A Way of Eliminating Errors When Using Bloom Filters for Routing in Computer Networks

Gökçe Çaylak Kayaturan *, Alexei Vernitski †
Department of Mathematical Sciences,
University of Essex
Colchester, UK
Email: *gcayla@essex.ac.uk, †asvern@essex.ac.uk

***Abstract*—A Bloom filter is a data type for storing sets. It can be considered as a data compression technique, but its more important feature is an extremely fast access to stored data. This is why it can be useful when calculation needs to be performed very quickly, for example, in an application to routing messages in a computer network. A well-known shortcoming of a Bloom filter are errors in the stored data. We present a way of labelling links in a computer network which prevents errors in Bloom filters in some routing scenarios and, therefore, results in a more efficient use of network resources.**

***Keywords*—Bloom filter; computer network; routing.**

I. Introduction

A Bloom filter is a data type for storing sets. Its great advantages are space efficiency (that is, the space used is very small) and time efficiency (that is, accessing the stored data is extremely fast). This is why using Bloom filters has been suggested for a number of applications. A Bloom filter is probabilistic in the sense that it involves the use of pseudorandom hashes and, therefore, comes with a non-zero probability of errors called false positives [1]. This is why a general theme in the studies of Bloom filters is reducing the number of false positives.

Some applications of Bloom filters [2][3][4][5] to network problems, especially data mining, reducing data traffic and security of data transmission between parties, motivate us to introduce a new kind of encoding method for the items of the set represented by a Bloom filter. A wide range of research of network applications of Bloom filters was also presented in [6] and [7]. The study [8] proposes a Bloom filter protocol to minimize the memory for the storage of summary caches in order to reduce the web traffic. One variant of a Bloom filter called compressed Bloom filter [9] also focuses on decreasing the size of the network messages transmitted. For less computational requirements a path architecture has been established in [10] to increase the security of network traffic flow via Bloom filter. Secure transmission of messages is addressed in the studies of cryptographic Bloom filters [11] introducing a coding method for transmitting large data using Bloom filters. Sharing information between parties via grid models was proposed by [12] and [13].

The model we consider is a rectangular-grid computer network with messages forwarded between computers within it. Similar models were considered in [12][14]. We remove randomness from Bloom filters by constructing an explicit method of encoding links in the network. Our approach makes the use of Bloom filters more secure as well as optimizes the size of data to reduce the network traffic. The main advantage of our coding system is that it we encode sets using Bloom filters without introducing any errors at all.

This paper is organised with seven sections in total. In Section 2 the general definition of a standard Bloom filter is given, and optimal parameters are defined. In Section 3 we introduce a new encoding technique for edges in a rectangular grid network. In Section 4 we explain how our encoding method restricts and controls the number of 1s of Bloom filter. In Section 5 we prove that there are no false positives. In Section 6 we compare our encoding system with the performance of the standard Bloom filter. Section 7 is the conclusion.

II. Standard Bloom filters

A set S with n elements is represented by a binary array of m bits. In order to make this possible, each item which potentially can be an element of S is represented by an m bit array in which k bits are set to 1. The positions of these bits are chosen using pseudorandom hashes. All other bits are set to zero. We shall refer to these arrays as Bloom filters of the items. The Bloom filter of a set S is constructed by applying the bitwise OR operation to all the Bloom filters of the elements of S. Note that binary OR operation takes two bits as inputs and produces a bit 1 if at least one of the inputs is 1, or produces a bit 0 when all inputs are 0. This compression of the data of a set shows that the Bloom filter has space efficiency. After the Bloom filter of a set S has been constructed, one can query whether an item x belongs to the set S or not by comparing the Bloom filter of the set with the Bloom filter of x in all bit positions. If at one of these positions the Bloom filter of x is greater than the corresponding positions of the Bloom filter of S, then we conclude that the tested item is definitely not in the set S. However, when the tested item is less than or equal to the Bloom filter in all bit positions, then this item probably is in the set S. Some of these items, called 'false positives', are seemingly in the set S but are not really in the set. Namely, there is a probability that an item $x \notin S$ might be recognised by the Bloom filter to be in the set S, and then it is called 'false positive'. Even though some errors in the set occur, presence of any element in the set is checked very quickly. Hence comes the time efficiency of the Bloom filter.

An approximate value of the probability of false positives in a Bloom filter is usually expressed is a function which depends on three parameters k, m, n that are the number of 1s of items of the set S, length of these items and number of items of the set, respectively. The study of Bloom [1] shows that the probability of an item being a false positive is as given in the approximate formula (1), which is obtained by a simple probability argument.

$$\left(1-(1-\frac{1}{m})^{kn}\right)^k \approx \left(1-e^{\frac{-kn}{m}}\right)^k \tag{1}$$

The probability of being a false positive was expressed in other ways by [6][8][9][15]. The smallest probability of a false positive is reached when $k = \ln 2 \times \frac{m}{n}$, as can be seen by taking the derivative of (1). In theory, the smallest probability may be obtained when k is exactly $\ln 2 \times \frac{m}{n}$, but in practice k must be an integer.

III. Encoding paths in a rectangular grid

The model we are considering is a computer network having a shape of a rectangular grid of size $M \times N$, that is, M links horizontally in each row and N links vertically in each column. We assume that there is a computer at each node of the grid. We assume that from time to time a computer in the network may need to send a message to another computer in the network; then the sender computer encloses a header with the message, which describes exactly what path the message must follow. For this purpose, each edge is allocated its own Bloom filter in advance, and the path is represented as the set of its edges, that is, the Bloom filter of the path is the bitwise OR of the Bloom filters of the edges constitution the path. We assume that only shortest paths in the grid are used to deliver messages between the computers.

Lemma 1. A directed shortest path between two distinct vertices in a rectangular grid consists of directed edges with at most two different directions.

Proof: Consider an imaginary straight line between two distinct vertices in a rectangular grid. According to Euclid geometry this straight line is the shortest distance between two given vertices. But the path consists of the edges between the vertices. This imaginary line can be best approximated by directed edges pointing in only two different directions, and hence the shortest path includes only such edges. ■

We introduce two systems of coordinates for the grid: the one starting from the bottom-left corner and the second starting from the top-left corner of the grid. Accordingly, we introduce two notations for each vertex, with the letter u in the former system of coordinates and with the letter v in the latter.

Each edge is represented by a Bloom filter with the length $m = 4 \times (M + N)$ bits. The Bloom filter consists of two equal length parts, which correspond to the two systems of coordinates. The vertex at the bottom left corner of the grid is denoted by $u_{(0,0)}$ and is the origin of the x/y coordinate system, with the coordinates of the vertices increasing in the direction of north-east. The point $v_{(0,0)}$, the origin of the other x/y coordinate system, is the vertex at the top left corner of the grid, and the coordinates of vertices increase in the direction of south-east.

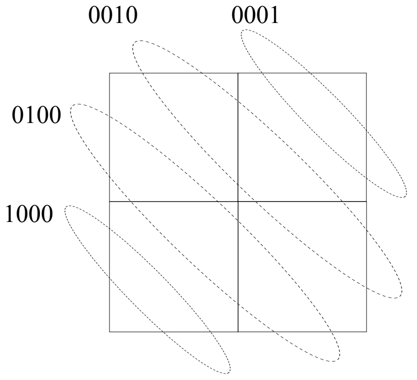

Figure 1. The allocation of the first halves of the Bloom filters of the edges in each cell.

Encoding an edge is based on the coordinates of the edges incidental with it and on the horizontal or vertical orientation of the edge in the grid. That is, when an edge is horizontal, the first and the second halves of the Bloom filter of the edge will be based on the vertex $u_{(i,j)}$, otherwise known as $v_{(i,N-j)}$, that is the left endpoint of the encoded edge.

However, if an edge is vertical, the vertices to encode it will be $u_{(i,j)}$, which is on bottom endpoint of the edge, and $v_{(i,N-j-1)}$, which is top endpoint of the same edge. Predictably, the first half of the Bloom filter of a vertical edge will be specified with the vertex $u_{(i,j)}$ and the second half will be constructed with $v_{(i,N-j-1)}$.

It is useful to imagine each half of the Bloom filter encoding an edge as a sequence of two-bit long blocks. There are $2(M+N)$ two-bit blocks in total in the Bloom filter, since each half of the encoded edges has $(M+N)$ blocks. Exactly one block of each half contains 1, and it is either 01 or 10 (as described further). All other blocks are 00. To encode a vertical edge, the bit 1 will be placed in the first positions of both $(i+j+1)$th and $(M+N+i+N-j-1+1)$th blocks in the first and the second half of the Bloom filter of the edge, respectively. To encode a horizontal edge, the bit 1 is placed in the second position in $(i+j+1)$th and $(M+N+i+N-j+1)$th blocks in the first and the second halves of Bloom filter of edge, respectively.

Let us look at an example. The first halves of the Bloom filters of the edges of a 2×2 grid are presented in Figure 1, and the second halves of their Bloom filters are presented in Figure 2. Consider every cell of both Figure 1 and Figure 2 as a rectangular grid with size 1×1. A horizontal edge of a cell is represented with the points $u_{(0,0)}$ and $v_{(0,1)}$, which are the left endpoint of the edge, in a rectangular grid with size 1×1. Then the block 01 will be placed in the $0+0+1 = 1$st and $1+1+0+1+1 = 4$th block positions among $2(1+1) = 4$ blocks. Note that the length of the Bloom filters of edges is $4(1+1) = 8$ bits. So the Bloom filter of this horizontal edge will look like 01000001. Besides, a vertical edge is represented with the points $u_{(0,0)}$ and $v_{(0,0)}$, which are the bottom and top endpoints of the vertical edge, respectively, in the grid with size 1×1. The block 10 will be situated in the $0+0+1 = 1$st and $1+1+0+0+1 = 3$th block positions. The Bloom filter

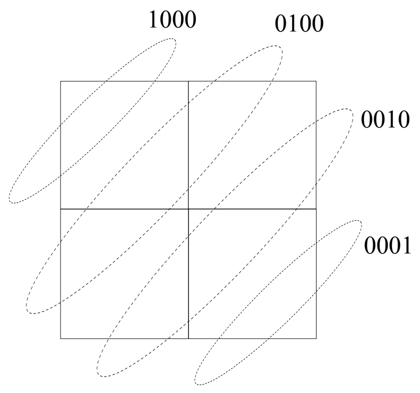

Figure 2. The allocation of the second halves of the Bloom filters of the edges in each cell.

of the this vertical edge will be as 10001000.

IV. An analysis of Bloom filters of paths

A. *Positions of* 1*s in the Bloom filters*

All possible shortest paths consist of $n \geq 1$ edges lying between two distinct vertices continuously. All Bloom filters of edges include two 1s and the Bloom filter of a path is obtained by applying OR operation to the encoded adjacent edges lying on the path together.

Lemma 2. Consider an undirected path whose direction it towards north-east (or, equivalently, south-west). Then the first half of the Bloom filter of the path contains a consecutive subsequence of blocks having the form 01 and 10.

Likewise, if we consider an undirected path whose direction it towards south-east (or, equivalently, north-west), then the second half of the Bloom filter of the path contains a consecutive subsequence of blocks having the form 01 and 10.

Proof: This is because the coordinates of vertices u of the path edges are increasing towards north-east and the blocks in the first halves of the Bloom filters of the edges are specified by the points of u. Likewise, if we consider the second halves of Bloom filters of edges of a path directed towards south-east, the coordinates of vertices v increase in the same way. ■

One can observe that some blocks in the second half of the Bloom filter of a path going north-east contain two bits 1 at the same time. Similarly, the first half of the Bloom filter of a paths going south-east might include some blocks 11. If all the edges of a specific path are turned in one direction (north or south or west or east), then all blocks which include one bit 1 in either half of the Bloom filter of the path come after each other consecutively without an interruption of a block such as 00 or 11.

B. *The number of* 1*s in the Bloom filters*

In our encoding the number of 1s in the Bloom filter of each encoded edge is 2 and they are not placed randomly but depend on the position of the edge. The Bloom filters of shortest paths in a rectangular grid are obtained without randomness being involved, hence, the number of 1s in these Bloom filters is restricted.

Lemma 3. The number of 1s of the Bloom filter of the paths in rectangular grid is not greater than $2n$ where the number of edges of the path is n.

Proof: By the lemma 2, the blocks 10 or 01 constitute a continuous sequence of blocks within the first or the second half of the Bloom filter of a path. So, these blocks 10 or 01 represents the number of edges in the path and there are n edges in total in a path. Hence, when the number of the bits 1 in one half of the Bloom filter of the path is n, the number of bits 1 will be $\leq n$ in the other half of the Bloom filter. Any half of Bloom filter of path does not necessarily have n 1s, since the 1s in one half of the Bloom filter of a path might be produced in the same positions by more than one edge. More precisely, the path might contain the edges that are encoded on the same block position with the same pair bits including 1. Therefore, the bits 1s of the Bloom filter of these paths are $\leq 2n$.

We may note that if the edges of a path are directed in only one way, then both halves of the Bloom filter of the path will contain n 1s. ■

V. Avoiding false positives

Theorem 4. All Bloom filters of edges are unique in a rectangular grid.

Proof: All edges of a grid of size $M \times N$ are encoded by $2(M+N)$ blocks and each block consists of two bits. Suppose two edges e and f are distinct and both are horizontal. So, the 1s of these edges will take place in second bit positions in corresponding blocks.

The vertices of the horizontal edge e will be $u_{(i,j)}$ and $v_{(i,N-j)}$ and the vertices of the horizontal edge f will be given with $u_{(k,l)}$ and $v_{(k,N-l)}$, when we accept the left top and bottom corners of the grid as the centres of two different x/y coordinate systems. Both halves of horizontal edges are encoded with the left endpoints of the edges. The blocks containing 1s will be placed in $(i+j+1)$th and $(M+N+i+N-j+1)$th block positions of the Bloom filter of edge e. Similarly, $(k+l+1)$th and $(M+N+k+N-l+1)$th blocks of the Bloom filter of edge f contain 1s.

Assume the edges f and e are represented by the same Bloom filter. That means both $i+j+1 = k+l+1$ and $M+N+i+N-j+1 = M+N+k+N-l+1$ occur at the same time.

$$i + j = k + l \tag{2}$$
$$i - j = k - l \tag{3}$$

Then, the equations $i = k$ and $j = l$ are obtained. This contradicts with the assumption that the edges e and f are distinct.

If we suppose that both edges e and f are vertical, then both edges will contain 1s on the first positions of the corresponding blocks. Assume these two edges are distinct and encoded by the same Bloom filter. Then the same equations concerning the coordinates of the vertices $u_{(i,j)}$, $v_{(i,N-j-1)}$ and $u_{(k,l)}$, $v_{(k,N-l-1)}$ as above can be constructed and obviously the same contradiction which is found for the edges lied horizontally will be obtained.

Now we suppose that one of the distinct edges is horizontal and the other is vertical. Namely, when e is vertical, then the bit 1 of the Bloom filter of e will only appear in the first positions of some blocks. Yet the bit 1 of the Bloom filter of the edge f will appear in the second position of some blocks, since f is horizontal. Hence even if the blocks including 1s of these two distinct edges are in the same block positions, these 1s are not placed on the same positions.

As a result of considering these three cases, we conclude that all edges of a grid are encoded uniquely in a given size grid. ■

Theorem 5. The Bloom filter of a path consisting of $n \geq 1$ edges in a rectangular grid model does not yield any false positives when links adjacent to the path are queried.

Proof: In the proof we shall concentrate on one fixed node of a path and demonstrate that no more than one link will be recognised by its Bloom filter as the next link of the path.

Consider the Bloom filter of a shortest path in a rectangular $M \times N$ grid. Suppose a message travelling along this path has arrived to some node via an edge e. It can continue travelling via any of the next three edges adjacent to e at one end, and the computer at the node needs to decide which one of them to choose. We shall see that the Bloom filter of the path $\beta(P)$ including edge e does not yield any false positives, that is, only one of these three edges is recognised.

An edge e encoded with $4(M+N)$ binary bits is represented by $\beta(e)$ that is divided into two bits length blocks called $\beta_1(e), \beta_2(e), \ldots, \beta_{2(M+N)}(e)$. The two individual bits constituting a block $\beta_p(e)$ will be denoted by $\beta_p^1(e)$ and $\beta_p^2(e)$. Note that the Bloom filters of the edges are divided into two equal parts and the positions of the blocks containing 1s of all Bloom filters of edges are specified with the coordinates u for the first half and coordinates v for the second half. When an edge is horizontal, the bit 1 is in the second positions of certain two blocks in the first and the second half of the Bloom filter, and the remaining blocks are all 00. The positions of 1s are determined by the coordinates $u_{(i,j)}$ and $v_{(i,N-j)}$ and are at the positions $\beta^2_{(i+j+1)}(e)$ and $\beta^2_{(M+N+i+N-j+1)}(e)$, where $(i+j+1)$ and $(M+N+i+N-j+1)$ represent the positions of the blocks. Note that we will use k instead of $(i+j+1)$ and l instead of $(M+N+i+N-j+1)$ to make the indices simpler.

If the edge e is vertical, then 1s of the blocks will be in the first positions of the corresponding blocks. These bits will be at positions $\beta_k^1(e)$ and $\beta_{l-1}^1(e)$, since the endpoint vertices of a vertical edge e are $v_{(i,j)}$ and $u_{(i,N-j-1)}$.

Let the three edges adjacent to the edge e at one end be f, g and h (as shown on Figure 3). We already know that $\beta(e) \neq \beta(f) \neq \beta(g) \neq \beta(h)$. Now, we will look for the false positives of the Bloom filters of the paths for these four possible directions.

Firstly, we will suppose that the edge e is on the path where the message moves east and the represented vertices of e will be $u_{(i,j)}$ and $v_{(i,N-j)}$ (see Figure 3).

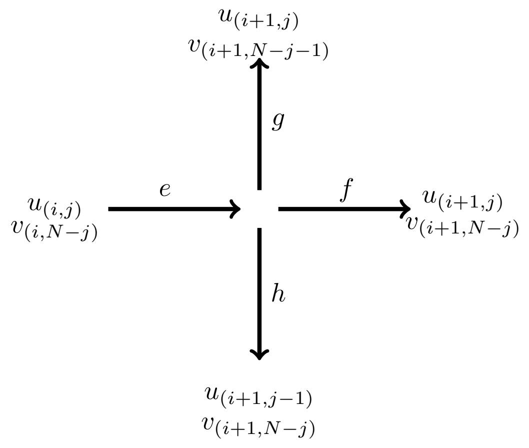

Figure 3. The edge e is supposed to lie on a path which is on the way of east and the next three adjacent edges of e.

The next edge the the message can follow after the edge e can be the one to the east, north or south, that is, the edges f, g, h, which are represented by the vertices $u_{(i+1,j)}$ and $v_{(i+1,N-j)}$, $u_{(i+1,j)}$ and $v_{(i+1,N-j-1)}$, $u_{(i+1,j-1)}$ and $v_{(i+1,N-j)}$, respectively. By the encoding method, we add the values of the points of represented vertices and 1 together to find the block positions of the pairs including 1s of the edges. So, the bits $\beta_k^2(e)$ and $\beta_l^2(e)$ will be 1. As a result of the encoding method, the bits $\beta^2_{(k+1)}(f)$ and $\beta^2_{(l+1)}(f)$, $\beta^1_{(k+1)}(g)$ and $\beta^1_{(l)}(g)$, $\beta^1_{(k)}(h)$ and $\beta^1_{(l+1)}(h)$ will be 1.

Note that since the edge e lies on the path, the bits of corresponding blocks of the Bloom filter of the path are $\beta^2_{(k)}(P) = 1$ and $\beta^2_l(P) = 1$.

The proof proceeds by considering several cases: that f is on the path, that g is on the path and that h is on the path.

If both $\beta^2_{(k+1)}(f) \leq \beta^2_{(k+1)}(P)$ and $\beta^2_{(l+1)}(f) \leq \beta^2_{(l+1)}(P)$ occur at the same time, then we say whether the path includes the edge f or f is a false positive of the path. But the bit of the path $\beta^1_{(k+1)}(P)$ is 0. Since if the path goes north-east, then the first half of the Bloom filter of path will contain the blocks which consist of one 0 and one 1 consecutively (see lemma 2). That means $\beta^1_{(k+1)}(g) = 1 > \beta^1_{(k+1)}(P) = 0$. Namely, the Bloom filter of the path is not greater than or equal to the Bloom filter of the edge g in all bit positions. As a result, when the Bloom filter of the edge f is smaller than or equal to the Bloom filter of the path P in all bits positions, more precisely $\beta(f) \leq \beta(P)$, then the edge g is definitely not on the path.

When both Bloom filters of e and f are smaller than or equal to the Bloom filter of path in all bits positions at the same time, then the two consecutive blocks $\beta_k(P)\beta_{(k+1)}(P)$ of the Bloom filter of the path will look like 0101. This is because the coordinates of the vertices used to produce the blocks including the bit 1 in the first half of the Bloom filter are increasing towards north-east, and the edges e and f are followed along the path towards the east and north. Under these circumstances, when the edge e lies on the path and $\beta(f) \leq \beta(P)$, then $\beta^1_{(k)}(P) = 0$. But $\beta^1_{(k)}(h) = 1 > \beta^1_{(k)}(P) = 0$, then we conclude that the edge h is not on path. The Bloom filter of edge h is not smaller than or equal to the Bloom filter of the path in all bits positions.

Now, let us consider another case, if $\beta(g) \leq \beta(P)$ in all bits positions, then we say the edge g lies on the path after the edge e or a false positive of the Bloom filter of the path. Again, when both Bloom filters of e and g are smaller than or equal to the Bloom filter of the path in all bits positions and assume the direction of travel towards the east and north, which are the ways that make the blocks including one 1 and one 0 in the first part of the Bloom filter of the path to lie consecutively. So $\beta^2_{(k+1)}(P) = 0$, but we obtain that $\beta^2_{(k+1)}(f) > \beta^2_{(k+1)}(P)$, and hence the edge f is not on the path. Subsequently, by comparing the first bit of the (k)th block in the Bloom filter of the path and the Bloom filter of the edge h, we will conclude that h is definitely not an edge of the path.

Hence, both $\beta(e) \leq \beta(P)$ and $\beta(g) \leq \beta(P)$ occur at the same time, then both $\beta(f)$ and $\beta(h)$ are not smaller than or equal to the Bloom filter in all bits positions.

Finally, if both $\beta(e) \leq \beta(P)$ and $\beta(h) \leq \beta(P)$ in all bits positions, then the second part of the Bloom filter of the path will contain the blocks consisting of one 0 and one 1, consecutively. The coordinates of vertices v increase towards south-east and e and h can be travelled towards east and south, respectively. But $\beta^2_{(l+1)}(f) > \beta^2_{(l+1)}(P)$ and $\beta^1_{(l)}(g) > \beta^1_{(l)}(P)$. Hence, when $\beta(h) \leq \beta(P)$, the edges f and g are definitely not on the path.

As a result, the edge e is followed by exactly one of the adjacent edges f or g or h. As shown above, when e lies on the path and any adjacent edges of e is smaller than or equal to the Bloom filter in all bits positions at the same time, then the other adjacent edges are definitely not on the path.

We considered the situation when the edge e has been travelled by the message eastwards. When the edge e is oriented some other way, so that message travels along it north or west or south, the same argument as above can be applied. ■

VI. The advantages of our approach

The standard Bloom filter is defined as a randomized data structure. If our model included randomness, we would certainly get some false positives. However, we use the standard Bloom filter approach (indeed, we use disjunction and comparison of binary arrays precisely in the way as it is done, when the Bloom filters are used) without randomness. The possible false positives of our use scenario are the edges adjacent to a fixed path which the messages must follow. If false positives existed, the messages would be sent along these adjacent edges instead of following only the specified path.

If one position per edge was used in the Bloom filter (in a naive attempt to avoid false positives), the length of the Bloom filter would be $2MN + N + M$, which is the total number of the edges of $M \times N$ sized grid. Yet the length of bloom filter in our model is $4(M + N)$, which is better, since $4(M + N) < 2MN + N + M$ for the big values of M and N.

In our model, the number of edges lying on a path takes its maximum value with $(M+N)$, when the two distinct vertices are the two opposite corners of the rectangular grid. Therefore, the number of edges on a path is $\leq (M+N)$. The length of all encoded edges and the Bloom filter of any path is a constant $4(M + N)$.

Let us see how many false positives the standard Bloom filter would produce if we used it to represent sets of this size. The formula for the optimal value of the number of the bits 1 of an item is obtained as $k = \ln 2 \times \frac{m}{n}$, when the probability of the false positives is minimized, [9]. The ratio of the length of the Bloom filter and the number of edges of the path of our model is $\frac{m}{n} = \frac{4(M+N)}{(M+N)} = 4$, where n takes its maximum value. Hence, the number of 1 of an edge is $k = \ln 2 \times 4 \approx 2,772$. Hence, in order to obtain the minimal probability of the false positive of the Bloom filter of the path, the number of 1 of the all Bloom filters of the edges can be chosen to be $k = 2$, like in our model.

When the number of edges on the path is maximal, then the probability of false positives of the Bloom filter of the path would have been obtained as (4).

$$\left(1 - e^{\frac{-2(M+N)}{4(M+N)}}\right)^2 \approx 0,1548 \qquad (4)$$

Of course, for some paths the number of edges on the path is less than $M+N$. For such shorter paths, the ratio $\frac{n}{m}$ decreases. Hence, on average the probability of false positives would be less than the equation (4). Hence, if the bits 1 would have taken place in the Bloom filter of the edges randomly, the probability of having no false positives in a path would be approximately $0.85^{2(M+N)}$, where $2(M + N)$ is the number of adjacent edges of the shortest path with maximum number of edges in the grid.

VII. Conclusion

This model uses active knowledge about the particular application of Bloom filters and combines space efficiency and time efficiency with one hundred percent accuracy by intelligently allocating Bloom filters to individual items. We shall continue this research by considering other network topologies.

References

[1] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," Communications of the ACM, vol. 13, no. 7, 1970, pp. 422–426.

[2] Y. Lu, B. Prabhakar, and F. Bonomi, "Perfect hashing for network applications," in Information Theory, 2006 IEEE International Symposium on. IEEE, 2006, pp. 2774–2778.

[3] C. E. Rothenberg, C. A. B. Macapuna, M. F. Magalhães, F. L. Verdi, and A. Wiesmaier, "In-packet bloom filters: Design and networking applications," Computer Networks, vol. 55, no. 6, 2011, pp. 1364–1378.

[4] L. Carrea, A. Vernitski, and M. Reed, "Yes-no bloom filter: A way of representing sets with fewer false positives for in-packet path encoding," 2014, submitted for publication.

[5] X. Yang, A. Vernitski, and L. Carrea, "An approximate dynamic programming approach for improving accuracy of lossy data compression by bloom filters," accepted, European Journal of Operational Research.

[6] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," Internet mathematics, vol. 1, no. 4, 2004, pp. 485–509.

[7] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and practice of bloom filters for distributed systems," Communications Surveys & Tutorials, IEEE, vol. 14, no. 1, 2012, pp. 131–155.

[8] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," IEEE/ACM Transactions on Networking (TON), vol. 8, no. 3, 2000, pp. 281–293.

[9] M. Mitzenmacher, "Compressed bloom filters," IEEE/ACM Transactions on Networking (TON), vol. 10, no. 5, 2002, pp. 604–612.

[10] T. Wolf, "A credential-based data path architecture for assurable global networking," in Military Communications Conference, 2007. MILCOM 2007. IEEE. IEEE, 2007, pp. 1–7.

[11] M. Mitzenmacher and G. Varghese, "Biff (bloom filter) codes: Fast error correction for large data sets," in Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on. IEEE, 2012, pp. 483–487.

[12] X. Li, L. Peng, and C. Zhang, "Application of bloom filter in grid information service," in Multimedia Information Networking and Security (MINES), 2010 International Conference on. IEEE, 2010, pp. 866–870.

[13] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid information services for distributed resource sharing," in High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on. IEEE, 2001, pp. 181–194.

[14] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," ACM SIGMOBILE mobile computing and communications review, vol. 7, no. 3, 2003, pp. 19–20.

[15] J. K. Mullin, "A second look at bloom filters," Communications of the ACM, vol. 26, no. 8, 1983, pp. 570–571.

[16] L. Carrea, A. Vernitski, and M. Reed, "Optimized hash for network path encoding with minimized false positives," Computer networks, vol. 58, 2014, pp. 180–191.