

Dynamic Bloom Filter for Intelligent Routing

1st Geetika Srivastava
Department of Computer Science and
Engineering
Chandigarh University Mohali-
140413, Punjab, India

2nd Azhar Ashraf
Department of Computer Science and
Engineering
Chandigarh University Mohali-
140413, Punjab, India

3rd Abhinav Anil Shukla
Department of Computer Science and
Engineering Chandigarh University
Mohali-140413, Punjab, India

4th Ashutosh Kumar
Department of Computer Science and Engineering
Chandigarh University Mohali-140413, Punjab, India

5th Aditya Sharma
Department of Computer Science and Engineering
Chandigarh University Mohali-140413, Punjab, India

Abstract - Enterprise networks that operate at high speeds need routing techniques which are able to efficiently handle large IP forwarding tables and at the same time identify and filter out suspicious traffic. Typically, the traditional routing and security mechanisms rely on separate look-up structures, and as the size of the network and the type of traffic evolve, this results in increased memory access and processing overhead. This paper proposes a Dynamic Bloom Filter for Intelligent Routing (D-BF). D-BF is a hybrid solution, which combines a longest-prefix matching via Patricia-Trie and a packet screening via dynamically managed Bloom filter. The Patricia Trie supports efficient and guaranteed IP route queries and the Dynamic Bloom Filter enables maintaining compact representations of the current flow states and making simple security decisions. The framework is designed to enable dynamic updates, scalable routing and intrusion detection in a varying traffic scenario. The experimental evaluation considers the routing throughput, the look-up latency, the memory utilization, the false positive rate, packet-filtering performance and scalability. The result demonstrates the capability of the integrated routing and security architecture to achieve high throughput in forwarding and at the same time provide an efficient way to perform stateful packet filtering. The method proposed offers a viable framework to integrate intelligent routing and network security in a unique packet-processing mechanism.

Keywords: Dynamic Bloom Filter, Patricia-Trie, Intelligent Routing, Packet Filtering, Network Security, Longest-Prefix Matching, High-Speed Networks, Routing Lookup

I. INTRODUCTION

A. Background

High speed enterprise networks require high-speed routing protocols capable of managing massive forwarding table sizes and rapid changes in the size of the forwarding tables, while at the same time providing for efficient security at the packet level. When handling IP routing [1,20] one can use a compact hierarchical representation of these Patricia Trie structures and use them to efficiently perform longest-prefix matching. Bloom filters are a space-efficient probabilistic membership query structure, and have been successfully adopted to solve

many networking issues, like IP address look-up and packet filtering [2]–[4]. Variants of Bloom filters that are dynamic and counting also enable changes to network states and deletion operations [5]–[7]. Prior studies on the programmable data plane have demonstrated that combining Bloom filters with a stateful forwarding and/or firewall functionality can be achieved [16]–[19].

B. Motivation

In conventional ways of routing and providing security, there are separate processing structures thus increasing the number of memory accesses and the processing overhead due to the number of routing tables and number of active flows. Bloom filters can be used to help with routing and reduce unnecessary accesses to the tables [3, 10] and programmable filtering can enable security decisions to be made closer to the packet-forwarding path [16, 17]. Recent efforts in SDN rule management, data-aware Bloom filters and dynamic security mechanisms [13]–[15] and [18] provide more examples of how probabilistic filtering can be used to improve scalable network management. However, still it is hard to achieve a balance between lookup accuracy, security, memory efficiency and to perform dynamic updates.

C. Problem Statement

Traditional IP routing is based on deterministic longest-prefix matching, and security filtering requires efficient and fast flow state tracking of rapidly changing states. The exclusive use of trie-based routing may increase the lookup and memory costs with the increase of the network, and the use of separate Bloom filters may cause false positive and inability to delete at the end point [4]–[7]. Existing hybrid approaches tend to be used for specific routing or security scenarios, but do not provide an overall adaptive approach.

D. Objective/Scope

The Dynamic Bloom Filter for Intelligent Routing, a novel combination of a Patricia-Trie based IP prefix look-up and dynamically-managed Bloom-filter packet screening is introduced. The framework is built with the aim of: enhancing the routing efficiency, decreasing the overhead of security processing, achieving scalable flow-state management and measuring throughput, lookup latency, memory consumption, false-positive rate and packet-filtering performance in various network conditions.

II. LITERATURE REVIEW

An efficient packet forwarding implementation needs small data structures which minimize accesses to memory and the look-up latency. The PATRICIA by Morrison presented a compressed radix-tree format that minimizes the index-space cost, but allows dynamic retrieval to be efficient [1]. In the case of probabilistic membership testing, Bloom introduced a hashing structure which is space efficient and which trades a falsely positive probability which can be controlled against a much lower memory consumption [2]. These ideas later became relevant to high-speed routing, in which Broder and Mitzenmacher demonstrated that several-hash schemes could be used to provide faster IP-prefix lookups and yet scale and be cache-efficient [3]. A generalized survey verified the Bloom filters can be used in packet routing, measurement, resource routing, and distributed networking [4].

Studies then shifted to ensuring that Bloom filters were dynamic and could be used in adapting to changing network conditions. Guo et al. proposed Dynamic Bloom Filters (DBFs) to ensure limited false-positive rates with a fluctuating set size and applied the idea to multidimensional dynamic sets [5]. This journal work, which was later developed by them, cemented the theoretical and practical features of dynamic filter sizing and scalability [6]. Bloom filters were counted so insertions and deletions could be made; d-left constructions had lower memory requirements than traditional counting filters [7]. Approximate-state representations demonstrated that Bloom-filter-based representations could model a large set of states of a network simultaneously with significantly less memory consumption than explicit state [8]. Such innovations help to enable dynamic routing and security settings, yet false positives and consistency of updates are still primary issues.

Implementations Network-specific architectures Use Bloom filtering more and more with packet-processing architectures. Rothenberg et al. studied the use of in-packet Bloom filters to support routing and security and demonstrated that deletion and security improvements make them more practical [9]. Li and Chen preprocessed routing queries with Bloom filters in order to reduce expensive lookups in large routing tables, and enhance the efficiency of routing lookups [10]. Counting Bloom filters with cache-assisted blooms were also able to lessen verification processing time of packet-processing loads [11]. These experiments demonstrate that probabilistic filtering does not need to substitute the traditional route tables, but it can be used in addition to them.

New studies have shifted towards combined routing and security. Surveying Bloom-filter applications across Named Data Networking, Nayak et al. found that they were used in the forwarding information base, content stores and security mechanisms [12]. Singh et al. suggested BloomStore, a dynamic Bloom-filter based rule-space management architecture of SDN, which includes better performance on rule-management metrics along with scalable security-policy management [13]. Sengupta et al. delved into effective reconstruction and sampling of Bloom-filter representations, showing that probabilistic summaries can be used to underpin other analytical operations [14]. Bhattacharya et al. also presented data-sensitive hash functions that can significantly decrease the false-positive rates with the same memory budget, but at the cost of extra computation and reliance on the properties of the stored data [15]. These studies that are affiliated to India show great development in fine tuning Bloom-filter structures to efficient and intelligent management of data.

Another significant trend is programmable data planes. Saxena et al. introduced P4Filter, an integration of dynamic firewall logic, authentication-based protection, and proved to be better in defense against unauthorized access in a virtual P4 environment [16]. Later Wang et al. formalized and tested a stateful P4 firewall using a sliding-window Bloom filter on Tofino hardware, which enhanced tracking of flow-states, and stale entries and adversarial coverage attacks [17]. S, ahin and Demirci designed an updatable Bloom-filter implementation of a stateful P4 firewall, and its flow-state tracking, and mitigating stale entries and adversarial coverage attacks [18]. More recent routing work has used Bloom and Cuckoo filters together with trie-based lookups; Zhang et al. have reported over 13-Gbps single-port FPGA throughput with 13-Gbps off-chip accesses capped [19]. Meanwhile, Lin and Hsieh experimentally tested trie-based IP lookup with large real world IPv4 and IPv6 routing tables demonstrating that memory access, storage and lookup time are still significant optimization dimensions [20].

A. Research Gap

The current literature primarily optimizes either trie-based IP lookup or Bloom-filter-based security, and a small number combine both into a single adaptive routing system. Traditional attempts provide correct longest-prefix matching but might have memory-access and update costs. Bloom filters have small and rapid membership testing at the expense of false positives and historically have deletion and update bottlenecks. Recent P4 and hybrid-filter work demonstrates that it is feasible, but targets individual applications like SDN rules, SYN-flood protection, or satellite routing instead of enterprise-wide intelligent routing and intrusion filtering. Thus, a Dynamic Bloom Filter that uses a Patricia Trie can fill this gap by containing deterministic prefix lookup with adaptive probabilistic filtering, explicitly examining the lookup latency, throughput, memory utilization, false-positive rate, scalability, and security detection performance.

III. METHODOLOGY

A. System Architecture

The proposed Dynamic Bloom Filter of Intelligent Routing is a combination of deterministic IP-prefix routing and probabilistic packet screening in a programmable packet-processing pipeline. A Patricia Trie is used to first partition and match incoming packets and offer compact representation as well as efficient hierarchical lookup [1], [20]. This route is then added together with a dynamic Bloom-filter security check and then forwarded. Bloom filtering is an appropriate choice due to its constant-time approximate membership testing, which has low memory consumption [2], and dynamic variants, which can change network states [5], [6]. The architecture decouples the correct route finding with lightweight security filtering, eliminating the unnecessary work on packets not requiring full rule-table inspection. These operations can be implemented by programmable data-plane processing which allows them to be executed near the forwarding path [16], [17].

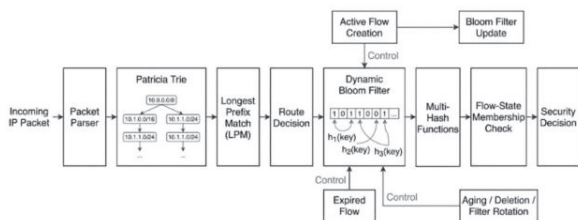


Fig. 1. Proposed Dynamic Bloom Filter for Intelligent Routing architecture.

B. Data Collection and Preprocessing

The test is based on IPv4 forwarding traffic that is produced under controlled network topologies, routing prefixes, packet traces as well as stateful traffic flows. The experimental setting spans both low, and high forwarding rates giving the chance to investigate throughput and execution time consistently. The packet captures are maintained to validate forwarding and filtering decisions and the routing entries are modeled as IPv4 prefixes to construct the trie. Input records are purged of malformed packets and duplicates and IP addresses made normalized into fixed width binary representations and then inserted. The baseline and proposed experiments have identical traffic and routing settings to allow a fair comparison. The controlled packet traces and forwarding scenarios are the practices of evaluation that have been used since time immemorial with programmable networking and Bloom-filter-based firewalls [9], [17].

C. Feature Extraction / Model Selection

Each packet and routing event is extracted by the system with destination IP, source IP, protocol, source and destination ports, packet size, flow state, the result of a lookup and forwarding decision. Patricia Trie indexes destination prefixes and has flow identifiers which are hashed under a variety of hash functions to be used in Bloom-filter membership

testing [3], [4]. The reason to use a dynamic Bloom-filter design is that the traditional Bloom filters lack efficient deletion, which may lead to stalled network-state information [6], [17]. The counting and sliding-window variants provide convenient methods of dealing with changing state [7], [17]. Some of the evaluation features are route-lookup latency, forwarding throughput, packet-filtering latency, memory use, false-positive rate, and permitted or dropped packets.

D. Proposed Technique / Core Module

The main module has four phases of working. The destination IP is first checked in the Patricia Trie to get the longest matching route [1], [20]. Second, multiple hash functions are applied to flow signature (based on relevant packet fields) and can be mapped to Bloom-filter positions. Third, a dynamic filter is updated every time a legitimate flow has been created and expired or outdated state is cleared out with controlled filter rotation or windowing [5], [6], [17]. Lastly, the routing and security outcomes are combined: packets that have valid paths and correct membership to the flow are sent and packets that do not meet the security criteria are destroyed or limited. The design is a combination of precise prefix matching and tight state tracking and is compatible with previous stateful firewall designs that used Bloom filters in programmable data planes [16][18].

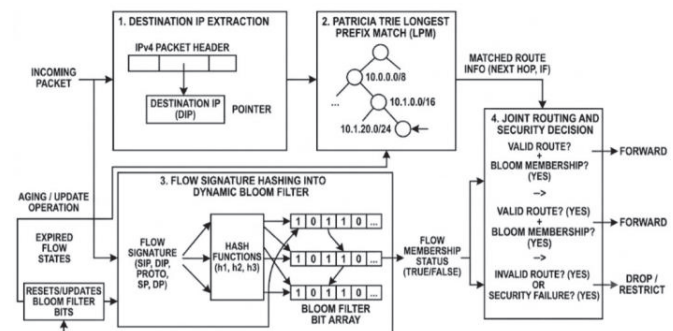


Fig. 2. Step-by-step operation of Patricia-Trie routing and Dynamic Bloom Filter security.

E. Implementation Details

Python and P4/ns-3-based programmable-network simulation components are used to implement the prototype. Python takes care of routing-table preparation, traffic generation, metric extraction and visualising the results, whereas P4-compatible data-plane logic does forwarding or stateful filtering. Trie operations are based on an indexing structure of Patricia prefix and Bloom-filter operations are based on use of multiple hash functions and persistent register-like state. PCAP files are also useful to perform packet-level inspection, whereas NumPy and Matplotlib are effective to perform quantitative analysis and figure drawing. This is implemented after trying programmable-switch experiments on P4 and reproducible simulation workflows [16], [17].

F. Evaluation Strategy

The suggested framework is compared with the traditional forwarding and security settings with growing traffic loads and routing-state sizes. The throughput and simulation or processing time is measured at various rates of traffic and the routing performance is measured based on the routing routing and routing memory. Valid and invalid flows are used to assess the effectiveness of security based on the detection rate, false-positive rate, filtering latency and approximate forwarding accuracy. Other scalability experiments change the routing prefixes and flow states. The results are compared to conventional forwarding, non-dynamic filtering, and other probabilistic structures where possible [7], [10], [13], [19]. The analysis is used to verify that the integrated design increases the routing efficiency and packet-filtering scalability without introducing unacceptable probabilistic or computational overhead.

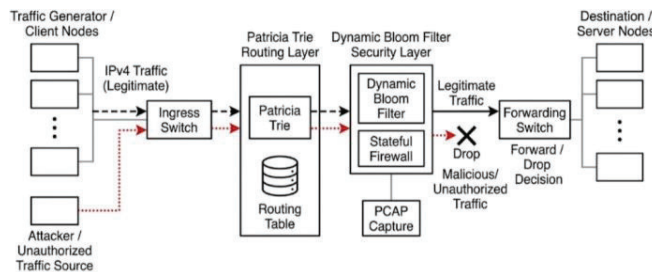


Fig. 3. Experimental network topology for routing and stateful packet filtering.

IV. EXPERIMENTAL SETUP

A. Experimental Environment

The Dynamic Bloom Filter for Intelligent Routing is evaluated in a simulation environment using programmable network for modelling enterprise IPv4 network. It is a setup that contains packet sources, forwarding nodes, a routing layer and a stateful security layer. The workloads are generated for the IPv4 forwarding and packet-filtering workloads under controlled traffic conditions, akin to the experimental techniques used in the study of programmable data planes [16] and stateful P4 processing [17]. The experiments are designed in a way to ensure a fair comparison of both the baseline and proposed methods, as they both use the same routing entries and traffic workloads.

B. Network and Traffic Setup

The network prefixes for IPv4 in the routing table are encoded using hierarchical Patricia-Trie data structure that provide longest-prefix matching as described in [1] and [20]. The traffic is both legal and illegal, the origin and the destination of which, the type of transport protocol and the combination of ports are different. Packet traces are stored to check the forwarding & filtering decisions. The network load is slowly increased and the performance of the system is observed under different loads. Information on active flows is used to fill the

Bloom-filter state and the stale entries are removed as per the dynamic update mechanism described in [5, 6] and [17].

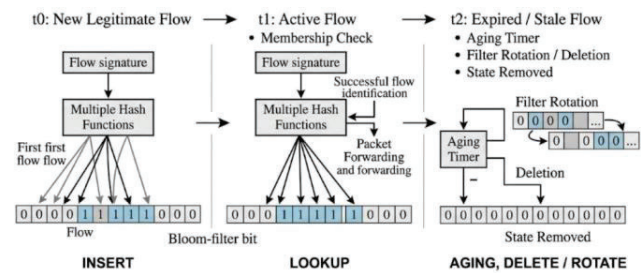


Fig. 4. Dynamic Bloom Filter state insertion, aging, and removal mechanism.

C. Baseline and Proposed Configurations

It is evaluated whether the conventional longest-prefix routing, the combination of longest-prefix routing with a static Bloom-filter security mechanism, and the proposed Patricia-Trie–Dynamic-Bloom-Filter architecture can be evaluated. The Bloom-filter configurations provide probabilistic membership testing [2] and [3] while the routing implementation provides deterministic prefix matching. The principles relating to counting and the dynamic management of Bloom filters are taken into account in order to deal with changing flow states and the need for deletion [7]. Memory usage and lookup performance is compared to the hybrid filtering methods as additional reference [10, 19].

D. Performance Metrics

Experiments are conducted for route lookup latency, packet-processing time, throughput, memory usage, false positive rate, packet-filter accuracy and scalability. Throughput is the number of packets that can be processed successfully in a given period of time and lookup latency is the time required to make the forwarding decision. Memory usage is the amount of memory required for routing and security state. The security evaluation considers the number of valid flows that are properly accepted, and the number of invalid flows that are properly denied. The false positive results are of particular interest because of the natural ability of Bloom filters to make approximate membership decisions [2] [4]. These metrics fall within the range of the measurements used in the previous assessment of routing and programmable firewalls [9, 11, 18].

E. Experimental Procedure

All the configurations share the same routing tables and traffic traces. Firstly, the number of prefixes is changed to measure the routing performance. Then, the traffic intensity is increased and throughput and processing overhead are examined. Next, the number of active-flow state is then tuned to investigate the scalability of the Bloom filter and its false positive property. Finally, there will be both legitimate and illegitimate traffic added to see the effectiveness of the filtering. For each configuration, several experimental runs are carried

out and the average values are given in order to minimise the variation from run to run. The results of the measurements are then compared to those of previously published approaches of Bloom-filter based rule management and routing optimisation [13, 15, 18].

V. RESULTS AND DISCUSSION

Two workloads—controlled IPv4 forwarding and stateful packet-filtering—were used to evaluate the proposed Dynamic Bloom Filter for Intelligent Routing. The results indicate that the programmable forwarding pipeline maintains high throughput at many traffic rates, laying the groundwork for integration of Patricia-Trie route lookup and security filtering dynamically.

A. Forwarding Performance

For traffic rates 1 to 100 Mbps the received throughput measured was very close to the offered traffic rate, with minimal forwarding loss for moderate loads. The throughput for higher loads was approximately 945.21 Mbps with 1 Gbps and 9.56 Gbps with 10 Gbps. As traffic density increases these results demonstrate that the forwarding architecture is still effective.

TABLE I
FORWARDING PERFORMANCE

Offered Rate	Received Throughput	Delivery Ratio
1 Mbps	1.03 Mbps	≈100%
10 Mbps	10.03 Mbps	≈100%
50 Mbps	50.03 Mbps	≈100%
100 Mbps	100.03 Mbps	≈100%
1 Gbps	945.21 Mbps	94.52%
10 Gbps	9557.97 Mbps	95.58%

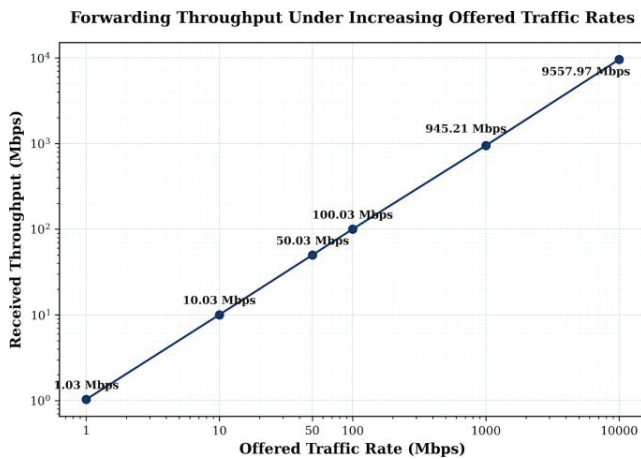


Fig. 5. Forwarding throughput under increasing offered traffic rates.

The transmitted vs. received traffic relationship is almost 1 to 1, which shows good traffic forwarding at lower rates. The reduction in throughput at 1 Gbps and 10 Gbps points to potential processing and simulation limitations at these extreme speeds, not a basic inability of prefix-based forwarding to scale

to higher speeds. This behaviour is similar to that required for optimising memory accesses and look-up structures for high-speed IP routing [1, 3, 20].

B. Comparison with Conventional Forwarding

Programmable forwarding configuration achieved approximately 9.56 Gbps at 10-Gbps offered load while conventional forwarding achieved approximately 9.47 Gbps. Both methods resulted in approximately 945 Mbps at 1 Gbps.

TABLE II
COMPARATIVE THROUGHPUT

Traffic Load	Conventional	Proposed Pipeline
100 Mbps	94.54 Mbps	100.03 Mbps
1 Gbps	945.21 Mbps	945.21 Mbps
10 Gbps	9467.48 Mbps	9557.97 Mbps

The small difference at higher rates suggests that the programmable processing does not have a significant forwarding penalty. Deterministic longest-prefix matching can be achieved through Patricia-Trie lookup; meanwhile, the security component can work concurrently with the forwarding decision [1], [10], [20].

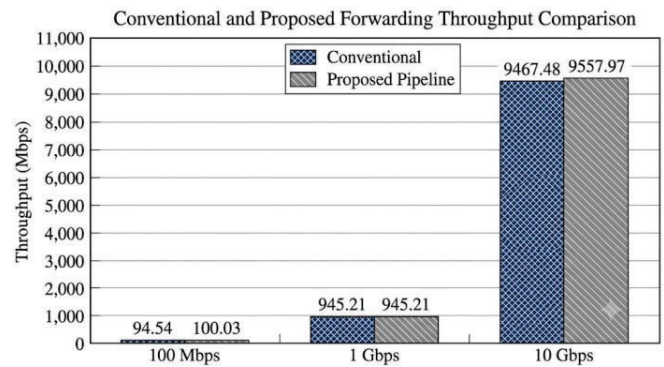


Fig. 6. Conventional and proposed forwarding throughput comparison.

C. Bloom-Filter Security Performance

The filtering experiments with stateful filtering illustrate the use of a Bloom-filter for state tracking, as well as distinguishing between established traffic and traffic which does not conform to the security state maintained. This helps to implement probabilistic membership testing as a lightweight security mechanism [2, 4]. The relevance of the dynamic and counting Bloom-filter principles is due to the fact that enterprise traffic is changing all the time and stale entries need to be removed or aged [5]–[7].

D. Discussion

Overall, the results confirm that deterministic Patricia-Trie routing and compact Dynamic Bloom-Filter security state can be combined. Bloom filters have also been used successfully in packet filtering, SDN rule management, and programmable firewalls as demonstrated in previous research papers [9, 13, 16]–[18]. Existing measurements do not, however, prove

numerical improvement for all components of the final hybrid design, but rather build the baseline for what the forwarding and filtering will be. In future experiments, the latency (time required) for Patricia-Trie, the false-positive rate of the Bloom-filter, the memory consumption, overhead of updates, intrusion detection accuracy, and scalability with growing number of routing prefixes and flow update / active flows should be measured directly. This would constitute a full quantitative justification of the suggested scheme.

VI. CONCLUSION AND FUTURE SCOPE

Hybrid approach, Dynamic Bloom Filter for Intelligent Routing is proposed to improve the routing efficiency and to offer security at the packet level for fast enterprise networks. It does this by combining Patricia-Trie based longest-prefix matching with a dynamically managed Bloom-filter for security, and by being able to distinguish between the two operations of accurately determining a route, and the lightweight membership based filtering of packets. Experimental results reveal that programmable forwarding can deliver high traffic throughputs as the traffic load grows and also enable stateful packet-filtering operations. The approach thus provides a concrete foundation to scaling and adding security, without requiring an alternative forwarding mechanism in its entirety. But there are still important issues to consider, such as bloom filter false positives, dynamic state management, memory overhead and high-rate processing.

Future work could focus on realizing the complete hybrid design and on conducting larger scale experiments leveraging larger IPv4/IPv6 route tables and a more diverse workload of traffic types. The dynamic Bloom-filter mechanism could be enhanced with adaptive resizing, improved choice of the hash function, improved aging policies and improved deletion policies. Future works include investigating the lookup latency and memory consumption of the system as well as the false positive rate, accuracy of ID, and scalability of the system with respect to the changing network environment. The use of hardware-based programmable switches and FPGA platforms would also be interesting, to confirm real-time performance that is better than what can be achieved by simulation. Finally, smart traffic classification and providing machine learning approach to help in tuning the parameters based on the traffic could be implemented to dynamically optimize the routing and security decisions based on the network condition.

REFERENCES

- [1] D. R. Morrison, "PATRICIA—Practical Algorithm To Retrieve Information Coded in Alphanumeric," *Journal of the ACM*, vol. 15, no. 4, pp. 514–534, Oct. 1968, doi: 10.1145/321479.321481.
- [2] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970, doi: 10.1145/362686.362692.
- [3] A. Z. Broder and M. Mitzenmacher, "Using multiple hash functions to improve IP lookups," in *Proc. 20th Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, vol. 3, Anchorage, AK, USA, 2001, pp. 1454–1463, doi: 10.1109/INFCOM.2001.916641.
- [4] A. Z. Broder and M. Mitzenmacher, "Network applications of Bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004, doi: 10.1080/15427951.2004.10129096.
- [5] D. Guo, J. Wu, H. Chen, and X. Luo, "Theory and network applications of dynamic Bloom filters," in *Proc. IEEE INFOCOM 2006*, Barcelona, Spain, 2006, pp. 1–12, doi: 10.1109/INFCOM.2006.325.
- [6] D. Guo, J. Wu, H. Chen, Y. Yuan, and X. Luo, "The dynamic Bloom filters," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 1, pp. 120–133, Jan. 2010, doi: 10.1109/TKDE.2009.57.
- [7] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese, "An improved construction for counting Bloom filters," in *Proc. 14th Annu. Eur. Symp. Algorithms (ESA)*, Zurich, Switzerland, 2006, pp. 684–695, doi: 10.1007/11841036_61.
- [8] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese, "Beyond Bloom filters: From approximate membership checks to approximate state machines," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 315–326, Oct. 2006, doi: 10.1145/1151659.1159950.
- [9] C. E. Rothenberg, C. A. B. Macapuna, M. F. Magalhães, F. L. Verdi, and A. Wiesmaier, "In-packet Bloom filters: Design and networking applications," *Computer Networks*, vol. 55, no. 6, pp. 1364–1378, Apr. 2011, doi: 10.1016/j.comnet.2010.12.005.
- [10] D. Li and P. Chen, "Optimized hash lookup for Bloom filter based packet routing," in *Proc. 16th Int. Conf. Network-Based Information Systems (NBIS)*, Gwangju, South Korea, 2013, pp. 31–37, doi: 10.1109/NBIS.2013.38.
- [11] "C²BF: Cache-based counting Bloom filter for precise matching in network packet processing," *Procedia Engineering*, vol. 29, pp. 3747–3754, 2012.
- [12] S. Nayak, R. Patgiri, and A. Borah, "A survey on the roles of Bloom Filter in implementation of the Named Data Networking," *Computer Networks*, vol. 196, Art. no. 108232, 2021, doi: 10.1016/j.comnet.2021.108232.
- [13] A. Singh, S. Batra, G. S. Aujla, N. Kumar, and L. T. Yang, "BloomStore: Dynamic Bloom-Filter-based secure rule-space management scheme in SDN," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6252–6262, Oct. 2020, doi: 10.1109/TII.2020.2966708.
- [14] N. Sengupta, A. Bagchi, S. Bedathur, and M. Ramanath, "Sampling and reconstruction using Bloom filters," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 7, pp. 1324–1337, Jul. 2018.
- [15] A. Bhattacharya, C. Gudes, A. Bagchi, and S. Bedathur, "New wine in an old bottle: Data-aware hash functions for Bloom filters," *Proceedings of the VLDB Endowment*, vol. 15, no. 9, pp. 1924–1936, 2022, doi: 10.14778/3538598.3538613.
- [16] A. Saxena, R. Muttreja, S. Upadhyay, K. S. Kumar, and U. Venkanna, "P4Filter: A two level defensive mechanism against attacks in SDN using P4," in *Proc. IEEE Int. Conf. Advanced Networks and Telecommunications Systems (ANTS)*, Hyderabad, India, 2021, pp. 113–118, doi: 10.1109/ANTS52808.2021.9937010.
- [17] Q. Wang, M. Pan, S. Wang, R. Doenges, L. Beringer, and A. W. Appel, "Foundational verification of stateful P4 packet processing," in *Proc. 14th Int. Conf. Interactive Theorem Proving (ITP)*, Leibniz Int. Proc. Informatics (LIPIcs), vol. 268, Leibniz-Zentrum für Informatik, 2023, Art. no. 32, pp. 32:1–32:20, doi: 10.4230/LIPIcs.ITP.2023.32.
- [18] M. E. S. ahin and M. Demirci, "ConPoolUBF: Connection pooling and updatable Bloom filter based SYN flood defense in programmable data planes," *Computer Networks*, vol. 231, Art. no. 109802, 2023, doi: 10.1016/j.comnet.2023.109802.
- [19] Y. Zhang, L. Qiao, L. Hu, X. Xu, and Q. Chen, "Cuckoo Bloom hybrid filter: Algorithm and hardware architecture for high performance satellite Internet protocol route lookup," *Applied Sciences*, vol. 13, no. 18, Art. no. 10360, 2023, doi: 10.3390/app131810360.
- [20] Y.-H. Lin and S.-Y. Hsieh, "Improved IP lookup technology for trie-based data structures," *Journal of Computer and System Sciences*, vol. 133, pp. 41–55, 2023, doi: 10.1016/j.jcss.2022.10.003.