ISSN : 2394-2975 (Online)
ISSN : 2394-6814 (Print)
Vol. 2, Issue 1 (Jan. - Mar. 2015)
International Journal of Advanced Research in Education Technology (IJARET)

Survey on Bloom-Filter Based Forwarding and Denial of Service Attack

[I]Roopa Patrimath, [II]Nirmala Y Bariker
[I]M.Tech Student, [II]Assistant Professor
[I,II]Dept.of CSE, Mangalore Institute of Technology and Engg., Mangalore, Karnataka, India

Abstract
This paper presents the Bloom-Filter Based Forwarding. Bloom-Filter is a probabilistic data structure for storing sets. That resolves the current problems in the network such as routing table growth and multicast scalability and Denial of Service attack. there are several protocol has been proposed such as source routing and delivery tree encoded in a packet header, based on the information stored in the packet header as a filter, nodes will forward the packet. But it has a several problems like path selection and security issues. So in order to overcome that path selection algorithm has been used and for security purpose Encryption techniques has been proposed.

Keywords
Bloom-Filter, DoS, Multicast, Network security.

I. Intoduction

In standard IP routing-table lookup[9], when user want to forward the data to any multicast groups or unicast the construction of routing table at each node is necessary and each node has to store the per flow, state information at every node. So it requires larger storage and the computation at each node which leads to the burden on the nodes. But it's a traditional approach which has been used in the network.

When user wants to process packets across the network without storing any state or per flow information at each node, the attractive approach has been proposed that's called Bloom-Filter based Forwarding. In-packet Bloom-filter-based forwarding is relatively simple The delivery tree is stored in the packet header as a set of forwarding-hop identifiers (FHIDs), which can be either nodes, links, or in–out interface pairs on the delivery tree.

The set of FHIDs in the delivery tree is encoded as a Bloom filter [10] data structure, which enables efficient testing of set membership. Network nodes forward packets by checking which potential FHIDs, e.g., in-out interface pair, nodes and links are in the Bloom filter. Bloom-Filter can provide better Scalability compare to the standard IP routing because in this it doesn't store any state or per flow information.

While forwarding the data in the network there is a possibility of denial of service attack, which may leads to the false positive results and that causes the data forwarding to choose different path which has not encoded in the filter by the forwarder. And it causes the data to loop in the network itself. And finally it causes the denial of service attack.

So in order to overcome the Dos Attack there are 3 security approaches has been used in the existing bloom-filter structure.

1) Limiting the number of items stored in the Bloom filters.
2) Centralizing Bloom filter computation and making forwarding-hop identifiers secret.
3) Cryptographically computed per-flow forwarding-hop identifiers.

Perhaps surprisingly nobody until today has thoroughly evaluated the security of these proposals. Bloom-filter-based forwarding needs further improvements on security before deploying on open networks. The security mechanisms Proposed in the existing system do increase the cost of DoS attacks

II. Bloom-Filter

Bloom-Filter is probabilistic data structure for storing sets. The main functions of the bloom-filter are element addition, membership testing but not the element removal. Data elements are set in the filter and filter is implemented as a bit array with fixed length of m and small k hash function is used on the data set and the corresponding k bits in the bit array are set to 1. And membership testing is also done by checking whether the corresponding k bits in bit array. If requested bits are set then it will give true result if not some time it will lead to false positive results.

Bloom-filter is said to be probabilistic data structure because some time it will lead to false positive results when performing membership testing. The rate of false positive grows if the number of *n* bits inside the filter is more.

So in order to avoid false positive results limit the number of bits to be set inside the filter. Because of their space efficiency, Bloom filters are typically used for storing large sets or when the memory usage needs to be minimized. the filters have to fit into network packet headers. Therefore, we consider unusually short filters of 256–1024 bits, which can store roughly 20–100 data items.

Three methods have been proposed for Bloom-filter-based Multicast forwarding:

1) Using Bloom filters in the multicast routers to reduce the space required for storing the multicast forwarding table [9].
2) Source routing where the multicast delivery-tree is encoded as a Bloom filter in the packet headers.
3) Storing the list of receivers in the packet header as a Bloom filter [11,12].

Where in the first approach each outgoing interface of a multicast router has a Bloom-Filter, it encodes the multicast-group addresses which are reachable through that interface.

False positives are acceptable because they only cause the multicast packets to be forwarded to some unnecessary paths.

Where in the second approach source tree or set of route through which data has to be processed is encoded in-packet Bloom-Filter. In this each out going link is assigned with m bit link identifiers and k pseudo-randomly selected bits are set. But the drawback is that function and inputs used to choose the bits vary in different proposals

Where in the third approach the set of receiver through which data has to be processed is set inside the filter and every node will open the filter and it will check whether it has a any other node to forward. Based on the information stored in the filter forwarding decision will be done. But main drawback in this is security and possibility of denial of service attack and it will also give the false positive results.

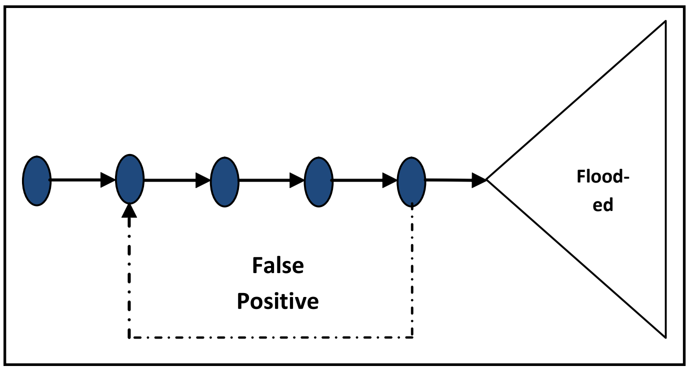

Fig.1: Forwarding Loop

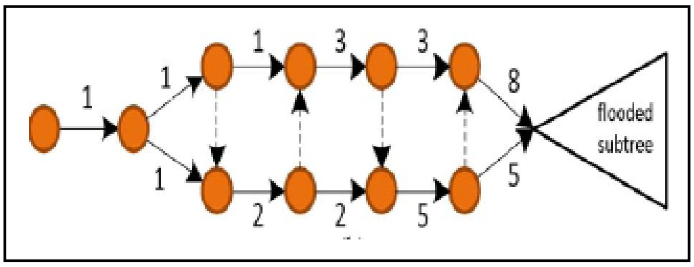

Fig. 2: Repeated flow duplication.

III. Related Work

Literature survey is mainly carried out in order to analyze the background of the current project which helps to find out flaws in the existing system & guides on which unsolved problems we can work out. So, the following topics not only illustrate the background of the project but also uncover the problems and flaws which motivated to propose solutions and work on this project. A variety of research has been done on power aware scheduling. Following section explores different references that discuss about several topics related to power aware scheduling

Revisiting IP Multicast: This paper revisits a much explored topic in networking – the search for a simple yet fully-general multicast design Many years of research into multicast routing have led to a generally pessimistic view that the complexity of multicast routing – and inter-domain multicast routing in particular – can only be overcome by restricting the service model (as in single-source) multicast [1]. This paper proposes a new approach to implementing IP multicast that will leads to a re-evaluation of this commonly held view [1].

Self-Routing Denial-of-Service Resistant Capabilities Using In-packet Bloom Filters: In this paper, it proposes and analyses an in-packet Bloom-filter-based source-routing architecture which is resistant to Distributed Denial-of-Service attacks. This approach is based on forwarding identifiers that act simultaneously as path designators, i.e. define which path the packet should take, and as capabilities, i.e. effectively allowing the forwarding nodes along the path to enforce a security policy where only explicitly authorized packets are forwarded. The compact representation is based on a small Bloom filter whose candidate elements (i.e. link names) are dynamically computed at packet forwarding time using a loosely synchronized time-based shared secret and additional in-packet flow information (e.g., invariant packet contents). The capabilities are thus exploitable and flow-dependent, but do not require any per-flow network state or memory look-ups, preliminary security analysis suggests that the self-routing capabilities can be an effective building block towards DDoS-resistant network architectures [2].

Forwarding anomalies in Bloom filter based multicast: Several recently proposed multicast protocols use in-packet Bloom filters to encode multicast trees. These mechanisms are highly scalable because no per-flow state is required in the routers and because routing decisions can be made efficiently by simply checking for the presence of outbound links in the filter. Yet, the viability of previous approaches is limited by the possibility of forwarding anomalies caused by false positives inherent in Bloom filters. This paper explores such anomalies, namely (1) packets storms, (2) forwarding loops and (3) flow duplication. This paper proposes stateless solution that increases the robustness and the scalability of Bloom filter-based multicast protocols. In particular, it shows that the parameters of the filter need to be varied to guarantee the stability of the packet forwarding, and which presents a bit permutation technique that effectively prevents both accidental and maliciously created anomalies. The solution to avoid such anomalies, the context of Bloom-Cast is proposed, which is a source-specific inter-domain multicast protocol, which uses analytical methods and simulations [3].

In-packet Bloom filter based data center networking with distributed Open Flow controllers: This paper discusses a novel data center architecture based on load-balanced forwarding with in-packet Bloom filters enabled by two support services that distribute the directory and topology state of Open Flow controller applications. By deploying an army of Rack Managers acting as Open Flow controllers, the proposed architecture promises scalability, performance and fault-tolerance. We conjecture that packet forwarding itself may become a cloud internal service implemented by leveraging cloud application best practices such as low-latency, highly-available, key-value storage systems. Moreover, we contribute to demystifying the argument that the centralized controller model of Open Flow networks is prone to a single point of failure and show that direct network controllers can be physically distributed; yielding a “sweet spot” between fully distributed and centralized networking paradigms [4].

Data center networking with in-packet Bloom filters: This paper describes a networking approach for cloud data center architectures based on a novel use of in-packet Bloom filters to encode randomized network paths. In order to meet the scalability, performance, cost and control goals of cloud infrastructures, innovation is called for at many areas of the data center environment, including the underlying switching topology and the packet forwarding paradigms. Motivated by the advent of high-radix, low-cost, commodity switches coupled with a substrate of programmability, our proposal contributes to the body of work re-thinking how to interconnect racks of commodity PCs at large. In this work, we present the design principles and the Open Flow based test-bed implementation of a data center architecture governed by Rack Managers, which are responsible to transparently provide the networking and support functions to cost-efficiently operate the DC network. We evaluate the proposal in terms of state requirements, our claims of false-positive-free forwarding, and the load balancing capabilities [5].

Implementing zFilter based forwarding node on a NetFPGA: In this paper, it describes about the NetFPGA based forwarding node implementation for this new, IP-less, forwarding fabric. The implementation requires removing the traditional IP forwarding implementation, and replacing it with the Bloom-filter matching techniques for making the forwarding decisions. To complete the work, we provide measurement results to verify the forwarding efficiency of the proposed forwarding system and we compare these results to the measurements from the original, IP-based forwarding implementation [6].
Secure in-packet Bloom Filter forwarding on the NetFPGA: In-packet Bloom-Filter allows one to forward source-routed packets with minimal forwarding tables; the Bloom-Filter is encoded with the identities of the links through which the packet needs to be forwarded. If the link identities are made content dependent, e.g. by computing the next-hop candidate link identifiers by applying a cryptographic function over some information carried in the packet header, the Bloom filters differ pseudo-randomly from packet-to-packet, making the forwarding fabric resistant towards unauthorized traffic. In this paper, it describes about the early implementation and testing of an in-packet Bloom filters forwarding node that implements cryptographically computed link identifiers. We have tested two different cryptographic techniques for the link-identity computation and thereby for making the forwarding decision. The algorithms have been implemented and tested on the Stanford NetFPGA. The performance and efficiency of the algorithms is also briefly discussed [7].

IV. Proposed System

Where in the existing system there was a no exact path selection technique which has been used. In the previous work they used cryptographically computed paths. But in this technique every time computation has been done at each node and filter will be keep changing every time. So it's difficult to compute filter every time. In another method user will encode the set of link identifiers as a forwarding paths by performing Logical OR operations. When filter reaches at each node it will perform logical AND operation to check the filter content.
So in this paper am using a shortest path algorithm to compute the filter for data forwarding. By using this technique when user wants to forward the data to intended destination. First it will create the filter of shortest paths and along with that filter message will be transmitted.
In the proposed method the percentage of DoS attack will be reduced by using Encryption technique.

DIJKSTRA'S ALGORITHM: This resolves the single-source shortest-path problem when all edges have non-negative weights. It is a greedy algorithm and similar to Prim's algorithm. Algorithm starts at the source vertex, s, it grows a tree, T, that ultimately spans all vertices reachable from S. Vertices are added to T in order of distance i.e., first S, then the vertex closest to S, then the next closest, and so on. Following implementation assumes that graph G is represented by adjacency lists.

DIJKSTRA's (G, w, s)
```
1. INITIALIZE SINGLE-SOURCE (G, s)
2. S ← { } // S will ultimately contains vertices of final shortest-path weights from s
3. Initialize priority queue Q i.e., Q ← V [G]
4. While priority queue Q is not empty do
5. u ←EXTRACT_MIN(Q) // Pull out new vertex
6. S ←S U {u} // Perform relaxation for each vertex v adjacent to u
7. For each vertex v in Adj[u] do
8. Relax (u, v, w)
```

Data Flow Diagram of proposed system:
A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design). On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

Sender side: At this stage first sender will specify the name of the sender and receiver through which data has to be processed. So first sender will compute the shortest path and that path will be hashed in terms of 0's and 1's that is termed as Bloom-Filter. Then the Bloom-Filter and User data or message is encrypted together and it's forwarded in the network as shown in the figure 3.

Receiver side: At this stage receiver will capture the encrypted packet from the network. And receiver will decrypt that packet to identify which is the next path in the filter. In this receiver can be a intermediate node. If there is a path to forward data then again message and filter will be encrypted and then it is forwarded. or else if current node itself is a receiver then the packet will be processed there itself as shown in the figure 4.

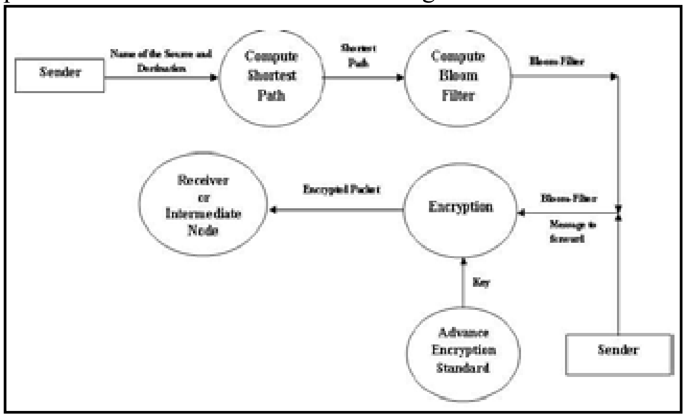

Fig. 3: Sender Side

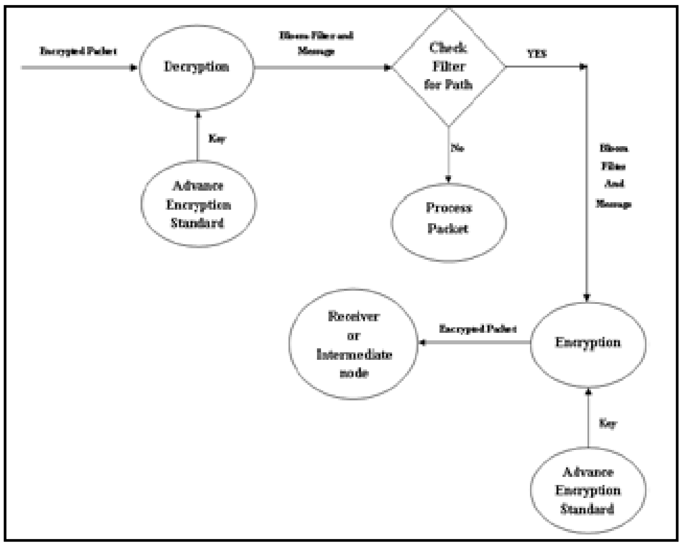

Fig. 4: Receiver Side

V. Conclusion

In this paper, discussion is made on what is Bloom-filter and how the filter will be constructed and which are the possible methods for constructing the bloom-filter and how the Denial of Service attack can be avoided. In the section of related work it will give the brief explanation regarding previous work on Bloom-Filter based Forwarding and DoS attack. In the Proposed system section it will give the brief explanation about algorithm which has been used and how it will protect from denial of service attack.

VI. Acknowledgment

I am very thankful to my guide Ms. Nirmala Y Bariker Assistant Professor, Department of Computer Science and Engineering, Mite for her cordial support, valuable information and guidance, to prepare this paper and also thankful to Prof. Dr. Nagesh H R, Head of the Department, Computer Science and Engineering, for his valuable and constructive suggestions during the planning and development of this work.

References

[1] S. Ratnasamy, A. Ermolinskiy, and S. Shenker, *"Revisiting IP multicast," Comput. Commun. Rev., vol. 36, no. 4, pp. 15–26, 2006*

[2] *C. Rothenberg, P. Jokela, P. Nikander, M. Särelä, and J. Ylitalo, "Self-routing denial-of-service resistant capabilities using in-packet Bloom-Filters," in Proc. Eur. Conf. Comput. Netw. Defense,2009, pp. 46–51.*

[3] *M. Särelä, C. E. Rothenberg, T. Aura, A. Zahemszky, P. Nikander, and J. Ott, "Forwarding anomalies in Bloom-Filter based multicast," in Proc.30th IEEE INFOCOM, 2011, pp. 2399–2407.*

[4] *C. Macapuna, C. Rothenberg, and M. Magalh aes, "In-packet Bloom-Filter based data center networking with distributed OpenFlow controllers," in Proc. IEEE GLOBECOM Workshops, Dec. 2010, pp. 584–588.*

[5] *C. Rothenberg, C. Macapuna, F. Verdi, M. Magalhães, and A. Zahemszky, "Data center networking with in-packet Bloom-Filters," in Proc.28th SBRC, Gramado, Brazil, 2010, pp. 553–566.*

[6] *J. Keinänen, P. Jokela, and K. "Implementing zFilter based forwarding node on a NetFPGA". in Proc. NetFPGA Dev. Workshop, 2009, pp. 1–8.*

[7] *A. Ghani and P. Nikander, "Secure in-packet Bloom -Filter forwarding on the NetFPGA". in Proc. 1st Eur. NetFPGA Dev. Workshop, 2010, pp. 1–7.*

[8] *M.Särelä, "BloomCasting for publish/subscribe networks," Ph.D. dissertation, Aalto Univ., Espoo, Finland 2011.*

[9] *B. Grönvall, "Scalable multicast forwarding," Comput. Commun. Rev.,vol. 32, pp. 68–68, January 2002.*

[10] *B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," Commun. ACM, vol. 13, pp. 422–426, Jul. 1970.*

[11] *X. Tian, Y. Cheng, and B. Liu, "Design of a scalable multicast scheme with an application-network cross-layer approach," IEEE Trans. Multimedia, vol. 11, no. 6, pp. 1160–1169, Oct. 2009.*

[12] *X. Tian, Y. Cheng, and X. Shen, "DOM: A scalable multicast protocol for next-generation Internet," IEEE Netw, vol. 24, no. 4, pp. 45–51, Jul.–Aug. 2010.*

Authors Profile

Roopa Patrimath completed the Bachelor's Degree in Computer Science & Engineering from Visvesvaraya technological University (VTU). Currently pursuing M.Tech degree in Computer Network Engineering at Mangalore Institute of Technology, Mangalore.

Nirmala Y Bariker completed bachelors and masters degree in Computer Science and Engineering. Currently working as Assistant Professor in Mangalore Institute of Technology, Mangalore. She has published 4 research papers in National and International conferences.