Journal of Theoretical and Applied Information Technology
30th September 2015. Vol.79. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SPACE-EFFICIENT AND ACCURATE FORWARDING LOOP DETECTION METHOD USING BLOOM-FILTER FOR FAST AND RELIABLE INTERNET ROUTING

GHADAH ALDABBAGH[1], HALABI HASBULLAH[2], KARAN VERMA[2], OMAIMAH BAMASAK[1]
[1]Dept. of Comp. Science, Faculty of Computing & IT, King Abdulaziz Univ., Saudi Arabia
[2]Dept. of Computer & Information Sciences, Universiti Teknologi PETRONAS, Malaysia
E-mail: galdabbagh@kau.edu.sa[1], halabi@petronas.com.my[2], karan.verma.phd@gmail.com[2], obamasek@kau.edu.my[1]

ABSTRACT

Link or router node failure in a network of Internet is a typical cause of traffic congestion due to the developed forwarding loop at the router. This failure has a significant impact on Internet performance, contributed from the inability of the affected router to find alternative link/route in fast manner and from the high probability of packet dropping during the attempt of re-routing. The existing Internet approach in handling this issue is to use TTL (time-to-live) of TCP/IP, by which a packet will be dropped whenever the TTL timer expires. However, this approach was found inefficient due to long convergence period. Hence, the effort now is to develop a faster re-routing mechanism, by reducing the possibility of forwarding loop incidents for any cases of link/node failures, whilst minimizing packet losses during the convergence period. This work proposes a novel detection method for possible forwarding loop incidents at a router with support of Bloom-filter. Bloom-filter is a probabilistic data structure that helps to ensure the availability of an item in a set, which never lead to false negative results, but may produce false positive results. With this Bloom-filter-based method, link's or node's failure information is attached at the packet header of a packet in a space-efficient manner and to accurately detect for possible incidents of forwarding loop when the packet is traversing through its route from source to destination. If the possibility can be more accurately detected, then packets losses can be minimized to very least during the convergence period and hence, fast and reliable routing shall be achieved. Through simulations, it was found that the proposed method of BF-*k*/2 has outperformed the other re-routing methods. It has not only efficiently used the limited space of the packet header, but also adaptively reducing the false positive probabilities for reliable routing.

Keywords: *Internet, Link/Node Failure, Forwarding Loop, Bloom-Filter.*

1. INTRODUCTION

For many reasons, a link or a router node in the network of Internet may fail at any point of time, leading to network unavailability, temporarily or permanently. The failure of network links or router nodes can disrupt Internet traffic for long periods of time, and leading to severe traffic congestion [1]. Traffic congestion at a router may arise not only from hardware failures and malicious traffics (such as DoS attacks), but also from legitimate usage spikes, e.g. flash crowds. One reason for traffic congestion is due to forwarding loop for a packet at a router, where a number of attempts were being made by a router to find an alternative link (and ultimately an alternative route), when the next outgoing link for the said packet is failed, or the router node itself is failed [17, 21]. If the network failure is caused by a router node, then the previous router node on that link may experience congestion due to the fact that the expected link for a packet toward the failed router node is now cannot be used to forward any packets. In either cases, the result is that a packet intended for a destination will keep looping at the router trying to find an outgoing link, until finally the packet may be discarded after a number of tries, ruled by the TTL (time-to-live) timer. If congestion has happened, the Internet-based service offerings will not be available to users, and thus not reliable for the users to use.

Hence, the general problem faced by the Internet is that it is currently suffering from reliability measures due to congestion, which is referring to the inability of the network to find alternative links/routes in fast manner, leading to packet looping and packet dropping, and thus

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

packets lost during its convergence period. Convergence period is defined as the time taken from the moment when the failure is detected to the moment when the alternative link/route is found [19]. Simply, congestion is the result of packet looping whenever a packet cannot be forwarded through its outgoing link [22, 27]. At the same time, other packets from different routers have arrived at the affected router, and leading to even worst traffic congestion scenario, with some packets sent from a source to an intended destination must be dropped when the TTL timer has expired. Therefore, congestion affects the performance of the router, and ultimately to the overall Internet performance. As a result, from user point of view, the Internet is considered as 'unreliable', as it has failed to provide the demanded services. This may lead to a conclusion that the Internet cannot be fully adopted as a universal communication infrastructure. However, the existing Internet method of providing solution to the traffic congestion problem, and hence the problem of forwarding loop/packet looping, by using the TTL approach is inefficient due to long convergence period and high packet losses.

To overcome the problem of link/node failures, and the subsequent problem of forwarding loop/packet looping at a router, a number of new re-routing protocols were proposed. This allows emergency re-routing of traffic flows in case of failure of link or router node, as depicted in Figure 1, to be performed. This helps traffic to continue reaching its destination from the point where the failure is detected, until the inter-domain routing algorithm re-converges to a solution that bypasses the affected link or router node.

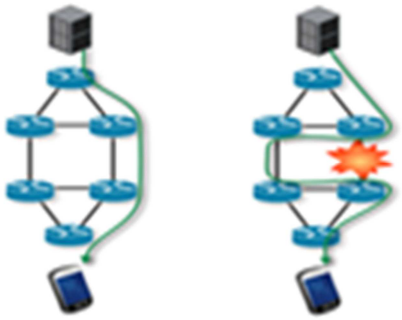

Figure 1: A Re-Routing Technique For Routing Reliability

In this research work, a general solution is proposed that addresses this reliability problem with the anticipation that it will lead to a better performing communication infrastructure of Internet. The proposed approach is based on a layered set of interventions aimed to increase reliability at each level of abstraction: network links, network routes, and application services [2, 18]. With this approach, a packet is expected to be sent and received reliably over the Internet. However, the focus of this work is to seek solution at the network-link level only, which is an inter-domain links, rather than at sources-destinations level. With this focused method, further advancement in the representation and dissemination of link's and node's failure information are needed to be developed. With that motivation, this research objective can be understood as seeking the answer to the following two general questions: how can probabilistic data structure be used to detect forwarding loop/packet looping incidents in a space-efficient and fast manner?; and how can the forwarding loop/packet looping detection results be more accurate?

In summary, from the network-link level abstraction, developing a space-efficient and a highly accurate detection method for incidents of forwarding loop/packet looping is proposed to resolve the problem of traffic congestion at a router node. Based on the review of literature, it was found that there are opportunities to research further on a more efficient and fast method of carrying failure information in the packet header of a packet, while providing routing reliability with the use of Bloom-filter [12, 28]. Based on the identified problems and the capability of Bloom-filter, the following research questions have now been developed:

a) How can failure information be carried in the packet header of a packet in a space-efficient manner, which can be used to detect the possibility of forwarding loop/packet looping incidents at a router node in a fast manner?
b) How can forwarding loop/packet looping incidents be detected as accurate as possible, in order to minimize the convergence period to a very minimal, or to eliminate it if possible?

By solving this reliability problem, Internet can be expected to satisfy the demands of user's requirements, which is getting more stringent day by day.

The general aim of this research work is to design and develop reliable links between nodes of Internet when they are communicating with each

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

other in a task of routing for sending data packets from a source to a destination. In achieving this aim and based on the research questions mentioned above, the followings are the specific objectives of this research work:

a) To exploit the storing and checking capability of Bloom-filter set membership data structure to carry failure information in the limited packet header's storage space in a space-efficient manner, and to detect for possible incidents of forwarding loop/packet looping at a router node at earliest possible.
b) To exploit the probabilistic capability of Bloom-filter to obtain as accurate as possible the detection of forwarding loop/packet looping incidents at a router, so that packet dropping can be reduced, or eliminated, and hence convergence-free re-routing can be achieved.

2. RELATED WORK

Even after decades of development, the Internet still suffers from reliability problems, which is referring to the inability of the network to efficiently find alternative link/route due to congestion without packet lost during its convergence period. However, if the failure recovery is solely left to the routing algorithm only, such as only relying on the TTL timer, then there can be severe data losses during its long convergence period. Hence, there is a serious need to develop new method of fast and reliable re-routing approach, by which the complete information about the network topology for re-routing decision shall no longer be carried along with the packet that traversing the network. Instead, to support applications that cannot cope with this extra delay during the convergence period, researchers have proposed forwarding plane resilience mechanisms for fast and reliable re-routing task. It is implemented at router level with only requiring minimum network resources, and thus lowest possible processing overhead. One of the most significant proposals is [29], which operates by transforming topology uncertainty (due to node and link failures) into traffic volume uncertainty (due to forwarding loop). Then, a re-routing scheme is developed to find efficient alternative link from the affected router that works for a traffic congestion case and with a given set of failure scenarios.

There are ways to explore alternative links in case of link or router node failures with much reduced resource requirements. Work by [30] showed that full failure recovery can be achieved by supplying negligible failure information in packets that are traversing from source to destination and with small extension to forwarding table. This research work also improved the state-of-the-art in forwarding plane full failure recovery by investigating polynomial-time algorithms to decompose the network into an appropriate basis of oriented cycles, which can then be used to implement emergency forwarding tables in routers. However, they have been purely implemented on off-line calculation basis, which the result is late-produced and may be not as accurate as needed.

It is interesting to observe from literature that many network solutions have reduced data processing times and networking costs by exploiting some probabilistic methods. Bloom-filter (BF) is one of the methods, which was conceived by Burton Bloom in 1970 [1]. BF is a probabilistic data structure that helps to ensure the availability of an item in a set. This data structure is used to execute member queries that never leads to false negative, but may be to false positive. An important operation of a BF includes membership testing and adding elements to a set. It is widely used in many applications, like peer-to-peer networking, databases, packet routing, resource allocation, and applications like spell checkers [5, 6]. In a routing task, BF may be used at each edge router node to filter incoming packets and to forward them to the downstream router nodes when meeting a set of specified conditions. When a packet is received by a router, the destination address in the packet is compared to a masked BF at the router, and then forwarded to matching interfaces. To implement this approach, a global level map is required, which was managed by a server in the network. In this approach, faster processing is achieved as the router does not store individual addresses, and hence the complete topology, but only in the masked-BF form [28]. However, the states of the global map must be updated instantaneously with each individual router in the network to reflect the current topology of the network. That is to say, the masked-BF of all the routers in the network must always be in synchronism with the global map.

Bloom-filter $B = (b_0, b_1, \ldots, b_{m-1})$ is an m-bit of array that represents an element set of $S = \{x_1, x_2, \ldots \ldots, x_n\}$. Initially all bits are set to 0, as shown in Figure 2(a). Elements can be added

Journal of Theoretical and Applied Information Technology
30th September 2015. Vol.79. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

into the BF by computing a set of array positions in the BF that are set to 1, as shown in Figure 2(b). The presence of an element is tested by checking if those array positions are set to 1. This means that new elements can always be inserted into a BF, but no elements can be deleted. With this testing capability, false positives are always possible, i.e. a membership test can return positive even if the element has not been added to the BF. However, false negatives are not possible to occur [1, 9].

Each element e is represented with k positions in the array. For example, k separate hash functions can be used to compute k array positions, each hash function giving output [0, m-1]. The element can be encoded as an m-bit long vector, in which the array positions denoted by the k hash values are set to 1, as shown in Figure 2(b). Two hash values in BF can collide, as shown in Figure 2(c), which the collision may indicate that an event has been detected. In relation to this, it is interesting to know of whether or not an incident of forwarding loop or packet looping has occurred at a router node. For example, if a set of excerpt queries is being performed for a number of attempts to find alternative links, but the results are always failed (due to collisions in the BF), this would in turn confirm that forwarding loop or packet looping has actually occurred at the router. As the figure shows, k = 5 hash values are inserted into the BF, but only 3 array positions are marked to 1 without collision, but collision has happened at position 8 in the array.

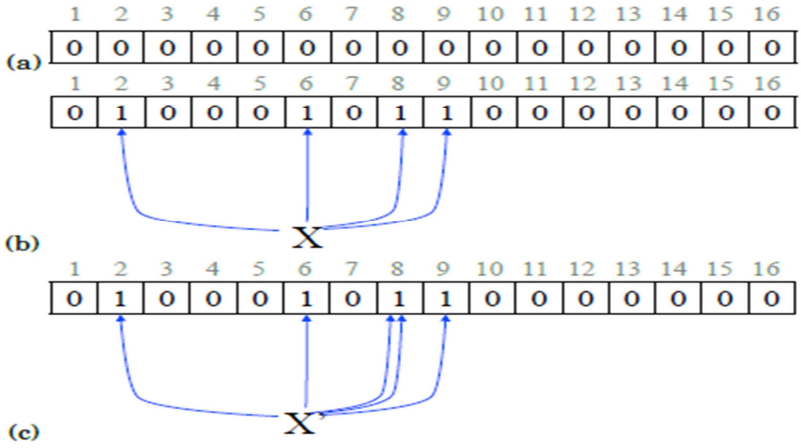

Figure 2: (A) Empty Bloom-Filter (B) X Is Added To Bloom-Filter By Setting The Hash Value Array Positions To 1, K = 4. (C) Shows A Hash Collision With Two Hashes For Element X' Both Yield Position 8, K = 5.

An element can be added onto a BF by bitwise ORing the element's m-bit vector together with the BF. The presence of an element is tested by checking if the k array positions are set to 1. This can be efficiently done with $F \in B : m \wedge e = e$, where m is the Bloom-filter and e the tested element in m-bit long form. Figure 3(a) shows a BF after elements X and Y have been added to it. The membership of an element, such as W, is tested by checking if each array position set to 1 in W is also set to 1 in the BF. The membership testing for W shows that W is not a member in the filter, since the bit in array position 10 is set to 0. When an element has not been added to the BF, but the array positions of the element are set to 1, a false positive happens. As an example, Figure 3(b) shows a false positive. Only two elements, X and Y, have been added to the BF. F is denoted by the array positions {2, 4, 8, 9}, which have all been set to 1 due to X and Y. Hence, membership testing will indicate that F is in the BF, while it has, in fact, not been added.

Following the membership checking capability of the BF, which may lead to false positive results as discussed earlier, then there is question of what is the chance of that false positive results to happen for any checking instances. This can be expressed as false positive probability (FPP), which is the probability that a membership test for an element will return true for an element not added to the BF. Element count of original data set $|S|$, and count of hash functions $|h|$ used to calculate the BF are the factors that determines the false positive probability.

A BF is a simple space-efficient randomized data structure representing a set that supports membership queries. Its space efficiency is achieved at the cost of a small probability of false

Journal of Theoretical and Applied Information Technology
30th September 2015. Vol.79. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

positives. A BF representing set $S = \{x_1, \ldots\ldots, x_n\}$ of n elements is described by an array of m-bits; initially, all are set to 0. It uses a k independent hash function $h_1, \ldots\ldots, h_k$ with a range of $[1-m]$. When a given hash function h_i is applied to input X_i, the result is a value between 1 and m. Since hash functions are uniform, the probability that this result is equal to a particular number b is $1/m$. Therefore, the probability of the bit at position b being 1 after one hash function is $1/m$, and the probability that it is 0 is $(1-1/m)$. The probability that it is 0 after all k hash functions are applied is $\left(1-{}^{1}/_{m}\right)^{k}$. Since there are n elements in the set, the probability that bit b is equal to 0 (after processing all n elements) is $\left(1-{}^{1}/_{m}\right)^{kn}$.

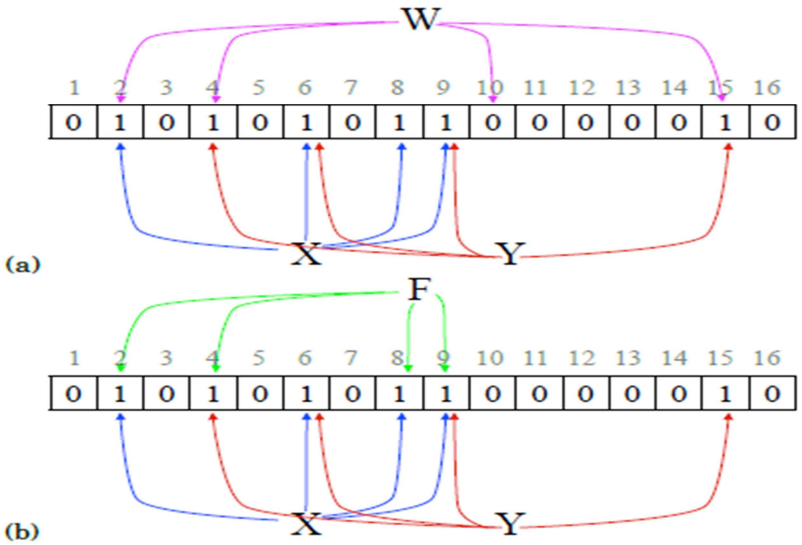

Figure 3: (a) Shows a Bloom-filter to which elements X and Y have been added. The corresponding array positions denoted by the blue and red arrows have been set to 1. The element W is not in the Bloom-filter, since the bit in array position 10 is 0. (b) Shows the same Bloom-filter and element F that has not been added to the Bloom-filter. However, the test for membership indicates that F has been added, since all the corresponding array positions are set to 1. F is a false positive.

Hence, $\left(1-\left(1-{}^{1}/_{m}\right)^{kn}\right)$ is the probability that a given bit b is set to 1 after all input elements $\{x_1, x_2, \ldots\ldots, x_n\}$ are processed as depicted by Figure 3. Since least false positive probability is desired, the probability for an arbitrary input y and corresponding k bits are 1 without y belonging to the set is needed. This false positive probability is:

$$f_p = (1-(1-\tfrac{1}{m})^{kn})^k \quad (1)$$

Similarly, equation (1) can be expressed as;

$$\approx \left(1-e^{\frac{-kn}{m}}\right)^k = exp\left(k\ln\left(1-e^{-kn/m}\right)\right) \quad (2)$$

Finally, equations (1) and (2) are combined,

$$\frac{dy}{dk} = \ln\left(1-e^{-kn/m}\right) + \frac{kn}{m}\frac{e^{-kn/m}}{1-e^{-kn/m}} \quad (3)$$

It can be shown that the expression $(1-e^{\frac{-nk}{m}})^k$ is minimized redundancy when $k = \ln 2 \cdot \left(\frac{m}{n}\right)$, giving a false positive probability f_p of:

$$f_p = (1-e^{\frac{-nk}{m}})^k = \left({}^{1}/_{2}\right)^k \approx (0.6185)^{m/n} \quad (4)$$

From Equation (4), it can be said that a false positive probability f_p depends on k and the ratio k/m. When the probability that some bits are 0, then $(1-{}^{1}/_{m})$. Using the k function and n elements in the BF, there is a need to set bits from m-bits as 1 for kn times. Therefore, after n elements are stored, the probability that the bits are still 0 f_{p_0} can be expressed as:

$$f_{p_0} = \left(1-\tfrac{1}{m}\right)^{kn} \approx e^{-kn/m} \quad (5)$$

and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$f_{p_1} = (kn)(1 - 1/m)(1 - 1/m)^{kn-1} \quad (6)$$

Hence, the probability that k/m bits is in collision free region is given by $(1 - f_{p_n})$.

With this derivation, it is said that Bloom-filter that used *k* hash function, i.e. (*n*+1)*k* is called BF-*k*. The hash addresses of the elements in the set and one element that being looked up should be independent to each other. Filtration is essential for the operation of Internet networks. Most of the well-known approaches to filtration for Internet focus on achieving high filtration accuracy for the entire network without addressing the detection of forwarding loop incidents. The BF of the bit vector representing an element is constructed by applying a fixed number of hash functions to the element. In other words, all of the bit-vectors that represent elements are constructed using the same number of hash functions. Varying the number of used hash functions has been proposed in the context of BF forwarding [12, 17]. These proposals are motivated by varying false positive rate in network nodes of different degrees: if the same values of are used at a low degree node and a high degree node, the false positive forwarding rate of the high degree node is greater.

Now, BF with its hash functions has been extended to detect the possible forwarding loop/packet looping incidents at a router node. BF has been proposed and used to construct a hash table that records TCP/UDP data flows with limited storage cost at the packet header of a packet.

Therefore, there is a chance to solve the issue of traffic congestion (caused by forwarding loop and packet looping) at a router node, by exploiting the limited storage space of the packet header of a packet that traverses over the network and passes through the router to its intended destination. With the use of BF and its variant, and the existence of a global map of the network, this is achievable where only encoded entry/access checking is required to detect the present of forwarding loop/packet looping incidents. Additionally, the detection accuracy for any incident can be improved when the false positive probability of BF is enhanced, which shall lead to a fewer packet losses for any traffic congestion scenario. All these are the aims in providing a fast and reliable Internet infrastructure.

3. METHODOLOGY

It is important to highlight that in this work, the focus of investigation will be on two points related to routing capability: 1) how to make use of the limited storage space in the packet header in carrying failure information of the links and router nodes, and 2) how to accurately detect for possible incidents of forwarding loop/packet looping at a router node. Therefore, it is not so much on the routing task itself; instead, they are the needed supporting components for fast and reliable routing.

The specific method in detecting forwarding loop/packet looping incidents at a router node is by attaching failure information at the packet header of a packet that is traversing from a source node to a destination node in the network of Internet, with the help of Bloom-filter. Additionally, the BF will also be used to accurately detecting these incidents, such that a lower packets dropping is achieved. As a platform for improvement, a previous work by [28] will be used as the basis for developing a convergence-free forwarding/routing scheme.

Based on this, the improvement will be made on the carrying capacity of the packet header to carry a 'representative' of the failure information in a space-efficient manner. The failure information will consist of two basic facts: 1) probability of forwarding loop/packet looping incidents at a router, and 2) false positive probability that each link from that affected router is associated with. However, the complete information of these two is only available in the global map, which is managed by an appointed server in the network, and not at each the individual router. All the routers in the network will compute locally these probability values, and updates synchronously with the global map, together with the latest state of the network topology. Each router node will have a masked BF, which the BF will only use encoded-bits to check the existence and status of the information in on-line fashion, thus providing fast detection mechanism with least overhead. Simply, with BF and its encoded entry/access to the complete information, the detection of forwarding loop/packet looping can be made faster and space efficient. This is an extension from that of previous works, in which the earlier only carries information about the failed links that the packet has traversed so far, and full failure information is being carried all the way in the packet. The extension is also due to the added value for the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

stored information, which they are now in statistical forms. Additionally, previously off-line computation was performed, leading to processing overhead at a router and delay in routing, while the proposed method will be able to perform on-line computation. Furthermore, accuracy in detecting any incidents of forwarding loop or packet looping shall lead to better performing Internet by reducing packets losses during its convergence period. The BF capability is again will be used to decrease the probability of false positive. The ultimate goal is to provide a convergence-free re-routing mechanism, thus reliable Internet services are obtained.

The only difference in the BF-k is that the incorrect deletion of a false positive item is undetectable, while the incorrect deletion of a multi-address item is detectable in advance. The problem is that when performing lookup for an element, this only has the element identifier and will have to try all set IDs to see if any of them is encoded in the filter. The number of different set IDs is in thousands, causing huge lookup overhead. So that to overcome such space cases of the hashing schemes, where the addressing spaces will be halved and then abolished is called BF-*k*/2.The space advantages are more difficult to sum up; again it depends on the error rate that is to be tolerated. It also depends on the potential range of the elements to be inserted; if it is very limited, a deterministic bit vector can do better. If the number of elements to be inserted cannot be estimated, it is better to use hash table or a scalable Bloom filter.

The proper management of Internet is a challenging task due to the mobility of nodes and their velocity as all devices work on open channels. These are very challenging security tasks since all of these characteristics markedly increase the possibility of threats and attacks [10]. There are three challenges for space efficient and detecting the attack traffic. First, accurate rules are needed to distinguish the attack model from the legitimate so that legitimate can still reach the victim while the attack is being space efficient functioned. Second, the attack model should use an intrinsic feature of the attack; otherwise the HBF mechanism will be evaded by a simple change of attack signature. Third, the space efficient rule must be simple (not complicated) and the HBF process should be computationally efficient, otherwise the HBF process will not be effective. Consequently, most detection schemes developed for Internet assume high numbers of available nodes within the network that act as intermediates [8]. The basic assumption for all schemes is that there is a limited number of attack paths, and not all legitimate shares a path with the attack. Without confidence in accurately differentiating attack from legitimate at a single location; all schemes try to detect attack paths based on space and HBF detection mechanisms.

A BF-*k*/2 is a simple space-efficient randomized data structure representing a set that supports membership queries. Its space efficiency is achieved at the cost of a small probability of false positives. A BF representing set $S = \{x_1, \ldots\ldots, x_n\}$ of n elements is described by an array of m-bits; initially, all are set to 0. It uses a k independent hash function $h_1, \ldots\ldots, h_k$ with a range of $[1–m]$. Here, it is assumed that hash functions are perfectly random (Golle et al. 2001). For each element $x \in S$, bits $h_i(x)$ are set to 1 for $(1 \leq i \leq k)$; checking to see equation 6.

It has been found that the value of k that minimizes the false positive probability of a single BF-k (i.e.$(1 - e^{\frac{-nk}{m}})^k$) also minimizes $\mathrm{BF} - \mathrm{k}/2$ approximately up to five decimal places based on our empirical results. Hence, the number of hash functions is set to $(1 - e^{\frac{-nk}{m}})^k$ in the BF-k/2 scheme, and the $BF - k/2 = \left(1/2\right)^k \approx (0.6185)^{m/n}$. The value of $\mathrm{BF} - \mathrm{k}/2$ as the ratio of $\frac{m}{n}$ varies from 1 to 10. It can be shown that when $\frac{m}{n} = 5$, it is about 0.16. When $\frac{m}{n} = 10$ drops to 0.016 only. (Note that when $\frac{m}{n} = 5$ and when $\frac{m}{n} = 10$, the false positive rates of BFICK are 0.39 and 0.15 respectively). If the nodes independently chose the k, each node could set the value so high that the average number of false positive is less than the one and packet storms are avoided. If internet router use globally decided values for generating forwarding hop identifies, the BF-k/2 forwarding scheme uses global *k*.

3.1 Improving space efficiency

It is proposed in this work a method to improve the storage capacity limitation of the packet header with the help of Bloom-filter. BF is a compact data structure for high-speed on-line membership checking against large data sets. Some BF variants can be used to improve the space efficiency issue [2]. However, it has been

Journal of Theoretical and Applied Information Technology
30th September 2015. Vol.79. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

proven that BF is not space optimal [3]. Therefore, the aim for space efficiency is that it should reduce the number of bits it takes to encode (represent) each member and its set ID, particularly in checking for the failure information. This is extremely important if the data structures are placed in on-die static random-access memory (SRAM). It should reduce the number of memory entries/accesses onto the SRAM on a per-packet basis in a router. Additionally, possibly the idea for space efficiency can be implemented with the use of large and sparse BF at the sender/receiver, and compress/decompress the filter before/after transmission.

To reduce the number of bits needed per entry/access, it should encode each entry/access just once, capturing both the member identifier and the set ID, i.e. (*e*, *Se*). However, the problem is that when performing a lookup for an element, this only has the element identifier and will have to try all set IDs to see if any of them is encoded in the filter. The number of different set IDs is in thousands, causing huge lookup overhead. The target is to create indirection in the lookup process, by separating the membership encoding and the set ID storage in two data structures, called the *index-encoder* and *set-id table* (abbreviated as *SID-table*), respectively. In the *index-encoder*, it encodes the membership of a member, as well as a small index that points out where to find the right set ID in the *SID-table*. This index may take a few different values (e.g., from 1 to 10).

The lookup process consists of two steps: given a member identifier, the first step tests whether the member is a member and checks few index values (instead of the set IDs in the thousands) to see which one is encoded in the *index-encoder*. Using the right index, the second step finds out where to fetch the set ID from the *SID-table*. In order to support efficient lookup, it encodes the primary index a in the *index-encoder* by two steps: 1) it hashes the member identifier e to a number g of blocks in the *index-encoder*, where g may be one or a small integer. It will then fetch these blocks to the processor. They can be logically thought of as a small BF, denoted as Ce, now residing in the processor for encoding a; 2) it hashes a (together with e) to k' bits in Ce and sets them to 1. The sequence in executing the encoding processes is shown as the following, and is depicted in Figure 4 and Figure 5.

i. It performs hash operations on the member's ID e and obtains a sequence of hash bits.
ii. Using the hash bits, it finds k candidate entries from the *SID-table*, and stores the set ID *Se* of e with a check-sum computed from e to one of the entries; the index of the entry is a.
iii. Using the obtained hash bits, g blocks are fetched from the *index-encoder*, which form a virtual Bloom-like filter Ce.
iv. Encodes (e, a) to Ce.

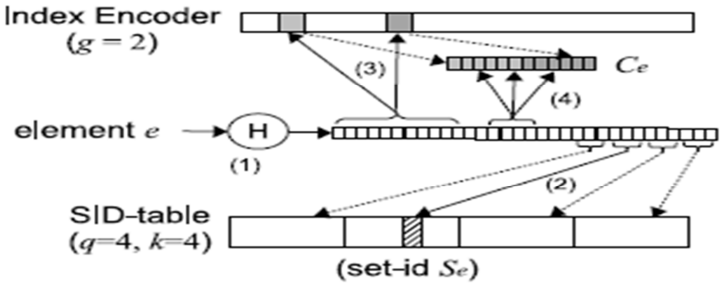

Figure 4: Insertion Member To Multi Set Membership Function

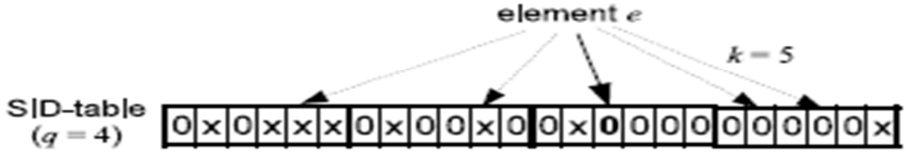

Figure 5: Candidate-To-Right Policy To Insert A Member To A SID-Table. An Entry Marked With X(0) Means It is used.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The effect of the above candidate-to-right policy (in one place only store only one bits, mean corresponding bits are checked, if all bits are equal to 0 or 1 then it can be said that the element belongs to the set, then a false positive is returned. It also avoids collision within space efficient data structure) may be amplified by allowing more than one value in one position. As shown in Figure 6, it follows the steps to determine the set ID. First, it generates a sequence of hash bits using element e. Using the hash bits, it locates and fetches g blocks in the *index-encoder*, i.e. to the processor. The processor has k units that test, in parallel, whether any candidate index i is encoded, for $1 \leq i \leq k$. Each unit hashes i to the k' bits in Ce and checks whether all the bits are 1s. If so, i is encoded in Ce.

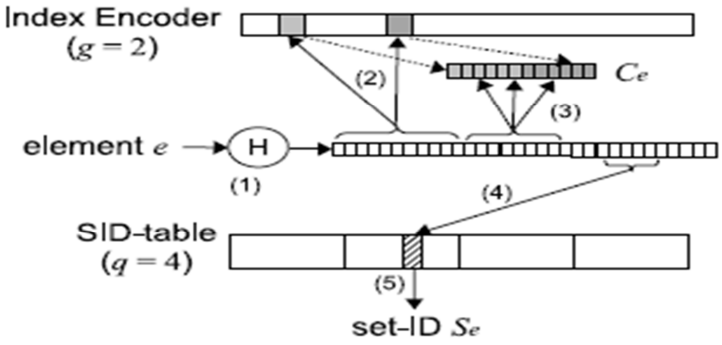

Figure 6: Looking Up A Member

An important factor from using this highly space-efficient (reduced) data structures scheme is that the size of the filter does not linearly dependent on the number of elements inserted. Once the BF is sufficiently saturated, it can save a copy of the filter and start afresh with a new one. Using this technique, the limited storage space at the packet header is expected to efficiently store the least requested information, but has the capability to compute the complete information at a router node in on-line fashion, leading to fast decisions being made at a router.

Hence, BF-*k* scheme needs improvement packet header overhead. We can reduce the two independent hash function requirement of the double hashing technique to a single hash computation based on e.g., CRC32 or BOB. This result can be applied to BF-k/2 networking applications with on-line element hashing instead of pre-computed element names. Moreover, the hash segmentation technique may be useful in other multiple-hashing-based data structures (e.g., d-left hash tables) that require hashing on a packet basis. The main idea of is to reduce the number *1*s by choosing the "best" set of hash functions. Besides our in-packet header scope, our approach differs in that we include the information of which group of hash functions was used (d value) in the packet itself, avoiding thereby the caveat of checking multiple sets. As for routing, when a packet is received, the destination address in the packet is compared to each BF on the router and forwarded to matching interfaces. This mechanism does not store any individual addresses in the router. There are two main ideas here: concentrate the k bits in one (or only few) cache blocks and precompute random bit patterns in order to save both hash bits and access time. While these Bloom-filter variants improve execution time at the cost of slightly increased FPR, the filters save space by engineering practical variants of the theoretically space optimal Bloom-filter replacements.

3.2 Improving detection accuracy for the Forwarding Loop incidents

A false positive of BF means that the BF predicts an element to be present even though it is not present in the set. The probability of such an error is called the False Positive Probability (FPP), and it is relatively small. On the other hand, a false negative means that the BF predicts an element to be not present even though it exists in the BF. Such False Negative Probability (FNP) is not present in the BF. As more numbers of elements are inserted into the BF, the bit vector saturates, and once that happens, the FPP increases. The saturated BF need to be preserved and a new empty BF must be created. Therefore, there are

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

two possible ways to reduce the increase of FPP: to decrease the number of entries/accesses (element insertions) used for membership checking, and to develop more efficient BF preserving method.

There is a need to reduce FPP to achieve accuracy in detecting forwarding loop/packet looping incidents, and hence to reduce to very minimal the dropped packets during convergence period. Hierarchical Bloom-filter (HBF) is proposed in this work, which is a method of BF preservation when a BF is saturated as the result of higher number of elements being inserted into BF for checking. With HBF, all the blocks present in the single packet are inserted into the BF in the form of a hierarchy. At level 1, the HBF behaves similar to the Block Bloom-filter (BBF), where initially the packet payload is split into blocks of equal size, and they are inserted into the BF along with their offset values. At level 2, adjacent blocks are concatenated forming a super block of double the size, and inserted into the same BF. The same process is followed for further higher levels, till all the blocks are inserted as one block.

Packet data is split into multiple blocks and each block is appended with 'Interface Id' from which packet arrived. After splitting the blocks from the packet payload, the first block is concatenated with offset value of 0, second block with offset value of 1. Then the modified blocks are passed through hash functions and stored in the BF. Hence, actual block data being stored into the block of BF would be appended with inbound interface (IP Address or MAC) or unique *interface-id* given to each interface. All combinations of offset are tried with initial block and incrementing offset is attached to other blocks in the sequence. If all the blocks in sequence are present in the BF, which may lead to collision, it would validate the existence of excerpt query. This would in turn confirm the forwarding loop or packet looping has happened at the router.

By having multiple hierarchical levels of the insertion of blocks into the BF, it will increase the querying accuracy and reduce collisions (reducing packets dropped). Even better, the querying accuracy increases as the HBF checks the offset values using only the set ID, which implies reduced collisions, as if there is any duplicate request came for the same set ID, HBF neglects the request. By identifying a number of routers, and if all the routers in the network tree process an excerpt query at the same time, this can also lead to packet drops, and affecting the network bandwidth. Hence, it would be better to minimize packet losses by an alternative packet attribution method.

The existing packet attribution process can be modified to maintain multiple BF (at the same time), one for each interface (separately). An extra attribute called *Interface Address* would be added to the existing BF data structure to identify the BFs that are associated with different interfaces. In this way, the modified attribution process initializes multiple BF at a time, one for each interface. Packets are segregated based on the interface, and packet payload can be stored into respective BF based on interface they arrived, which then determine the next router. This is depicted in Figure 7. Each block is concatenated with the corresponding offset number, and then inserted into the BF.

Block = *Content* || *Inbound Interface-id* || *Offset*,

where *Content* = *Block content* and *Offset* = 0 < (*Packet length* / Block size)

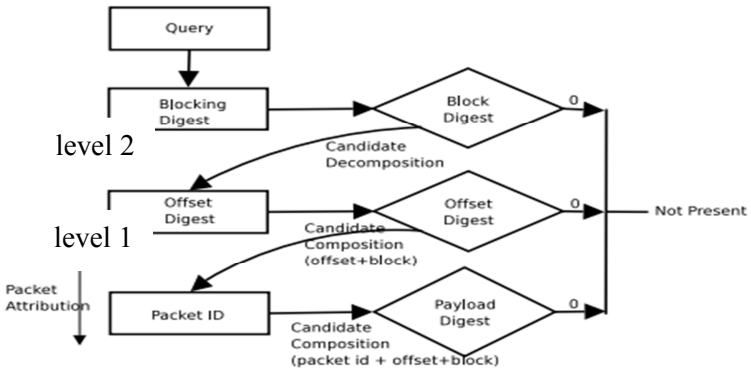

Figure 7: Hierarchical Bloom-Filter With Its Attribution Process

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

How BF-*k*/2 improve detection accuracy for incidents of forwarding loop/packet looping probabilistically filters a certain percentage of the received messages based on its computing capacity, and then reports all the invalid messages detected. HBF detection technique first extracts the IP addresses of the incoming network traffic. It then determines whether the source IP address has been seen previously or is a new IP address. The resulting time series of the rate of previously unseen or new IP addresses is then analyzed by the levels to identify whether the system is under attack. The system being protected against DoS attacks is represented in terms of two independent states: not under attack, which is the system state when receiving non-attack normal traffic, and under attack, the system state when receiving DoS attack traffic. Low-rate attacks can be as harmful as the high-rate ones, yet even more dangerous due to the fact that they are difficult for routers and counter-DoS mechanisms to detect. Where the BF is used to keep track of the set of nodes visited. Each node has a corresponding mask that can be ORed into the BF as it passes; if the filter does not change, there may be a loop. False positives may lead to packets incorrectly being dropped because of an assumed loop. The authors discuss ways to limits the negative effects of false positives in this context.

Bloom filters can yield an acceptably low rate of false positive drops if we use failures and reprieves. We demonstrate that Bloom filters still retain good loop detection accuracy, especially for small loops. The results in this section were obtained analytically, using the simulation data derived in the previous section. Again, we do not consider network effects such as packet loss. Therefore, these results represent an upper bound on the number of redundant packets allowed by a particular loop detection technique. To simplify the analysis, we only consider multicast packets. For a loop of size k, this implies that TTL's allow for d-k redundant packets or d=k -1 redundant packets per link:

$$Redundant = d/k - 1 \quad (1)$$

For the Bloom filter mechanism, packets are dropped once the failures and reprieves are exhausted. In the absence of reprieves, each failure allows a looping packet to traverse an extra loop of k hops. Therefore, the number of redundant packets. To minimize the overhead of loop detection, system comprises a set of containment layers that are tailored to the needs of different kinds of protocols. Restricted protocols use the routes of an underlying multicast routing protocol such as OSPF rather than compute their own. Packets are dropped once the failures and reprieves are exhausted. In the absence of reprieves, each failure allows a looping packet to traverse an extra loop of k hops. Therefore, the number of redundant packets per link is simply the number of failures. The packet's Bloom filter is bitwise ORed with the interface's Bloom mask. If the Bloom filter does not change, then the packet might be looping. Possible responses to failing the Bloom test include dropping the packet, and using failures and reprieves to deter false positives. We can reduce the Bloom filter space requirement if we are willing to forego perfect detection accuracy. The basic idea is to permit a small number of Bloom collisions before dropping a packet, there by trading off detection accuracy for reduced false positives

4. RESULTS ANALYSIS AND DISCUSSION

The main outcomes to be analyzed and discussed from this results section are the ability of the BF to efficiently carry failure information in the limited space of packet header and to accurately determine the false positive probability for reliability in term of packets dropping during the convergence period. To achieve this, the proposed method is compared against the established available methods, such as TTL (time-to-live) and FCP (failure-carrying packet), through a set of simulations. Also, the performance of the proposed method is validated with a range of practical and randomly generated topologies. Table I defined the simulation parameters.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table I: Simulation parameters

Parameter	Typical Value
No. of nodes	20, 50
Node speeds	10, 20, 30 ms
Simulation time	400 sec
Environment size	1500 x 800 meter
Packet sizes	1.0 MB, 2.0MB
Data transmission size	1400 bytes
Packet type	TCP/UDP
Antenna model	Omni-directional Antenna
Traffic type	CBR
Visualization tool	NAM

4.1 Improvement in space efficiency of the packet header

In this section, efficient use of the limited packet header storage space is evaluated by comparing the performance of the proposed BF-*k*/2 with other contemporary methods. The main aim is to provide proofs that the BF-*k*/2 proposed solution has significantly improved the usage of the storage capacity of the packet header, and hence supporting the fast Internet re-routing mechanism.

a) Packet attribution

Through simulation, the performance of HBF-with packet attribution is compared against Block-BF and Hierarchical-BF with respect to their storage size, compression ratio, and processing time. As shown in Table II, the proposed HBF-with packet attribution approach requires less storage space, while achieving higher compression ratio and faster processing time as compared to the other two solutions. Lesser processing time implies that the convergence period is shorter. In overall, the proposed HBF-with packet attribution approach has improved the space efficiency in carrying the failure information at the packet header, and has also providing faster processing time for any incidents of forwarding loop/looping packet incidents at a router.

Table II: Storage size, compression ratio, and processing time for different BF approaches

No.	BF approaches	Storage size (KB)	Compression ratio	Processing time (ms)
1	Block Bloom-filter	10,978.8	72.635	48,018
2	Hierarchical Bloom-filter	10,977.7	72.643	21,061
3	Hierarchical Bloom-filter with packet attribution	6,123.7	130.222	21,059

b) Memory access counts

The number of memory accesses to SRAM at a router node to forward packets in an attempt to find alternative links is having impact on the limited storage space of the packet header. A lower count is required as it will reflect minimum forwarding efforts for finding alternative links. In Figure 9, variation in memory access counts with the increase in load factor (a scenario of traffic congestion) is observed. A general observation indicated that as the number of traffic load (due to congestion) increases at a router node, the required memory access counts is also increases because now more attempts are needed to search for alternative links during its convergence period. The previous solutions of TTL, FCP (failure-carrying packets), and BF-*k* (Bloom-filter with *k* value) for convergence period showed that the memory access counts increases exponentially with the increase in load factor. Simply, all these

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

methods are not effective in handling congestion, especially at earlier stage, where sharp increases in memory accesses are experienced, and maintaining that high memory accesses throughout the convergence period. However, with the proposed solution of BF-*k*/2, the memory access counts are significantly much reduced. Importantly, it is steadily maintaining a consistence lowest memory access counts regardless of the traffic loads being experienced. It showed that the limited storage capacity of the packet header has been successfully used to carry the required failure information (in encoded forms), and the proposed method of BF-*k*/2 has been able to exploit powerfully the memory access to encode its bits representation to check for the required failure information from the global map via the local router. With this improvement, it can detect incidents of forwarding loop/packet looping in a fast and efficient manner. Ultimately, it implies that faster or shorter convergence period is achieved.

With the significant performance improvement made by BF-*k*/2 method as compared to the others, then it is real possible that a convergence-free re-routing can be obtained at the affected router. As can be seen from Figure 8, regardless of the traffic loads, the memory access is maintaining a lower count. This indicates that the number of trials in the attempts to find alternative link is keep low, possibly with only one time attempt, and hence convergence free period. As the router must encounter situation of traffic congestion at any point of times, the BF-*k*/2 is readily available to serve the re-routing requirement in a convergence-free manner.

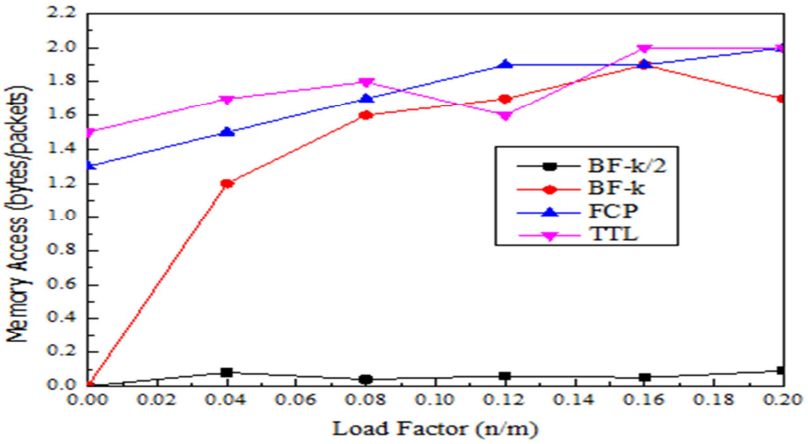

Figure 8: Memory Access Performance For N = 25000 And M = 64.

c) Packet header overhead

It is important to see the overhead being experienced by the packet header in handling a convergence period in a traffic congestion scenario. Figure 9 showed the performance of TTL, FCP, BF-*k*, and BF-*k*/2 with respect to their packet header overhead over a fixed convergence time period, which was set to 10000 milliseconds. It is observed that BF-*k*/2 has experienced an almost consistent packet header overhead throughout the convergence period, while the others have experienced a heavier packet header overhead. At time 0 msec., each of the method is consuming an overhead, in which BF-*k*/2 has consumed the least overhead of about 4 bytes/packet/second. It then starts to decrease sharply until 1000 msec. for each of the method. It follows that each method decreases steadily until the set time of 10000 msec. However, BF-*k*/2 has shown a flatter decrease as compared to the others. It means that BF-*k*/2 offered much less packet header overhead, and thus, a faster convergence period. Importantly, BF-*k*/2 has efficiently used the limited storage space to carry the full failure information at the header of the packet. The maximum header size during the simulation run is 8 bytes, assuming each failure header takes 2 bytes. It is important to note that the failure information is only inserted into the header when link or router node failure is detected.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

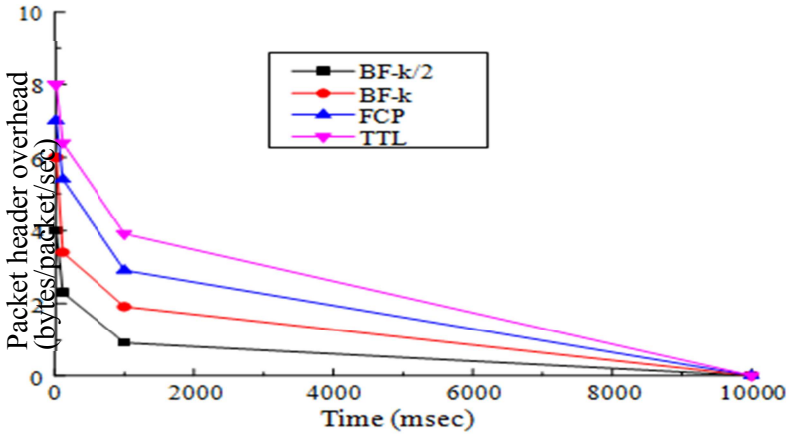

Figure 9: The Effect Of Packet Header Overhead On Space Efficiency

d) Packet loss rate

It can be expected that any re-routing schemes will experience packet losses during their convergence period. The lower packet losses implied that better reliability of the re-routing scheme. Figure 10 showed the packet lost rate against the number of failures per second for any failed links or failed router nodes. It can be observed that the lost rate for BF-*k*/2 has increased to 0.009 packet/sec as compared to BF-*k*, FCP and TTL schemes (which are 0.026 packet/sec, 0.028 packet/sec, 0.029 packet/sec, respectively) at the early stage when the number of failures is small. These early stage packet losses happened due to the fact that this is the first time that a forwarding loop or packet looping incident is detected, and when the packets cannot be forwarded to its alternative links, then it may be dropped. However, after that early stage packet losses and when the number of failures getting higher, the lost rate from BF-*k*/2 scheme is significantly reduced regardless of the number of failures recorded at that point of time, while the others have increased their lost rates as the number of failures increase. It is also observed that the packet lost rate for BF-*k*/2 reducing steadily over the number of failures, while the other schemes of TTL, FCP and BF-*k* have shown significant increases over the number of failures. Hence, it can be concluded that BF-*k*/2 has performed much better than the other schemes when the number of failures getting higher. This is achievable due to efficient use of the limited space storage of the packet header, where the memory access for membership checking through the use of BF-*k*/2 scheme is being applied as discussed above.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

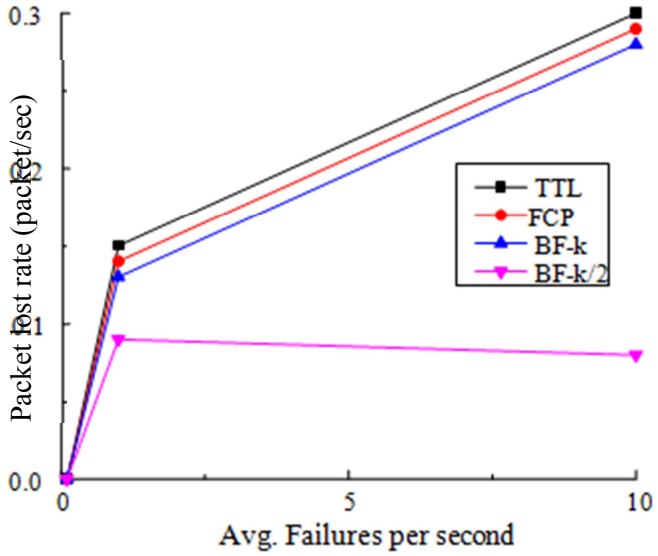

Figure 10: The Effect Of Loss Rate On Space Efficiency

4.2 Detection accuracy for looping incidents

In this section, the performance of the proposed BF-*k*/2 method in term of its detection accuracy is compared with others contemporary methods. The proofs are provided to support the claim that BF-*k*/2 is performing better in providing a reliable re-routing mechanism for the Internet.

a) Detection accuracy vs. Number of looping incidents

It can be expected that when the number of forwarding loop/packet looping incidents is high, then the detection accuracy will be better. As proposed, the performance of BF-*k*/2 scheme is evaluated by simply comparing with its initial derivative of BF-*k* scheme under different numbers of looping incidents. This is done when BF-*k*/2 performance is checked in terms of its detection accuracy as the number of forwarding loop/packet looping incidents increased at a router node. As can be observed from Figure 12, BF-*k*/2 is achieving higher detection accuracy for any number of looping incidents as compared to its counterpart of BF-*k*. Therefore, it is suggested to use BF-*k*/2 scheme in detecting the looping incidents when traffic congestion has occurred so that higher accuracy can be achieved.

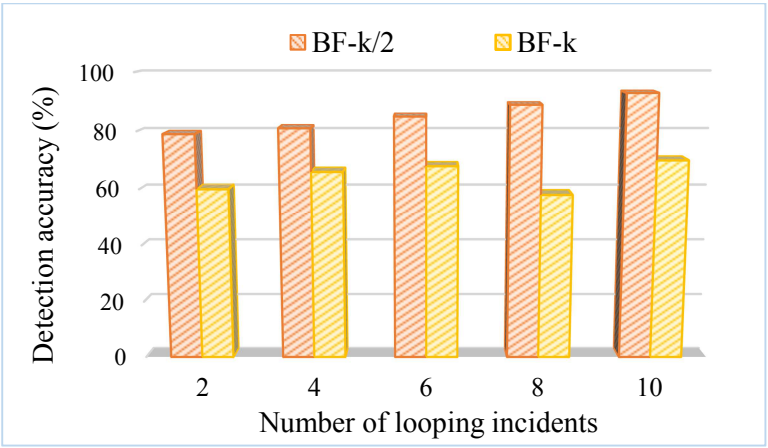

Figure 12: Detection Accuracy Between The Existing BF-K And The Proposed BF-K/2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

b) Detection accuracy vs. false positive probability

It can be understood that traffic congestion may happen at a router at any point of time. The main task now is to understand further this behavior so that the reason for the cause is more visible. Intuitively, any incidents of forwarding loop or packet looping must be triggered by an instant of traffic congestion at a router node, which the congestion must be caused by the forwarding attempts onto a failed outgoing link, at least once. If the incidents of forwarding loop/packet looping can be accurately detected, then the chance of false positive to happen can be reduced to minimum, leading to a convergence free re-routing.

Figure 11 gives a representation that there must be a limit point between the ability to accurately detect incidents of forwarding loop/packet looping and the possibility of obtaining a reasonable false positive results. In general, during a congestion period, when the number of looping incidents is low and the traffic density is also low, then the accuracy in detecting the looping incidents is low but with high false positive probability (FPP). Inversely, when the number of looping incidents and the traffic density are high, then the detection accuracy is high with lower FPP. Therefore, there must a limit point for the detection accuracy and FPP to perform at their best during a congestion period, and while a convergence process is running. This is practically true in the sense that when the detection accuracy cannot be provided, then the FPP to occur is higher. From Figure 11, it is observed that the limit point for detection accuracy and FPP for the proposed method of BF-*k*/2 is achieved when the number of looping incidents is at about 10 and the traffic density is at about 65%. That is to say, the proposed BF-*k*/2 method is expected to perform at its best when the traffic load (congestion level) is less than 65% and when the number of looping incidents is less than 10 incidents – these are the limit. Above these values, the two requirements will be compromised severely. By understanding this behavior, then the proposed BF-*k*/2 method can always be tuned to work with lower number of forwarding loop/packet looping incidents and with less traffic loads.

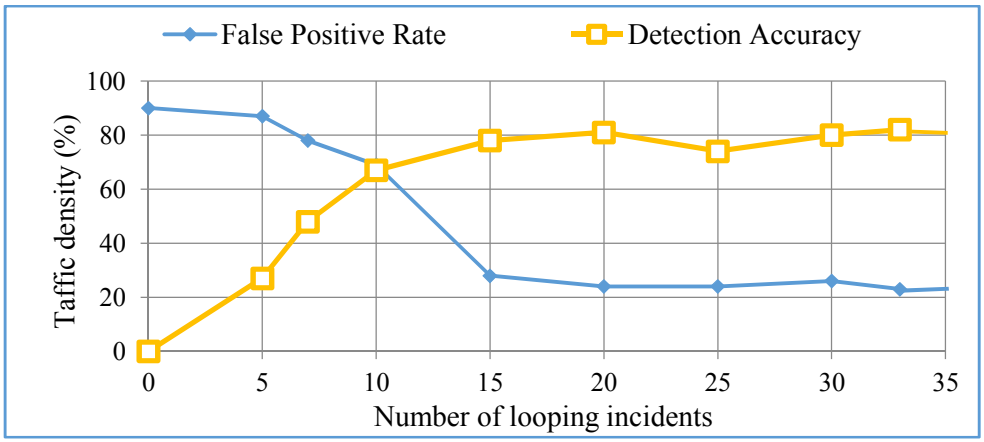

Figure 11: A Balance Between Loop Detection Accuracy And False Positive Results

c) Packet lost rates over a convergence period

It is also important to see what would be the number of packets dropped when the proposed BF-*k*/2 is run over a time of convergence period as compared to TTL, FCP and BF-*k* schemes. This is for the reason that when a scheme is offering lower packet lost over a shorter convergence period, it would be a better choice in handling the forwarding loop/packet looping issue. From Figure 13, it can be observed that the packet lost rate per link with BF-*k*/2 method is much less than the TTL, FCP and BF-*k*. Importantly to note that BF-*k*/2 is offering much stable packet lost rates over a period of convergence process, hence, its performance is much more predictable as compared to the others. With this predictable behavior, BF-*k*/2 is then the best choice to handle any case of traffic congestion. Ultimately, the convergence period will be faster with higher reliability.

Journal of Theoretical and Applied Information Technology
30th September 2015. Vol.79. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

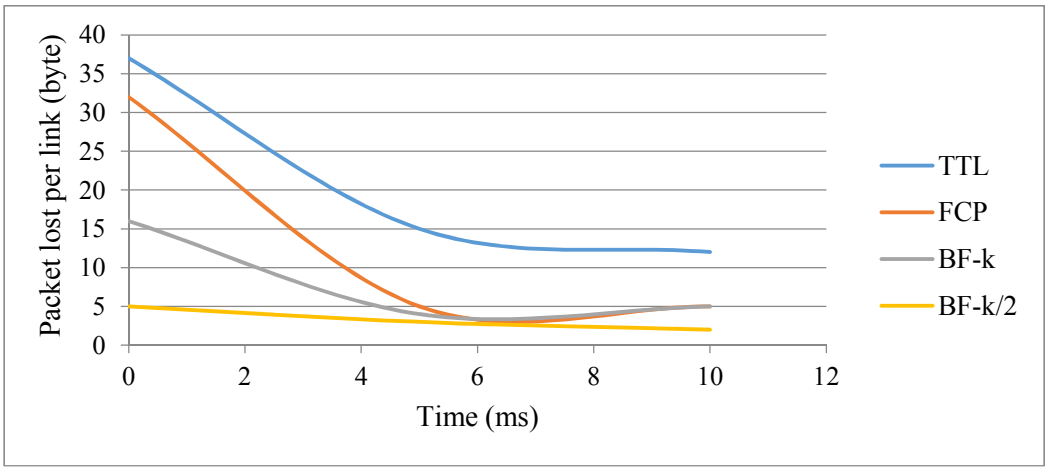

Figure 13: Packet Lost Over A Failed Link During A Convergence Period

d) Convergence period with different topology setting

Also, it is interesting to see the performance of the proposed BF-*k*/2 method with respect to its detection accuracy in different topology setting. As can be seen from the earlier discussion that when the incidents of forwarding loop/packet looping can be accurately detected, then the convergence period will be shorter. For this purpose, comparison was made between BF-*k*/2 with TTL, FCP and BF-*k* in different topologies of Abilene [9], Exedus [11] and Random [13]. Generally, Abilene, Exedus and Random are the kind of topology that high-performance backbone network, peering network, and scale-free network. Figure 14 showed the convergence period of the BF-*k*/2 as compared with TTL, FCP and BF-*k*. As can be observed, the BF-*k*/2 method has always out-performed the other methods in terms of convergence period with different network topologies.

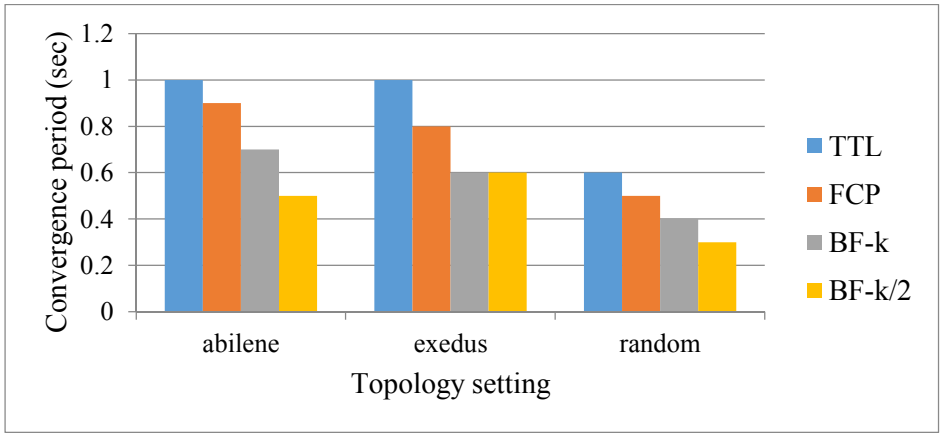

Figure 14: Impact Of Topology On Convergence Period

e) Processing times

Lastly, processing time of BF-*k*/2 is compared with BF-*k,* FCP and TTL in three different interface-id. From Figure 15, it is observed that BF-*k*/2 has performed much better than the other re-routing schemes. BF-*k*/2 has obtained the lowest processing time …..

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

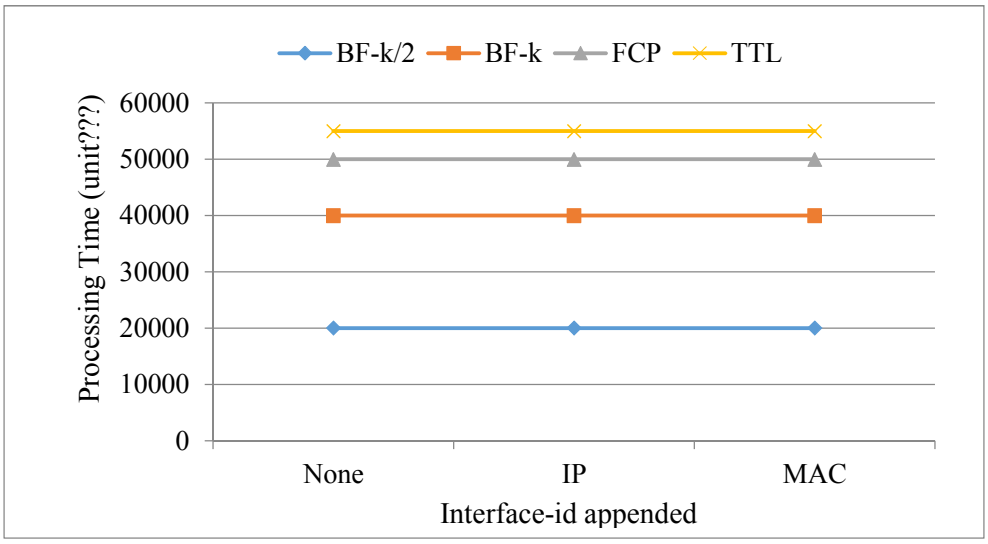

Figure 15: Processing Time With Different Interface-Id

5. Conclusion

In this research work, a novel method has been developed to provide full failure recovery through forwarding plane re-routing. It is a network-link abstraction, where space-efficient of packet header and accurate detection of packet looping probability has been proposed and evaluated. The main theoretical contribution in this approach was the design of a scheme that efficiently uses the limited storage space of the packet header and that accurately detects any possible incidents of forwarding loop or packet looping at a router node, regardless of traffic load being experienced at a router node in question. In achieving the said contributions, the Bloom-filter capability in performing membership checking has been extended to handle more complex cases of routing functions as it is demanded from the nowadays highly congested Internet traffic scenario.

It was found that Bloom-filter has also been able to be used as a space-efficient technique for data storage into a narrow storage capacity of a packet header. It was found that Bloom-filter is the best data structure, which can be used to provide reliability in detecting forwarding loop incidents at a router to as high accuracy. However, it is relatively a new area and topic to use BF in routing decisions.

It is when compared with the other previously reported contemporary and BF-based re-routing schemes for reliability, the proposed BF-*k*/2 method has not only retains reliability properties, but also bears smaller packet loss ratio and less transmission overhead, especially when the traffic is heavy. With the BF-*k*/2 method, it has been observed that the detection for forwarding loop/ packet looping incidents has increased its as compared to the existing methods. Hence, the proposed method has a great potential to be used with any Internet routing protocols, with least resource requirements and less overhead in mitigating the forwarding loop/packet looping problem due to traffic congestion at a router node in the network. With this reliability results, the provisioning of Internet toward a universal communications infrastructure is more promising than before.

ACKNOWLEDGEMENT

This paper has been funded by the National Plan for Science, Technology and Innovation (MAARIFAH) – King Abdulaziz City for Science and Technology - the Kingdom of Saudi Arabia – award number (12-INF 2723-03). The authors also, acknowledge with gratitude the Science and Technology Unit in King Abdulaziz University for technical support.

REFERENCES:

[1] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. ACM of the Communications, 13(7), p. 422-426, 1970.

[2] G. Carl, G. Kesidis, R. R. Brooks, and S. Rai. Denial-of-service attack-detection

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

techniques. IEEE Internet Computing, 10(1) , p. 82-89, 2006.

[3] C. Douligeris and A. Mitrokotsa. DDoS attacks and defense mechanisms: classification and state-of-the-art. Computer Networks, 44(5), p. 643-666, 2004.

[4] E. Page. Continuous inspection schemes. Biometrika, 41, p. 100-115, 1954.

[5] S. Geravand and M. Ahmadi. Bloom filter applications in network security: A state-of-the-art survey. Computer Networks, vol. 57, pp. 4047-4064, 2013.

[6] M. Sarela, C. E. Rothenberg, T. Aura, A. Zahemszky, P. Nikander, and J. Ott. Forwarding. Anomalies in Bloom filter-based multicast. In Proceedings of the IEEE INFOCOM, p. 2399-2407, 2011.

[7] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz. Theory and Practice of Bloom Filters for Distributed Systems. IEEE Communications Surveys & Tutorials, 14(1), p. 131-155, 2012.

[8] P. Francois and O. Bonaventure. Avoiding transient loops during the convergence of link-state routing protocols. IEEE/ACM Transactions on Networking, 15(6), p 1280-1292, 2007.

[9] M. Särelä, C. E. Rothenberg, A. Zahemszky, P. Nikander, and J. Ott. BloomCasting: security in Bloom filter based multicast. Springer in Information Security Technology for Applications, p 1-16, 2012.

[10] K. El Defrawy and G. Tsudik. PRISM: Privacy-friendly routing in suspicious MANETs (and VANETs. In Proceedings of the IEEE International Conference on Network Protocols (ICNP), p 258-267, 2008.

[11] P. P. Lee, T. Bu, and T. Woo. On the detection of signaling DoS attacks on 3G wireless networks. 26th IEEE International Conference on Computer Communication (INFOCOM), p 1289-1297, 2007.

[12] P. P. Lee, T. Bu, and T. Woo. On the detection of signaling DoS attacks on 3G/WiMax wireless networks. Computer Networks, 53(15), p 2601-2616, 2009.

[13] D. Comer. Network Systems design Using Network Processors. Prentice Hall, 2003.

[14] R. Jain. Characteristics of destination address locality in computer networks: a comparison of caching schemes. Computer Networks and ISDN Systems, 18(4), p 243-254, 1990.

[15] S. Iyer, R. K. Rao, and A. Shelat. Classipl: an architecture for fast and flexible packet classification. IEEE Network, 15(2), p 33-41, 2001.

[16] S. McCreary, and k. claffy. Trends in wide area IP traffic patterns a view from Ames Internet exchange. In Proceedings of the ITC Specialist Seminar, Monterey, 2000.

[17] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer. Single-packet IP traceback. IEEE/ACM Transactions on Networking, 10(6), p 721-734, 2002.

[18] [19] J. Saltzer, D. Reed, and D. Clark. End-To-End Arguments In System Design. ACM Transactions on Computer Systems, 2(4), p 277-288, 1984.

[20] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, Fast and Scalable Layer Four Switching In Proceedings of the ACM SIGCOMM, p 191-202, 1998.

[21] Broder, A. and Mitzenmacher, M. Network Applications of Bloom Filters: A Survey. Internet Mathematics 1 (4), p 485–509, 2002.

[22] B. Zhou, R. Zhu, Y. Zhang, and L. Cheng,. An Efficient Data Fingerprint Query Algorithm Based on Two-Leveled Bloom Filter. Journal of Multimedia, 8, p 73-81, 2013.

[23] A. Pagh, R. Pagh, and S. S. Rao. An optimal Bloom filter replacement. In Proceedings of the 16th Annual ACM-SIAM symposium on Discrete algorithms, p 823-829, 2005.

[24] A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. Internet mathematics, 1, p 485-509, 2004.

[25] R. Chikhi and G. Rizk. Space-efficient and exact de Bruijn graph representation based on a Bloom filter. Algorithms for Molecular Biology, 8, p 1, 2013.

[26] M. Rhu, M. Sullivan, J. Leng, and M. Erez. A locality-aware memory hierarchy for energy-efficient GPU architectures. In Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture, p 86-98, 2013.

[27] M. Shand, S. Bryant, S. Previdi, C. Filsfils, P. Francois, and O. Bonaventure. Framework for Loop-Free Convergence Using the Ordered Forwarding Information Base (oFIB) Approach. 2013.

[28] K. Lakshminarayanan, M. Caesar, M. Ragan, Achieving convergence-free routing

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

using failure-carrying packets. In proceeding of ACM SIGCOMM, 2007.

[29] Y. Wang, H. Wang, A. Mahimkar, R. Alimi, Y. Zhang, L. Qui, and Y. R. Yang, R3-Resilient Routing Reconfiguration, in Proceeding of ACM SIGCOMM, 2010.

[30] S.S. Lor, R. Landa, R. Ali, and M. Rio, Packet re-cycling: eliminating packet losses due to network failure, in Proceeding ACM HotNets, 2010.