Future Generation Computer Systems 39 (2014) 55–66

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Fast dynamic internet mapping

Mehiar Dabbagh, Naoum Sayegh, Ayman Kayssi *, Imad Elhajj, Ali Chehab

Department of Electrical and Computer Engineering, American University of Beirut, Beirut 1107 2020, Lebanon

HIGHLIGHTS

- Experiments show that routing changes have regular temporal behavior.
- We propose a methodology to extract and exploit these temporal regularities.
- Our methodology achieves up to 66% reduction in a number of probing packets.
- Our methodology also achieves 90% reduction in the scanning time.
- Our approach overcomes the problems that arise in the presence of load balancers.

ARTICLE INFO

Article history:
Received 31 January 2013
Received in revised form
28 December 2013
Accepted 21 April 2014
Available online 1 May 2014

Keywords:
Traceroute
Internet topology
Load balancers

ABSTRACT

Probing network devices using the Traceroute tool is a most popular network monitoring technique for troubleshooting and network mapping. Distributed scanning systems perform Traceroute periodically in order to detect routing anomalies. We propose in this paper a six-step methodology for creating a more efficient profile-based probing strategy that reduces both the number of probes and the time needed to map the network topology. Our proposed methodology takes the existence of load balancers into account when building the profile-based strategies and thus overcomes any inconveniences that these load balancers may cause. The basic idea behind our methodology is to examine how often routing changes occur in the routing path at the different times of the day. This insight will be used to provide a high probing weight for parts of the routing path that change frequently and a lower probing weight for parts that change rarely during the different time periods of the day. Since routing changes may occur more frequently at certain periods of the day, we propose an approach to determine the duration of the periods when the routing changes occur with a similar frequency. A profile-based probing strategy is later assigned to each one of these periods separately in order to further reduce the required number of probes. The experimental results show that our approach exploits temporal regularities in the routing changes and achieves great savings in the number of probes. In fact, our approach achieves a 66% reduction in the number of probes as compared to classical Traceroute that is launched periodically. This is an important enhancement for systems that scan the Internet repeatedly. Furthermore, we show another enhancement that cuts down scanning time by 90%. This is achieved by scanning multiple hop levels at the same time after marking the probes in order to match them with the received responses.

1. Introduction

Traceroute [1] is a popular tool that is widely used to map the topology of IP networks for many purposes. It can be used to monitor Internet connectivity in order to discover and diagnose errors in network configurations. Furthermore, it helps network managers decide where to place routers [2]. From a network mapping perspective, there are many cases where one needs to obtain a map for the connected networks.

Network managers deploy distributed scanning systems that perform Traceroute periodically in order to monitor the network [3]. A distributed scanning system consists of a number of scanners (also called monitors) that are distributed in different geographical locations within the network. These scanners scan the network periodically by sending a large number of Traceroute probes towards different destinations. The reported paths to reach these destinations from the different scanners are later analyzed and anomalies are reported to the administrators. Three requirements are desired in distributed scanning systems in order to become efficient.

* Corresponding author. Tel.: +961 3965258.
E-mail addresses: mmd43@aub.edu.lb (M. Dabbagh), nss21@aub.edu.lb (N. Sayegh), ayman@aub.edu.lb (A. Kayssi), ie05@aub.edu.lb (I. Elhajj), chehab@aub.edu.lb (A. Chehab).

http://dx.doi.org/10.1016/j.future.2014.04.006
0167-739X/

1.1. Low probing overhead

Although these scanning systems are very useful for network monitoring and traffic engineering, they are associated with an overhead cost. The scanning systems send a large number of probes periodically that waste the available resources as the sent probes do not carry any information but are necessary for network monitoring. In fact, a large amount of monitoring traffic may cause traffic congestions that affect the performance of the network [4]. Furthermore, each packet forwarding operation is associated with an energy cost as the router needs to search the routing table to determine the next hop level in addition to the energy needed to transmit the packet on the link [5]. Thus it is important to reduce the energy and performance overheads associated with network monitoring.

1.2. Scalability and low scanning time

Another problem associated with distributed scanning systems in addition to the waste of resources is that these systems need a considerable time to scan large networks. The reason behind this limitation is that Traceroute starts probing the first hop level and waits to receive a reply for the sent probe before it moves to probing the next hop level. This makes the scanning procedure slow and impractical for large networks.

1.3. Accuracy

The wide deployment of network load balancers throughout the Internet has made classical "tracerouting" prone to errors. These load balancers forward packets to different interfaces based on certain policies in order to balance the IP traffic among the links and hence to optimize resource utilization. As a result, classical Traceroute results in inaccuracies as it may report non-existing links between routers and fails to discover all the paths between a source and a destination on the Internet. This has motivated researchers to explore other techniques to provide more accurate and thorough maps of the network topology. It is worth noting that load balancers are not a special case that rarely exists in Internet networks. According to [6], 89% of all Internet paths are affected with at least one load balancer. Thus it is important to take the existence of load balancers into account in scanning systems in order to have accurate mapping of the network.

There are three types of load balancers used on the Internet [7]:

(a) Per-packet load balancers: packets are forwarded to one of the outgoing interfaces randomly. This guarantees that traffic is divided evenly among the outgoing links but does not ensure that packets will be received in order.

(b) Per-flow load balancers: packets are assigned a flow based on the values of certain fields in the IP, UDP and/or TCP protocol headers. A per-flow load balancer chooses the same outgoing interface for packets belonging to the same flow. This ensures that packets belonging to the same flow will follow the same path and thus should be received in order. Generally, five fields (called the five-tuple fields) are used to identify a flow: the source and destination IP addresses, the source and destination port numbers, and the protocol field.

(c) Per-destination load balancers: these load balancers can be viewed as a special case of per-flow load balancers. A per-destination load balancer determines the outgoing interface based on the destination IP address only.

The above three requirements have motivated researchers to explore different techniques to reduce both the number of probes and the scanning time and to obtain accurate network maps.

We focus in this paper on enhancing the routing path scanning approaches by producing a more efficient profile-based scanning strategy that takes all these three requirements into account. This strategy takes the existence of load balancers into account when it builds the profile of the routing changes in the studied path and thus overcomes any inaccuracy that these load balancers may cause. Our strategy also significantly reduces the required number of probes in addition to the scanning time for systems that run Traceroute regularly. This paper extends our previous work [8], where we provided the general framework for building a profile-based probing strategy. Since routing changes may occur with a different frequency throughout the different times of the day, we provide in this paper a method for determining the beginning, the end and the duration of the periods where routing changes occur with a similar frequency. A separate profile-based probing strategy is proposed next for each one of these periods separately in order to achieve further savings in the number of probes and to make the built profile more accurate.

The rest of the paper is organized as follows: in Section 2, we review state-of-the-art approaches that were proposed to reliably map network topology in the presence of load balancers. We also cover the enhancements that were added to these techniques in order to reduce the probing traffic on the links and show how our work is different from what was previously proposed. In Section 3, we explain our proposed methodology for extracting routing changes and for adjusting probing weights in order to produce a more efficient profile-based probing strategy. In Section 4, we explain the proposed algorithm to determine the duration of the time periods where the routing changes occur with a similar frequency. This is one of the steps of our proposed methodology but is explained in detail in a separate section due to its importance in our analysis. In Section 5, experiments are conducted to illustrate our methodology and results are analyzed to estimate the savings in time and in the number of probes that our approach achieves. In Section 6, we present some conclusions and future work.

2. Related work

We distinguish two directions in the previous work that addresses the enhancement of network topology mapping. The first direction focuses on solving the problems that arise due to the existence of load balancers along the scanned paths. The second direction works on reducing the number of probes that are required for the scanning activity. We explain these two directions in the following sub-sections.

2.1. Methods for reliable network mapping

The authors of [6,7,9,10] proposed solutions for more reliable network mapping in the presence of load balancers. Classical Traceroute [11] works by sending three probes to each hop. These probes are not identical and might have different five-tuple fields. Therefore, problems with Traceroute occur if a per-flow load balancer exists along the path to the destination. In this case, packets will be assigned to different flows (since they have different five-tuple fields) causing false links between routers in the obtained map. A solution to overcome the wrong connections in the presence of per-flow load balancers was presented in [7]. The authors studied the fields that are used by such load balancers to identify a flow. Their study revealed that ToS, ICMP checksum, and ICMP code fields are also used to identify a flow, in addition to the five-tuple fields. After determining the fields that affect per-flow load balancers, the authors suggested the Paris Traceroute algorithm as a modified version of classical Traceroute. Paris Traceroute keeps all the fields that are used to identify a flow constant while probing towards the destination. As a result, all the probes will be assigned to the same flow and thus no false links will be reported. However, this proposed method fails if a per-packet load balancer exists along the scanned paths. Another drawback of the Paris algorithm is that it only returns a single path from the source to the destination.

An attempt to discover all the paths between a source and a destination was presented in [9]. The authors proposed an algorithm called MDA (Multipath Detection Algorithm). The basic idea behind MDA is to send a number of probes with different flow identifiers at hop-level L. If a reply from two interfaces is received, then more probes are sent to L in order to make sure that no third interface at L exists. The procedure is repeated until no new interfaces are detected. Next, MDA iterates by scanning the paths using these identifiers for the next levels until the destination is reached. A main advantage of MDA is discovering all the paths between the source and the destination. However MDA requires a large number of probes to detect all the paths. Furthermore, it does not determine the type of the detected load balancers.

The authors of [10] worked on determining mathematically the number of probes that are sent using MDA such that the probability of missing an interface does not exceed 5%.

Another enhancement to MDA was suggested in [6]. The authors proposed an approach to determine the type of the detected load balancers when the MDA algorithm is used. Determining the load balancer type is done by sending six identical probes to an interface succeeding the load balancer. If the same interface replies all the time, then the load balancer is not a per-packet load balancer. This is based on the observation that the probability of forwarding six identical packets to the same interface in the case of a per-packet load balancer is very low (at 4%). If the load balancer is not a per-packet balancer, one needs next to distinguish whether the load balancer is a per-flow or per-destination load balancer. Thus, six new probes are sent with the same destination but with different port numbers resulting in different flow identifiers. If the same IP interface replies all the time, then the load balancer is a per-destination load balancer. Otherwise, the load balancer is a per-flow balancer.

The work in [12–16] also addressed the problem of reliably mapping the network. The authors proposed to probe the network from multiple scanners. Inaccuracies due to the existence of load balancers are then resolved by analyzing and comparing the collected paths from the different scanners.

The work in [17–20] investigated the problem of selecting the location of the scanners in distributed scanning systems. The authors suggested different approaches based on graph theory to place the scanners such that all the network destinations are reachable within the network. This selection also allows resolving accuracies in any link by comparing the obtained maps from the different locations.

2.2. Methods for reducing the number of probes

The works in [21–23] provide solutions to reduce the probing traffic. The authors in [21] proposed Tracetree, an algorithm that reduces the redundant probing packets when a single scanner is scanning multiple destinations. The basic idea is to start probing the different destinations from the last link and then to scan backwards. The probing stops when a previously detected interface is encountered. Tracetree reduces the number of probing packets significantly for a single scanner. However, further opportunities for reducing redundant probes are possible in distributed scanning systems. A distributed scanning system consists of multiple scanners that are scanning the paths of a large list of IP destinations. An example of a distributed system is Skitter [3]. In fact, Skitter is composed of 18 scanners that are distributed around the world. The scanners send the detected paths to a central repository that combines them together in order to have one big map of the Internet. Tracetree is not the perfect solution to reduce probing traffic for such distributed systems since it focuses on reducing redundant probes in each scanner separately and does not consider the common detected paths among the different scanners.

In [22], the authors showed that a further reduction in the number of probes could be achieved for distributed scanning systems. In such systems, multiple scanners located at different geographical locations scan the paths towards the same destination. In fact, the Internet topology in this case can be viewed as a tree rooted at the destination where scanners are the leaves of the tree. We thus refer to the proposed approach in [22] by Reverse tree. If the scanners in the distributed scanning system communicate with each other, then a coordinated forward probing strategy can be used to scan the network with a small number of probing packets. A scanner stops scanning if it discovers a node that was previously detected by another scanner for the same destination.

A compromise between forward and backward probing was suggested in Doubletree [22] for distributed scanning systems. In order to map the path to reach a destination D, Doubletree starts probing from the middle node and continues probing forward and backward. The backward probing in scanner S_i stops if it discovers an interface that was previously detected by it. On the other hand, forward probing is stopped if S_i discovers an interface that was already detected by another scanner S_j while scanning the path of the same destination D. Details on how DoubleTree may be implemented are provided in [24].

Although Tracetree and Doubletree reduce the probing traffic significantly, they do not address the problems that arise due to the existence of load balancers. To address these problems while keeping the probing traffic small, Fast Mapping was suggested in [23]. Fast Mapping is a framework that uses both MDA and Tracetree. Fast Mapping starts by finding all the paths to reach a destination using MDA. Then, it runs Tracetree periodically in order to detect any change in the paths to the destination. Once a change is detected, Fast Mapping launches MDA again to remap the new paths.

In this paper, we investigate whether path changes follow certain patterns. We examine how often routing changes occur in different parts of the path during different times of the day. Based on the collection and analysis of Traceroute traces, a profile is built for each part of the path indicating how often and when this part changes. This profile is used to set more probing weight for parts that tend to change more frequently than others. On the other hand, lower probing weight will be assigned to the parts that change rarely. This reduces both the probing overhead and the scanning time as it avoids the frequent mapping of the parts of the paths that do not change frequently.

Table 1 provides a quick comparison about what problems in distributed scanning systems that each of the previous approaches that were mentioned in the section covers. Although previous methods were proposed to reduce probing overhead, our work is different from all previous works [3,21–24] that attempted to reduce the number of probes as our approach exploits regularities in routing changes in order to reduce the number of probing packets. Our work takes the existence of load balancers into account when reducing the number of probes and this is a main advantage over the techniques proposed in [3,21,22,24]. Furthermore, we propose in this paper an enhancement that reduces the scanning time significantly. No previous work to the best of our knowledge has addressed reducing the scanning time before. The basic idea behind our enhancement is to scan multiple destinations at the same time instead of waiting to receive a reply before probing the next hop. It is also worth noting that the methods in [3,21–24] can be applied on top of our approach to further reduce probing overhead.

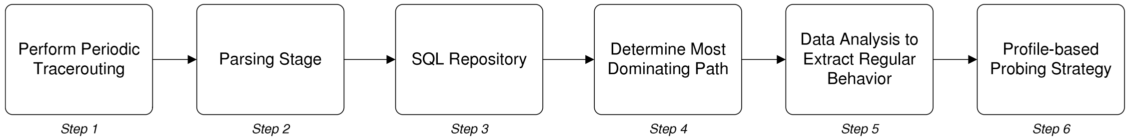

Fig. 1. Proposed profile-based methodology for adjusting probing weights for the different parts of the routing path.

Table 1
Comparison of our approach with the previous work in terms of the addressed problems in distributed scanning systems.

Approach	Reference number	Accurate mapping with load balancers	Reducing probing overhead	Reducing scanning time
Traceroute	[11]	×	×	×
Paris Traceroute	[7]	✓	×	×
MDA	[6,9,10]	✓	×	×
Trace analysis	[12–16]	✓	×	×
Tracetree	[21]	×	✓	×
Reverse tree	[22]	×	✓	×
Double tree	[24]	×	✓	×
Fast mapping	[23]	✓	✓	×
Proposed approach	–	✓	✓	✓

3. Proposed methodology

We explain in this section our methodology for extracting regular routing path changes and for adjusting the probing weight for the different parts of a routing path based on how often routing changes occur in order to minimize the number of probes needed for detecting path changes.

The following notation is used in this paper: we refer to the routing path as R. The length of the routing path is referred to as L. We refer to each hop level in the routing path by h. Thus:

$$R = \{h_1, h_2, \ldots, h_L\}$$

where h_i represents the IP interface at hop level i.

We propose a six-step methodology in order to detect regular routing path changes and to adjust probing weights as shown in Fig. 1.

Step 1: *perform periodic Tracerouting*. In the first step, Paris Traceroute is performed periodically every 15 min in order to collect raw data for our analysis. We chose Paris Traceroute since it is an open source and it does not report false links in the presence of load balancers along the scanned path as explained in Section 2. Our experiments revealed that it is not very common that more than one routing change occurs in 15 min, which explains the reason behind the choice of 15 min as a time interval between the performed scans.

Step 2: *parsing stage*. The second step is a parsing stage whereby we extract the IP address of each probed hop from the raw data.

Step 3: *storing parsed data in an SQL repository*. In the third step, the parsed data is stored in an SQL repository. This facilitates the processing that will be performed in the next steps.

Step 4: *determine the most dominating path*. In the fourth step, we determine the most dominating routing path. This is done by comparing at each hop level all the reported IP addresses. The dominating IP address (h_{di}) for a hop level i will be the one that occurred most frequently in all the traces at that level. The dominating path will consist of all the dominating IP addresses at each hop level. We refer to the dominating path as:

$$R_d = \{h_{d1}\, h_{d2}, \ldots, h_{dL}\}.$$

The dominating path will be used as a reference to detect changes. A change in the hop level i is detected if an IP address different than (h_{di}) responds to the sent probe.

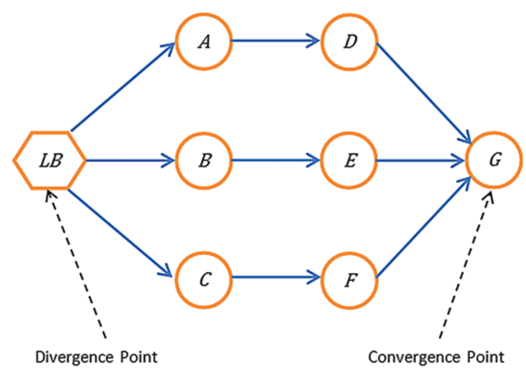

Fig. 2. Illustration of the divergence and convergence point when a per-packet load balancer (LB) exists along the path to reach the destination. The hop levels where (A, B, C) and (D, E, F) are located are discarded from our analysis since they keep changing randomly.

Step 5: *data analysis*. The fifth step is the major step of our profile-based probing framework and it consists of:

(a) Detecting per-packet load balancers: the MDA algorithm that was explained before in Section 2 is run once in order to detect per-packet load balancers. This is to overcome the limitation of Paris Traceroute since it does not detect this type of load balancers. These load balancers forward packets randomly to one of the outgoing interfaces and a different interface will be always reported at certain hop levels due to the existence of per-packet load balancers. These different reported interfaces do not reflect an alternation in the network connectivity. Thus all the hops between the divergence and convergence points that are identified by MDA should be excluded when trying to find regular changes as they keep changing randomly. Fig. 2 illustrates the divergence and convergence points when a per-packet load balancer (LB) exists along the path to reach the destination. Circles represent routers and arrows represent links between these routers. The router (LB) is the only router that is a per-packet balancer whereas the rest are normal routers. As mentioned above, when probing the hop level that is after (LB), we receive a reply from either (A) or (B) or (C) as packets will be forwarded randomly. The same thing applies for (D), (E) and (F) and thus we remove the two hop levels that are after the divergence point in our study since they change randomly and this change is not due to any alternation in the routing tables that may be of interest for network monitoring.

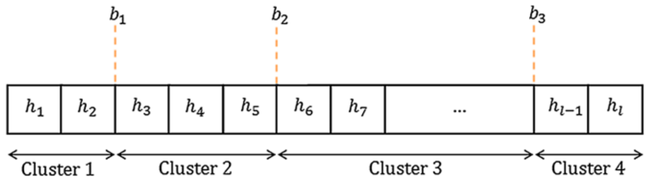

Fig. 3. Dividing the routing path into four clusters using three boundaries (b1, b2 and b3) where (h_i) refers to the hop level (i) and (l) is the length of the routing path to reach the destination.

(b) Separating the collected traces based on the time the Traceroute was launched into a number of periods. We refer to the ith period as (T_i) where $i = 1, 2, \ldots, N_p$ and N_p is the number of periods. We analyze each one of these periods separately in our following steps and we assign a different probing strategy for each one of these time periods. The objective of dividing the traces into a number of periods is to examine regularities in each one of these periods separately. This is beneficial if certain parts of the routing path tend to change more frequently during a certain time interval of the day and thus should be given higher probing weight at that interval as compared to a lower probing weight in the remaining intervals. This helps in saving the number of required probes to scan the routing path. The algorithm that we propose to determine the beginning, the end, the duration and the number of these intervals is explained in details in Section 6. The following steps of our methodology are applied to the data of each time interval separately.

(c) Calculating the probability that an IP address different from the dominating IP address replies to the sent probe at each hop level. This probability can be calculated as shown in Eq. (1):

$$pr\,(h_i) = \frac{n_i}{N_i} \tag{1}$$

where $pr\,(h_i)$ is the probability that an interface different from the dominating interface (h_{di}) replies to the probe at hop level i.
n_i: the number of times an IP interface different than the dominating IP (h_{di}) replies at hop level i.
N_i: the total number of collected probes at hop level i.

(d) Dividing the routing path into K clusters: in this step of our analysis, the routing path is divided into K clusters where a cluster consists of a number of consecutive hop levels. The number of clusters or partitions K is supplied by the scanner. For each time period T_i, we assign each cluster a probing weight that is proportional to how often the hop levels in that cluster change during time period T_i. The objective of this clustering is to exploit regularities in the changes of the routing path in order to reduce the number of probes that are sent to the hop levels that rarely change. In our clustering approach, the routing path is not divided statically into equal clusters. We rather choose the boundaries that determine the divided K parts such that the number of required probes for scanning the routing path is minimal. We apply this step on the collected traces for each of the periods separately, resulting in a division of the routing path into K clusters that might be different for each one of these periods based on how frequent routing changes occur at each hop level during each period. We show in Section 4 the effect of varying the value of the parameter K on the reduction in the number of probes.

Fig. 3 provides an example of how the routing path of length (L) is divided into four clusters using three boundaries (b_1, b_2, b_3) where (h_i) refers to the ith hop level. The same probing weight will be assigned for all the hop levels within the same cluster

We explain now how to determine the boundaries such that the number of probes for scanning the path is minimal. Let $b_1, b_2, \ldots, b_{K-1}$ be the boundaries that divide the routing path into K clusters $P_1, P_2, \ldots, P_K$. We define the probing weight for part P_j as shown in Eq. (2):

$$\text{Weight}\left(P_j\right) = \text{roundup}\left(pr\left(P_j\right) \times N_{\max}\right) \tag{2}$$

where: weight $\left(P_j\right)$ is the probing weight for cluster P_j.
$pr\,(P_j)$: is the maximal probability for a hop level in part P_j to change as shown in Eq. (3):

$$pr\left(P_j\right) = \text{Max}\,(pr\,(h_i)) \quad \text{where } h_i \in P_j. \tag{3}$$

The minimal value of $pr\left(P_j\right)$ is equal to 0.1. This is due to the fact that the probability that a cluster remains the same cannot be zero since it is always possible that a change in the routing path occurs.

$N_{\max}$: is the maximum number of probes that the scanner affords to send per hop in the time interval of the day. This parameter is supplied by the scanner.

If every cluster P_i has a number of hops n_i, the total number of probes required to scan the routing path, N_{probes}, can be calculated as shown in Eq. (4):

$$N_{\text{probes}} = \sum_{i=1}^{K} n_i \times \text{Weight}\,(P_i)\,. \tag{4}$$

We define the problem now as finding the location of the boundaries $b_1, b_2, \ldots, b_{K-1}$ such that the total number of probes (N_{probes}) is minimal. This optimization problem aims at finding $[\text{argmin}_{b_1, b_2, \ldots, b_{K-1}} (N_{\text{probes}})]$ where the objective function is N_{probes} and the decision variables are $b_1, b_2, \ldots, b_{K-1}$. Since the number of hops between any two Internet nodes is usually not larger than 40, we try all the possible positions of $b_1, b_2, \ldots, b_{K-1}$ and store the positions of these boundaries such that N_{probes} is minimal. The complexity of such a brute force search can be identified as follows: for a path that consists of (L) hop levels, there exist ($L - 1$) boundaries that need to be tried. So in order to divide the routing path into (K) cluster, we need to try ($L-1$) choose (K) different divisions. Finally we pick the division that divides the path into (K) clusters while N_{probes} is minimal.

Step 6: *Profile-based probing strategy*. In this step we perform Paris Traceroute based on the calculated probing weights in each time interval of the day.

In order to speed up the scanning process, we implement a novel enhancement that reduces the scanning time. We start first by explaining why traditional Traceroute techniques are slow and then present our proposed enhancement for a fast network mapping.

Classical Traceroute techniques send a single probe and wait to receive a reply before resuming the scan of the next hop level. The Traceroute probe is actually a packet sent by the scanner with a TTL value that is equal to the hop level that needs to be mapped. For example, in order to know the IP address at the third hop level on the routing path to reach a certain destination, the scanner sends a packet destinated to that destination with a TTL value equal to 3. Every router that receives the sent probe reduces the TTL value of the probe by one. The router checks next the resulting value of the TTL field. If it is not zero, then the packet is forwarded to the following hop level based on the routing table. Otherwise (if the resulting TTL value is equal to zero), the router drops the packet and generates a TTL-exceeded ICMP error message with the router's IP address as a source of the ICMP message. The ICMP message is destinated to the sender of the probe (to the scanner) which allows the scanner to know the IP address at the third hop level. The ICMP message carries the first 64 bits of the original message.

In order to reduce the scanning time, our proposed enhanced approach sends multiple probes to scan all the hop levels at the same time. The problem becomes matching each received TTL-exceeded ICMP error message with the probe that caused that message. In other words, we need to know the original TTL value of the probe that triggered the received ICMP error message so that we infer the hop level that is associated with each IP source address of the received ICMP messages. We propose in our approach the use of the identification field in the IP header as a marker to differentiate among the different probes that are sent simultaneously. As a result, the probe that the scanner sends with a TTL value equal to 3 will always have an IP identification value that is equal to 3. The IP identification field is not changed when the packet is propagated from router to another. Since the ICMP reply carries the first 64 bits of the original message, then it carries the IP identification field whose value reveals to the scanner the hop level associated with the IP source address of the received ICMP message. The choice of the identification field as a marker is based on the fact that this field does not affect the decision of per-flow load balancers as it is not one of the five-tuple fields and thus this enhancement does not result in false links in the obtained results.

A recent RFC [25] prohibits the use of the identification field for uses other than fragmentation tracking. When this Proposed Standard becomes Internet Standard, our approach could still be implemented to achieve fast mapping without violating the RFC by making a simple modification as follows. Instead of forcing the identification field to carry a value that is equal to the TTL value of the sent probe, we let the operating system of the scanner choose an identification value. The chosen value is compliant with the aforementioned RFC. A network sniffing tool running on the scanner side during the scanning process allows extracting the chosen identification value for each sent probe. A table is then constructed to store both the TTL value and the identification field for each sent probe. When the ICMP time-exceeded messages are received, the scanner uses this table to map the identification value of each probe to its corresponding TTL value. The scanner is thus able to map correctly the received ICMP messages with their triggering probes and the proposed approach allows scanning multiple hop levels simultaneously without violating the Proposed Standard.

4. Determining the intervals of time with a similar routing change behavior

We explain in this section how to determine the beginning, the duration, and the number of time intervals during the day in which routing changes occur with a similar frequency. In our previous work [8], we divided the traces into four equal periods where the duration of each one is six hours. A probing strategy was next assigned to each one of these periods based on the frequency of routing changes in each one of these periods. The choice of equal four periods was made to simplify the framework. Instead of dividing the traces statically into four equal periods, we show in this section the algorithm to determine the number of these periods along with the start, end and the duration of each period such that we end up with a number of periods that have a similar frequency of routing changes. We follow later the steps of the proposed methodology in Section 3 and apply the steps on the data of each one of these periods in order to achieve higher saving of probes and make the network monitoring system more dynamic.

The proposed algorithm to determine the time periods with a similar routing change behavior consists of four steps and is shown in Fig. 4. These steps are:

(a) *Initialization step: equal periods division*. Initially we set the number of periods to be 24 were the duration of each period is one hour. We extract from the SQL repository the traces for each hour separately (i.e. we take all the collected Traceroutes to reach the same destination at each hour separately). We refer to the datasets that have these traces by D_j where $j = 1, 2, \ldots, 24$. The dataset (D_1) for example consists of all the traces that are collected between the time 0:00 and 1:00. The dataset (D_2) consists of all the traces between the time 1:00 and 2:00 and so on.

Algorithm:
Determining Periods with Similar Routing Changes

Input:
- D, a data set containing all the collected traces.
- $Path = \{h_1, h_2, \ldots, h_L\}$, the path to reach a destination consists of L hop levels we refer to each hop level by h_i.
- $DominatingPath = \{hd_1, hd_2, \ldots, hd_{dL}\}$, the dominating path to reach the destination where hd_i is the dominating IP interface at hop level i and L is the length of the dominating path.
- ε, clustering threshold

Output:
- $T_1, T_2, \ldots, T_N$, Periods where the routing changes occur with a similar frequency during each period.
- $T_i = hour{:}minute \rightarrow hour{:}minute$ the start, the end and the time duration of each period.

Method:
(1) //Step1: Initialization
(2) Set $N = 24$ //Consider 24 equal periods $T_1, T_2, \ldots, T_{24}$
(3) Divide D into $D_1, D_2, \ldots D_N$
(4) //Step2: Calculate the probability that $h_j \neq hd_j$ for each D_i
(5) **For** $j = 1$ **to** N {
(6)   **For** $i = 1$ **to** L {
(7) n_{ij}: number of times ($h_i \neq hd_i$) during T_j
(8) N_{ij}: total number of probes sent to h_i during T_j
(9)   Calculate $pr(h_{ij}) = n_{ij}/N_{ij}$
(10)   }
(11) }
(12) //Step3: Represent each period by a tuple
(13) **For** $j = 1$ **to** N
(14) $T_j = \{pr(h_{1j}), pr(h_{2j}), pr(h_{3j}), \ldots, pr(h_{Lj})\}$
(15) //Step4: Periods merging stage
(16) **For** $j = 1$ **to** N {
(17) $k = j \bmod (24) + 1$
(18)   Calculate $Distance(T_j, T_k)$
(19)   **If**($Distance(T_j, T_k) < \varepsilon$) **then** {
(20)   Merge T_j and T_k
(21) $N = N - 1$
(22)   }
(23) }

Fig. 4. The proposed algorithm to determine the time periods with a similar frequency of routing changes.

(b) *Calculate the probability that an IP different than the dominating IP replies for the sent probe*. For each dataset D_j we calculate the probability that an IP different than the dominating IP replies at each hop level. The probability that an interface different than the dominating interface replies to the sent probe at hop level (i) during the period (j) can be calculated as shown in Eq. (5):

$$pr\left(h_{ij}\right) = \frac{n_{ij}}{N_{ij}} \tag{5}$$

where $pr\left(h_{ij}\right)$ is the probability that an interface different from the dominating interface (h_{di}) replies to the probe at the hop level i during the period j.

n_{ij}: is the number of times an IP interface different than the dominating IP (h_{di}) replies at the hop level i during the period j.

N_{ij} : is the total number of collected probes at the hop level i during the period j.

(c) *Represent each period by a tuple*. We represent the period j by a tuple T_j that consists of L features:

$$T_j = \{pr(h_{1j}), pr(h_{2j}), pr(h_{3j}), \ldots, pr(h_{Lj})\}$$

where $pr(h_{ij})$ is the probability that an interface different from the dominating interface (h_{di}) replies to the probe at the hop level i during the period j.

As a result, we end up with 24 tuples $(T_1, T_2, \ldots, T_{24})$. Each tuple consists of L features where each feature represents the probability that an IP different than the dominating IP replies at the hop level i during that period. This representation will help in comparing how frequent routing changes occur within each hour and will be used to identify the periods with a similar routing changes.

(d) *Periods merging*. We calculate the Manhattan distance between each two consecutive tuples. If the distance is lower than a threshold (ε), we merge these two periods together as the routing changes in both periods are almost similar. If not, then we keep them separate. The logic behind this is that if routing changes occur during two periods with the same frequency, then we can treat them as a one period.

The Manhattan distance between two consecutive tuples T_x and T_y is calculated as shown in Eq. (6):

$$\mathrm{dist}(T_x T_y) = \sum_{i=1}^{L} \left| pr(h_{ix}) - pr(h_{iy}) \right| \quad (6)$$

$$\text{where:} y = x \bmod(24) + 1 \quad (7)$$

y represents the index of the tuple that follows the tuple x in time.

The threshold (ε) is a parameter supplied by the scanner. Experiments in Section 6 show the tradeoffs of choosing a high or a low threshold.

5. Experiments and results

We implement the methodology proposed in Section 3 to study the changes in the routing paths to reach six Internet addresses. These addresses are actually the web server IP addresses of six universities where these web servers are known to be distributed geographically in different locations around the world. The list of the studied Internet addresses is shown in Table 2.

We developed the code to implement our methodology using the C# programming language. For the first step of the methodology, Paris Traceroute is performed periodically every 15 min over eight days resulting in 760 traces for the routing path for each address. We follow the steps explained previously and as shown in Fig. 1: the collected traces are parsed to extract the IP addresses (Step 2), then the data is stored in the SQL repository (Step 3) and the dominating paths are determined based on all the collected traces (Step 4).

The MDA algorithm is run next as part of Step 5 from the framework in Fig. 1 in order to detect any per-packet load balancers along the paths. Our results show that no per-packet load balancers exist along the paths to the chosen destinations. This is not surprising since according to [6], only 2.1% of Internet paths are affected by per-packet load balancers, while close to 40% is affected by per-flow load balancers. If MDA reported a per-packet load balancer along one of the paths, we would exclude the hop levels that are between the divergence and the convergence points of the per-packet load balancer as was explained in Fig. 2 in Section 3.

Next and as part of Step 5, all the collected traces are separated based on the time the Traceroute was launched into a number of groups where each group contains all the traces that were performed during the same period of the day. The objective of this division is to examine regularities in each one of these periods

Table 2
List of probed Internet addresses.

Domain name	IP address	Location
www.ucla.edu	169.232.55.224	United States
www.uchile.cl	200.89.70.188	Chile
www.aucegypt.edu	213.181.237.41	Egypt
www.polytechnique.fr	129.104.30.4	France
www.sydney.edu.au	129.78.5.11	Australia
www.u-tokyo.ac.jp	133.11.114.194	Japan

separately so that a different probing strategy is applied during each period. The question now is how to identify the beginning, the number and the duration of these periods. A simple approach would be to consider four equal periods where the duration of each period is six hours. A more advance solution is to use the algorithm proposed in Section 4 to determine the number and the duration of these periods so that we profile routing changes more accurately which leads as shown later into further reduction in the number of probes.

We start Section 5 by presenting the results for the simple approach where equal periods (each of six hours) are considered first. Next, we move to the more advanced approach where the algorithm explained in Section 4 is applied to identify the periods with a similar behavior.

The rest of Section 5 is organized as follows: in the first subsection we present a case study of the proposed six-step methodology where the number of periods is fixed to four equal periods and where we divide the path into four clusters $(K = 4)$. The boundaries that identify these four clusters are determined such that the number of required probes is minimal in each time period. In the second subsection, we keep the periods fixed but we study the effect of changing the number of clusters (K) on the overall reduction in the number of probes. In the third subsection, rather than having fixed periods, we identify the duration and the number of the periods with a similar routing changes. Finally, in the last subsection, we estimate the speedup that is gained by scanning multiple hop levels simultaneously after using the IP identification field as a marker to match the probes with the received responses.

5.1. Equal periods division—case study with four path clusters

We present in this subsection the results of our six-step methodology for four equal periods $(T_1, T_2, T_3,$ and $T_4)$ that represent the four quarters of the day. We divide in this experiment the routing path into four clusters $(K = 4)$ where a single cluster consists of a number of consecutive hop levels. Thus in order to have four clusters, we need to have three boundaries that divide the routing path into four clusters as explained in Fig. 3. For each position of the boundaries we estimate the probing weight based on Eq. (2) and then we calculate the number of probes using Eq. (4) with $N_{\max} = 6$, which means that a trace is performed once every hour as a maximum. We try all the possibilities of the boundaries and keep the ones with the minimal value for N_{probes}.

Figs. 5–8 show the obtained four clusters for one of the studied addresses (corresponding to www.ucla.edu). The x axis in the figures represents the hop level and the y axis represents the probability that an IP address different than the dominating IP address replies to the sent probe. Each cluster is marked by a different color and the maximal probability for the hops in one cluster to change is marked by a circle.

We can observe from Figs. 5–8 that routing changes do not occur with the same probability for all the hop levels. As an example, the first eleven hops in all the four periods rarely change. On the other hand, hops 12–15 change with a high probability. A monitor that is interested in detecting the routing changes that occur for a routing path should not waste probes on hops that change rarely

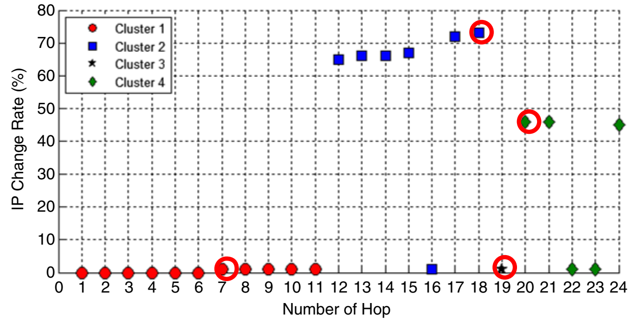

Fig. 5. The clusters for the routing path to reach (www.ucla.edu) for the time period T_1.

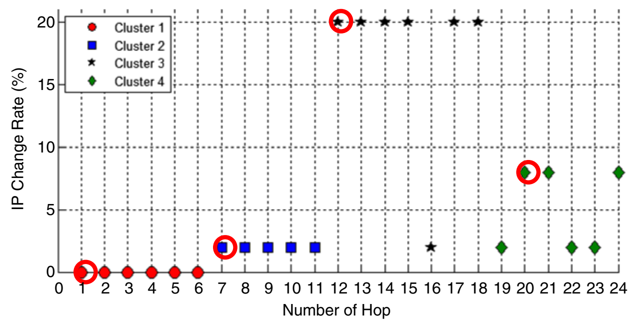

Fig. 6. The clusters for the routing path to reach (www.ucla.edu) for the time period T_2.

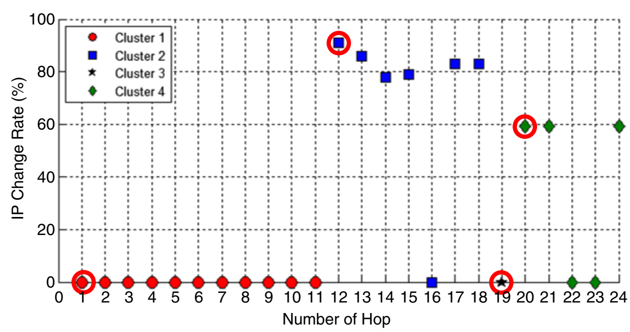

Fig. 7. The clusters for the routing path to reach (www.ucla.edu) for the time period T_3.

but should rather send more probes to the hops that have high probability to change.

We can observe by comparing Fig. 6 to Figs. 5, 7 and 8 that the hop levels from seven to eleven tend to change more frequently during period T_2 as compared to the other quarters of the day. This shows that certain routing changes tend to occur at certain times of the day and have regular behavior. These regularities caused different clusters in the second quarter (T_2) compared to the other quarters. In the four periods, our algorithm chooses these clusters such that the number of required probes is minimal. Similar observations can be drawn from analyzing the results for the routing paths of the other five destinations. We include only results to www.ucla.edu due to space limitation.

Table 3 shows the number of probes that our profile-based probing strategy will assign for each cluster in each quarter of the day for the routing change to the address of www.ucla.edu such that the total number of probes is minimal. It is worth noting that the number of assigned probes during T_2 in Table 3 is low due to the fact that routing changes rarely occur during T_2 compared to the other periods. This can be inferred by noticing that the change rate during T_2 in Fig. 6 for all the hop levels is much lower than

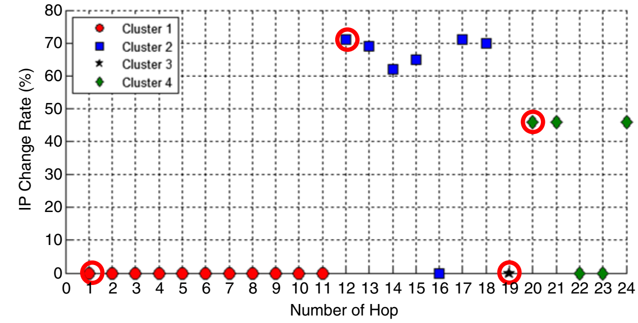

Fig. 8. The clusters for the routing path to reach (www.ucla.edu) for the time period T_4.

Table 3
Number of probes assigned for each cluster during each quarter of the day.

	T_1	T_2	T_3	T_4
Cluster 1	11	6	11	11
Cluster 2	35	5	42	35
Cluster 3	1	14	1	1
Cluster 4	15	6	20	15
All clusters	62	31	74	62

the change rate during the other periods in Figs. 5, 7 and 8. In fact, the maximum change rate during T_2 in Fig. 6 is 20% whereas the change rate is much higher for all the hop levels during the other periods (Figs. 5, 7 and 8) as it goes above 70%.

To assess the savings in the number of probes that our strategy achieves, we compare the total number of required probes in the four quarters of the day that our methodology needs to the total number of probes if the scanner is treating all the hop levels of the routing path equally and is sending the maximal number of probes ($N_{\max} = 6$) in each quarter of the day. We introduce μ, as the reduction in the number of probes coefficient, which is defined as shown in Eq. (8):

$$\mu = 1 - \frac{\sum_{i=1}^{4} N_{\text{probes},\,i}}{\sum_{i=1}^{4} N_{\text{Probes(Max)},\,i}} \tag{8}$$

where $N_{\text{probes},\,i}$ is the number of probes during period T_i.
$N_{\text{Probes(Max)},\,i}$: is the maximal number of probes during period T_i.
$N_{\text{Probes(Max)},\,i}$ is obtained for:

$$\text{Weight}\left(P_j\right) = N_{\max}, \qquad pr\left(P_j\right) = 1.$$

Our calculations for www.ucla.edu show that the reduction in the number of probes coefficient μ is 0.60 on average, which means that our approach achieves 60% savings when compared to a static Traceroute that does not take into account time and hop level change regularities. In fact, the reduction in the number of probes for $K = 4$ for the six addresses was 66% on average, which proves that a great deal of savings in the number of probes can be achieved by using a profile-based methodology.

5.2. *Equal path division—effect of the number of path clusters on the number of probes*

To study the effect of the number of clusters (K) and its impact on the total reduction in probes μ, we divide the routing path of the six probed addresses into two, three, four, and five clusters. The location of the boundaries that form the clusters in all the cases is determined such that the total number of required probes is minimal.

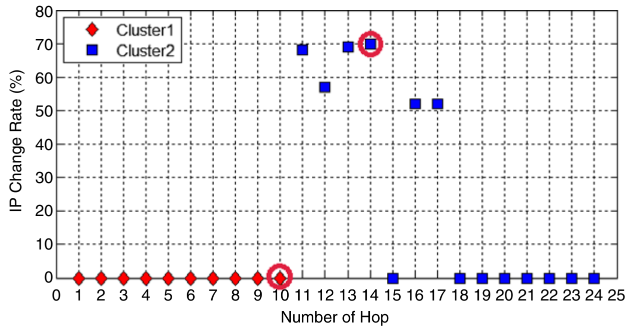

Fig. 9. The two clusters for the routing path (www.u-tokyo.ac.jp) for time period T_1.

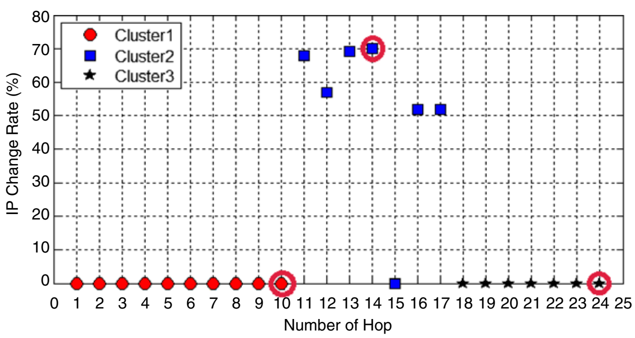

Fig. 10. The three clusters for the routing path to reach (www.u-tokyo.ac.jp) for time period T_1.

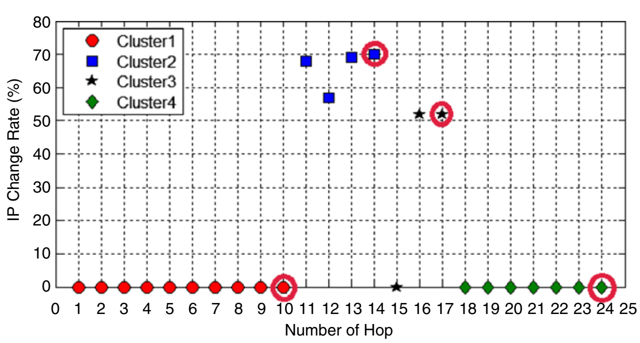

Fig. 11. The four clusters for the routing path to reach (www.u-tokyo.ac.jp) for time period T_1.

Figs. 9–12 show an example of such a division where the routing path to reach (www.u-tokyo.ac.jp) was divided into two, three, four, and five clusters at time period T_1. We can note from Fig. 9, corresponding to two clusters, that all hop levels in Cluster 2 (hops 11–24) will be assigned a high probing weight because the hop level number (14) has a high probability to change. As a result, hops 18–24 will also be assigned a high probing weight even though they do not have a high probability to change. When the routing path is divided into three clusters as shown in Fig. 10, the hop levels from 18 to 24 are given lower probing weight as their maximal probability to change is low. Consequently, the number of required probes for scanning the path will be reduced and each cluster will be assigned a probing weight that is proportional to its change rate. The same discussion holds for Figs. 11 and 12 where the routing path is divided into four and five clusters, respectively. We conclude that as the number of clusters increases, routing path changes can be exploited further to achieve greater savings in the number of probes.

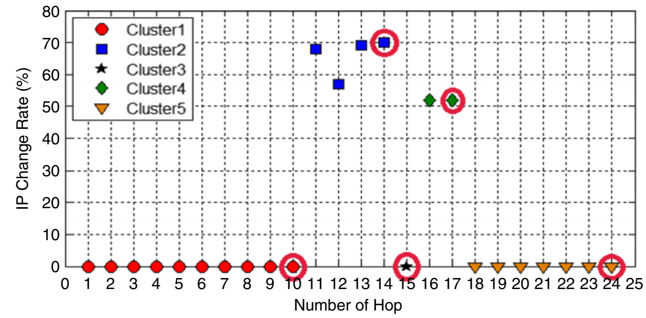

Fig. 12. The five clusters for the routing path to reach (www.u-tokyo.ac.jp) for time period T_1.

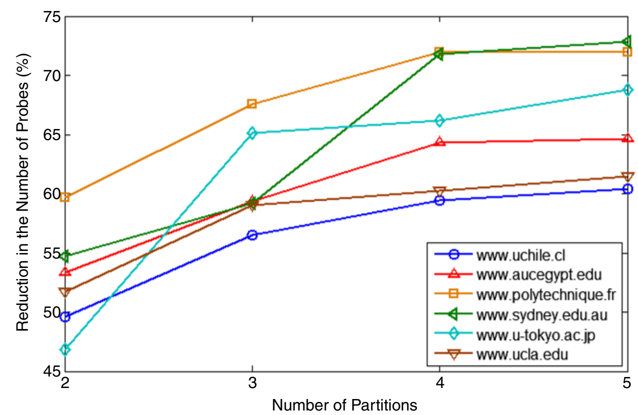

Fig. 13. Cluster size effect on reduction in the number of probes.

This conclusion can be further confirmed from Fig. 13, where we show the reduction in the number of probes versus the number of clusters K for the six probed addresses. We observe from the figure that as the number of clusters increases, the reduction in the number of probes improves. We also note that there is a significant reduction in the number of required probes when we compare $K = 2$ to $K = 4$. We also note that the reduction is less significant when we compare $K = 4$ and $K = 5$. Dividing the paths into four clusters seems to give the best savings without the additional overhead of a fifth cluster.

5.3. Dynamic time intervals

Rather than dividing the traces into a fixed number of periods with equal durations, we apply in this subsection the algorithm in Fig. 4 to determine the number and the duration of these periods, or time intervals. We show the results of applying the algorithm on the collected traces to reach (www.u-tokyo.ac.jp) as a case study. Initially we set the number of periods to be equal to 24 where the duration of each period is one hour. Fig. 14 shows the Manhattan distance after representing the collected traces during each hour by a tuple. The first bar in Fig. 14 represents the Manhattan distance between the 1st tuple and 2nd tuple, the second bar represents the Manhattan distance between the 2nd tuple and the 3rd tuple and so on. It is worth noting that the 24th bar represents the Manhattan distance between the 24th tuple and the 1st tuple as the traces collected between 0:00 and 1:00 that are represented by the 1st tuple follow in time the traces collected between 23:00 and 0:00 that are represented by the 24th tuple.

The threshold (ε) determines whether to merge two consecutive periods or not, and is a parameter supplied by the scanner. Two periods are merged into a single period if the difference in the Manhattan distance between these two periods is less than the threshold (ε). The following two examples illustrate the tradeoffs

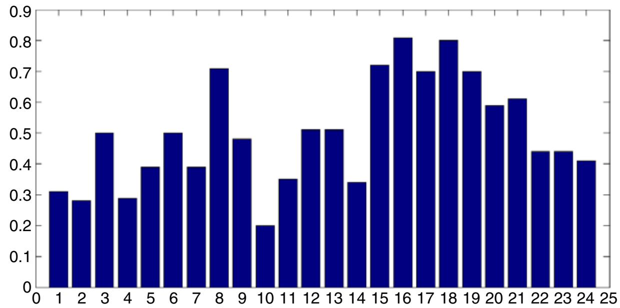

Fig. 14. The Manhattan distance between each two consecutive tuples where each tuple represents the frequency of routing changes during each hour.

of choosing a large or a small threshold. Fig. 15 shows the obtained periods when the threshold is set to $\varepsilon = 0.7$. Each color in Fig. 15 represents a different period and we therefore have four periods in this case. The x axis represents the hour of the day and helps in identifying the duration of these periods. The first period with a black color for example starts from the hour 18:00 and ends at 08:00. The second period starts from 08:00 and ends at 15:00 and so on. The routing changes within each period occur with a similar frequency; therefore we follow the steps of our methodology and divide the routing path during each period into $(K = 4)$ clusters. The boundaries that determine these clusters may be different during each period based on the routing changes during that period. Our results show that the total number of probes decreases by 5% when the periods are divided dynamically compared to the case where the periods are divided statically into four equal periods.

If we decrease the value of the threshold and set it to $\varepsilon = 0.6$, we obtain seven periods instead of four as shown in Fig. 16.

As the threshold ε decreases, the number of periods increases. The more periods we have, the finer the granularity of our analysis and the higher the savings in terms of probes. This is based on the fact that a different probing strategy will be assigned later to each one of these periods based on the frequency of routing changes during each period. However, as the number of periods increases, the processing overhead also increases as we need to analyze the data of each period separately and adjust a suitable probing strategy for each.

Fig. 17 shows the decrease in the number of periods as the threshold ε increases. The threshold varies between the smallest and largest values of the Manhattan distances in Fig. 14. When the threshold is equal to the minimum Manhattan distance of Fig. 14, we obtain 24 separate periods as no merging occurs. The number of periods decreases as the threshold increases until we obtain a single period where the threshold is equal to the maximum Manhattan distance between the tuples. In this case, the scanner assigns a single probing strategy for the whole day and ignores the temporal regularities of routing changes. Of course this is not efficient as many redundant probes will be sent.

A natural question would be how to tune the value of the merging threshold ε, or how to select the number of periods that will be considered in our framework. In order to determine the number of periods, a similar approach to the one explained in Section 5.2 can be used to show the reduction in the number of probes versus the number of periods (instead of the number of partitions). One would then search for the number of periods that strikes an optimal balance between reducing the number of probes and maintaining an affordable processing overhead.

The previously proposed approach for tuning the threshold provides the solution with optimal tradeoffs. However, the main problem with such an approach is that it requires an extensive search. A more practical approach would be to let the scanner determine how many periods it can afford to process. The value

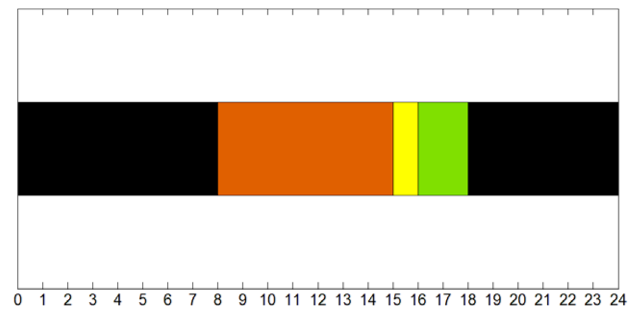

Fig. 15. The obtained periods by applying our proposed algorithm when $\varepsilon = 0.7$. Each period is marked by a different facecolor. The duration of each period can be inferred from the x-axis which represents the hour of the day.

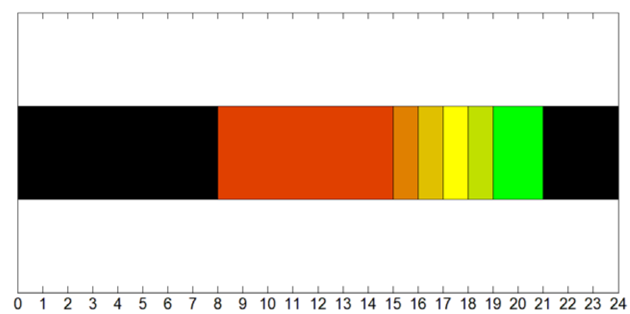

Fig. 16. The obtained periods by applying our proposed algorithm when $\varepsilon = 0.6$. Each period is marked by a different facecolor. The duration of each period can be inferred from the x-axis which represents the hour of the day.

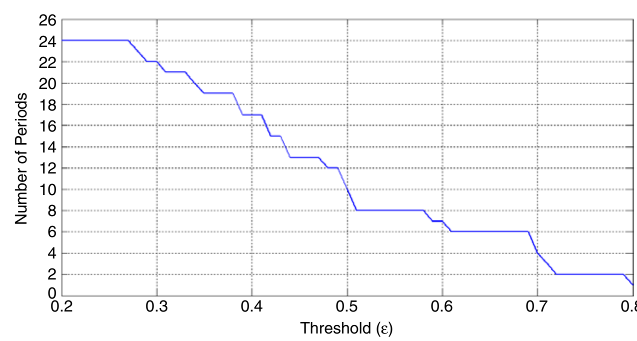

Fig. 17. The number of periods as a function of the threshold (ε).

of the threshold □ can then be determined from Fig. 17 that produces the specified number of periods. The routing changes for the periods produced by that threshold have high intra-period similarity and low inter-period similarity.

5.4. Speedup by scanning multiple hop levels

In the above experiments, we showed how to reduce the number of probes by taking routing regularities into account. We show in this subsection how to speed up the scanning process which is another requirement for scanning systems. We estimate in this subsection the savings in time that can be achieved by scanning multiple hop levels simultaneously instead of scanning the path hop by hop. The IP identification field is used as a marker to match the received replies with the corresponding probes as explained in Step 6 in Section 3.

Table 4 shows the time needed to scan the routing path to reach the destinations that were tested using both our identification-marking approach and Paris Traceroute. We repeat each measurement three times and the average time is taken for each destination. The results shown in Table 4 show that our

Table 4
A scanning time comparison between Paris Traceroute and our proposed identification-marking approach.

Destination	Path length (hops)	Scanning time (s)	
		Paris Traceroute	Identification-marking approach
www.ucla.edu	31	124.82	12.24
www.u-tokyo.ac.jp	27	95.74	10.71
www.sydney.edu.au	23	78.47	9.23
www.aucegypt.edu	20	63.12	7.71
www.polytechnique.fr	19	62.87	7.70
www.uchile.cl	19	52.99	7.69

approach achieves great reduction in the scanning time compared to Paris Traceroute. In fact, our approach reduces the scanning time by 85%–90%. We notice from Table 4 that the reduction in scanning time is higher for paths with a larger number of hops (i.e. longer paths). This is due to the fact that the longer the path, the larger the number of hops that need to be scanned and the larger the scanning time for Paris Traceroute. As the length of the path increases, our approach also needs larger time to scan the path as we need to wait for more hop levels to reply before we identify the whole path. However, our approach scales well for long paths as the probes are sent simultaneously to all the hop levels.

We compare the obtained paths using our identification-marking approach with those obtained using Paris Traceroute. Similar to Paris Traceroute, our approach does not report false links when per-flow load balancers exist along the path. This is based on the fact that the IP identification field is not one of the fields that affect the choice of outgoing interface in load balancers. This proves that our approach is as accurate as Paris Traceroute but has the advantage of reducing the scanning time significantly.

6. Conclusions and future work

In this paper, we proposed an efficient profile-based routing path scanning methodology that minimizes the number of probes and the scanning time while taking the problems that arise in the presence of load balancers into account. Our six-step methodology examines how often routing changes occur in the routing path through different periods of the day. Based on the collected traces, we set higher probing weight for the parts of the routing path that change more frequently. Our methodology also groups the hop levels into path clusters and adjusts a probing weight for each cluster. These clusters are determined such that the total number of probes for scanning the whole path is minimal. An algorithm was also proposed to determine the duration and the number of periods that have a similar frequency of routing changes such that higher reduction in the number of probes is achieved.

Our experimental results show that routing changes have regular temporal behavior and that certain hop levels have higher probability to change than others. Our proposed methodology extracts and exploits these two aspects and produces significant savings in the number of probes. We also examined the effect of increasing the number of path clusters on the total reduction in the number of probes. Our results proved that as the number of clusters increases, greater savings in the number of probes can be achieved. We also showed that further savings in the number of probes can be achieved if we determine periods dynamically based on the routing changes at the different times of the day. Furthermore, we have also proposed an approach to reduce the scanning time by probing multiple hop levels at the same time using the identification field in the IP header as a marker to match each probe to the received response. Our experimental results showed that this enhancement saves 90% of the time needed to scan the path compared to classical and Paris Traceroute. As part of future work, we plan to compare our approach with further approaches such as Fast Mapping in terms of reduction in the probing overhead and saving of scanning time.

Acknowledgments

The authors would like to acknowledge the support of the LNCSR and the AUB ECE Department.

References

[1] V. Jacobson, Traceroute, 1989, available at: ftp://ee.lbl.gov/traceroute.tar.z.
[2] B. Donnet, T. Friedman, Internet topology discovery: a survey, in: IEEE Communications Surveys and Tutorials, 2007.
[3] B. Huffaker, D. Plummer, D. Moore, K. claffy, Topology discovery by active probing, in: Proc. Symposium on Applications and the Internet, SAINT, 2002.
[4] R. Bejtlich, The Tao of Network Security Monitoring: Beyond Intrusion Detection, Addison-Wesley Professional, 2004.
[5] V. Sivaraman, A. Vishwanath, Z. Zhao, C. Russell, Profiling per-packet and per-byte energy consumption in the NetFPGA Gigabit router, in: IEEE INFOCOM, 2011.
[6] B. Augustin, T. Friedman, R. Teixeira, Measuring load-balanced paths in the internet, in: Proc. IMC, 2007.
[7] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, R. Teixeira, Avoiding traceroute anomalies with Paris traceroute, in: Proc. Internet Measurement Conference, 2006.
[8] M. Dabbagh, N. Sayegh, A. Kayssi, Efficient Internet Mapping in the Presence of Network Load Balancers, in: The 11th IEEE International Conference on Ubiquitous Computing and Communications, IUCC, Liverpool, 2012.
[9] B. Augustin, T. Friedman, R. Teixeira, Multipath tracing with paris traceroute, in: Proc. Workshop on End-to-End Monitoring, E2EMON, 2007.
[10] D. Veitch, B. Augustin, R. Teixeira, T. Friedman, Failure control in multipath route tracing, in: INFOCOM 2009, IEEE, 2009, pp. 1395–1403.
[11] V. Paxson, End-to-end internet packet dynamics, IEEE/ACM Trans. Netw. 5 (5) (1999).
[12] Z. Mao, et al. Towards an accurate AS-level TracerouteTool, in: Proc. ACM SIGCOMM, 2003.
[13] D. Magoni, J.J. Pansiot, Analysis of the autonomous system network topology, ACM SIGCOMM Comput. Commun. Rev. 31 (3) (2001) 26–37.
[14] L. Li, D. Alderson, W. Willinger, J. Doyle, A first-principles approach to understanding the internet's router-level topology, in: Proc. ACM SIGCOMM, 2004.
[15] P. Mahadevan, et al., The internet AS-level topology: three data sources and one definitive metric, ACM SIGCOMM Comput. Commun. Rev. 36 (1) (2006) 17–26.
[16] X. Zhao, et al. An analysis of BGP mulitple origin AS (MOAS) conflicts, in: Proc. ACM SIGCOMM Internet Measurement Workshop, IMW, 2001.
[17] W. Theilman, K. Rothermel, Dynamic distance maps of the internet, in: IEEE INFOCOM, 2000.
[18] F. Li, M. Thottan, End-to-end service quality measurement using source-routed probes, in: IEEE INFOCOM, 2006.
[19] Y. Bejerano, Rajeev Rastogi, Robust monitoring of link delays and faults in IP networks, in: IEEE INFOCOM, 2003.
[20] M. Stemm, R. Kaltz, S. Seshan, A network measurement architecture for adaptive applications, in: IEEE INFOCOM, 2000.
[21] M. Latapy, C. Magnien, F. Oueédraogo, A radar for the internet, in: Proc., the First International Workshop on Analysis of Dynamic Networks, ADN, in conjunction with IEEE ICDM 2008, 2008.
[22] B. Donnet, P. Raoult, T. Friedman, M. Crovella, Efficient algorithms for large-scale topology discovery, in: Proc. ACM SIGMETRICS, 2005, p. 327.
[23] I. Cunha, R. Teixeira, C. Diot, Measuring and characterizing end-to-end route dynamics in the presence of load balancing, in: Proc. PAMas, 2011.
[24] B. Donnet, B. Huffaker, T. Friedman, K. Claffy, Evaluation of a large-scale topology discovery algorithm, in: Proc. IEEE Int'l. IP Operation and Management (IPOM) workshop, 2006.
[25] J. Touch, Updated specification of the IPv4 ID field, RFC 6864, 2013.

Mehiar Dabbagh received his B.S. degree from the University of Aleppo, Syria, in 2010 and the M.S. degree from the American University of Beirut (AUB), Lebanon, in 2012, all in Electrical Engineering and Computer Science. During his Master's studies, he worked as a research assistant in Intel-KACST Middle East Energy Efficiency Research Center (MER) at the American University of Beirut (AUB), where he developed techniques for software energy profiling and software energy-awareness. Currently, he is a Ph.D. student in Electrical Engineering and Computer Science at Oregon State University (OSU). He is also a research assistant in the Networking and Communications group at (OSU). His research interests include Networks and Security, Energy-Aware Computing and Data Mining.

Naoum Sayegh received his B.S. degree from the University of Aleppo, Syria in 2010, and his M.S. degree from the American University of Beirut (AUB), Lebanon in 2013, majoring in Electrical Engineering and Computer Science. During his Master's studies, he worked as a research assistant in the industrial security team at the American University of Beirut (AUB), where he worked on developing test-beds to assess SCADA security and designing Intrusion Detection Systems (IDS) to prevent attacks on such systems. His research interests include Networks and Security, Wireless sensor networks, traceroutes, Power, and Communications.

Ayman Kayssi was born in Lebanon. He studied electrical engineering and received the B.E. degree, with distinction, in 1987 from the American University of Beirut (AUB), and the MSE and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1989 and 1993, respectively. He received the Academic Excellence Award of the AUB Alumni Association in 1987. In 1993, he joined the Department of Electrical and Computer Engineering (ECE) at AUB, where he is currently a full professor. In 1999–2000, he took a leave of absence and joined Transmog Inc. as a chief technology officer. From 2004 to 2007, he served as a chairman of the ECE Department at AUB. He teaches courses in electronics and in networking, and has received AUB's Teaching Excellence Award in 2003. His research interests are in information security and networks, and in integrated circuit design and test. He has published more than 160 articles in the areas of VLSI, networking, security, and engineering education. He is a senior member of IEEE, and a member of ACM, ISOC, and the Beirut OEA.

Imad H. Elhajj received his Bachelor of Engineering in Computer and Communications Engineering, with distinction, from the American University of Beirut in 1997 and the M.S. and Ph.D. degrees in Electrical Engineering from Michigan State University in 1999 and 2002, respectively. He is currently an Associate Professor with the Department of Electrical and Computer Engineering. Dr. Elhajj is the vice-chair of IEEE Lebanon Section, senior member of IEEE and a member of ACM. Imad received the Most Outstanding Graduate Student Award from the Department of Electrical and Computer Engineering at Michigan State University in April 2001, the Best Paper award at the IEEE Electro Information Technology Conference in June 2003, and the Best Paper Award at the International Conference on Information Society in the 21st Century in November 2000. Dr. Elhajj is recipient of the Teaching Excellence Award at the American University of Beirut, June 2011.

Ali Chehab received his Bachelor degree in EE from AUB in 1987, the Master's degree in EE from Syracuse University in 1989, and the Ph.D. degree in ECE from the University of North Carolina at Charlotte, in 2002. From 1989 to 1998, he was a lecturer in the ECE Department at AUB. He rejoined the ECE Department at AUB as an Assistant Professor in 2002 and became an Associate Professor in 2008. He received the AUB Teaching Excellence Award in 2007. His research interests include: Wireless Communications Security, Cloud Computing Security, Multimedia Security, Trust in Distributed Computing, Low Energy VLSI Design, and VLSI Testing. He received research grants from local and international companies such as Intel and Telus and he participated in European Tempus projects. He has about 120 publications. He is a senior member of IEEE and a member of ACM.