ERROR DETECTION AND CORRECTION USING BLOOM FILTERS

P.Spandana[1] **P.Poornima [2]** **Dr. K.Siva Kumara Swamy[3]**
spandana.palley@gmail.com[1] padaraju_poorni@yahoo.com[2]
[1]PG Scholar, Dept of ECE, Sphoorthy Engineering College, Nadargul, Saroor Nagar, Ranga Reddy, Telangana.
[2]Assistant Professor, Dept of ECE, Sphoorthy Engineering College, Nadargul, Saroor Nagar, Ranga Reddy, Telangana.
[3]Associate Professor, HOD, Dept of ECE, Sphoorthy Engineering College, Nadargul, Saroor Nagar, Ranga Reddy, Telangana

Abstract:-The Bloom filter (BF) is a well-known randomized data structure that answers set membership queries with some probability of false positives. In an attempt to solve many of the limitations of current network architectures, some recent proposals rely on including small BFs in packet headers for routing, security, accountability or other purposes that move application states into the packets themselves. In this paper, we consider the design of such in-packet Bloom filters (iBF). Our main contributions are exploring the design space and the evaluation of a series of extensions (1) to increase the practicality and performance of iBFs, (2) to enable false-negative-free element deletion, and (3) to provide security enhancements. In addition to the theoretical estimates, extensive simulations of the multiple design parameters and implementation alternatives validate the usefulness of the extensions, providing for enhanced and novel iBF networking applications.

Index Terms— Bloom filters (BFs), error correction, and soft errors.

I. Introduction

Since the seminal survey work by Broder and Mitzenmacher, the Bloom filter (BF) has increasingly become a fundamental data aggregation component to address performance and scalability issues of very diverse network applications, including overlay networks, data-centric wireless networks , traffic monitoring, and so on. With the caveat of one-sided errors, the use of Bloom filters turns memory and computational expensive operations into simple; resource-friendly set membership problems.

In this work, we focus on the subset of distributed networking applications that use packet-header-size Bloom filters to share some state (i.e. information set S) among network nodes. The specific state carried in the Bloom filter varies from application to application, ranging from secure credentials to IP prefixes and link identifiers, with the shared requirement of a fixed-size packet header data structure to efficiently verify set memberships. The commonality of recent inter-networking proposals is relying on Bloom filters to move application state to the packets themselves in order to alleviate system bottlenecks (e.g. IP multicast, source routing overhead), enable new in-network applications (e.g. security) or stateless protocol designs . We refer to the BF used in this type of applications as an in-packet Bloom filter (iBF). In a way, an iBF follows a reverse approach compared to a traditional standalone BF implementation: iBFs can be issued, queried, and modified by multiple network entities at packet processing time. These specific needs benefit

from additional capabilities like element removals or security enhancements. Moreover, careful design considerations are required to deal with the potential effects of false positives, as every packet header bit counts and the actual performance of the distributed system is a key goal

II.Literature Survey

Rapid advances in next-generation sequencing (NGS) technology have led to exponential increase in the amount of genomic information. However, NGS reads contain far more errors than data from traditional sequencing methods, and downstream genomic analysis results can be improved by correcting the errors. Unfortunately, all the previous error correction methods required a large amount of memory, making it unsuitable to process reads from large genomes with commodity computers.

A Bloom filter is a simple space-efficient randomized data structure for representing a set in order to support membership queries. Bloom filters allow false positives but the space savings often outweigh this drawback when the probability of an error is controlled. Bloom filters have been used in database applications since the 1970s, but only in recent years have they become popular in the networking literature. The aim of this paper is to survey the ways in which Bloom filters have been used and modified in a variety of network problems, with the aim of providing a unified mathematical and practical framework for understanding them and stimulating their use in future applications.

A Bloom Filter is a space-efficient randomized data structure allowing membership queries over sets with certain allowable errors. It is widely used in many applications which take advantage of its ability to compactly represent a set, and filter out effectively any element that does not belong to the set, with small error probability. This report introduces the Spectral Bloom Filter (SBF), an extension of the original Bloom Filter to multi-sets, allowing the filtering of elements whose multiplicities are below a threshold given at query time. Using memory only slightly larger than that of the original Bloom Filter, the SBF supports queries on the multiplicities of individual keys with a guaranteed, small error probability. The SBF also supports insertions and deletions over the data set. We present novel methods for reducing the probability and magnitude of errors. We also present an efficient data structure (the *String-array index*), and algorithms to build it incrementally and maintain it over streaming data, as well as over materialized data with arbitrary insertions and deletions. The SBF does not assume any a priori filtering threshold and effectively and efficiently maintains information over the entire data-set, allowing for ad-hoc queries with arbitrary parameters and enabling a range of new applications. The SBF, and the String-array index data structure are both efficient and fairly easy to implement, which make them a very practical solution to situation in which filtering of a given spectrum are necessary. The methods proposed and the data structure were fully implemented and tested under various conditions, testing their accuracy, memory requirements and speed of execution. Those experiments are reported within this report, as well as analysis of the expected behavior for several common scenarios.

III.Networking applications

iBFs are well suited for applications where one might like to include a list of elements in every packet, but a complete list requires too much space. In these situations, a hash-based lossy representation, like a BF, can dramatically reduce space, maintaining a fixed header size, at the cost of introducing false positives when answering set membership queries. From its original higher layer applications such as

dictionaries, BFs have spanned their application domain down to hardware implementations, becoming a daily aid in network applications (e.g., routing table lookups, DPI, etc.) and future information-oriented networking proposals [12]. As a motivation to our work and to get some practical examples of iBF usages, we first briefly survey a series of networking applications with the common theme of using small BFs carried in packets.

Data path security

The credential-based data path architecture proposes the following network router security feature. During the connection establishment phase, routers authorize a new traffic flow request and issue a set of credentials (aka capabilities) compactly represented as bit positions of a BF. The flow initiator constructs the credentials by including all the router signatures into an iBF. Each router along the path checks on packet arrival for presence of its credentials, i.e., the k bits resulting from hashing the packet 5-tuple IP flow identifier and the routers (secret) identity. Hence, unauthorized traffic and flow security violations can be probabilistically avoided in a stateless, per hop fashion. Using 128 bits only, for typical Internet path lengths, the iBF-based authorization token reduces the probability that attack traffic reaches its destination to a fraction of a percent.

Wireless sensor networks

A typical attack by compromised sensor nodes consists of injecting large quantities of bogus sensing reports, which, if undetected, are forwarded to the data collector(s). The statistical en-route filtering approach proposes a detection method based on an iBF representation of the report generation (collection of keyed message authentications), that is verified probabilistically and dropped en-route in case of incorrectness. The iBF-based solution uses 64 bits only and is able to filter out 70% of the injected bogus reports within 5 hops, and up to 90% within 10 hops along the paths to the data sink.

IP traceback

The packet-marking IP traceback method proposed relies on iBFs to trace an attack back to its approximate source by analyzing a single packet. On packet arrival, routers insert their mark (IP mask) into the iBF, enabling a receiver to reconstruct probabilistically the packet path(s) by testing for iBF presence of neighboring router addresses.

Loop prevention

In Icarus, a small iBF is initialized with 0s and then filled as forwarding elements add their Bloomed interface mask (setting k bits to 1). If the OR operation does not change the iBF, then the packet might be looping and should be dropped. If the Bloom filter changes, the packet is definitely not looping. 2.5. IP multicast revisiting the case of IP multicast, the authors of proposes inserting an iBF above the IP header to represent domain-level paths of multicast packets. After discovering the dissemination tree of a specific multicast group, the source border router searches its inter-domain routing table to find the prefixes of the group members. It then builds an 800-bit shim header by inserting the path labels (ASa: ASb) of the dissemination tree into the iBF. Routers receiving the iBF check for presence of next hop autonomous systems and forward the packet accordingly.

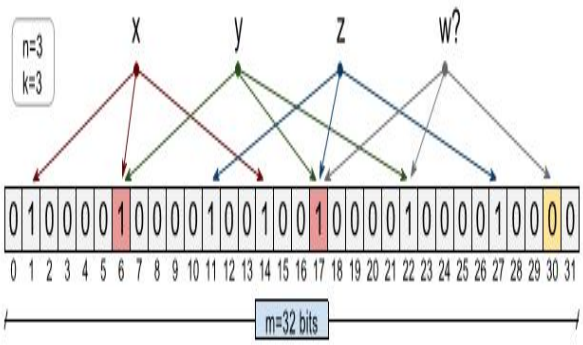
Source routing & multicast

The LIPSIN forwarding fabric leverages the idea of having interface identifiers in BF-form (m-bit Link ID with only k bits set to 1). A routing iBF can be constructed by OR ing the different

Link IDs representing a source route. Forwarding nodes maintain a small Link ID table whose entries are checked for presence in the iBF to take the forwarding decision. In a typical WAN topology and using 256-bit iBFs, multicast trees with around 40 links can be constructed to reach up to 24 users while maintaining the false positive rate (3%) and the resulting forwarding efficiency within reasonable performance levels.

IV.Proposed system

The basic notation of an iBF is equivalent to the standard BF that is an array of length m, number of independent hash functions k, and inserted elements n. On insertion, the element is hashed to k hash values and the corresponding bit positions are set to 1 (see example in Fig. 1). On element check, if any of the bits determined by the hash outputs is 0, we can be sure that the element was not inserted (no false negative property). If all the k bits are set to 1, we have a probabilistic argument to believe that the element was actually inserted. The case of collisions to bits set by other elements causing a non-inserted element to return 'true' is referred to as a false positive. In the example of Fig. 1, a false positive for w would be returned if all three hashes would map to 1s. For the sake of generality, we refer simply to elements as the objects carried in the iBF. Depending on the application, elements may take different forms such as interface names, IP addresses, certificates, and so on. False positives manifest themselves with different harmful effects such as bandwidth waste, security risks, computational overhead, etc. Thus, a system design goal is keeping false positives to a minimum.



A nice property of hash-based data structures is that they do not depend on the form of the inserted elements. Independent of its size or representation, every element carried in the iBF contributes with at most k bits set to 1. In order to meet the line speed requirements of iBF operations, one design recommendation is to have the elements readily in a pre-computed BF-form (m-bit vector with k bits set to 1), avoiding thereby any hashing at packet processing time. Element insertion becomes a simple, parallelizable bitwise OR operation. Analogously, an iBF element check can be performed very efficiently in parallel via fast bitwise AND and COMPARE operations. A BF-ready element name, also commonly referred to as element footprint, can be stored as an bit vector of size m or, for space efficiency, it can take a sparse representation including only the indexes of the k bit positions set to 1. In this case, each element entry requires only klog2m bits.

In this section, we describe three useful extensions to basic in-packet Bloom filter designs in order to address the following practical issues:

(i) Performance: Element Tags exploit the notion of power of choices in combining hashing-based element names to select the best iBF according to some criteria, for instance, less false positives. (ii) Deletion: Deletable regions introduce an additional header to code collision-free zones, enabling thereby safe (false-negative-free)

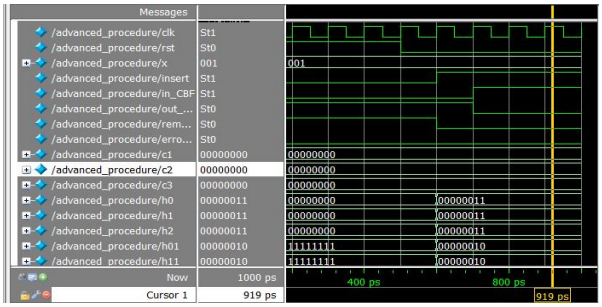
element removals at an affordable packet header bit space.

(iii) Security: Secure constructs use packet-specific information and distributed time-based secrets to provide protection from iBF replay attacks and bit pattern analysis, preventing attackers from misusing iBFs or trying to infer the identities of the inserted elements.

Conclusion

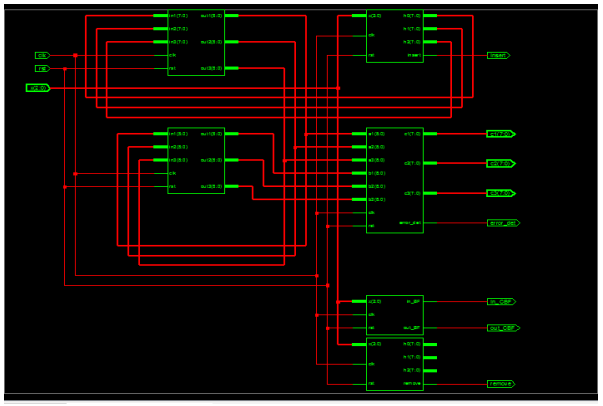
This paper explores an exciting front in the Bloom filter research space, namely the special category of small Bloom filters carried in packet headers. Using iBFs is an appealing approach for networking application designers choosing to move application state to the packets themselves. At the expense of some false positives, fixed-size iBFs are amenable to hardware and present a way for new networking applications. We studied the design space of iBFs in depth and evaluated new ways to enrich iBF-based networking applications without sacrificing the Bloom filter simplicity. First, the power of choices extension shows to be a very powerful and handy technique to deal with the probabilistic nature of hash-based data structures, providing finer control over false positives and enabling compliance to system policies and design optimization goals. Second, the spaceefficient element deletion technique provides an important (probabilistic) capability without the overhead of existing solutions like counting Bloom filters and avoiding the limitations of false-negative-prone alternatives. Third, security extensions were considered to couple iBFs to time and packet contents, providing a method to secure iBFs against tampering and replay attacks. Finally, we validated the extensions in a rich simulation set-up, including useful recommendations for efficient hashing implementations. We hope that this paper motivates the design of more iBF extensions and new networking applications.

Simulation Result:

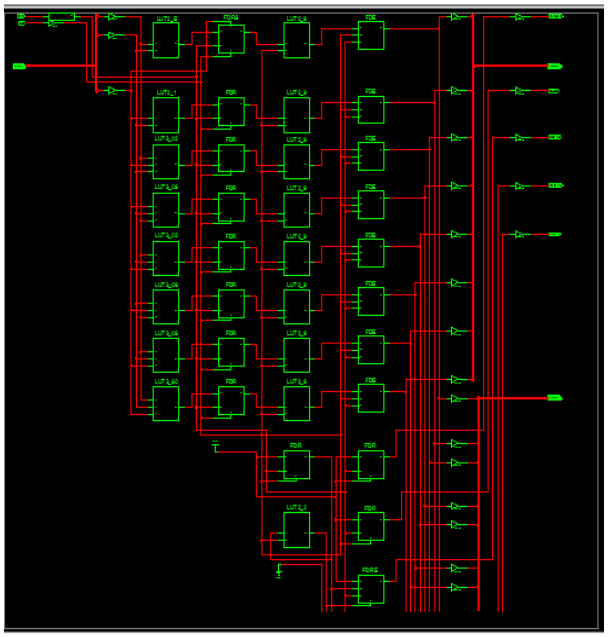


Synthesis Results:

RTL schematic:



Technology Schematic:



Design Summary:

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilizatio
Number of Slices	12	4656	(
Number of Slice Flip Flops	19	9312	(
Number of 4 input LUTs	18	9312	(
Number of bonded IOBs	34	232	14
Number of GCLKs	1	24	4

References

[1] A.Z. Broder, M. Mitzenmacher, Network applications of Bloom filters: A survey, Internet Math. 1 (4) (2003).

[2] B.H. Bloom, Space/time trade-offs in hash coding with allowable errors, Commun. ACM 13 (7) (1970) 422–426.

[3] H. Cai, P. Ge, J. Wang, Applications of Bloom filters in peer-to-peer systems: Issues and questions, in: NAS'08: Proceedings of the 2008 International Conference on Networking, Architecture, and Storage, Washington, DC, USA, 2008, pp. 97–103.

[4] P. Hebden, A. Pearce, Data-centric routing using Bloom filters in wireless sensor networks, in: M. Palaniswami (Ed.), Fourth International Conference on Intelligent Sensing and Information Processing (ICISIP-06), IEEE Press, Bangalore, India, 2006, pp. 72–78.

[5] T. Wolf, Data path credentials for high-performance capabilitiesbased networks, in: ANCS'08: Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ACM, New York, NY, USA, 2008, pp. 129–130.

[6] F. Ye, H. Luo, S. Lu, L. Zhang, S. Member, Statistical en-route filtering of injected false data in sensor networks, in: INFOCOM, 2004, pp. 839–850.

[7] S. Ratnasamy, A. Ermolinskiy, S. Shenker, Revisiting IP multicast, in: Proceedings of ACM SIGCOMM'06, Pisa, Italy, 2006.

[8] P. Jokela, A. Zahemszky, C. Esteve, S. Arianfar, P. Nikander, LIPSIN: line speed publish/subscribe inter-networking, in: Proceedings of ACM SIGCOMM'09, Barcelona, Spain, 2009.

[9] R.P. Laufer, P.B. Velloso, D. d. O. Cunha, I.M. Moraes, M.D.D. Bicudo, M.D.D. Moreira, O.C.M.B. Duarte, Towards stateless single-packet IPtraceback, in: The 32nd IEEE Conference on Local Computer Networks (LCN'07), Washington, DC, USA, 2007, pp. 548–555.

[10] A. Whitaker, D. Wetherall, Forwarding without loops in icarus, in:Proceedings of OPENARCH 2002, 2002.

[11] T. Aura, P. Nikander, Stateless connections, in: ICICS'97, SpringerVerlag, London, UK, 1997, pp. 87–97.

[12] C.E. Rothenberg, F. Verdi, M. Magalhães, Towards a new generation of information-oriented internetworking architectures, in: First Workshop on Re-Architecting the Internet, Madrid, Spain, 2008.

[13] P. Bose, H. Guo, E. Kranakis, A. Maheshwari, P. Morin, J. Morrison, M. Smid, Y. Tang, On the false-positive rate of Bloom filters, Inf. Process. Lett. 108 (4) (2008) 210–213.