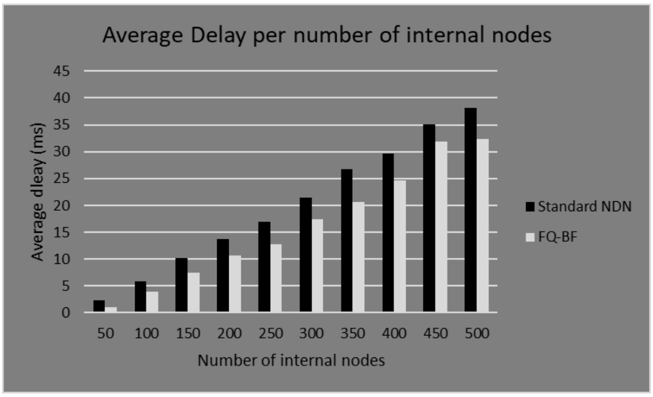

Figure 8. Comparative analysis of average delay based on the number of internal nodes (number of packets = 100)

VI. CONCLUSION AND FUTURE WORK

The FQ-BF is observed to be a highly suitable content search approach in NDN. It leads to a significant reduction in problems like Packet loss, timeout routing delay and increased number of queries in NDN. FQ-BF is able to reduce the number of queries by a significant difference of 60% as compared to standard NDN, in turn giving the improved number of satisfied interest packets (an increase of 20%) and lesser routing delay (average 3-4 milliseconds). Thus the FQ-BF can satisfy the requirements for improving the content search in NDN. The results of this paper are evaluated on a simulator environment, the FQ-BF can further be deployed on a real network to compare the performance of FQ-BF against benchmark approaches. More advanced and latest BF's also can be explored to further better the performance of the approach.

REFERENCES

1. History of Computers and Computing, Internet, Maturing of the Net, TCPIP, www.history-computer.com/Internet/Maturing/TCPIP.html
2. L. Zhang, D. Estrin, J. Burke, V. Jacobson, J.D. Thornton, D.K. Smetters, B. Zhang, G. Tsudik, K.C. Claffy, D. Krioukov, D. Massey, C. Papadopoulos, T. Abdelzaher, L.Wang, P. Crowley, E. Yeh, Named Data Networking (NDN) Project, 2010, http://named-data.net/project/annualprogress-summaries/ (2010)
3. Z.A. Jaffri, Z. Ahmad, M. Tahir, "Named Data Networking (NDN), new approach to future Internet architecture design: A survey", Int. J. Inf. Commun. Technol. (IJ-ICT) 2(3), pp. 155–165 (2013)
4. Divya Saxenaa, Vaskar Raychoudhurya, Neeraj Surib, Christian Beckerc, Jiannong Caod, "Named Data Networking: A survey", Computer Science Review (Science Direct), Volume 19, pp. 15-55 (2016)
5. V. Jacobson, D.K. Smetters, D. James Thornton, P.F. Micheal, B.H. Nicolas, B.L. Rebeeca, "Networking named content", In: Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, DOI: 10.1145/1658939.1658941, pp. 1-12 (2009)
6. B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," In: Communications of the ACM, vol. 13, pp. 422-426 (1970)
7. Singh, A., Garg, S., Kaur, R., Batra, S., Kumar, N., & Zomaya, A. Y. (2019). Probabilistic data structures for big data analytics: A comprehensive review. Knowledge-Based Systems, 104987.

8. Fan, L., Cao, P., Almeida, J., & Broder, A. Z. (2000). Summary cache: a scalable wide-area web cache sharing protocol. IEEE/ACM Transactions on Networking (TON), 8(3), 281-293.
9. Mitzenmacher, M. (2002). Compressed bloom filters. IEEE/ACM Transactions on Networking (TON), 10(5), 604-612.
10. Concas, F., Xu, P., Hoque, M. A., Lu, J., & Tarkoma, S. (2019). Multiple Set Matching and Pre-Filtering with Bloom Multifilters. CoRR.
11. Arshdeep Singh*1 & Gurjit Singh Bhathal2. (2019). BLOOM FILTER APPLICATIONS IN NAMED DATA NETWORK: A COMPREHENSIVE REVIEW. GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES, 6(8), 1–12. http://doi.org/10.5281/zenodo.3362262
12. Yossi Kanizo, Isaac Keslassy, "The variable-increment counting bloom filter", In: IEEE/ACM Transactions on Networking (TON), Volume 22 Issue 4, pp. 1092-1105 (2014)s
13. Elham Safi, Andreas Moshovos, Andreas Veneris, "L-CBF: A Low-Power, Fast Counting Bloom Filter Architecture", In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volume: 16, pp. 628-638 (2008)
14. Ali Marandi, Torsten Braun, Kave Salamatiany, Nikolaos Thomos, "BFR: a Bloom Filter-based Routing Approach for Information-Centric Networks", arXiv:1702.00340v1 [cs.NI] (2017)
15. Michele Tortelli, Luigi Alfredo Grieco, Gennaro Boggia, Kostas Pentikousis, "COBRA: Lean intra-domain routing in NDN", In: 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC), DOI: 10.1109/CCNC.2014.6994403, Las Vegas, NV, USA (2014)
16. Ali Marandi, Torsten Braun, Kave Salamatiany, Nikolaos Thomos, "Pull-based Bloom Filter-based Routing for Information-Centric Networks", arXiv:1809.10948v1 [cs.NI] (2018)
17. Zhuo Li, Kaihua Liu, Member, IEEE, Yang Zhao, Yongtao Ma, "MaPIT: An Enhanced Pending Interest Table for NDN with Mapping Bloom Filter", IEEE COMMUNICATIONS LETTERS, VOL. 18, pp. 1915-1918 (2014)
18. Yi Wang, Tian Pan, Zhian Mi, Huichen Dai, Xiaoyu Guo, Ting Zhang, Bin Liu, "NameFilter: Achieving fast name lookup with low memory cost via applying two-stage Bloom filters", In: 2013 Proceedings IEEE INFOCOM, DOI: 10.1109/INFCOM.2013.6566742, Turin, Italy (2013)
19. Wei Quan, Changqiao Xu, Jianfeng Guan, Hongke Zhang, Luigi Alfredo Grieco, "Scalable Name Lookup with Adaptive Prefix Bloom Filter for Named Data Networking", IEEE COMMUNICATIONS LETTERS, VOL. 18, pp. 102-105 (2014)
20. Hayoung Byun, Hyesook Lim, "A New Bloom Filter Architecture for FIB Lookup in Named Data Networking", Appl. Sci. 2019, 9, 329; doi:10.3390/app9020329 (2019)
21. Ju Hyoung Mun, Hyesook Lim, "Cache sharing using bloom filters in named data networking", Journal of Network and Computer Applications 90 (2017), 74–82 (2017)
22. Huichen Dai, Yi Wang, Hao Wu, Jianyuan Lu, Bin Liu, "Towards Line-Speed and Accurate On-line Popularity Monitoring on NDN Routers", In: 2014 IEEE 22nd International Symposium of Quality of Service (IWQoS), DOI: 10.1109/IWQoS.2014.6914318, pp. 178-187 (2014)
23. Emmanuel A. Massawe, Suguo Du, Haojin Zhu, "A Scalable and Privacy-Preserving Named Data Networking Architecture based on Bloom Filters", In: 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops, DOI 10.1109/ICDCSW.2013.32, pp. 22-26 (2013)
24. Schnurrenberger, "The content name collection, cnc." [Online]. Available: http://www.icn-names.net/, 2017 [Accessed: 10 February 2019]
25. E. Fredkin, Trie memory, ACM Commun. 3 (9) (1960) 490–499.
26. S. Dharmapurikar, P. Krishnamurthy, D.E. Taylor, Longest prefix matching using bloom filters, in: Proceedings of the International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2003, pp. 201–212.

AUTHORS PROFILE

ARSHDEEP SINGH, The author is a research scholar and pursuing the M. Tech in Computer Science and Engineering from Punjabi University, Patiala, India. He had received B.Tech. Degree from Chitkara Institute, Rajpura, India, in Computer science and engineering, in 2011. He is having over 8 years of industrial experience in reputed Multi-National companies holding different positions. He has developed many tools as part of his role as a developer to aid the industrial practices. He also worked on managing and improving the multiple projects for International clients as part of his role as DevOps engineer. He has many publications in the area of NDN and Big Data. His research interests are Probabilistic Data Structures, Named Data Networking, Big Data.

GURJIT SINGH BHATHAL The author is an Assistant Professor (Stage-II) at the Department of Computer Science and Engineering at Punjabi University Patiala. He has over 18 years of experience in India and abroad in the field of Information Technology. His research interests in the area of security including, Cloud security, Big Data Security and Cyber Security. He has completed his B. Tech in Computer Science and Engineering from SLIET Longowal, M. Tech from Punjabi University and Ph.D. from Punjabi University. He is a Microsoft Certified Engineer (MCSE) and IBM certified of Big Data and Data Privacy Fundamentals. He has more than 60 Publications including International journals, National and International conferences and three books in his credit. Mr. Gurjit Singh Bhathal recently awarded Outstanding Scientist in Computer Science and Engineering in 4th Annual Research Meet - ARM 2018.