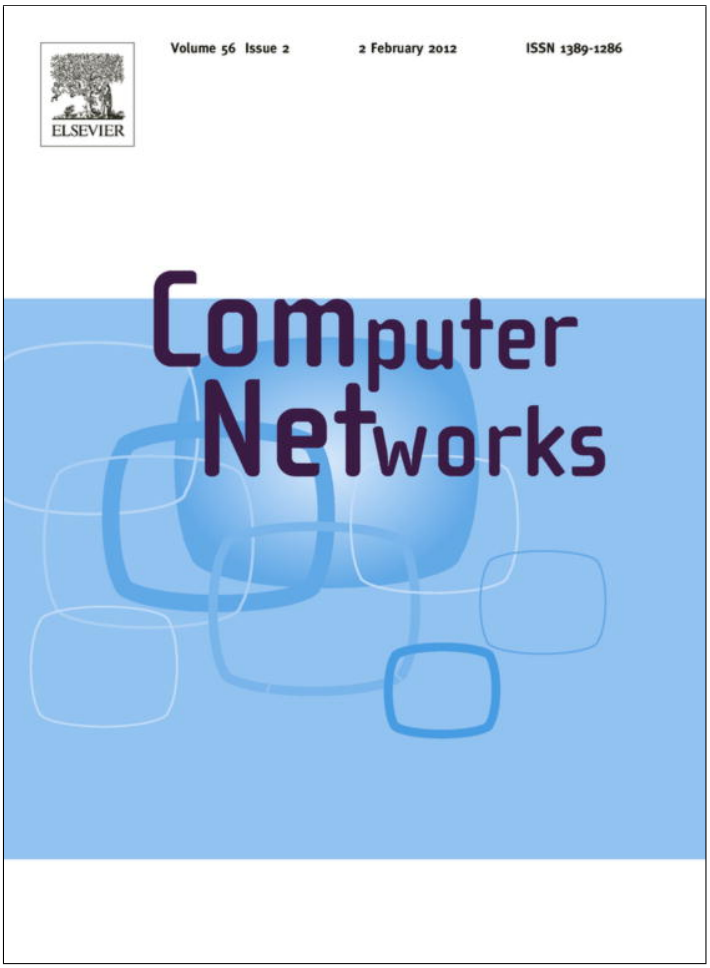

Computer Networks 56 (2012) 858–869

Contents lists available at SciVerse ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Path similarity evaluation using Bloom filters

Benoit Donnet [a,*], Bamba Gueye [b], Mohamed Ali Kaafar [c]

[a] Université catholique de Louvain, Louvain-la-Neuve, Belgium
[b] Université de Liège, Liège, Belgium
[c] INRIA Rhone-Alpes, Grenoble, France

ARTICLE INFO

Article history:
Received 9 December 2010
Accepted 8 November 2011
Available online 25 November 2011

Keywords:
Traceroute
Topology
Bloom filter
Similarity

ABSTRACT

The performance of several Internet applications often relies on the measurability of path similarity between different participants. In particular, the performance of content distribution networks mainly relies on the awareness of content sources topology information. It is commonly admitted nowadays that, in order to ensure either path redundancy or efficient content replication, topological similarities between sources is evaluated by exchanging raw traceroute data, and by a hop by hop comparison of the IP topology observed from the sources to the several hundred or thousands of destinations.

In this paper, based on real data we collected, we advocate that path similarity comparisons between different Internet entities can be much simplified using lossy coding techniques, such as Bloom filters, to exchange compressed topology information. The technique we introduce to evaluate path similarity enforces both scalability and data confidentiality while maintaining a high level of accuracy. In addition, we demonstrate that our technique is scalable as it requires a small amount of active probing and is not targets dependent.

1. Introduction

Intuitively, the *path similarity* between two nodes is defined as the IP paths overlap when those two nodes infer their paths towards an arbitrary third one. Two nodes are considered as being similar if they observe a large portion of overlapping paths (or path segments) towards a set of destinations.

Path similarity is very useful for many Internet applications, ranging from efficient distributed systems deployment to content location selection. In the context of Content Distribution Networks, for instance, providers could use path similarity to achieve both optimal performance path selection and path redundancy insurance, the path selection being obtained through similarity, while redundancy by non-similarity.

Similarity and non-similarity might find also a suitable usage in the deployment of large-scale measurement infrastructure. In particular, in the context of the Internet topology discovery [1] based on *traceroute* [2], it is important that monitors are well diversified in the network. An infrastructure such as the recently introduced *Archipelago* [3] or the Internet monitoring project *grenouille.com* [4] would benefit from similarity/non-similarity when deploying a new monitor. Indeed, if two vantage points share a large path similarity, it is obvious that they will collect redundant data, making their contribution marginal. On the contrary, if two monitors are quite non-similar, data collected should lead to a broader view of the network. In the same way, monitoring distributed systems from a set of vantage points can benefit from path similarity information in order to diversify monitoring locations [5–8].

Finally, as previously mentioned by Hu and Steenkiste, path similarity might be used in the context of available

* Corresponding author. Tel.: +32 (0)10 478718; fax: +32 (0)10 450345.
E-mail addresses: benoit.donnet@uclouvain.be (B. Donnet), gueye@run.montefiore.ulg.ac.be (B. Gueye), mohamed-ali.kaafar@inrialpes.fr (M.A. Kaafar).

1389-1286/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2011.11.003

bandwidth estimation [9]. Indeed, if several monitors are similar, it is very likely they will share the same bottleneck, and as most of the end-hosts encounter bottleneck at the first or the last four hops [10], it would be sufficient to collect bandwidth information from a single of these monitors.

Path similarity between monitors is evaluated based on actively collected data. A monitor probes a portion of the Internet using traceroute and sends the discovered topology to other monitors in the system for comparison. It should however be noted that current "raw" paths comparison for route similarity only achieves desirable accuracy, reliability and sensitivity properties at the expense of scalability and high overhead issues [6,9]. In other words, several monitors that would like to compare mutual similarities may need to exchange a huge amount of "raw" traceroute data, which can prove to be very onerous in terms of overhead and may lead to severe under performances of the network, especially when considering thousands of monitors, probing millions of IP destinations.

Further, exchanging "raw" traceroute data reveals most, if not all, of the network topology information of monitors, that are often controlled by Content Distribution Networks (CDN) [11,12]. Such an information is of primary economic and security importance for CDNs. Encryption could of course help with such confidentiality issue, but it does not solve any of the scalability issues.

Such non-scalable and cumbersome properties are then a compelling case for a lossy coding technique that would be used to exchange compressed data while maintaining both accuracy and confidentiality about path similarity.

In this paper, we propose to apply *Bloom filters* [13] for allowing monitors to exchange path information. A Bloom filter is a lossy summary technique based on a bit vector and a set of hash functions. While, Bloom filters have found numerous usages, particularly in networking due to their bandwidth saving capabilities [14,15], our approach uses them for encoding links discovered by a monitor during its probing (i.e., tracerouting) phase. The compressed path information is then exchanged between monitors and the bit vectors are compared by each monitor for evaluating its similarity with others. Scalability is achieved through the exchange of much lower amount of data, while confidentiality is maintained without encryption, as hop by hop information that would reveal networks topology is hashed by the Bloom filter. We provide methods for comparing two Bloom filters and infer, from this comparison, the similarity level between monitors.

Nevertheless, if a Bloom filter has the advantage of compressing the information, it comes with the drawback of triggering false positives. It would be a matter of concern if the bandwidth advantages of Bloom filters would be overcome by those false positives. A tradeoff must thus be found between compression, false positives, and similarity accuracy.

In this paper, we tackle also issues related to such tradeoffs. Based on real traceroutes we collected from PlanetLab monitors towards sets of thousands of destinations,[1] we examine the benefits of applying Bloom filters and compare the efficiency of similarity results obtained by exchanging compact traceroute data. We first demonstrate that our Bloom filter-based approach allows one to reduce bandwidth consumption by at least four times in terms of exchanged information compared to traditional proposals quantifying path similarity. We also examine the effects of the Bloom filter parameters on the similarity results, showing that, as long as we consider a reasonable compression ratio, we provide accurate similarity results.

We finally demonstrate that our approach is scalable, reducing so the risk of the probing phase from various monitors to turn out into a distributed denial-of-service (DDoS) attack. Indeed, we show that our technique is independent from the number of traceroutes performed, as well as from the overlapping of destinations (i.e., monitors in the system do not necessarily require to traceroute exactly the same destinations).

The remainder of this paper is organized as follows: Section 2 provides the required background on Bloom filters and explains how they can be applied to study Internet paths similarity; Section 3 formally describes the similarity metrics we introduce in this paper; Section 4 evaluates our lossy similarity technique; Section 5 positions this paper regarding the state of the art; finally, Section 6 concludes this paper by summarizing its main contributions and discussing future directions.

2. Bloom filter

In this section, we briefly remind the Bloom filters theory (Section 2.1) and, then, explain how they can be used for coding path similarity information (Section 2.2). Note that we reuse the notations previously introduced by Broder and Mitzenmacher [14].

2.1. Theory

A *Bloom filter* [13] is a vector v of m bits that codes the membership of a subset $A = \{a_1, a_2, \ldots, a_n\}$ of n elements of a universe U consisting of N elements. Typically, the size of the universe is not specified [13,14]. However, Bloom filters are only useful if the size of U is much larger than the size of A.

The idea is to initialize this vector v to '0', and then take a set $H = \{h_1, h_2, \ldots, h_k\}$ of k independent hash functions $h_1, h_2, \ldots, h_k$, each with range $\{1, \ldots, m\}$. For each element $a \in A$, the bits at positions $h_1(a), h_2(a), \ldots, h_k(a)$ in v are set to '1'. Note that a particular bit can be set to 1 several times. This is illustrated in Fig. 1.

To check if an element b of the universe U belongs to the set A, all one has to do is check that the k bits at positions

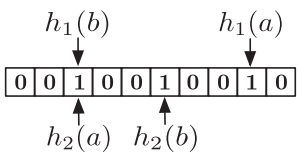

Fig. 1. A Bloom filter with two hash functions.

[1] Our dataset is freely available. See http://planete.inrialpes.fr/similarity_data/Traceroute-Similarity.tar.gz.

$h_1(b), h_2(b), \ldots, h_k(b)$ are all set to 1. If *at least* one bit is set to 0, we are sure that b does not belong to A. If *all* bits are set to 1, b possibly belongs to A. There is always a non-zero probability that b does not belong to A. In such a case, a *false positives* is raised.

In order to calculate the false positive rate, one can assume that all hash functions map each item in the universe into a random number uniformly over the range $\{1, \ldots, m\}$. As a consequence, the probability that a specific bit is set to 1 after the application of one hash function to one element of A is $\frac{1}{m}$ and the probability that this specific bit is left to '0' is $1 - \frac{1}{m}$. After all elements of A are coded in the Bloom filter, the probability that a specific bit is always equal to '0' is

$$p_0 = \left(1 - \frac{1}{m}\right)^{kn}. \quad (1)$$

As m becomes large, $\frac{1}{m}$ is close to zero and p_0 can be approximated by $e^{-\frac{kn}{m}}$.

The probability that a specific bit is set to '1' can thus be expressed as

$$p_1 = 1 - p_0. \quad (2)$$

The false positive rate can then be estimated by the probability that each of the k array positions computed by the hash functions is 1. f_P is then given by

$$f_P = p_1^k = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k. \quad (3)$$

The false positive rate f_P is thus a function of three parameters: n the size of subset A, m the size of the filter, and k the number of hash functions. Fig. 2 illustrates the variation of f_P with respect to the three parameters individually (when the two others are held constant). Obviously, and as can been seen in these graphs, f_P is a decreasing function of m and an increasing function of n. Now, when k varies (with n and m constant), f_P first decreases, reaches a minimum and then increases. There are two countervailing factors: using more hash functions gives us more chances to find a 0 bit for an element that is not a member of A, but using fewer hash functions increases the fraction of 0 bits in the array.

In networking, Bloom filters find a suitable usage in overlay and peer-to-peer networks, resource routing, packet routing, and measurement infrastructures. Although Bloom filters allow false positives, for many applications the space savings outweigh this drawback when the probability of an error is sufficiently low (see Mitzenmacher and Broder for details [14]).

The question arises now on how to use Bloom filters in the context of path similarity, and hence profit from the compression benefits while maintaining sufficiently accurate similarity detection.

2.2. Application to similarity

Any entity (let us call it *monitor*) wanting to evaluate its path similarity with others considers a set of probing targets (i.e., the *destinations*) and launches traceroutes towards those targets. Once the traceroutes have been collected, a monitor has a list of links, i.e., hop-by-hop connections.

This set of links is then encoded in a Bloom filter, as described in Section 2.1. That is, our universe U is supposed to be the set of all possible pairs of IP addresses in the Internet while the subset A is the links discovered by the monitor during the exploration phase. All links are then mapped to k positions in the bit vector using H, the set of hash functions. The resulting bit vector is then exchanged between the various monitors. How a monitor can compare two bit vectors and retrieve path similarity information is discussed in Section 3.

This scheme has the obvious advantage of completely hiding topological information collected by a monitor. Of course, it is still possible to determine which links are encoded in the filter but it requires to test the whole universe, i.e., $2^{32} \times 2^{32}$ tests. And the risk of false positives when querying the filter cannot guarantee to retrieve the exact set of links. The compression advantage of a Bloom filter is difficult to determine a priori and must be rather evaluated on a case study basis. This will be done in Section 3.

Coding hop-by-hop information in a Bloom filter and retrieving path similarity information from it might work if and only if two Bloom filters are comparable. This requires the set of hash functions used must be the same for all monitors. And, by extension, all the bit vectors must have the same size.

Such a situation requires that all monitors in the system reach an agreement on the Bloom filter tuning, given that a trade-off must be found between a good compression ratio of the links set and the false positive rate. If the monitors use the same set of destinations (or, at least, probe the

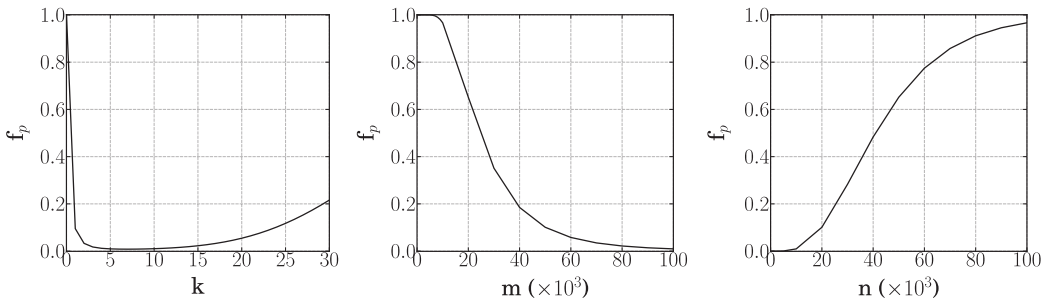

Fig. 2. f_P as a function of k, m and n.

same number of destination), one can infer the average of the number of links discovered. This average might then be used to tune the bit vector size.

3. Similarity metrics

In the remainder of this paper, we refer by $\mathbb{M}$ to the set of monitors and M_i denotes any monitor in $\mathbb{M}$ measuring paths towards a set of destinations $\mathbb{D}$.

Hu and Steenkiste [9] define path similarity as the percentage of links shared by routes from two monitors, M_i and M_j, to the set of destinations $\mathbb{D}$. This metric, called *RSIM* is defined as follows:

$$RSIM(M_i, M_j, \mathbb{D}) = \frac{\sum_{d\in\mathbb{D}} 2 \times \text{Common}(M_i, M_j, d)}{\sum_{d\in\mathbb{D}} \text{Total}(M_i, M_j, d)}. \quad (4)$$

Intuitively, closer to one $RSIM(M_i, M_j, \mathbb{D})$, the more similar paths issued from M_i and M_j.

This metric however assumes that both upstream and downstream paths share a unique path from monitors to destinations and vice versa. The $RSIM(\cdot,\cdot,\cdot)$ metric considers "raw" traceroute data sharing. As it does not require any lossy compression technique, it will then be used as reference point for comparison with Bloom filters.

If monitors have to rely on Bloom filters to compare their similarity, new metrics comparing those filters should be added. In the following, we introduce such metrics and compare their respective performance in Section 4.

Let $\vec{m}_i$ be the bit vector for monitor M_i, constructed as described in Section 2.2.

Since the filters that the monitors compare are a set of bit vectors, a straight way to compare them is to use the Hamming distance between them. Such a distance counts the number of positions where vectors elements differ. In other words, the Hamming distance measures the minimum number of bits that need to be substituted to change one Bloom filter into the other. To consider a ratio of the differences that do exist between the filters, we normalize by the bit vector size of the compared filters. We define then the *relative Hamming distance* as follows:

$$rH(M_i, M_j) = \frac{H(\vec{m}_i, \vec{m}_j)}{|\vec{m}_i|}. \quad (5)$$

where $H(\vec{m}_i, \vec{m}_j)$ provides the Hamming distance between both bit vectors. Put simply, the relative Hamming distance between two Bloom filters, of the same size and created with the same hash functions, can be used as a measure of the non-similarity of the underlying sets (IP links from monitors to a set of destinations). A trend of $rH(M_i,M_j)$ towards 0 (respectively 1) implies that paths from these two monitors are similar (respectively non-similar).

To observe the number of positions where both vectors $\vec{m}_i$ and $\vec{m}_j$ are identical, we compute the *Relative Inverse Hamming distance* as:

$$\overline{rH(M_i, M_j)} = \frac{\overline{H(\vec{m}_i, \vec{m}_j)}}{|\vec{m}_j|}. \quad (6)$$

where $\overline{H(\vec{m}_i, \vec{m}_j)}$ is the inverse of the Hamming distance. The larger $\overline{rH(M_i, M_j)}$, the more similar paths issued from M_i and M_j.

It is important to note though that the Hamming distance, in case of binary sequences comparison (as it is the case for our Bloom filters) is equal to a *XOR* operation. In other words, the relative Hamming distance provides the percentage of cases where links are coded with different values in the same positions. The inverse relative Hamming distance provides the percentage of cases where the filters contain the same values at the same positions (either the value is set to '1' or '0').

However, recall from Section 2, that the Bloom filters are initialized to '0', and that elements are set to '1' and added to positions that correspond to the hash of that particular element. In order to alleviate the impact of the initialization process on the comparison of the filters, one can compare the positions identically set to '1' in both filters, as an indication of similarity. We introduce the so-called *Bloom Distance*, defined as the following:

$$B(M_i, M_j) = \frac{One(And(\vec{m}_i, \vec{m}_j))}{|\vec{m}_i|}. \quad (7)$$

where $And(\vec{m}_i, \vec{m}_j)$ performs a logical AND between both vectors and $One(\cdot)$ counts the number of '1' in the bit vector in argument.

In this case, the optimum similarity value is given by the theoretical probability of filling the bit vector with ones, which can be easily calculated using Eqn. 2. The closer $B(M_i,M_j)$ to this probability, the more similar paths issued from M_i and M_j are.

4. Analysis

In this section, we evaluate the performance of our three metrics for retrieving path similarity information from Bloom filters. We first discuss our methodology, in particular how we actively collected data (Section 4.1). We next describe the performance metrics we use throughout this evaluation (Section 4.2). Section 4.3 discusses the results while Section 4.4 provides a summary of the main achievements of this section.

4.1. Methodology

For the purpose of our studies in this paper, we collected data using traceroute from 30 PlanetLab machines towards sets of 1000 destinations. The measurement campaign was done between March 31st, 2009 and April 5th, 2009. All these experiments were run concurrently so as to experience the same network conditions. We used the native PlanetLab traceroute. Our dataset is freely available.[2] Regarding the geographical situation of PlanetLab monitors used during the probing phase, Europe was the most represented continent, as shown in Table 1, followed by America and Asia (a single monitor in China).

The traceroute destinations were randomly selected within the *Archipelago* set of 3,000,000 destinations [3]. Archipelago is skitter' successor and has been deployed since September 2007. Destinations in Archipelago are

[2] See http://planete.inrialpes.fr/similarity_data/Traceroute-Similarity.tar.gz

Table 1
PlanetLab monitors geographical situation.

Continent	# Monitors
Europe	25
America	4
Asia	1

selected from all routed/24's. As we will be, in the following, interested in the impact of overlapping destinations, i.e., monitors in the system share entirely the same set of destinations or a given proportion of destinations in their set of targets, we defined nine different destinations sets. The geographical repartition of destinations is provided in Table 2. Note that the geolocation of traceroute targets has been done using *IP2Location* [16]. We see on Table 2 that most of the destinations are located in America, followed by Europe and Asia. The proportion of destinations in Oceania and Africa is roughly negligible.

From our dataset, we removed one monitor, located in Europe (Italy), as it was unable to perform correctly the traceroutes: it stopped discovering interfaces after the second hop. Note that, for the rest of the monitors, on average, 85% of the paths were incomplete, i.e., did not terminate at the destination.

When tracerouting, some routers along the path might reply with invalid address, typically because of mis-configuration, or might not respond to probes. For our study, we chose to remove links containing at least one invalid address and links containing at least one non-responding node. The addresses that we consider as invalid are a subset of the special-use IPv4 addresses described in RFC 3330 [17]. Specifically, we eliminate visits to the private IP address blocks 10.0.0./8, 172.16.0.0/12, and 192.168.0.0/16. We also remove the loopback address block 127.0.0.0/8. On average, 15% of links discovered by a monitor were categorized as invalid.

Regarding Bloom filters, the hashing was emulated with random numbers. We simulated randomness with the Mersenne Twister MT19937 pseudo-random number generator [18].

4.2. Performance metrics

To characterize the performance of our similarity tests, we use the classical false/true positives/negatives indicators. Let us first define specifically what would be defined as a monitor similar to another one. First, we use the $RSIM(\cdot,\cdot,\cdot)$ metric as a reference of path similarity, as it uses 'raw data' exchange to compare routes in a hop by hop way (see Eqn. 4, Section 3). Since our Bloom filters-based algorithms compact exchanged data, and as such, the metrics we introduce alike should be evaluated considering raw data similarity as a reference. We then consider that monitors routes are actually similar if the $RSIM(\cdot,\cdot,\cdot)$ metric returns the maximum value. It can also happen that we look for the n most similar monitors to another one. In this case, we consider the set of $RSIM(\cdot,\cdot,\cdot)$ values corresponding to the highest n values, and we refer to those monitors by $\mathcal{TOP}_n$.

Table 2
Traceroute destinations geographical situation.

Continent	Destination proportion (%)
Africa	0.95
America	41.11
Asia	29.27
Europe	26.62
Oceania	2.03

When looking for the monitor that is the most similar to its observations, a monitor M_i looks for the monitor M_j which corresponds to $max_{M_j \in \mathbb{M}} RSIM(M_i, M_j, \mathbb{D})$. When looking for the set of $\mathcal{TOP}_n$ similar monitors, the most similar monitors involve those which returns the first top n $RSIM(\cdot,\cdot,\cdot)$ values.

A *negative* is a non-similar monitor, according to the $RSIM(\cdot,\cdot,\cdot)$ metric, which should therefore be rejected by the similarity test. In others words, a negative is a monitor for which the $RSIM(\cdot,\cdot,\cdot)$ value compared to another monitor is not the maximum (or not within the set of n values if addressing the $\mathcal{TOP}_n$ similarity). On the other hand, a *positive* is a monitor that has been considered as similar to another monitor by the $RSIM(\cdot,\cdot,\cdot)$ metric, i.e., that has the maximum $RSIM$ value compared to the set of other monitors. The number of negatives (respectively positives) in the population comprising all the monitors comparisons is $\mathcal{P}_N$ (respectively $\mathcal{P}_P$).

For each metric, we identify a false positive as the case when a non-similar monitor has been wrongly identified by the specific metric as similar to the monitor in consideration. A false negative is a similar monitor that has been wrongly rejected by the specific metric as a non-similar monitor. True positives (respectively true negatives) are positives (respectively negatives) that have been correctly reported by the specific metric and therefore identified as similar monitors to the monitors to which they are compared. The number of false negatives (respectively false positives, true negatives, and true positives) reported by the metrics is $\mathcal{T}_{FN}$ (respectively $\mathcal{T}_{FP}$, $\mathcal{T}_{TN}$, and $\mathcal{T}_{TP}$).

We use the notion of *false negative rate* (FNR) which is the proportion of all the similar monitors that have been wrongly reported as non-similar (negatives) by the metric. The FNR is defined as:

$$\mathrm{FNR} = \frac{\mathcal{T}_{\mathcal{FN}}}{\mathcal{P}_P}. \tag{8}$$

The *false positive rate* (FPR) is the proportion of all the non-similar monitors that have been wrongly reported as similar (positives) by the metric and is defined as

$$\mathrm{FPR} = \frac{\mathcal{T}_{\mathcal{FP}}}{\mathcal{P}_N}. \tag{9}$$

Similarly, the *true positive rate* (TPR) is the proportion of similar monitors that have been rightly reported as similar by the metric.

$$\mathrm{TPR} = \frac{\mathcal{T}_{\mathcal{TP}}}{\mathcal{P}_P}. \tag{10}$$

Finally, the *true positive test fraction* (TPTF) is the proportion of positive tests that correctly identified similar monitors:

$$\mathrm{TPTF} = \frac{\mathcal{T}_{TP}}{(\mathcal{T}_{TP} + \mathcal{T}_{FP})}. \quad (11)$$

4.3. Results

In this section, we discuss the relevance of using Bloom filters for compacting path similarity information as well as the accuracy of our metrics for retrieving path similarity information from Bloom filters. We first show the advantage of Bloom filters in terms of compression (Section 4.3.1) and, next, evaluate factors that can influence the accuracy of our metrics for detecting compact path similarity. These factors are: the Bloom filter parameters (Section 4.3.2), the proportion of destinations overlap (Section 4.3.3), and the number of destinations probed by monitors (Section 4.3.3). In order to avoid biased statistical study, as these factors are interdependent, the factor being studied changes and the remaining are fixed. For example, evaluation of Bloom filter parameters may ignore changes in the overlapping proportion.

4.3.1. Bloom filters gain

Fig. 3 provides an insight into the gain of using Bloom filters instead of exchanging a list of links between monitors. The horizontal axis gives the 29 monitors, while the vertical axis gives the amount of bits sent by a given monitor. The curve labeled "list" is calculated based on the average number of links discovered by a given monitor over the nine destinations sets. This mean is then multiplied by 64, i.e., two 32-bits IP addresses. We determined the 95% confidence interval for the mean based on the Student t distribution. However, these confidence intervals, although being plotted in Fig. 3, are too tight to clearly appear.

As expected, the usage of Bloom filter provides an interesting compression ratio. With a vector made of 45,000 bits (a value selected for providing a very low false positive rate), we are already able to reduce the bandwidth consumption by a factor of 4. Note that, obviously, using a smaller bit vector will provide a stronger compression ratio. It is also worth to notice that some additional savings are possible by applying the compression techniques described by Mitzenmacher [15].

However, it would be a matter of concern if an higher compression ratio comes with an accuracy loss in the path similarity information. This is exactly the point we investigate in the following section.

4.3.2. Bloom filter parameters

In this section, we study the impact of varying the Bloom filter parameters on the similarity accuracy.

As explained in Section 2, a Bloom filter is driven by three key parameters: m, the bit vector size, k, the number of hash functions, and n, the number of elements to record in the Bloom filter. In our case, when considering a complete overlap of destinations between monitors and each monitor probing a set of 1000 destinations, on average, 4100 links have to be inserted in the filter (i.e., $n = 4100$).

Fig. 4 shows the evolution of the theoretical false positive rate (the colorbar in log-scale) when n is fixed (i.e., 4100) while k (horizontal axis) and m (vertical axis, in log-scale) vary. We see that for small vector size (i.e., less than 50,000 bits), the false positive rate is very high, whatever the quantity of hash function used. On the contrary, for a very large vector size (i.e., higher or equal to 500,000 bits), the false positive rate is very low. However, in that case, it comes at the expense of a smaller compression rate. Note that using a large set of hash function also increases the required computation time.

Fig. 4 provides thus a way to select parameters k and m according to the specific needs of the application. If the application does not care about computation time, a large number of hash functions associated with a sufficiently large bit vector would lead to a very low false positive rate. On the contrary, if the application is sensitive to computation, a lower k value is required. The application sensitivity on false positive will then lead the choice of the bit vector size.

For the reminder of this paper, we will consider the following values for m: 4500, 45,000, and 450,000 while considering k as 1, 3, 5, and 10. Those values are motivated by the fact that they provide a reasonable compression ratio (at least for $m = 4500$, 45,000), a low computation time (at least for $k = 1,3,5$), and the theoretical performance (in term of false positive rate) is good according to Fig. 4.

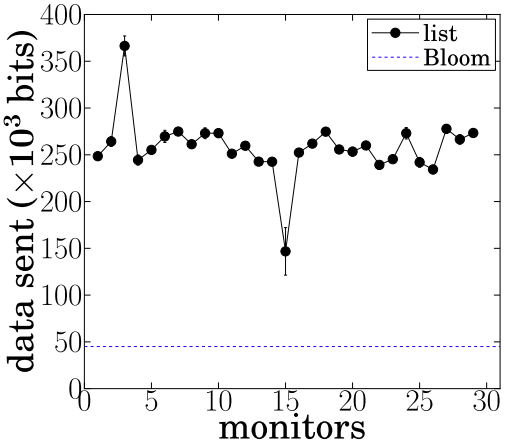

Fig. 3. Compression with Bloom filters.

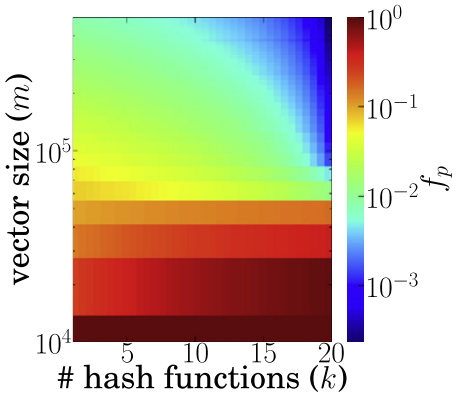

Fig. 4. Bloom filter false positive rate when inserting 4100 elements.

Finally, it is worth to notice that the false positive rate depicted in Fig. 4 is theoretical. Indeed, it assumes (as standard Bloom filter does) that all elements in the Universe share the same likelihood to be queried for a membership test. However, in practice, some elements might more frequent than others. And it might be a concern if such an element triggers a false positive. However, this problematic of evenness/unevenness of elements in Bloom filters is out of the scope of this paper. Interested readers might have a look at [19,20].

For each filter parameter, we plot the TPTF values for the $\mathcal{TOP}_1$ and $\mathcal{TOP}_5$ most similar monitors. This is given by Fig. 5. Each TPTF value is obtained as explained in Section 4.2, i.e., the total number of true positives and false negatives over all monitors for a given similarity metric. It is worth to notice that Figs. 5a and 5b are classified into four portions according to the used number of hash functions ($k \in \{1,3,5,10\}$). Each point located within a given portion, for a given similarity metric, depicts m, the bit vector size. That means that for instance the first dotted point (respectively second, and third), within a given portion in Fig. 5a, corresponds to a m value of 4500 (respectively 45,000, and 450,000).

The main observation retrieved from Fig. 5 is that, an acceptable compression ratio (e.g., $m = 45{,}000$) always provides high TPTF. We see that the proportion of positive tests that are true positives is constantly high, regardless of the number of hash functions chosen, for moderate to quite significant compression ratios used by the filters.

Figs. 5a and Fig. 5b exhibit a slightly worse performance for a vector size of 450,000 bits compared to a size of 45,000 bit. This might appear as being counterintuitive. Indeed, theoretically, a vector size of 450,000 should provide a better result (in term of false positive) compared to a Bloom filter that has 45,000 bits, as depicted in Fig. 4. This performance discrepancy might be explained by an artifact of our measurements. Each monitor encodes path information (i.e., pairs of IP addresses representing a link between two adjacent routers), which is obtained by a traceroute, into a Bloom filter. Nevertheless, when tracerouting, some routers along the path might reply with invalid address (such as a non-publicly routable IP address) or might not respond to probes. In such a case, a set of links along the path will be missed. Further, it should also be noted that, in our studied data set, some traceroute measurements do not terminate at the destination (i.e., the destination does not reply to probes due to a firewall, for instance). In so doing, the theoretical false positive rate computed based on Eqn. 3 is different with respect to the one obtained frow raw traceroutes. The "discrepancy" obtained is probably due to the missing links along the path.

However, the proportion of correct positive tests decreases each time we consider a too high compression ratio. It should be noted that if a small bit vector size is used ($m = 4500$), the proportion of true positives is not acceptable. Even if in the first portion, using $k = 1$, we observe a high TPTF, we still consider the test performance as non-acceptable since the false positive rates of the filter in this case are too high.

In the light of this, we can conclude that as long as the compression ratio is not too high, whatever the number of hash functions is (i.e., even $k = 1$), the three metrics we propose produce a large number of positive tests, catching most of the similarities among monitors. For the remainder of this paper, we arbitrarily tune a Bloom filter with $m = 45{,}000$ and $k = 5$.

4.3.3. Destination sensitivity

In this section, we evaluate how sensitive are the $B(\cdot,\cdot)$, $rH(\cdot,\cdot)$, and $\overline{rH(\cdot,\cdot)}$ metrics to destinations. We evaluate this sensitivity on two planes: the destinations choice between monitors and the number of destinations probed. The destinations choice is expressed by the overlapping of destinations sets between monitors. To this end, we picked nine different destinations sets, each of them corresponding to a certain proportion of destinations overlap. We considered an overlap of 0% (i.e., all the monitors probe different destinations), 5%, 10%, 25%, 50%, 75%, 90%, 95%, and 100% (i.e., all the monitors probe the same destinations set).

Fig. 6 illustrates how our metrics are sensitive to the choice of destinations with respect to different overlapping proportion, each monitor probing a set of 1000 destinations. Each curve plots the TPTF values for the nine destinations overlaps.

As expected, a large overlapping proportion results in more accurate metrics, classifying correctly a larger portion

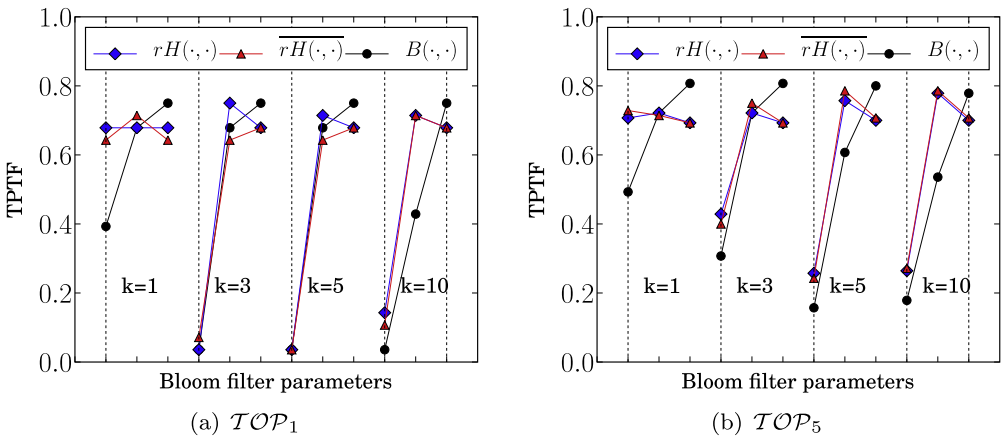

Fig. 5. Effects of Bloom filter parameters for a full overlap of destinations.

of monitors as similar. This is exemplified in Fig. 6a with the increase trend of the $B(\cdot,\cdot)$ curve. Despite the fact that the true positive test fraction curves clearly exhibit positive slopes, one should note that these rates increase much slower than the increase in overlapping proportions. That is to say that as long as we consider a non-completely independent set of destinations, that is probed by monitors, it is very likely that the three metrics achieve constant similarity detection, that are roughly equal to 70%.

To further evaluate the impact of overlapping proportion on the Bloom filter-based similarity metrics, and to evaluate the efficiency of the similarity classification test, we plot in Fig. 7 the *receiver operating characteristic* (ROC) curves for different overlapping proportions.

These plots show, for each metric, the point corresponding to the false positive rate (FPR) along the *X*-axis and to the true positive rate (TPR) along the *Y*-axis, with one portion for each destinations overlapping (90%, 95%, and 100%). The TPR and the FPR values are obtained by comparing the similarity computed over all monitors by $rH(\cdot,\cdot)$, $\overline{rH(\cdot,\cdot)}$, and $B(\cdot,\cdot)$ metrics with respect to $RSIM(\cdot,\cdot,\cdot)$. Obviously, the closer to the upper left corner of the graph a point, the better, since such points correspond to high true positive rates (i.e., a high proportion of positives being reported as such by the test) for low false positive rates (i.e., a small proportion of negatives incorrectly reported as positives).

We observe in Fig. 7 that, from this perspective, as already suggested in Fig. 6, our metrics for evaluating the path similarity through Bloom filters perform very well. Indeed, The FPR is very low (i.e., less than 2%) while the TPR is high (i.e., above 65%). In particular, $rH(\cdot,\cdot)$ and $\overline{rH(\cdot,\cdot)}$ reach 80% of TPR when looking for the $\mathcal{TOP}_5$ most similar monitors for a full overlap of destinations.

The ROC curves shown in Fig. 7 are computed over all monitors by our three metrics with respect to $RSIM(\cdot,\cdot,\cdot)$ which considers raw traceroute data sharing. It should then be noted that the obtained results are also observable from raw traces. We do not claim then that our findings are related to the Bloom filter encoding, but we do show through these results that the property exhibited by RSIM, consisting of being barely insensitive to destinations overlap, is preserved after the Bloom filter encoding.

Comparing the three metrics, in both Fig. 6 and 7, we see that $B(\cdot,\cdot)$ performs worst compared to $rH(\cdot,\cdot)$ and $\overline{rH(\cdot,\cdot)}$. Although, in Fig. 7a, the false positive rates exhibited by the test in all the metrics are roughly similar – such values are too small to notice any significant difference-,

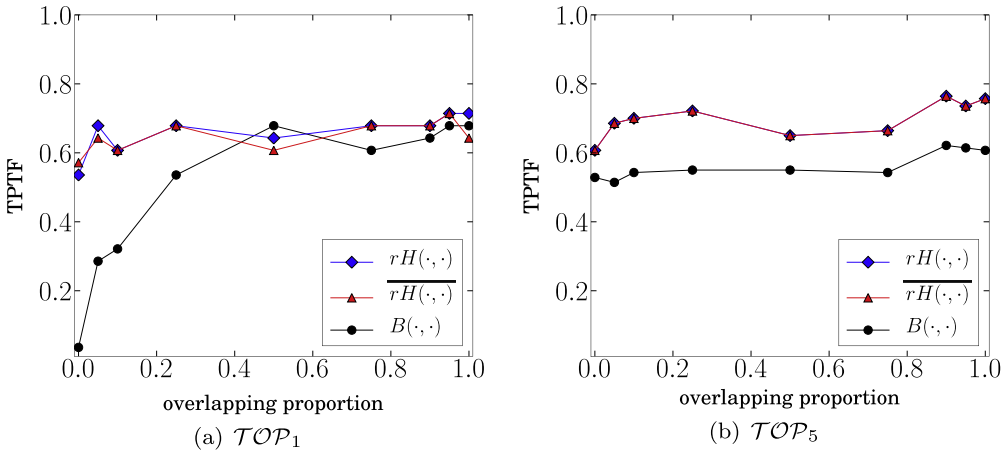

Fig. 6. Impact of destination sets on similarity.

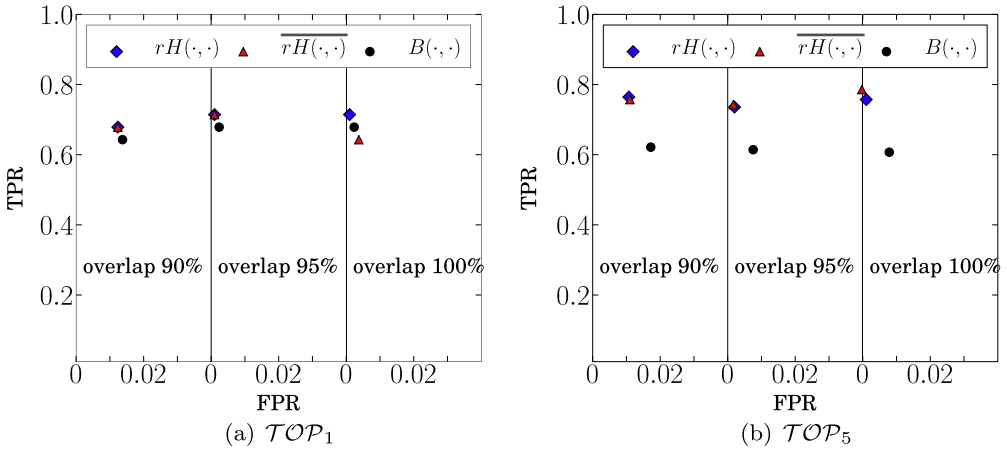

Fig. 7. Destinations measurability.

better performances of the $rH(\cdot,\cdot)$ and $\overline{rH(\cdot,\cdot)}$ metrics are particularly obvious for $\mathcal{TOP}_1$ (Fig. 6a). We see that $B(\cdot,\cdot)$ is quiet sensitive to the destinations choice, detecting on average 15% less true positives than the two other metrics. On the contrary, $rH(\cdot,\cdot)$ and $\overline{rH(\cdot,\cdot)}$, provide mostly a constant true positive test fraction of 70% with little sensitivity to destinations choice.

The reason that either $rH(\cdot,\cdot)$ and $\overline{rH(\cdot,\cdot)}$ outperform $B(\cdot,\cdot)$ is due to the fact that the $B(\cdot,\cdot)$ metric takes into account partial information in the filters (only bits set to 1′) whereas $rH(\cdot,\cdot)$ and $\overline{rH(\cdot,\cdot)}$ metrics consider the whole bit vector, a bit set to '0' being a valuable information as it indicates that no information has been recorded in that position in the filter.

So far, the path similarity study consisted in each monitor probing a set of one thousand destinations. It would also be important to determine if the number of destinations to be probed has some influences on the path similarity between monitors. This might potentially lead to a scalability issue. Indeed, it might be a concern if, for providing quite accurate results, a large set of destinations must be probed, traceroute being known to be intrusive. On the contrary, if probing a small set of destinations is enough to obtain accurate results, this would be an incentive for the deployment of any system requiring the estimate of similarity.

Actually, the number of destinations parameter needs to be studied from two sides: a first side is related to the coding of paths towards the set of destinations, and hence the impact of the number of destinations on the exchanged Bloom filters among monitors. The first question to answer is then how the number of destinations can impact the accuracy of the metrics we propose to detect similarity between monitors. The second aspect of the problem is much more related to path similarity itself, and what we need to answer is "what is the number of destinations monitors need to probe in order to achieve desirable similarity results, in terms of high true positive detection of similarity and low false positives".

To study the first aspect, we consider a full overlap of destinations and vary the number of destinations probed. We take into account the following destinations set cardinality: 1, 10, 25, 50, 75, 100, 250, 500, 750, and 1000. Fig. 8 shows the TPTF (vertical axis) when the number of destinations probed varies (horizontal axis). In this first experiment, similarity metrics we propose are again considered having as a reference the similarity as returned by the $RSIM(\cdot,\cdot,\cdot)$ metric. Note that the TPTF values in Fig. 8 show how accurate $rH(\cdot,\cdot)$, $\overline{rH(\cdot,\cdot)}$, and $B(\cdot,\cdot)$ are if monitors probe different set of destinations, while assuming that such a choice is not impacting the $RSIM(\cdot,\cdot,\cdot)$ metric itself.

The first interesting observation is that the accuracy of similarity detection in terms of true positives test fraction and false positive rates (not shown), is slightly constant if we consider a minimum number of 500 destinations. This gives to the three metrics a high utility if we consider that the overhead gained by not exchanging traceroute data is quadrupled when using Bloom filters. As expected, when monitors probe a small set of destinations, the accuracy is low. However, acceptable similarity detection can be considered when monitors probe sets of overall a hundred of destinations. This observation is generalized when looking for the $\mathcal{TOP}_5$ most similar monitors. In essence, when considering a very small number of destinations, the similarity as computed by the metrics $rH(\cdot,\cdot)$, $\overline{rH(\cdot,\cdot)}$, and $B(\cdot,\cdot)$ are skewed by outliers inside the set of destinations. However, a set of destinations that are close to 100, is closely representative enough of the overall population of traceroute that a monitor can perform, being not shadowed by outliers while preserving a high degree of generality.

The choice of the optimal number of destinations to be probed would then require a compromise between the number of destinations where the Bloom filter-based metrics are not impacted, and the number of destinations where similarity using "raw" data would actually provide accurate results.

Tables 3 and 4 provides the proportion of monitors finding the same $\mathcal{TOP}_1$ (Table 3) most similar monitor and the same $\mathcal{TOP}_5$ (Table 4) most similar monitors when probing different sets of destinations. Each column (respectively row) in both tables represents the number of destinations probed by a monitor. Each cell T_{ij} gives the proportion of monitors that return the same most similar monitors when

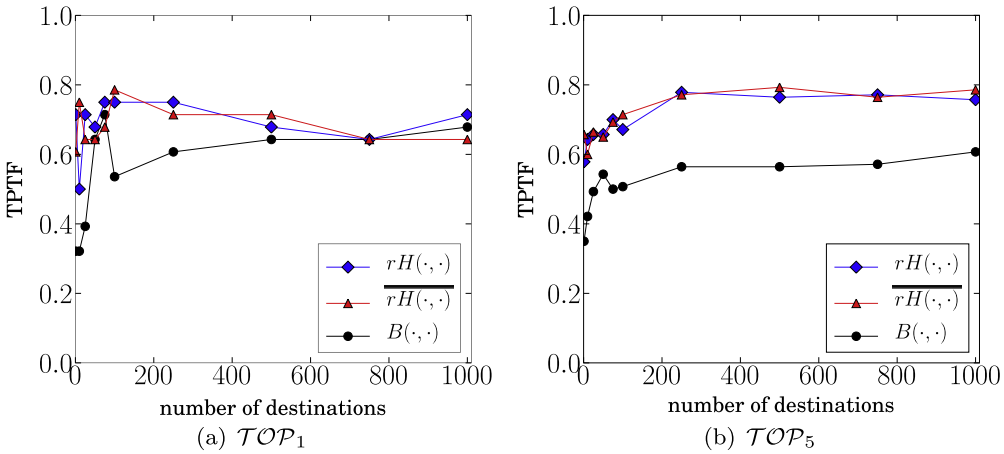

Fig. 8. Effects of the number of destinations probed.

Table 3
$\mathcal{TOP}_1$: incidence of the number of destinations ($\overline{rH(\cdot,\cdot)}$).

# Destinations	1	10	25	50	75	100	250	500	750	1000
1	1.00									
10	0.42	1.00								
25	0.42	0.53	1.00							
50	0.42	0.50	0.75	1.00						
75	0.42	0.53	0.85	0.89	1.00					
100	0.39	0.53	0.67	0.78	0.78	1.00				
250	0.39	0.53	0.71	0.71	0.75	**0.89**	1.00			
500	0.39	0.50	0.64	0.71	0.67	**0.82**	**0.82**	1.00		
750	0.39	0.50	0.67	0.78	0.71	**0.85**	**0.85**	**0.89**	1.00	
1000	0.39	0.46	0.60	0.71	0.64	**0.82**	**0.82**	**0.82**	**0.89**	1.00

probing a set of i destinations or a set of j destinations. Table 3 and 4 indicates results for the $\overline{rH(\cdot,\cdot)}$ metric.

A first global look at Table 3 and 4 suggests that, for obtaining accurate results, a certain amount of probing must be done, on the order of hundred of destinations. Probing so would allow one to obtain, in 80% of the cases, the same $\mathcal{TOP}_1$ most similar monitor as larger probing campaign. On the contrary, retrieving the $\mathcal{TOP}_5$ most similar monitors implies more probing, at least 250 destinations, for reaching the same accuracy level than $\mathcal{TOP}_1$.

A conclusion from Table 3 and 4 is that once a certain threshold of destinations probed has been reached (100 for $\mathcal{TOP}_1$ and 250 for $\mathcal{TOP}_5$), it is not necessary to burden more the network as it will not increase so much the similarity accuracy. This fact is illustrated, in Table 3 and 4, by bold values.

Note that we found similar results for the two other metrics, i.e., $rH(\cdot,\cdot)$ and $B(\cdot,\cdot)$.

4.4. Summary

In Section 4.3, we evaluated the behavior of our metrics for comparing several large sets of information encoded as Bloom filters. Those sets were made of topological data (i.e., links between routers) collected through traceroute.

During our analysis, we mostly focused on two planes: Bloom filters and destination sensitivity. For both planes, we showed that the metrics $rH(\cdot,\cdot)$ and $\overline{rH(\cdot,\cdot)}$ provide globally better performance results than $B(\cdot,\cdot)$.

Table 5 aims at highlighting the main lessons learned from Section 4.3. Regarding the Bloom filter plane, the number of hash functions does not impact the accuracy. On the contrary, the bit vector size, m, has some influence. A too high compression ratio would lead to bad performance. For the destination plane, it is sufficient to probe between 100 and 250 destinations, while the overlap of destinations sets between monitors does not influence the performance accuracy.

Table 5
Summary

Plane	Accuracy	
Bloom filter	k	$\emptyset$
	m	>4500
Destination	Overlap	$\emptyset$
	Quantity	[100–250]

Despite, on average, 85% of the paths were incomplete in our data set, all above observations indicate that the data set used in this paper can plausibly represent a rich cross-section of the whole situation on today's Internet, and thus allow us to quantify path similarity with high accuracy.

5. Related work

Since the late 1990s, Internet topology discovery has been an extensive research field [1]. This research activity focused mostly on developing efficient traceroute-like tool or on modeling the Internet topology. Although there are many potential applications, the problem of path similarity

Table 4
$\mathcal{TOP}_5$: incidence of the number of destinations ($\overline{rH(\cdot,\cdot)}$)

# Destinations	1	10	25	50	75	100	250	500	750	1000
1	1.00									
10	0.39	1.00								
25	0.40	0.54	1.00							
50	0.37	0.53	0.67	1.00						
75	0.38	0.52	0.71	0.75	1.00					
100	0.41	0.50	0.67	0.73	0.76	1.00				
250	0.33	0.46	0.62	0.68	0.70	0.71	1.00			
500	0.31	0.47	0.61	0.68	0.69	0.68	**0.86**	1.00		
750	0.32	0.48	0.60	0.70	0.67	0.66	**0.82**	**0.89**	1.00	
1000	0.33	0.47	0.61	0.71	0.67	0.68	**0.83**	**0.88**	**0.93**	1.00

does not appear to have been the subject of much study. Only a few work has been done on path similarity or diversity.

For instance, Teixeira et al. evaluate the IP-level path diversity between *Points of Presence* (PoPs), i.e., a collection of routers owned by an AS in a specific location (city or suburb). Based on real ISP dataset, Teixeira et al. found that all pairs of PoPs have, at least, two disjoint paths between them. However, Teixeira et al. do not discuss if this diversity might be observed at the Internet scale, neither how a given application might use this diversity.

Hu and Steenkiste propose a metric, *RSIM*, for evaluating end-to-end path similarity. This metric is based on the number of common links between two monitors. A specificity of RSIM is that it is a bi-directional metric, i.e., it provides the similarity between monitor *A* and *B*, but also between *B* and *A*. An application of RSIM is for scalable bandwidth estimation [21]. However, it is not explained how this similarity information might be exchanged between monitors. Pathak et al. use path similarity at the AS and router level for evaluating the asymmetry of delay in the Internet [22]. Roughly, they propose a metric somewhat equivalent to RSIM but without taking into account the bi-directionality of links.

Works has also been done on identifying relays for alternative paths in large scale networks to increase route diversity [7,8]. Indeed, multipath routing has been proposed to better reduce maximum load on nodes and congestion. The use of alternative paths improves the quality of service of communication across the Internet. For instance, Agapi et al. seek to find good relay nodes by using path similarity-based synthetic coordinates [8]. The key idea is based on the assumption that if a relay is suitable for a given path, it is also likely to be good for other similar paths. They define path similarity between paths P_1 and P_2 as the probability that a relay that is good for P_1 is also good for P_2. Note that a set of relay could be exchanged between nodes using Bloom filters. This has not been investigated by Agapi et al.

Bloom filters have been extensively studied, particularly in networking due to their capacities in bandwidth savings when membership information must be exchanged between monitors [14]. Recent works used Bloom filters for exchanging topology information between traceroute-like monitors [19,23]. However, authors do not investigate how these Bloom filters might be used to evaluate path similarity.

6. Conclusions and future work

In this paper, we presented a Bloom filter-based approach to measure and exchange Internet path similarity between monitors. Our solution aims at considerably decreasing the amount of data sent through the network and ensuring the secrecy of topology information, while maintaining a high level of confidence in the measurability of path similarity.

It is based on compacting traceroute data through the use of Bloom filters, exchange those Bloom filters, and then compute similarities or dissimilarities based on a set of metrics to compare each pair of monitors' Bloom filters. Our approach does not rely on destination overlapping, and as such is unaffected by the choice of the set of destinations that the monitors can probe. In fact, we have shown that for different percentages of destinations overlaps, and for even an acceptable low number of probed destinations, our metrics still provide very good performance, distinguishing clearly between similar monitors paths.

To the best of our knowledge, this is the first such general method capable of providing accurate results while maintaining a very low overhead in exchanging traceroute information. Optionally, our method allows one to hide the network topology from monitors to the set of destinations they are monitoring. Since hops are hashed through the Bloom filters, topological information is then hidden and immune from any malicious data sniffing activity. Encryption of raw data provides also efficient confidentiality for data, but does not resolve the problem of overhead.

In practice, we introduced a simple way to study path similarity between monitors through compacted traceroute information. We choose to illustrate this study through traceroute information, which is at the very core of any Internet topology. This leads us to believe that our proposed similarity detection approach can effectively identify similar monitors in very many CDNs monitors deployment without any major software change. However, the way to code "raw" information and the metrics introduced to compare Bloom filters are generic enough to allow for their application on very many other topology characteristics information needed to be exchanged. We could, for instance, include information describing available bandwidth estimation for paths, as complementary to traceroute paths, and encode such an information into Bloom filters. Again, the gain in terms of reduced overhead would allow an easier and more practical information exchange among monitors. Note that, despite we performed measurement in an IPv4 environment, our technique is independent of the used IP version. This means that in a world where IPv6 would be largely deployed, our technique would present stronger interest in term of compression.

The measurements collected in this paper were deliberately tested on monitors and destinations chosen at random in the real Internet, with a realistic high percentage of non-complete traceroute, although their similarities or dissimilarities increase with traceroute paths fullness. Despite the possible non-optimal similarity measurements resulting from such a choice, the results obtained show the effectiveness of our method in characterizing path similarity with very low overhead. Nevertheless, given the gain afforded by our approach, one can envisage that CDNs may readily want to deploy monitors within their network to offer enhanced services to their customers, and to select well suited set of controlled destinations inside the core network. Such business driven strategic deployment can only improve the measurability of our similarity metrics and thus improve the path similarity accuracy, with much higher similarity detection rates than the lower bound reported in this paper. We have also shown that the false

positive rates are very low, promising not considering monitors to be similar while it is not actually the case. This feature can be exploited when deploying redundant backup content servers. In this case, we insure (with very high confidence) that the backup server, if solicited, will continue to serve customers under the same conditions than the previous server.

The schemes presented in this paper are generic and might be applied in any context in which large sets of information must be exchanged and compared. Future work should reveal the efficiency of our techniques under those contexts. We further aim at evaluating them in a geolocalisation context.

Acknowledgements

This work has been partially supported by the European Commission-funded 223936 ECODE project and the European Commission-fund 27489 ANA project.

Mr. Donnet's work is supported by the FNRS (Fonds National de la Recherche Scientifique, rue d'Egmont 5 – 1000 Bruxelles, Belgium.).

References

[1] B. Donnet, T. Friedman, Internet topology discovery: a survey, IEEE Communications Surveys and Tutorials 9 (4) (2007) 2–15.
[2] V. Jacobson et al., Traceroute, Main Page, UNIX, 1989, See source code: <ftp://ftp.ee.lbl.gov/traceroute.tar.gz>.
[3] K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, Internet mapping: from art to science, in: Proc. IEEE Cybersecurity Applications and Technologies Conference for Homeland Security (CATCH), 2009.
[4] A. Schmitt et al., La météo du net. <http://www.grenouille.com/> (ongoing service since 2000).
[5] P. Radoslavov, H. Tangmunarunkit, H. Yu, R. Govindan, S. Shenker, D. Estrin, On Characterizing Network Topologies and Analyzing their Impact on Protocol Design, USC–CS–TR 00–731, Computer Science Department, University of Southern California (February 2000).
[6] R. Teixeira, K. Marzullo, S. Savage, G. Voelker, In search of path diversity in ISP networks, in: Proc. ACM SIGCOMM Internet Measurement Conference (IMC), 2003.
[7] T. Fei, S. Tao, L. Gao, R. Guerin, How to select a good alternate path in large peer-to-peer systems? in: Proc. IEEE INFOCOM, 2006.
[8] A. Agapi, T. Kielmann, H. Bal, Synthetic coordinates for disjoint multipath routing over the Internet, in: Proc. CoreGRID Symposium, 2007.
[9] N. Hu, P. Steenkiste, Quantifying Internet end-to-end route similarity, in: Proc. Passive and Active Measurement Workshop (PAM), 2006.
[10] N. Hu, O. Spatscheck, J. Wang, P. Steenkiste, Optimizing network performance in replicated hosting, in: Proc. Web Caching and Content Distribution Workshop (WCW), 2005.
[11] S. Ratnasamy, M. Handley, R. Karp, S. Shenker, Topologically-aware overlay construction and server selection, in: Proc. IEEE INFOCOM, 2002.
[12] S. Bakira, Approximate server selection algorithms in content distribution networks, in: Proc. IEEE International Conference on Communications (ICC), 2005.
[13] B.H. Bloom, Space/time trade-offs in hash coding with allowable errors, Communications of the ACM 13 (7) (1970) 422–426.
[14] A. Broder, M. Mitzenmacher, Network applications of Bloom filters: a survey, Internet Mathematics 1 (4).
[15] M. Mitzenmacher, Compressed Bloom filters, IEEE/ACM Transactions on Networking 10 (5).
[16] Hexasoft Development Sdn. Bhd, IP address geolocation to identify website visitor's geographical location. <http://www.ip2location.com>.
[17] IANA, Special-use IPv4 addresses, RFC 3330, Internet Engineering Task Force (September 2002).
[18] M. Matsumoto, T. Nishimura, Mersenne Twister: a 623-dimensionally equidistributed uniform pseudorandom number generator, ACM Transactions on Modeling and Computer Simulation 8 (1) (1998) 3–30.
[19] B. Donnet, B. Baynat, T. Friedman, Retouched Bloom filters: allowing networked applications to trade off selected false positives against false negatives, in: Proc. ACM CoNEXT, 2006.
[20] J. Bruck, J. Gao, A. Jiang, Weighted bloom filter, in: Proc. IEEE Internationl Symposium on Information Theory (ISIT), 2006.
[21] N. Hu, P. Steenkiste, Exploiting Internet route sharing for large scale available bandwidth estimation, in: Proc. Internet Measurement Conference (IMC), 2005.
[22] A. Pathak, H. Pucha, Y. Zhang, C. Hu, M. Mao, A measurement study of Internet delay asymmetry, in: Proc. Passive and Active Measurement Conference (PAM), 2008.
[23] B. Donnet, T. Friedman, M. Crovella, Improved algorithms for network topology discovery, in: Proc. Passive and Active Measurement Workshop (PAM), 2005.

Benoit Donnet received his MS degree in computer science from the Institut d'Informatique of the Facultes Universitaires Notre Dame De La Paix (Namur – Belgium) in 2003. Mr. Donnet received his Ph.D. Degree in computer science form the Universite Pierre et Marie Curie in 2006. He is currently FNRS fellow at the Universite catholique de Louvain in the IP Networking Lab (http://inl.info.ucl.ac.be). His research interests are in Internet measurements, focusing on scalable measurement techniques, Bloom filters, and traffic engineering techniques.

Bamba Gueye received the B.Sc. in Computer Science from the Universite Cheikh Anta Diop de Dakar, Senegal. He received the M.Sc. degree in Networking in 2003 and the Ph.D. degree in computer science in 2006, both from the Universite Pierre et Marie Curie, France. He is currently a post doc researcher at the University of Liege in the Research Unit in Networking group (http://www.run.montefiore.ulg.ac.be). His research interests include Internet measurements characterizing the Internet and measurement-based geolocation.

Mohamed Ali Kaafar received his Eng. degree and MS degree in computer science and Comp. Networking from the Ecole Nationale des Sciences Informatique in Tunisia in 2003 and 2004. Dr. Kaafar received his Ph.D. degree (performed at INRIA, Fr) in computer science form the Universite Nice Sophia Antipolis in 2007. He is currently a permanent research scientist at INRIA Rhone-Alpes Grenoble (http://planete.inrialpes.fr). His research interests include Internet Security, Anomaly detection and Internet Measurements.