A radar for the internet

Frédéric Ouédraogo, Clémence Magnien, Matthieu Latapy

A Radar for the Internet

Frédéric Ouédraogo [1,2] , Clémence Magnien [1] and Matthieu Latapy[1]

[1] *LIP6 - CNRS & UPMC (Université Pierrre et Marie Curie), Paris, France*
[2] *Université de Ouagadougou, Ouagadougou, Burkina Faso*

Alors que la plupart des recherches sur la mesure de l'Internet visent à obtenir une vue aussi complète et précise que possible de sa topologie, nous proposons dans cet article une nouvelle approche qui se focalise sur une vision locale, *ego-centrée*, afin d'en observer la dynamique. Nous concevons et implémentons un outil de mesure ego-centrée appelé `tracetree`, et définissons le radar, qui consiste en une série de mesures ego-centrées. Nous menons des mesures de ce type pendant plusieurs semaines en continu avec une fréquence élevée (toutes les quelques minutes) depuis une centaine de sources. Nous discutons en détail l'influence de certains paramètres clés, puis nous montrons par une comparaison avec `traceroute` les améliorations apportées.

Keywords: Internet topology, dynamics, measurement, traceroute

1 Introduction

In order to obtain some insight on the internet topology *dynamics*, we propose here a new measurement approach: we focus on what a machine sees of the topology around itself, which we call an *ego-centered view*. It basically is a routing tree measured in a `traceroute`-like manner (similar to multicast trees, as studied in [Pan07]). We design and implement an ego-centered measurement tool named `tracetree`, optimized against redundancy (see Section 2). With an appropriate choices of parameters (see Section 3) ego-centered measurements may be performed very efficiently; it is therefore possible to repeat them in periodic rounds, which we call radar measurement. One obtains in this way data [Rad] which contain much information on the *dynamics* of the topology, at a frequency significantly higher than what was previously done.

2 The `tracetree` Tool

One may collect ego-centered views directly by using `traceroute` to a set of destinations. This approach however has serious drawbacks [DRFC05]: the measurement load is highly unbalanced between nodes and the obtained data is redundant. Even worse, this implies that the obtained information is not homogeneous, and thus much more difficult to analyse rigorously. Finally, though the measurement would intuitively produce a routing tree, the obtained view actually differs significantly from a tree (this is illustrated in Figure 1). Again, this makes the analysis (visualisation of the data, for instance) more intricate.

One may avoid these issues by performing tree-like measurements in a backward way [DRFC05, Moo04]: given a set of destinations, we first probe the last link on the path to each of them, then the link before it, and so on; when two or more paths reach the same node then the probing towards all corresponding destinations, except one, stops (such measurements require the distance of the destinations, which is not trivial [Moo04] and which we do not detail here). However, naive such measurements encounter serious problems because of routing changes and other events (see Figure 1). Our `tracetree` algorithm solves this: the tree nodes are no longer IP addresses, but pairs of IP addresses (or stars in case of timeouts) and TTLs at which they were seen (see Figure 1). This algorithm is very simple, and ensures that the obtained view is a tree. It sends only one packet for each link, which is optimal. It also ensures that each link is discovered exactly once, which gives an homogeneous view of the topology and balances the measurement load.

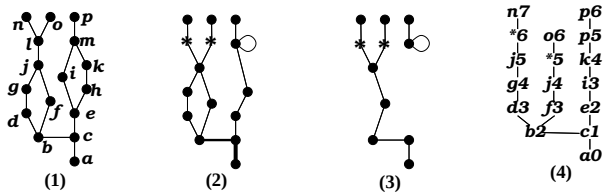

Figure 1: Typical outputs of various measurements schemes. (1) – The real topology. We will perform measurements from monitor *a* towards destinations *n*, *o*, and *p*. We suppose that *l* does not answer to probes, that *b* is a per-destination load balancer, forwarding traffic towards *n* to *d*, and traffic towards *o* to *f*, and that *e* is a per-packet load balancer forwarding packets alternately to *i* and *h*. Such situations are frequent in practice. **(2) – Measurement with `traceroute`.** Three routes are collected, leading to a higher load on links close to the monitor (represented by links thickness). **(3) – Naive tree measurement.** Due to per-packet load-balancer *e*, the node *m* is seen twice; the probing stops when it is seen the second time, and we obtain a disconnected part. **(4) – Measurement with `tracetree`.** The nodes are pairs of IP addresses and TTL, with redundancy in the addresses, but one necessarily obtains a connected tree.

Algorithm 1: `tracetree`

Input: set D of destinations, with $d \in D$ at distance i_d.
to_probe $\leftarrow$ empty queue; *to_receive* $\leftarrow \emptyset$; *seen* $\leftarrow \emptyset$; **foreach** $d \in D$ **do** add (d, i_d) to *to_probe*
while *to_probe not empty* **or** *to_receive* $\neq \emptyset$ **do**
  if *to_probe not empty* **then**
    pop (d, ttl) from *to_probe* and send a probe to it; add $(d, ttl, current\_time())$ to *to_receive*
  if *an answer p to a probe to* (d, ttl) *arrived, with* $(d, ttl, \_) \in$ *to_receive* **then**
    print *p.source ttl d*
    if $(p.source, ttl) \notin$ *seen* **then**
      add $(p.source, ttl)$ to *seen*; push $(d, ttl-1)$ in *to_probe* **if** $ttl > 1$
  for $(d, ttl, t) \in$ *to_receive* **if** *timeout exceeded* **do**
    remove (d, ttl, t) from *to_receive*; **print** * *ttl d*; push $(d, ttl-1)$ in *to_probe* **if** $ttl > 1$

With the `tracetree` tool, we have the ground material to conduct radar measurements: given a monitor and a set of destinations, it suffices to run periodic ego-centered measurements.

3 Influence of Parameters

First notice that many parameters (including the monitor and destination set) may have a deep impact on the obtained data. Testing all combinations of these parameters is totally out of reach.

We propose the following approach to estimate their impact and keep the study rigorous. We first choose a set of seemingly reasonable parameters, which we call *base parameters*. They consist of a set of 3000 destinations for each monitor, a 2s timeout and a 10 minutes delay between measurement rounds. Then we conduct measurements with these parameters from several monitors in parallel. On some monitors, called *control monitors*, we keep these parameters constant; on others, called *test monitors*, we alternate periods with base parameters and periods where we change a single parameter. Control monitors show if the dynamics observed from test monitors are due to changes of parameters or to events on the network.

Figure 2 (left) shows the impact of the delay between rounds. It is clear from the figure that reducing this delay has no significant impact on the observed behavior. In particular, the variations in the number of IP addresses seen, though they have a higher time resolution after the delay reduction, are very similar before and after it. Moreover, the control monitor shows that the base time scale is relevant, since improving it does not reveal significantly different dynamics.

Figure 2 (center) shows the impact of the number of destinations. As expected, increasing this number leads to an increase in the number of observed IP addresses. Notice however that the two increases are not proportional. indeed, increasing the number of destinations may induce too high a network load and lead to

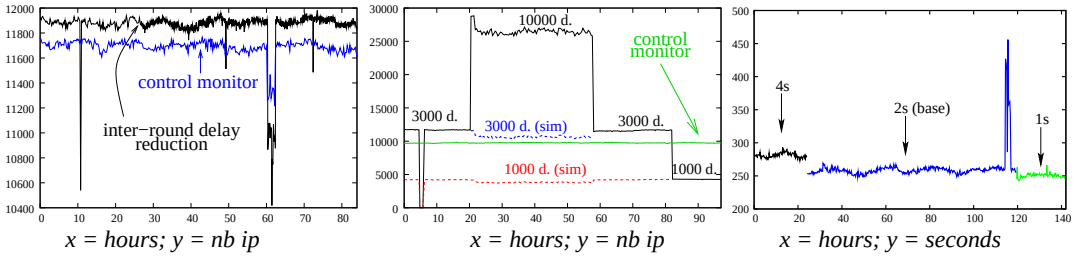

Figure 2: Impact of measurement parameters. Number of distinct IP addresses viewed (left and center) and round duration (right) at each round as a function of time (in hours) **Left: impact of inter-round delay.** The bottom plot corresponds to a control monitor with the base parameters; the other monitor starts with the base parameters, and about 27 hours later we reduce the inter-round delay from 10 minutes to 1. **Center: impact of the number of destinations.** The plot close to $y = 10\,000$ corresponds to a control monitor with the base parameters. The other plain-line plot is produced by a monitor which starts with a destination set D of size 3 000 (base value), changes to a set D' of 10 000 destinations containing D about 20 hours later, goes back to D about 40 hours later, and finally turns to a subset D'' of size 1 000 of D a bit more than 20 hours later. In addition, the dotted plots correspond to simulations of what we would have seen from this monitor with D during the measurement using D' (obtained by dropping all nodes and links which are on paths towards destinations that are not in D), and what we would have seen with D'' during the measurements using D or D' (obtained similarly). **Right: impact of timeout value.** The monitor starts with a timeout value of 4s, then we change this value to 2s about 21 hours later, and finally we change this value to 1s about 120 hours later.

a relative loss of efficiency: since some routers answer to ICMP packets with a limited rate only, overloading them makes them invisible to our measurements. Simulating measurements with 3 000 or 1 000 destinations from measurements with 10 000 destinations indeed display a smaller number of IP addresses than direct measurements with these numbers of destinations; this confirmis that using this high a number of destinations leads to a relative loss of efficiency (the control monitor proves that this is not due to a simultaneous topology change). Importantly, this does not occur in simulations of 1 000 destination measurements from ones with 3 000, thus showing that the load induced with 3 000 destinations is reasonable, to this regard.

Figure 2 (right) shows the impact of the timeout value. As expected, decreasing the timeout leads to a decrease in the round duration. However, it also causes more replies to probe packets to be ignored because we receive them after the timeout. A good value for the timeout is a compromise between the two. We observe that the round duration is only slightly larger with a timeout of 2s than with a timeout of 1s (contrary to the change between a timeout of 4 to 2s). The base value of the timeout (2s) is therefore appropriate, because it is sufficienly large and does not lead to a large round duration.

We also considered the other parameters, and other observables (like the number of stars seen at each round, and the number of reply packets received after the timeout) and the conclusion was the same: the base parameters meet our requirements.

4 Comparison with `traceroute`

As explained in Section 2, one of our key goals is to perform significantly better than direct use of `traceroute`. We therefore compare measurements performed with `traceroute` and `tracetree` in terms of numbers of IP seen and number of packets sent.

Figure 3 (left) first shows that simulating a `tracetree` measurement from a `traceroute` one is relevant: the simulated plot perfectly fills the gap in the direct `tracetree` measurement plot. Moreover, `traceroute` captures no significant additional information: it only sees about 3% of additional IP addresses, and the observed dynamics is almost unchanged. However, the difference in the number of packets sent at each round is important: `traceroute` sends twice as much packets as `tracetree`.

This has a deep impact, visible on Figure 3 (center). As expected, the number of IP seen as a function of the number of rounds is higher with `traceroute` than with `tracetree`: any `traceroute` round gathers slightly more data than the corresponding `tracetree` round (below 1 %, here). It is however much more interesting to compare them in terms of numbers of packets sent (reflecting the load induced on the network). To this regard, `tracetree` is much more efficient than `traceroute`: it reaches 14 100 distinct

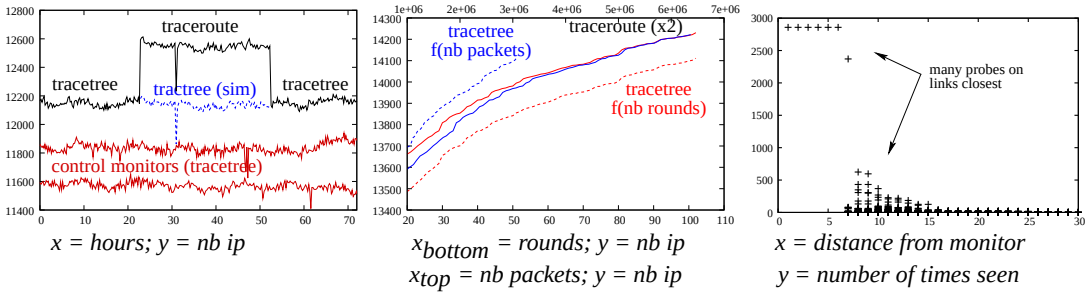

Figure 3: Comparison between `traceroute` and `tracetree`. Left: number of distinct IP addresses viewed at each round from three monitors: two control ones (the first, close to $y = 11\,600$, has a set D of 3000 destinations and the second, close to $y = 11\,800$, another set D' with same size) and one (with same destination set D) on which we temporarily turn to a `traceroute`-like measurement (top plot); on the period of `traceroute`-like measurement, we simulated a `tracetree` measurement (by keeping only one path from the source to any node), and we plot the number of distinct IP addresses seen in this simulation (dotted line). **Midle:** number of distinct IP addresses viewed since the beginning with a `traceroute` measurement (plain lines) and a `tracetree` measurement simulated from it (dotted lines); a pair of plots shows this number as a function of the number of rounds (given on the bottom horizontal axis); the second pair shows it as a function of the number of packets sent (top horizontal axis); to improve readability, we cut the part of the plots corresponding to the 20 first rounds and to the 10^6 first packets, respectively. **Right:** Number of times each link is discovered by `traceroute` as a function of the distance from monitor. Each point represents a link, its x-coordonate is its distance from monitor and y-coordonate is the number of times it is seen by `traceroute`.

IP addresses with around 3 millions packets, while `traceroute` needs around 4.5 millions packets.

Moreover, the load induced by `tracetree` is balanced among links: exactly one packet is sent to discover each link. As illustrated in Figure 3 (right), this is not true for `traceroute`: some links are probed thousands of times (those close of monitor), which induces a high load on them.

Finally, our `tracetree` tool is more efficient than `traceroute`, and is appropriate for repeatedly running it in a radar-like way with a reasonable network cost.

5 Conclusion and Perspectives.

We focused here on limited portions of internet topology to make it possible to study its dynamics. We designed and implemented a new tool to capture efficiently such portions at a time scale of a few minutes, and we carefully analysed its behavior.

We used this tool to conduct a wide set of measurements. We used destinations chosen by sampling random valid IP addresses answering to an ICMP Echo Request. We ran measurements continously during several weeks from more than one hundred monitors scattered around the world (Planetlab), with a time scale of a few minutes. We provide the obtained data to the community [Rad], which is a significant contribution in itself: this data is a rich source for the analysis of the dynamics of internet topology, which is our key perpective.

References

[DRFC05] B. Donnet, P. Raoult, T. Friedman, and M. Crovella. Efficient algorithms for large-scale topology discovery. In *Proc. ACM SIGMETRICS*, Jun. 2005.

[Moo04] T. Moors. Streamlining traceroute by estimating path lengths. In *Proc. IEEE International Workshop on IP Operations and Management (IPOM)*, Oct. 2004.

[Pan07] J.-J. Pansiot. Local and dynamic analysis of internet multicast router topology. *Annales des télécommunications*, 62:408–425, 2007.

[Rad] Program and data. `http://www-rp.lip6.fr/~latapy/Radar-sub/`.