SPIRA: A Network-Friendly Topology Discovery Protocol

Mohamed Karim Sbai, Mohamad Jaber, Chadi Barakat

SPIRA: A Network-Friendly Topology Discovery Protocol

Mohamed Karim Sbaï
Telecom Bretagne
mohamed.sbai@telecom-bretagne.eu

Mohamad Jaber
INRIA - France
mohamad.jaber@inria.fr

Chadi Barakat
INRIA - France
chadi.barakat@inria.fr

***Abstract*—The Internet being very large and rapidly evolving, it is always difficult to maintain a real-time view of its topology without continuously flooding it with a large number of concurrent probe packets. Although there have been considerable research efforts to reduce the number of these probes (e.g. reducing redundancies), the congestion and network overhead they cause have often been overlooked. In this paper, we propose SPIRA, a network-friendly protocol to discover the Internet topology. Our protocol regulates the throughput of probes as a function of the observed delay and loss measurements. Starting from a monitoring computer and a set of destinations, a cartography of intermediate routers (IP addresses and coordinates) and links between them (interfaces and delays) is deduced in a short time and with a minimal overhead. We evaluate the performance of our protocol using real experiments on the PlanetLab testbed.**

I. Introduction

The Internet is a very large network composed of thousands of routers and domains, and millions of end-user computers. Routers in the Internet are connected together with communication links that have different characteristics and use a variety of technologies. As the number of routers is growing exponentially and the characteristics of links (delay, bandwidth) are continuously changing, the topology of the Internet is becoming complex, dynamic and quickly growing. The cartography of this topology is an essential network monitoring and management data. The cartography of routers and links helps for instance in understanding the evolution of the Internet and in elaborating extensive statistics, which are used in taking management decisions, developing simulation models, and detecting anomalies. Two main approaches have been proposed in the literature for Internet topology discovery. The first one is based on passive measurements by observing the traffic circulating between nodes or looking into routing tables (e.g. BGP routing tables [2]). This approach requires access rights to these measurements and a long time to collect the required information and process it. It also requires having many monitoring points distributed across the Internet. Unlike the first approach that does not inject any extra-packet in the network, the second one is based on active measurements; it consists in sending probes from few monitoring nodes, called *vantage points*, to a large set of destinations and then in collecting answers and aggregating them. This second approach is largely used because it does not require access rights to the core of the Internet. Yet, the complexity of the Internet makes it very difficult to maintain an up-to-date view of its topology without continuously flooding its routers and links with a large number of probe packets. Thus, active measurements can be aggressive and can steal non negligible bandwidth from concurrent application traffic. They can even be a source of congestion on some bottleneck links. Still, active measurements for topology discovery are a promising approach for its ease of deployment. Classical topology discovery methods are generally based on the *traceroute* tool (RFC 1393). A traceroute run is composed of a sequence of packets that are sent from a monitor, or vantage point, to an IP destination in order to discover the IP addresses of the intermediate routers on the path to this destination and the delay of links between them. By sending many traceroutes from one monitor to a large number of destinations, one can obtain the shortest path tree, composed of routers and links, whose root is the vantage point and whose leafs are the destinations. After merging the trees obtained by several vantage points, one can get a sample of the Internet topology. It is known that during the active measurement of the topology by a vantage point, each router of the shortest path tree is discovered as many times as the number of destinations it connects to. Several methods have been proposed in the literature to eliminate this redundancy. For example, the authors in [13] propose TraceTree an egocentric measurement tool that sends less packets than Traceroute to discover a tree topology from one monitor to a set of destinations. Although these methods reduce the redundancies, they neglect the impact of sending simultaneously many probe packets on the network congestion level. In fact, on one hand, if one increases the probing rate to save topology discovery time, he will increase the load on the network and may cause congestion on some of its bottleneck links. The probing traffic can even become aggressive towards the concurrent data traffic, which may distort the measurement results themselves (e.g. link delay measurements). On the other hand, if the probing rate is very low, it will take the user a long time to discover the topology of the Internet, with the risk of having the characteristics of this topology changing in the meantime. Finding the right rate at which to send probe packets has been largely disregarded by the measurement community, and is the main objective of our work.

Existing tools [4], [12] mostly propose to fix the rate of traceroutes to some value judged not aggressive for the network, and hence sacrifice probing time to avoid congestion. This rate is not adapted as a function of network conditions. Other works, e.g. [10], propose to balance the load of probes

between destinations in order not to focus on the same network area at the same time. This technique is efficient in reducing the aggressiveness of the measurement traffic close to the destinations, however, when the bottleneck is close to the vantage point, which is most probably the case, the aggregate probe traffic still needs to be controlled as a function of network conditions to avoid congestion and to fully utilize the available resources. In this paper, we propose SPIRA, a network-friendly topology discovery protocol. Unlike existing methods, our protocol adapts the rate of traceroute probes to the observed packet loss rate and measured packet delay. The objective is to avoid network congestion and to fully utilize the available resources, while being friendly with competing traffic. In its current version, SPIRA aims at discovering all routers and links along the shortest path tree from a single vantage point to a preconfigured set of destinations. Its extension to the case of many vantage points will be the subject of a future research. SPIRA runs at the vantage point and requires no collaboration from the destinations, apart from replying to probe packets. It maintains a set of nodes to probe. This set starts initially with the destinations then gets enlarged as long as new routers are known to exist. Once a node is probed and discovered, it is removed from the list. Note that a router can be easily probed even if we do not know its IP address, it is enough to know one of the destinations downstream it and the number of hops separating the vantage point from this router. The aim of the probing of this router becomes then the discovery of its IP address and the measurement of the delay between it and the vantage point. SPIRA discovers the presence of routers by sending ICMP echo request packets [8] to destinations and measuring the number of hops. When the number of hops separating the vantage point from a destination is obtained, a number of routers equal to this number of hops is added to the list of nodes to probe. The addresses of routers are first unknown but as said before, they can still be recognized by the IP address of the downstream destination and the number of hops. Later, when a probe is sent to a router, its IP address is discovered together with the delay separating it from the vantage point. This allows to infer the delay between this discovered router and the next hop router towards the destination. If the IP address of a discovered router has been already seen and attributed to another node of the topology, the two nodes are merged together in the measured topology. All routers, discovered and undiscovered, connecting the vantage point to these two merged routers are considered as being the same. In this way, we ensure that redundancies are reduced to the minimum. Note that since the traceroutes are sent from the same vantage point, routers will respond with the same IP address. In case they reply with different IP addresses, alias resolution techniques as the ones proposed in [10], [11] have to be used. The main contribution of SPIRA is the implementation of a rate control algorithm for probing the list of nodes-to-probe. This algorithm is based on a sliding window that increases and shrinks as a function of network conditions, and that decides on the number of nodes the vantage point can probe at any time before getting any reply. Once a reply is received, the window slides and a new echo request packet is sent. We implement our algorithm in a way inspired from TCP algorithms [1] for two main reasons. First, TCP algorithms have proven their efficiency in controlling the congestion in the Internet. Second, we want SPIRA to be friendly with concurrent traffic while efficiently using the available bandwidth in network bottleneck links. This way, one can use SPIRA to discover the topology of the Internet seen by a vantage point in a short time while yielding a minimum overhead on the network and being friendly with other users' application traffic. Our experience with SPIRA [9] tells us that it can perform a better congestion control if nodes to probe (destinations and routers) are ranked in a way that respects the Internet topology. A good ranking is the one that clusters together nodes located behind the same bottleneck link. Such clustering smooths the variations of SPIRA congestion window and provides a better rate adaptation. Different information can be used to cluster nodes together as the BGP information, the domain names, the geographical positions, etc. We propose in this work the use of Internet coordinate systems. Vivaldi [5] a distributed coordinates system is used for the calculation of these coordinates. Note that SPIRA can work without this clustering, and hence without coordinates. The only drawback is that its performance gets reduced. When the coordinates of destinations are available, we propose a complementary module that computes, for free, the coordinates of the routers in the topology as well. In fact, we compute the coordinates of routers by incrementally minimizing the error between measured inter-router delays and delays computed with the virtual coordinates. We implemented SPIRA using C++ under Linux and we ran extensive experiments on the PlanetLab platform [7] to evaluate the performance of the solution. The results confirm that our protocol can considerably reduce the number of sent packets compared to simple methods. Furthermore, the results illustrate the benefits of using SPIRA congestion control in reducing the discovery time and in minimizing the overhead on the network. In the next section, we describe the probing mechanism of our solution. The congestion control part of SPIRA is described in Section III. In Section IV, we explain how we use coordinates of destinations to enhance the congestion control in SPIRA and to derive the coordinates of routers. Section V summarizes our experimental set-up and obtained results. Section VI concludes the paper and gives some future research directions.

II. SPIRA PROBING MODULE

The probing module is the most important module in SPIRA and is installed in the vantage point. It is a sort of collector that probes the nodes of the topology and analyses their answers. SPIRA maintains a list of nodes to probe that keeps evolving during the session. We call it the *probing list* in the sequel. Nodes in this list, which are either routers or IP destination hosts, are always ranked according to the clustering algorithm to be presented in Section IV. Initially, SPIRA is provided with the IP addresses of a set of destinations. During the topology discovery session, it discovers the presence of routers and adds them to the *probing list*. To probe a node and discover its delay, number of hops and IP address, the monitor

(or vantage point) sends an ICMP echo request packet [8] including in its payload the time by which the packet has been sent. We call this field of the probe payload the *TimeStamp*. When receiving such a packet, the probed node responds by an ICMP echo reply packet including in its payload the header of the probe packet. This report from the probed node allows the vantage point (i) to discover the IP address of the probed node, (ii) to know the TTL of the ICMP echo request packet when it arrived to its destination, which allows to calculate the hop count, and (iii) to extract the *TimeStamp* of the corresponding probe packet, which allows to identify the probe packet and to measure the round-trip time. Probed nodes can be destinations or unknown intermediate routers. When receiving an answer from a destination, the monitor discovers the number of intermediate routers existing on the path to this destination. Since the monitor does not yet know the IP addresses of these routers, it recognizes each router by the couple (destination IP, hop count). These routers are added to the *probing list* and ranked according to the clustering module described next. When the turn of these routers comes, they are probed and their identity discovered. To probe an intermediate router in the absence of its IP address, the monitor cannot use an ICMP echo request packet destined directly to this router. Instead, for an intermediate router located at *h* hops from the vantage point and on the path to some known IP destination, the monitor sends an ordinary IP packet with a TTL equal to *h* to this destination, where the first bytes of the payload contains the *TimeStamp*. When the packet reaches the target router, its TTL becomes equal to zero. Here, the router sends an ICMP Time Exceeded message to the monitor. This allows the monitor to discover the IP address of the router and to compute the delay to it. By deduction from previous measurements, it can also compute the delay between it and the downstream router towards the destination. If the newly discovered router has already been visited by the probing module (IP address already seen), the monitor eliminates future redundancies by merging the remaining routers that exist on the path between the vantage point and this router with those of the already visited router. This shortens considerably the probing list and minimizes the number of sent packets.

III. Probing rate control and congestion avoidance

SPIRA probing rate control is designed with the main purpose of being network friendly. The rate at which probes are sent is adapted to the observed packet loss rate and delay. Inspired from our previous work on information collection [9], we implement a probe-clocked window-based congestion control similar to the one of TCP [1]. The monitor (or vantage point) maintains one variable *cwnd* indicating the congestion window size in number of probes. *cwnd* models the maximum number of probes the monitor can inject into the network without receiving any answer. SPIRA adapts *cwnd* to the observed delay and loss rate of probes and proposes two algorithms for that:

- **Slow Start:** The monitor begins the session by setting *cwnd* to an initial integer value *RS*. It thus sends *RS* probes to *RS* different IP destinations. After some time, ICMP answers start to arrive, some of them arrives on time and others are delayed or lost. A timely answer indicates that the network is not congested and that the monitor can further increase *cwnd*. In Slow Start, the congestion window is increased as follows upon the reception of an ICMP reply: $cwnd = cwnd + 1$. This yields a doubling of the probing rate for every *cwnd* probes or every Round-Trip Time (RTT). The window continues growing until network congestion is detected. Here, *cwnd* is divided by 2 and the monitor enters a Congestion Avoidance phase.
- **Congestion Avoidance:** It represents the steady state of our protocol. During it, the monitor increases slowly *cwnd* to probe the network for more capacity. As a linear increase of *cwnd* is aimed, upon each timely answer, *cwnd* is increased as follows: $cwnd = cwnd + \frac{RS}{cwnd}$. When congestion is detected, *cwnd* is divided by 2 and a new Congestion Avoidance phase is started.

Up to now, we have not explained how the monitor detects network congestion. To obtain this information, SPIRA maintains a timer that is set to an estimation of the RTT. This timer is scheduled at the beginning of the session and rescheduled every time it expires. Its value is updated as a weighted-moving average of the measured RTT between the vantage point and the probed nodes, augmented by four times the mean deviation of the RTT as in TCP. An answer is considered as being lost if it does not arrive before the expiration of the timer that follows the transmission of the probe. SPIRA calculates then the loss rate of probes over time windows equal to the timer value. If the loss rate becomes greater than some preconfigured threshold *CT*, the network is supposed to be congested and the window reduction procedure is invoked. When the loss rate of probes reaches a high value close to one *SCT*, the network is supposed to be severely congested, the window is closed to *RS* probes and slow start is invoked as in TCP. Using this mechanism, the rate of probes is adapted to network congestion and the probing traffic is friendly with other concurrent Internet traffic. We are aware that losing probes can mean that there router has disabled ICMP packets. That is why, we select *CT* and *SCT* slightly higher to ones we selected for data collection in our protocol TICP [9].

IV. Node clustering and coordinate calculation

A better congestion control can be obtained if nodes to probe are ranked in a way to respect the network topology. To reach this finality, we propose a complementary module that supposes that the monitor is provided with the coordinates of destinations. The coordinates of routers are not needed, they are calculated on the fly and provided for free by our protocol. The coordinates of destinations can be obtained thanks to an Internet coordinate system like Vivaldi [5]. This is a decentralized, low overhead, adaptive synthetic coordinate system, that computes coordinates able to predict network latencies with low error. We choose Vivaldi as it minimizes the number of measurements needed to compute coordinates and it is a distributed and incremental approach that converges in a reasonable time. One can still use other information to rank nodes to probe (e.g. domain names, AS number, geographical location), or not use any topology-aware ranking. The cost to

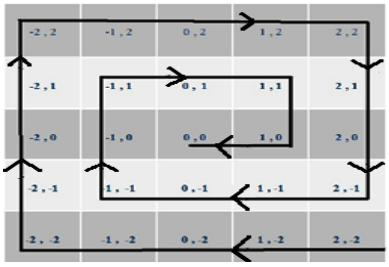

Fig. 1: Node clustering and probing order

pay is a less efficient congestion control. Following the distributed version of Vivaldi, each node maintains an estimation of its own coordinates and is connected to a set of other nodes. Periodically, it picks up a random node among its neighbors and asks it for its coordinate estimation. Then, it computes the Euclidean distance between itself and this selected node and compares it to the measured network delay. Based on this comparison, the node can decide on the shift to perform in its coordinates to reduce the estimation error. All nodes keep applying this algorithm in a decentralized manner until the whole system stabilizes. SPIRA simply uses the coordinates of destinations calculated by Vivaldi and is not intended to update them. However, given these coordinates, SPIRA calculates the coordinates of intermediate routers by simulating the Vivaldi algorithm for each router. Vivaldi simulation for routers is performed at the vantage point. The Vivaldi simulator runs (fast) in parallel and in loop over the delay matrix of routers to compute their coordinates and keep them updated. The latter matrix includes all known and estimated delays among routers, and is updated every time a new measurement is done. The initial coordinates of routers are taken as if they are equidistant on the path from the monitor to the downstream destination based on the measured number of hops. In case of node merging (because they are concluded to model the same router), we take the average of coordinates of the merged nodes. This way we guarantee that coordinates of routers are altogether updated upon each new information and this is for the best estimation of their coordinates. The main advantage of using coordinates for destinations and routers is that our probing module can order them in a way to respect the network topology and can hence probe them in this order. For an efficient ranking, we cluster the nodes in the *probing list* in a geometric grid. A cluster is a square-zone in the Vivaldi geometric space. The central square contains the vantage point. The coordinates of nodes decide on the cluster to which they belong. The monitor crosses then the nodes in a spiral way from the farthest to the nearest. Once a spiral ends, another spiral starts until all the shortest path tree is discovered. Figure 1 shows an example of a clustered space. As illustrated in [9], this clustering ameliorates the estimation of RTT used for scheduling the timer of the congestion control mechanism. It helps SPIRA to experience a smooth variation of network conditions resulting in a better congestion control. Furthermore, and as a result of using the coordinates of destinations, we get for free the coordinates of routers which are very useful to draw the Internet topology on a 2D plane and to identify the location of routers on it.

V. Experiments and results

In this section, we evaluate the performance of SPIRA by running real experiments on the PlanetLab testbed [7]. We mainly show the gain in terms of probing delay and number of probes.

A. Experimental set-up

We implement SPIRA using the C++ programming language. For now we target vantage points that use the Unix/Linux operating system. Our package is composed of two main modules running simultaneously and communicating together: the *probing module* and the *Vivaldi simulator module*. A computer located in our laboratory at INRIA is used as vantage point or monitor, and 500 hosts of the PlanetLab platform [7] are used as destinations. For scenarios where coordinates of hosts are needed, we run the classical Vivaldi [5] algorithm between the PlanetLab nodes. When the stable regime of Vivaldi is reached, the resulting coordinates are provided in an XML file to the monitor before the beginning of the probing session. For the clustering algorithm, the side of a cluster square is taken equal to 100ms, which was proven in [9] to be the most efficient value for PlanetLab. In the following paragraphs and if not explicitly mentioned, coordinates are supposed to be used for the congestion control. It is only in paragraph V-D that we conduct some experiments not using the coordinates mainly for comparison purposes. Results presented in this paper are averages on several runs conducted at different periods of the day and different days from 2007 to 2010.

B. First results

We run a first set of experiments to study the resulting topology between our machine at INRIA and the other PlanetLab nodes. Figure 2 shows the number of discovered routers as a function of the number of destinations. Clearly, the number of routers increases when the number of destinations grows. After some number of destinations, the increase in the number of routers slows down. Indeed, the paths to these additional destinations mostly share the same routers with the already probed paths. To discover the full topological tree from a monitor, one does not need to probe an infinity of destinations but only needs to probe a sufficient number of them that is enough to sample intermediate routers. The following step is to verify whether we eliminate intra-monitor redundancies (i.e. routers that are probes several times by the same monitor). We draw in Figure 3 a comparison of the number of probe packets between a topology discovery mechanism using simple traceroutes and our probing solution. One can easily notice that the number of packets sent increases exponentially with the number of destinations for the basic traceroute method. However, when our protocol is used, the number of sent packets is always close to the number of routers. Hence, we conclude that SPIRA presents a good solution for the intra-monitor redundancy problem. In the following paragraphs,

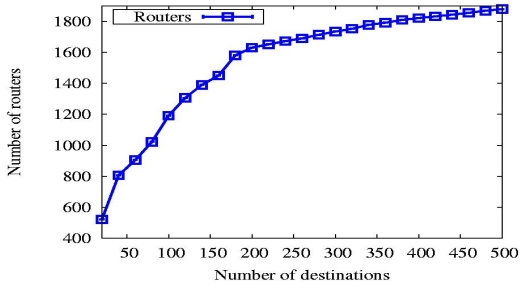

Fig. 2: Routers Vs. Destinations

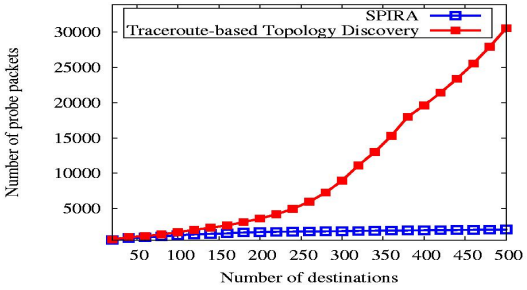

Fig. 3: Probes Vs. Destinations

we go one step further in evaluating the performance of our congestion control mechanism and in studying the gain brought by coordinates.

C. Evaluation of SPIRA congestion control

To evaluate the performance of our congestion control mechanism, we compare it to a set of experiments running a simple version of SPIRA that implements a fixed size congestion window. At the opposite of SPIRA, these experiments are supposed to run a fixed probing rate independent of network conditions (probe loss rate and delay). We study for different sizes of the window both the overhead and the probing speed, and we compare it to our adaptive method. In Figure 4 , we show the number of probes sent as a function of the number of destinations for three values of *cwnd* compared to our adaptive protocol. Fixing the congestion window can be seen as a mimic of a behavior equivalent to standard rate-limiting as done, for instance, by author in [12]. We consider a small, a medium and a large congestion window. This corresponds respectively to *cwnd* equal to 2, 10 and 20 probes. Having a small window means that the probing rate is very low. In this case, the network congestion is negligible and probes are rarely lost leading to the smallest number of sent packets. However, for a medium and large fixed window, the monitor is continuously probing nodes with a high rate. In this case, the congestion and the overhead are so important that the monitor retransmit many packets to probe nodes whose initial packets were lost. SPIRA, having an adaptive congestion window, can slow down or increase the probing rate to fully utilize the available bandwidth without overloading the network. In this way, we can reduce considerably the number of retransmissions. Note how the number of probe packets sent by SPIRA follows the number of discovered routers (Figure 2) and is very close to that of a small size window. Figure 5 illustrates the above observations by plotting the number of probes as a function of *cwnd* for 200 PlanetLab destinations. The change in the slope at a congestion window equal to 10 for the fixed window case comes from the fact that probed destinations in each window becomes more and more spread leading to more available bandwidth. As we will see later, this will be equally reflected by the probing time. Although a small congestion window yields the minimum overhead, one can expect that it has the longest probing session duration. Using a large window shortens this duration, but this is far from the optimal because of the number of lost probes.

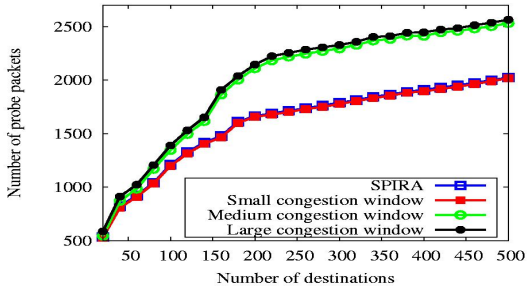

Fig. 4: Probes Vs. Destinations

We study in Figure 6 the probing session duration for the same values of *cwnd* as before and we compare the results to our adaptive method. One can observe that the session duration for a small window is indeed the longest. However, for large and medium windows, the session durations are shorter as the probing rates are higher. Nevertheless, our probing method has the shortest probing time because it is able to find the optimal speed of sending probes without loosing many of them. Figure 7 proves this observation by plotting the probing session duration as a function of the congestion window *cwnd*. From the above results, we can conclude that our protocol SPIRA is a network-friendly and efficient solution for topology mapping at the router level. It engenders a negligible overhead on the network while having the shortest probing session.

D. Benefits of using coordinates

In this paragraph, we study whether knowing the coordinates of destinations and computing those of intermediate routers enhances the performance of the congestion control mechanism of SPIRA. In case these coordinates are used, the destinations in the *probing list* are ordered and clustered following their coordinates. In Figure 8, we plot the number of probes as a function of the number of destinations for our protocol with and without the clustering of nodes. The figure clearly shows that the overhead is lower for the version using coordinates since it yields less congestion and consequently less probe packets. The usage of Vivaldi does not increase the number of probes sent in the network since the topology discovery measurements are used by the Vivaldi simulator module to compute router coordinates.The coordinates of destinations are supposed to be stable and constant during a probing session.

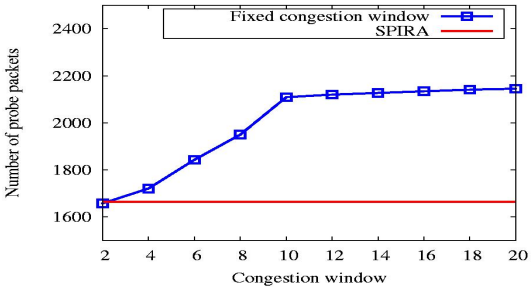

Fig. 5: Probes Vs. Congestion window

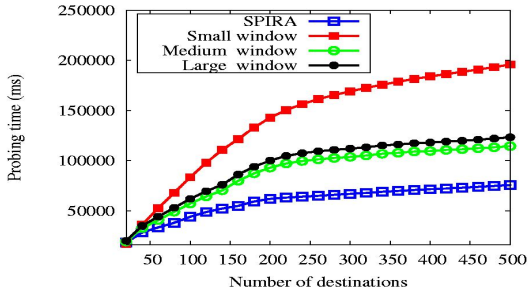

Fig. 6: Probing time Vs. Destinations

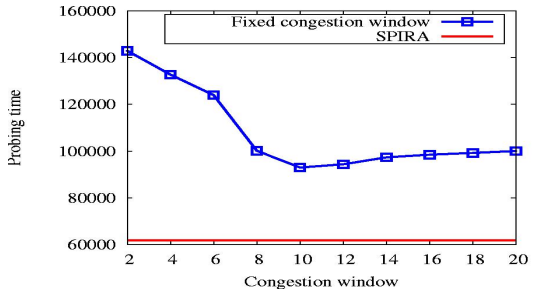

Fig. 7: Probing time Vs. Congestion window

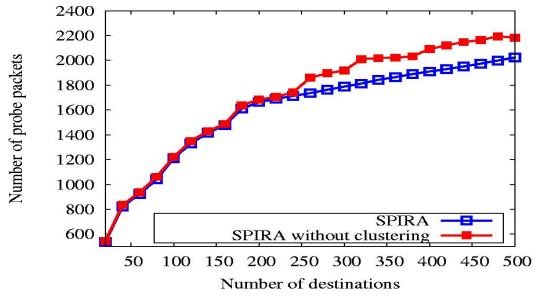

Fig. 8: Impact of clustering on number of probes

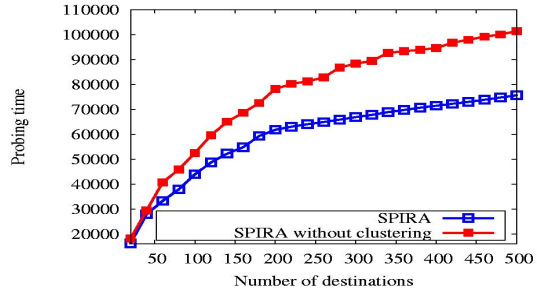

Fig. 9: Impact of clustering on probing time

Considering the probing time, Figure 9 shows that using coordinates yields the shortest session. This can be explained by the fact that the probing rate is adapted in an efficient way to the available bandwidth without causing a lot of delayed or lost packets.

VI. Conclusions and Perspectives

To infer the Internet topology, one may need to send a large amount of probes and hence cause an important overhead on the network. In this paper, we propose SPIRA, a network-friendly topology discovery protocol that adapts the probing rate to the observed probe losses and delays. When our protocol is provided with coordinates of a set of hosts, it can also compute the coordinates of the routers in the same geometric space. Our future work will focus on eliminating the inter-monitor redundancies by supposing some cooperation between monitors. The extension of SPIRA to topologies at the Autonomous System level is also an interesting research direction.

References

[1] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581 (Proposed Standard), Apr. 1999. Obsoleted by RFC 5681, updated by RFC 3390.

[2] D. G. Andersen, N. Feamster, S. Bauer, and H. Balakrishnan. Topology inference from BGP routing dynamics. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, Marseille, France, Nov. 2002.

[3] P. Barford, A. Bestavros, J. Byers, and M. Crovella. On the marginal utility of network topology measurements. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, pages 5–17, Nov. 2001.

[4] Cooperative Association for Internet Data Analysis (CAIDA). The Skitter project. http://www.caida.org/tools/measurement/skitter/.

[5] F. Dabek, R. Cox, M. F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. In *Proceedings of ACM SIGCOMM*, pages 15–26, 2004.

[6] B. Donnet, P. Raoult, T. Friedman, and M. Crovella. Deployment of an algorithm for large-scale topology discovery. In *IEEE Journal on Selected Areas in Communications*, 24(12):2210–2220, 2006.

[7] L. Peterson, S. Muir, T. Roscoe, and A. Klingaman. PlanetLab Architecture: An Overview. Technical Report PDN–06–031, PlanetLab Consortium, May 2006.

[8] J. Postel. Internet Control Message Protocol. RFC 792 (Standard), Sept. 1981. Updated by RFCs 950, 4884.

[9] M. K. Sbai and C. Barakat. Experiences on enhancing data collection in large networks. In *Computer Networks*, 53(7):1073–1086, May 2009.

[10] B. Augustin, X. Cuvellier, B. Orgogozo and F. Viger Avoiding traceroute anonmalies with Paris traceroute. In *IMC*, October 2006

[11] A. Bender, R. Sherwood and N. Spring Fixing Allys growing pains with velocity modelling In *IMC*, 2008

[12] M. Luckie Scamper: a Scalable and Extensible Packet Prober for Active Measurement of the Internet In *Internet Measurement Conference*, 2010

[13] M. Latapy , C. Magnien and F. Oudraogo A Rader for the Internet In *International Workshop on Analysis of Dynamic Networks*, 2008

[14] C. Launois , S. Uhlig and O. Bonaventure Scalable Route Selection for IPv6 Multihomed Sites In *Networking*, 2005