Seventh FRAMEWORK PROGRAMME
FP7-ICT-2009-5 - ICT-2007-1.6
New Paradigms and Experimental Facilities

SPECIFIC TARGETED RESEARCH OR INNOVATION PROJECT

Deliverable D3.2

"Measurement-based Topology Modeling"

Project description
Project acronym: **EULER** Project full title: **Experimental UpdateLess Evolutive Routing** Grant Agreement no.: **258307**
Document Properties
Number: **FP7-ICT-2009-5_ICT-2007-1.6_(258307)_D3.2** Title: **Measurement-based Topology Modeling** Responsible: **UCL** Editor(s): **Julien Hendrickx, Jean-Charles Delvenne** Contributor(s) listed in alphabetical order: **Albert Cabellos (UPC/CAT), Jean-Charles Delvenne (UCL), Julien Hendrickx (UCL), Matthieu Latapy (UPMC), Dimitri Papadimitriou (ALB), Fabien Tarissan (UPMC)** Dissemination level: **Public (PU)** Date of preparation: **Dec.2011** **Version: 1.0**

Contents

Chapter 1

Introduction and Motivations

It appeared a decade ago that the Internet topology has features which make it very different from classical assumptions and models. As these properties were highly counter-intuitive, they received much attention, and much effort has been devoted to understand them and capture them into relevant models. Nowadays, measuring and modeling Internet-like topologies has become a well-recognized area of research in itself. Indeed, it makes no doubt that features observed in Internet measurements should be part of our modeling effort. However, gaining accurate and reliable knowledge of actual properties of the Internet topology is challenging. As stated by W.Willinger (in [74], p.586), "A very general but largely ignored fact about Internet-related measurements is that what we can measure in an Internet-like environment is typically not the same as what we really want to (or what we think we actually measure)". Indeed, the Internet is a huge and complex system composed of many different (sometimes buggy) protocol implementations, heterogeneous and poorly documented devices, operators implementing different (and often not measurement-friendly) policies, etc. As a consequences, data obtained from measurements is partial, and often biased. Going further, understanding such data and deriving appropriate conclusions is a challenge in itself and it relies on a deep knowledge of the actual deployment of infrastructures, both nowadays and in the past as old devices/software/protocols are mixed with the most modern ones. As a consequence, current knowledge of Internet topology remains limited, as well as confidence in previously published studies.

The classical approach for Internet topology measurement consists in collecting as much data as possible (using for instance traceroute of Border Gateway Protocol (BGP) tables) and in constructing from it a view of the topology by merging the obtained data. Although this approach may seem reasonable, in most cases network measurements give partial views of the networks under concern, contain erroneous data for the reasons cited above, and may moreover be intrinsically biased. For instance, it is shown in [51, 6] (both experimentally and formally) that the degree distribution observed on measurements of the Internet topology may be significantly biased by the measurement procedure. Similar problems occur for other properties and other networks [37, 52, 68, 69]. This has crucial consequences for the field. For instance, the results claiming that the Internet is very resilient to failures but sensitive to attacks [10, 13, 19, 20] rely on the assumption that the Internet has a power-law degree distribution with a given exponent, which is observed in measurements [31]. The fact that such degree distributions may be observed even if the underlying network has a totally different degree distribution [51, 6]) makes the relevance of these results unclear. This leads to difficult discussions and analyses of

the extent to which the observed degree distribution may be trusted [24, 37].

This problem is nowadays widely acknowledged: as network measurements rely on intricate procedures which give limited views of the network, the obtained views may have properties induced by the measurement procedure, and thus differ significantly from the ones of the studied network. However, only very limited and unsatisfactory solutions exist to cope with this problem: despite a few exceptions [52, 68, 69], most results on this topic are negative and show that the observed properties should not be trusted [51, 6, 52]. One approach could be to derive the properties of the network of interest from the observed ones (rather than simply consider that the observed ones are true), but this turns out to be very difficult. Another approach could be to conduct larger measurements, and this is indeed done (Caida [1], Dimes [66]). However, the network to measure generally evolves faster than our ability to measure it, and thus such approaches, even though they provide interesting data, do not solve the problem.

In order to overcome these difficulties, we propose here a new approach which consists in identifying properties which play a key role in the project, and then design and implement dedicated measurement strategies. We moreover use collected data to infer properties of interest. This is in sharp contrast with classical approaches.

Whereas measurements classically aim at obtaining a sample which is representative of the whole, we propose to design measurement procedures which, given a property of interest, will be able to give a precise and reliable estimation of this property. Different measurement strategies may be designed for a same complex network, each one addressing the reliable estimation of a different property (without collecting any sample representative of the whole). This drastically changes the way we design and conduct measurements, as they no longer aim at producing the largest possible samples but rather at producing samples which may be very limited in size while making it possible to rigorously deduce a property of the network.

In this task T3.2, we apply this approach to several issues: Section 2.1 presents a method for measuring the neighborhood of Internet core routers by sending traceroute probes from many monitors distributed in the Internet towards a given target router (see Section 3.1.1). After some intricate analysis, this enable to propose for the first time a reliable estimation of the degree distribution of core routers on the Internet. Section 2.2 focuses on the dynamics of Internet by using the Radar tool (see Section 3.1.2) to capture consecutive snapshots of routing trees and then extract new interesting properties of this object. This allows in a second step to propose new models able to reproduce those properties. Section 2.3 intends to complete the analysis of Radar data by providing tools for detecting events in the dynamics. It consists mainly in monitoring a set of statistical properties which will enable to detect any outliers in the evolution of the network. Section 2.4 investigates the same issue but with an emphasis on the stability of AS path. This part was implemented in a tool providing relevant information on BGP routing stability (see Section 3.1.3). Finally, Section 2.5 is devoted to the study of overlay networks triggering traffic on the Internet with a particular focus on peer-to-peer (P2P) networks. We analyze in this section how the propagation of information may be captured into specific models. The objective is again to provide efficient generator for such traces in order to help the simulation/emulation phase in WP4.

Once the particular aspects of the Internet network have been analyzed, each with their particular methodology, we want to use them in order to create or validate global models for the Internet network and its dynamics. For example the measured degree distribution allows to validate or invalidate random graphs models of the Internet, regarding their prediction on degree distribution. This can be achieved by standard statistical techniques such as χ^2 test, which however must be adapted to the present situation. Indeed standard statistical tests take as inputs a set of input often assumed to be i.i.d., while here we only have the degree distribution of one sample graph, measured from the real Internet, thus strict independence of the individual degrees cannot hold (for instance the sum must be even). In this context, the theory of statistical learning becomes useful, for instance to learn how observable events (i.e., outliers on observed quantities) correspond to their hidden causes, such as the loss of connectivity of a core router. The adaptation of these classical theories to the specifics of the Internet is a challenge, discussed in Section 4 of this document.

The combination of a new measurement approach and methodology, tailored for individual properties, with a reconstruction of a global model from those reliable measured properties with rigorous statistical techniques can, we believe, lead to a deep and trustworthy understanding of the Internet, directly useful for the WP2 and WP4.

Chapter 2

Measured Quantities

2.1 Degree distribution

2.1.1 Objective

The degree distribution of a network topology (i.e. for each integer k the fraction p_k of nodes with k links) is one of its most basic properties. It has a strong impact on key features of the network like efficiency of protocols and robustness to failures and attacks. Until the seminal papers [59, 31] in the late 90's, it was commonly assumed that the degree distribution of the Internet followed a homogeneous law, generally modeled by a Poisson distribution. Based on actual measurements, though, these papers gave evidences of the fact that it may be much more heterogeneous, better modeled by a power-law. Since then, much effort is devoted to studying this distribution and capturing it in appropriate models of the Internet.

However, recent works (see Section 2.1.2 below) have shown that the observed degree distribution of the Internet may be significantly biased by the measurement process. Indeed, it is deduced from partial maps obtained through intricate and unreliable measurements which tend to give biased views.

Our objective here is to propose and implement a new approach in order to rigorously estimate the degree distribution of the Internet, in a much more reliable way than before. For this purpose, we first focus on the physical structure of the core Internet: nodes are routers and their degree is their number of interfaces (i.e. IP addresses). Instead of building maps of this topology, we design a method to accurately estimate the degree of a given router; then we build a process that allows to select core routers uniformly at random, and estimate their degree. We therefore obtain an accurate estimate of the degree of a set of random nodes in the considered topology, representative of the whole.

2.1.2 State of the art and contribution

The classical approach to estimate the Internet degree distribution in general is to build a map of it using traceroute-based measurements (each run of the traceroute tool gives in principle a path in the topology and the map is built by merging large numbers of such paths) [59, 31, 5]. The degree distribution of the map is usually assumed to be representative of the actual one.

However, it has been shown that such measurements lead to intrinsically biased views [51, 6, 25, 37, 39, 52], and in particular that the observed degree distributions may be a consequence of the measurement approach more than a true property of the underlying network. More precisely, whereas it is often assumed that the degree distribution of the Internet is heterogeneous and close to a power-law distribution Lakhina et al. [51] experimentally showed that estimating the degree distribution by the classical method leads to the observation of an heterogeneous distribution even if the underlying topology has a Poisson degree distribution. This result was proved formally in [6] in the case of a single source, and extended to the case of other graph topologies.

As a consequence, current knowledge of the degree distribution of the Internet topology, one of its main properties, is unsure and subject to controversy. This calls for the development of new approaches, while most current effort is devoted to the collection of larger and larger maps with the expectation that larger maps will be more representative.

We have shown in [22] using simulations that another approach is appealing. It consists in estimating the degree of specific target nodes by sending probes from distributed monitors. By selecting random nodes, one may then obtain an estimate of the degree distribution of the network. The simulations presented in this paper are very promising: they show that this approach succeeds in accurately estimating the degree distribution of various topologies and is free from bias. Though this work remains merely based on simulations, our goal here is to implement it and conduct actual measurements in the Internet core, and so to obtain a reliable estimate of its true degree distribution. We first present our method below, then the measurement we conducted and the obtained results. This work is still in progress, so we highlight key direction for current and future work.

2.1.3 Methodology

Our approach relies on our ability to accurately measure the degree of a given core router and to uniformly sample core routers (independently from their degree). We then measure the degree of such random routers and obtain the degree distribution of a representative set of routers. If the set is large enough, this degree distribution is close to the one of the whole network.

2.1.3.1 Measuring the degree of a given router

When a machine in the Internet receives an invalid packet from a sender S, it is in general supposed to answer to S with an error message using the dedicated ICMP protocol. Many tools, including traceroute and many alias resolution tools, rely on this feature to generate errors which provide information regarding the network. An important feature of such error messages is that they are in principle sent using the interface of the machine that routes packets towards S. Therefore, by generating an error on a target t, a monitor m obtains an interface of t. If many monitors distributed in the Internet do so, then they will probably obtain *all* interfaces of the target, and so its degree.

We implement here this approach with *UDP-ping*, a tool that we designed and implemented, which sends a UDP packet to a target on an unallocated port. See Section 3.1.1 for details on the implementation and behavior of this tool.

Notice that not all routers answer to such probes, and that some routers always use the same interface to answer. Likewise, some routers may answer with random interfaces. We will handle such issue below. The key point here is that, given an appropriate set of monitors and a correct target, we are able to accurately estimate its degree. This is already an improvement over previous situation, where no such tool existed and where the degree of a node could only be guessed from maps obtained with traceroute-based measurements.

2.1.3.2 Sampling targets

With the ability to reliably estimate the degree of a single router as explained above, one may estimate the degree distribution of a set of nodes (by applying the method to each node in the set). If this set is a set of nodes taken uniformly at random in the core Internet (i.e. all the nodes have the same probability to be chosen) then the obtained degree distribution is itself an approximation of the degree distribution of the core Internet, and the larger the sample, the better the approximation. However, choosing uniformly at random a node in the core Internet is not possible in general. We explain here how to bypass this issue.

First notice that it is easy to sample a random IP address, as this is noting but a 32 bit integer. But a random IP address is *not* a random node, as routers may have several interfaces (aka IP addresses) and then the probability to sample a router by sampling a random address is proportional to its number of interfaces. As a consequence, if we know the degree distribution (p_k) of nodes corresponding to IP addresses sampled uniformly at random, then the degree distribution of the set of *all* nodes is nothing but (p_k/k).

In summary, although we are unable to sample uniformly at random nodes in the core Internet, we are able to sample uniformly at random IP addresses and infer from the degree distribution of the corresponding nodes the one of the whole core Internet.

In addition, the random IP addresses we sample may belong to routers outside the core, to end-hosts, or even to routers in the core behaving incorrectly (they do not answer to probes or always use the same interface to answer). We easily identify IP addresses which do not belong to core routers: we see only one interface for them during our measurements, and simply discard them (as a consequence, we obtain no estimate of p_1, the fraction of nodes of degree 1). Notice that core routers which behave incorrectly also lead to nodes observed with degree 1, or even 0, and discarding them is correct. This induces no bias on the observed degree distribution if their behavior is not correlated to their degree, which seems a reasonable assumption.

Finally, we obtain a practical method to sample a set of uniformly random IP addresses of core routers, and to infer from their degree distribution the one of the whole core Internet. The quality of this estimate obviously depends on the quality of our set of monitors and on the size of our set of IP addresses. In-depth study of the achieved

quality is a challenging task which belongs to future work, but we will discuss below ways to do so.

2.1.4 Measurement

We implemented our method and ran it on the Internet using a set of 625 monitors provided by the PlanetLab infrastructure. We targeted 3 millions of IP addresses chosen uniformly at random among the ones answering UDP-ping probes (we sampled uniformly at random a huge set of 32 bit integers, sent a UDP-ping packet to the corresponding IP addresses, and kept only the answering ones). Building this target list took approximately 10 hours. We then sent it to each monitor, which ran UDP-ping towards each of the 3 million targets, in a random order (to avoid the arrival of too many probes in a short period of time at a same target, which would look like an attack of the target). We repeated the whole process 3 times in a row with the same targets, in order to gain insight on the dynamics of the observed degrees, the reliability of an observation, and to study possible impact of previous measurements on the next ones. Each iteration lasted less than 4 hours. The complete measurement, from the beginning to the end, lasted less than 24 hours, which is much shorter than usual Internet measurements (this is important to avoid possible bias due to the dynamics of the Internet).

Let us insist on the fact that conducting such measurements is a challenge in itself. First, we rely on a set of distributed, heterogeneous and not always stable monitors. More importantly, the load induced on the network to obtain such quick measurements is not negligible: on the monitor side, the induced traffic is non-trivial (in particular, we send packets to an *abnormal* number of IP addresses) and on the target side packets are received from many unknown distributed sources, which may look like a distributed denial-of-service attack or a port scan (we avoided this last risk by targeting always the same random port). It must therefore be clear that such measurements must be very carefully controlled, and that repeating them or significantly increasing their size is extremely difficult. Future work may be done in this direction by exploring the possibility to conduct much slower measurements while being able to avoid bias due to the dynamics of the Internet. This is an important goal, as this would open the way to the study of the representativity of each measurement, to the study of the dynamics of the degree distribution, and to better estimates thanks to larger sets of targets.

Each of the three iterations of measurement above produced an estimate of the degree of each node with an interface in our set of 3 million targets. However, as explained above, a post-processing of this data is necessary before obtaining an estimate of the degree distribution of the core Internet.

First, we removed suspicious, inaccurately measured or invalidly targets:

- targets from which at least one monitor received more than one answer to a probe (the target is mis-configured, a packet duplication occurred during the measurement, the address is a multi-cast address, etc); we assumed that such behavior is not correlated to the degree of the target, and so removing such targets induces no bias;
- targets from which abnormally few answers were received (significantly less than

average targets), i.e. targets for which we obtain insufficient information;

- targets for which we observed only one single interface, indicating that the target machine is not in the core or is not properly configured.

Finally, we obtain a set of 5500 targets, for which we have a reliable estimate of their degree, leading to an estimate degree distribution (p_k). This sample is biased as explained above, and our final estimate of the degree distribution of the whole Internet core is (p_k/k)

2.1.5 Results

We display in Figures 2.1, 2.2 and 2.3 the degree distributions obtained from our measurements before and *after* the post-processing and bias correction described above.

The obtained degree distribution is, up to our knowledge, the first estimate obtained with a method designed and implemented for this purpose. All previous attempts relied on traceroute-like measurements which are known to produce unreliable estimates. It makes no doubt that the estimate we obtain here is much more reliable than any previously published one.

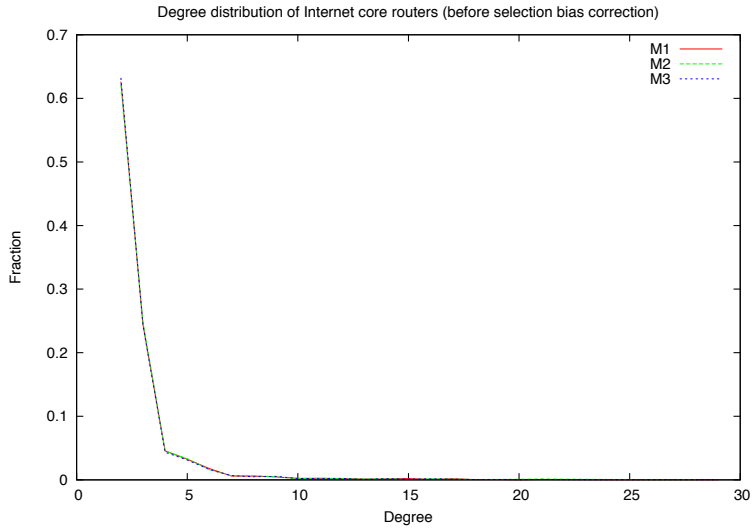
In order to explore further this reliability of this estimate, we studied its dependence over the set of monitors and targets (including the *number* of monitors and targets). We have observed that changing the target sets has negligible impact. We have also shown that a significant number of our monitors are well distributed and that removing half the monitors has negligible impact on the results. Likewise, removing half the targets has negligible impact. Finally, we have shown using extensive simulations that the degree estimation for each node is reliable; the probability to observe a target with a low degree (like here) with such a number of monitors is extremely low if its actual degree is very large.

To go even further, one may use statistical tools like re-sampling and statistical tests. Measurements with other/more monitors and long-term measurements with significantly more targets are also promising directions. We also expect other methods to be designed to produce other estimates to which we could compare our own. We may then obtain more insight on the quality of our estimate and on the actual degree distribution of the Internet, the main property used to model it.

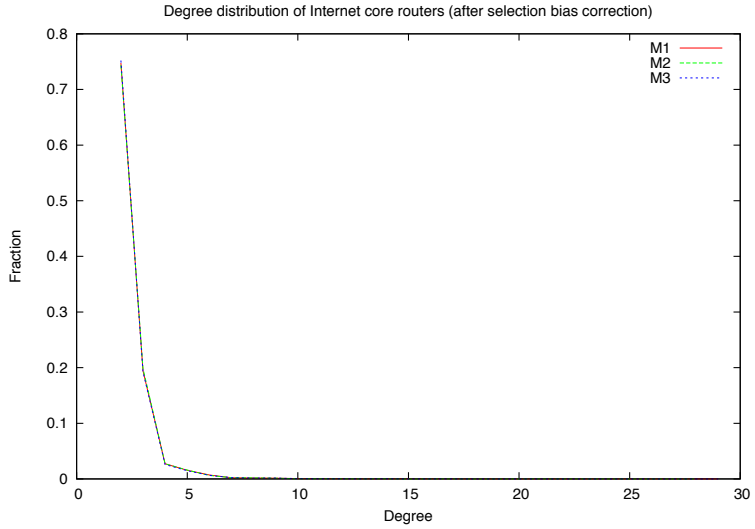
2.2 Dynamics of IP-level Internet topology

Many works have studied the Internet topology at various levels, but few have investigated the key question of its dynamics. One of the main reasons is due to the difficulty to obtain maps of the Internet at a frequency and a scale that would allow a study of its dynamics. As a consequence, available studies focus on long-term evolution of the Internet [57]. However, insight on the real-time dynamics of the Internet is mandatory to build relevant models of the topology and its dynamics, which in turn are crucial for simulations and protocol design.

Some insight on the Internet topology dynamics was obtained recently by studying the dynamics of *routes* as seen by traceroute. In particular, it was shown that load-balancing is prevalent[72]: some routers, called *load-balancers*, send packets for a given

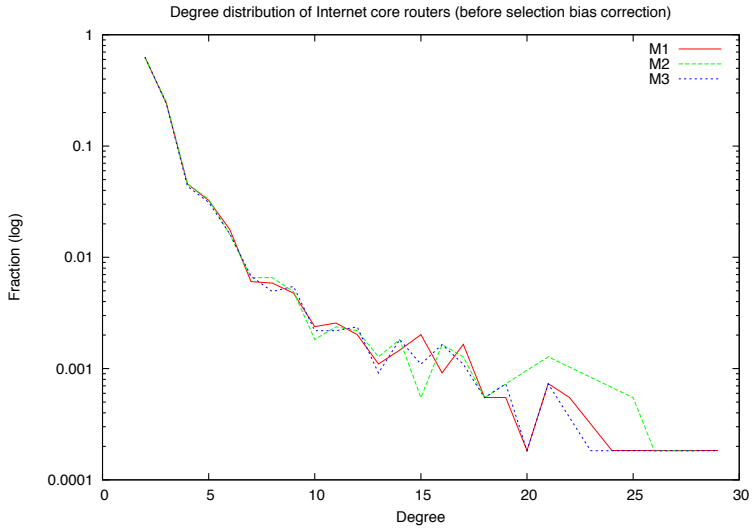


(a)

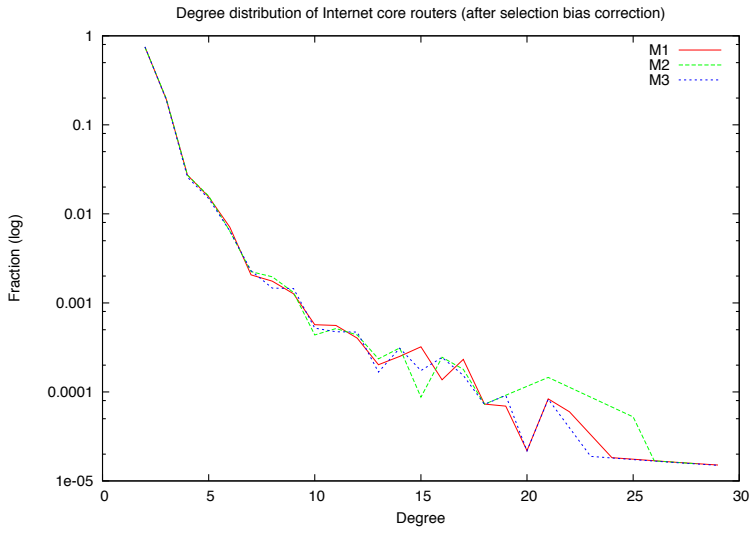


(b)

Figure 2.1: Observed degree distribution before (a) and after (b) the correction of the selection bias (lin-lin)

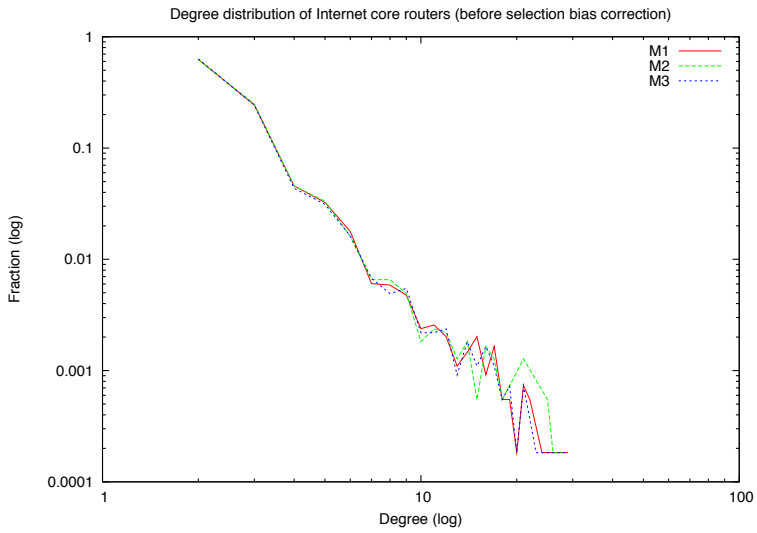


(a)

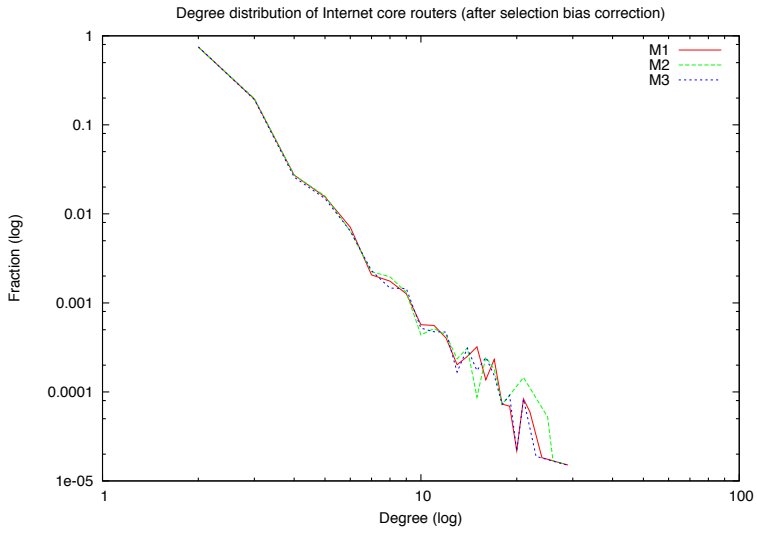


(b)

Figure 2.2: Observed degree distribution before (a) and after (b) the correction of the selection bias (lin-log)



(a)



(b)

Figure 2.3: Observed degree distribution before (a) and after (b) the correction of the selection bias (log-log)

destination on different paths. This in itself induces a dynamics of observed routes, which comes in addition to the dynamics due to routing table updates, and addition/removal of physical devices. The dynamics of the Internet is therefore a mix of several kinds of dynamics, very different in nature but difficult to distinguish from measurements.

We use here measurements of routing trees (i.e. the routes from a monitor to a given set of targets) obtained with tracetree, which can be measured every few minutes [53]. We use the dynamics observed in such measurements as a proxy to the actual Internet dynamics. We identify key properties of the observed dynamics and design a very simple model able to reproduce them. The goal of this model is not to be realistic in the sense that it would reproduce the root causes of the dynamics; instead, it aims at showing that some simple features of the model are sufficient to explain some unexpectedly observed phenomena. As a consequence, future models of the dynamics of Internet topology should include these features as building blocks.

2.2.1 Motivation and Objective

This work focuses on the Internet IP-level topology and proposes a first step towards realistic modeling of its dynamics. We first perform periodic measurements of routing trees from a single monitor to a fixed destination set. By analyzing this data, we identify invariant properties of its dynamics. We then propose a simple model for the underlying mechanisms of the topology dynamics. Simulations show that it effectively captures the observed behaviors, thus providing key insights of relevant mechanisms governing the Internet topology dynamics.

Studying the structure of the Internet topology is an important and difficult question. No map of the Internet being available, researchers have to conduct costly measurement campaigns, and deal with the fact that the obtained data can be biased. Studying the dynamics of this topology is therefore an equally hard, if not harder, problem.

In this subtask, we use an orthogonal approach to obtain insight on the dynamics of the IP-level routing topology. We study ego-centered views of this topology, i.e. routing trees from a given monitor to a given set of destinations. Such trees are basically composed of the routes from the monitor to each of the destinations. Each such route may be measured using traceroute, but we designed a new tool, named tracetree, which performs this measurement much more efficiently [53]. It makes it possible to measure a routing tree from a monitor to 3000 destinations in typically a few minutes, which limited network load (much smaller than traceroute). This in turn makes it possible to repeat such measurements periodically at a relatively high frequency (minutes), thus capturing the dynamics of routing trees [53]. We use here this dynamics as a proxy to the dynamics of the Internet topology.

Previous work has shown that the objects observed this way are highly dynamic: that they evolve much more quickly than what was previously expected [55]. Here, we analyze in depth the dynamics of measured routing trees. We show that two features play a key role in these dynamics: load-balancers, and the evolution of the routing topology.

Based on these observations, we propose a simple model for the routing dynamics in

the Internet. We show that it accurately reproduces the behaviors observed in real data, thus providing an explanation for these observations. Our model is basic and has an explanatory value rather than being a model allowing precise and realistic simulations. In particular, it does not take into account the long term evolution of the topology with the addition/removal of nodes and links (due new material, failures, etc). However, it represents a crucial first step towards the modeling of the Internet IP-level topology and its dynamics.

2.2.2 State of the art and Related work

Study of the dynamics of the Internet topology has been tackled both by analyzing the dynamics of individual routes [63, 65, 50, 47] and from a more global perspective, mainly at the *as* - or *ip*-level [34, 16, 57, 61, 58]. Load-balancing has also been acknowledged for playing an important role in the dynamics of routes as measured with traceroute-like tools [72]. Cunha *et al.* [23] used a method for measuring load-balanced routes, *i.e.* routes containing one or more load-balancing routers, and study their dynamics. Most related to our characterization of the evolution of the Internet topology is the work by Oliveira *et al.* [57], which analyzed the *as* topology and shows that real topology dynamics can be modeled as constant-rate birth and death of links and nodes.

Work on modeling the Internet topology and its dynamics can be roughly divided between approaches aiming at reproducing global network characteristics through simple mechanisms, and approaches aiming at realistically mimicking the evolution mechanisms of the topology, *e.g.* reproducing the criteria taken into account by ASes for creating peering or customer-provider links [15, 73, 62]. Tangmunarunkit et *al.* [70] found that network generators based on local properties, such as the degree distributions of nodes, can capture global properties of the topology. This shows that simple, not necessarily realistic mechanisms such as the ones we introduce below may be relevant for capturing important properties.

Whereas most existing works focus on the long-term evolution (*e.g.* from the Internet birth to current times, or its evolution over years) of the AS topology, we are concerned here with the short- to medium-term evolution of the *routing* topology at the IP-level. The two topologies are closely related but not identical objects. In particular, routing changes can occur in the absence of changes in the AS topology. Finally, our model does not take into account node birth and death, which would be necessary for modeling the long-term topology evolution.

2.2.3 Methodology

2.2.3.1 Data collection

We use the *tracetree* tool (see [53] and Section 3.1.2) to collect two datasets of periodic measurements of ego-centered views. The first one, called *woolthorpe*, was collected from a monitor in Pierre and Marie Curie University in Paris towards a set of 3,000 destinations. The collection frequency is of one measurement round every 15 minutes approximately. It started in December, 2010 and ended in June, 2011, which leads to a total of 17,450 rounds. The second one, called *ovh*, was collected from a French server hosting company, using 500 destinations in order to increase the measurement frequency,

which is of one round every one and a half minute approximately. It started in October 2010 and ended in September 2011, which represents a total of 318,000 rounds. These datasets are publicly available [38]. The frequencies were empirically determined, and chosen to be as high as possible while avoiding perturbations of the network (see [53] for details on how to conduct measurements with tracetree).

2.2.3.2 Observed dynamic features

We present below two key characteristics of the dynamics observed in these datasets. We focus on the dynamics of the set of observed nodes, which certainly is the most basic property and so constitutes a relevant entry point. We conducted preliminary analysis regarding link dynamics, which indicate similar behaviors.

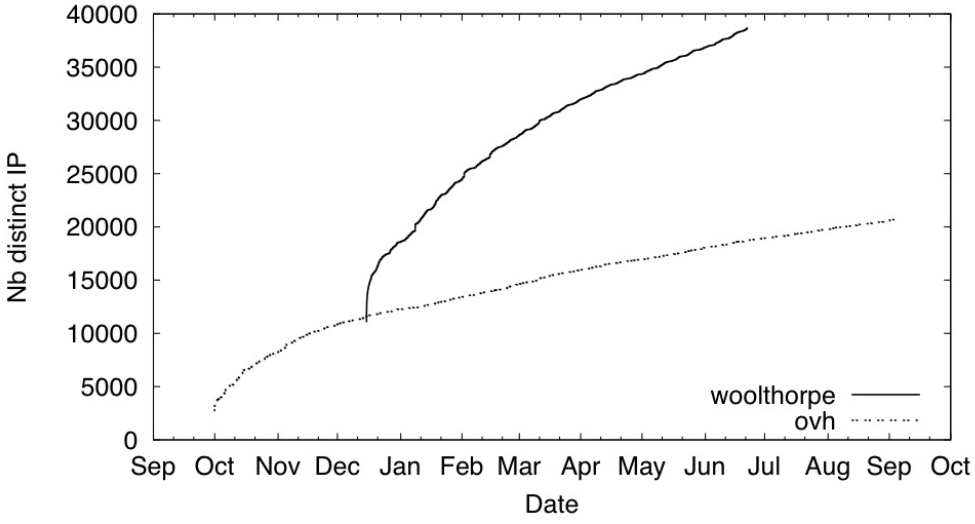
Notice also that previously measured datasets are available for different durations at different times since 2008. We performed our analysis on a representative set of these datasets and made similar observations to the ones we present here. This shows the generality of our observations.

Renewal of IP addresses. A previous study of the same type of data has shown that the set of observed IP addresses does not stabilize with time [55]. Instead, it was observed that new IP addresses that had never been observed before were continuously discovered at an important rate. These observations were made on two-months-long measurements. Figure 2.4(a) shows that it is also the case for very long lasting measurements. It presents the number of IP addresses observed since the beginning of the measurement, for both datasets. A dot (x, y) in these figures means that y different IPs have been discovered between the round 1 and x. We see that, after an initial fast growth, the plot increases linearly, for extended periods of time (although it will eventually not grow forever as the number of IP addresses is bounded).

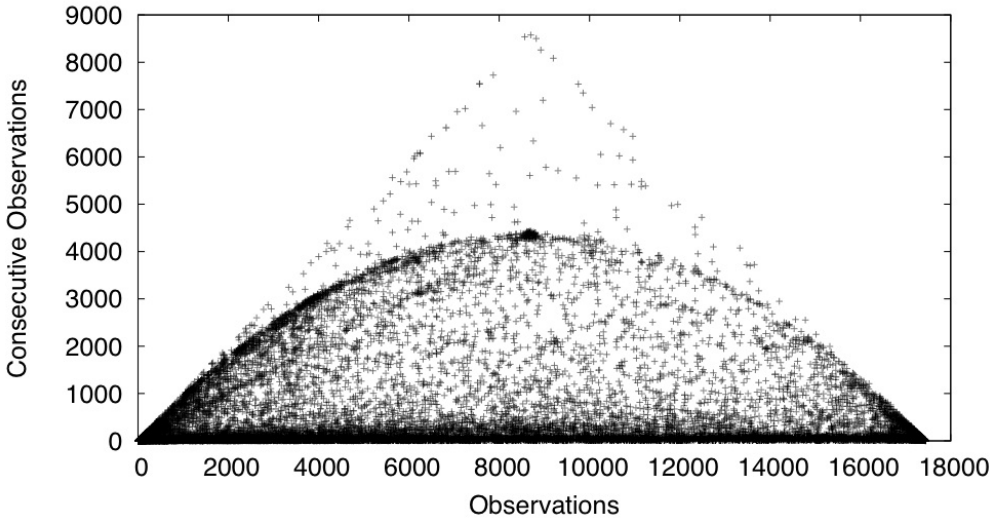
Observation of IP addresses. To analyze more in depth the dynamics of the ego-centered views, for each IP address, we compute two quantities. Its number of observations is simply the number of distinct rounds it was observed in. We also study its number of consecutive observations, i.e. the number of groups of consecutive rounds in which it consistently appears. For example, an IP address which was observed on rounds 1, 3, 4, 7, 8, 9, and 10 corresponds to 7 observations and 3 consecutive observations.

Figure 2.4(b) presents the correlation between these quantities for the woolthorpe dataset. Each dot corresponds to an IP address, and its coordinates are its number of observations on the x-axis and its number of consecutive observations on the y-axis. The plot presents a clear geometric shape which can be explained in the following way. There can be at most as many consecutive observations as observations (therefore $y <= x$); conversely, there cannot be more consecutive observations than the number of rounds in which the corresponding address was not observed (therefore $y <= r - x$, r being the total number of measurement rounds). This defines the borders of the triangle.

The presence a large number of IP addresses close to the parabola can be explained by load-balancing routers. If a load-balancing router spreads traffic among k paths, each router belonging to any of these paths has a probability $p = 1/k$ of being observed at each round, leading to rp observations approximately.



(a)



(b)

Figure 2.4: (a) Properties of the observed dynamics and the renewal of IP addresses over time (a) and vs consecutive observations (b).

A given round then corresponds to the first of consecutive observations with the probability p that it was observed in this round, multiplied by the probability $1 - p$; p that it was not observed in the previous round. Multiplying this probability by r gives the number of expected consecutive observations, which is then equal to $rp(1 - p))$ and is the equation of the parabola. This is a simplification of the real case in which a router may belong to paths used by several load balancers, themselves belonging to paths used by other load balancers. In practice, an IP address belonging to load-balanced paths can have any probability p, $0 < p < 1$, of being observed.

IP addresses above the parabola tend to blink more than expected: those which are at the tip of the triangle are observed exactly every other round. Investigation showed that this kind of behavior is caused by the tracetree tool design in presence of per-destination load-balancing. Faced with two load-destination balanced routes to destinations d_1 and d_2, the tool indeed tends to alternate between measuring the route to d_1 and the one to d_2 between consecutive rounds.

We can also observe a large number of dots close to the $y = x/2$ line. Manual investigation showed that they correspond to addresses that are observed only during a finite part of the measurement, and have a probability $p = 1/2$ of being observed, due to load balancing. If such an IP address is observed with a probability $1/2$ during k rounds, its number of observations will indeed be $x = k/2$, and its expected number of consecutive observations will be $y = k(1/2)2 = x/2$.

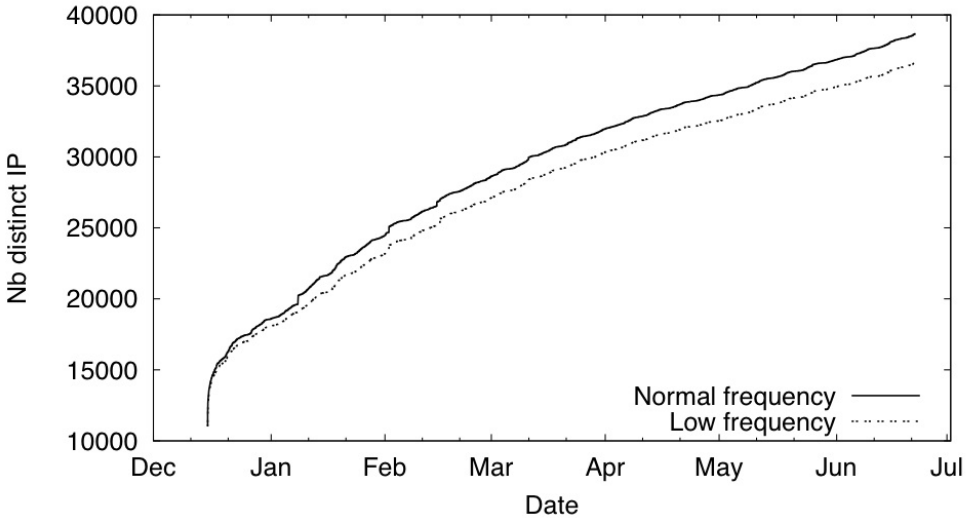
Finally, a large number of IP addresses are close to the x-axis, which means they are mainly observed during consecutive rounds.

Due to space constraints, we do not present the corresponding plot for the ovh dataset. It has the same aspect than the one for the woolthorpe dataset, except that there are no points significantly above the parabola. Manual inspection indicates that this is caused by the very long duration of these measurements. Indeed, if we take into account only a small part of these measurements, then the corresponding plot is identical to the one for the woolthorpe dataset. IP addresses above the parabola are caused by per-destination load balancing. Over time, the routes to these destinations evolve, which removes the tendency of the tool to observe them every other round.

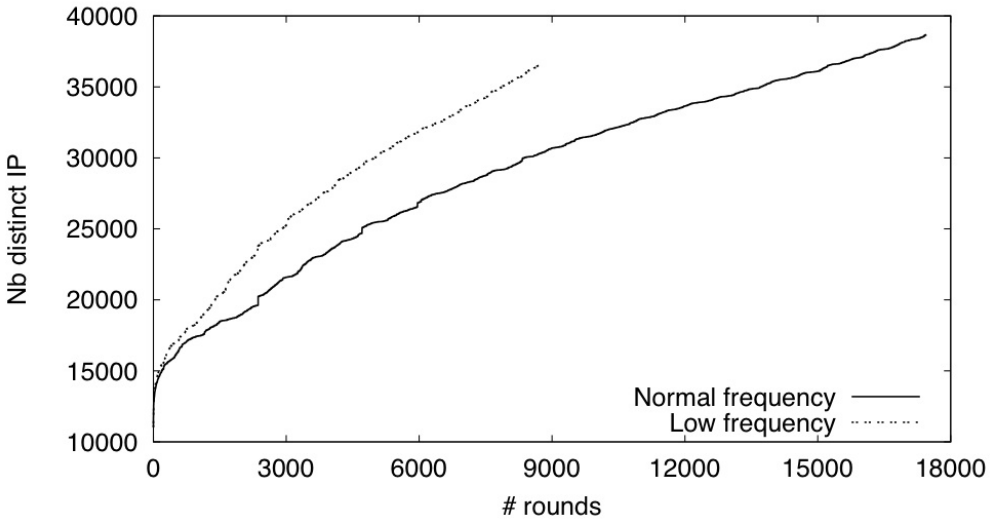
2.2.4 Impact of different dynamics

It is acknowledged that load-balancing routers play a significant role in the observed dynamics of routes with *traceroute*-like measurements [4]. Previous work also suggests that routing dynamics play a key role in the continuous renewal of IP addresses we observe [55]. This section identifies the strong role played by these factors in our observations.

These two factors play different roles. Intuitively, in presence of routing changes, the longer a measurement lasts, the more IP addresses we will observe (because more changes will have occurred). But in presence of load balancing, on the contrary, performing more measurement rounds will lead to observe more IP addresses, independently of the time elapsed between consecutive rounds (This is of course only true under certain



(a)



(b)

Figure 2.5: Influence of frequency on the number of discovered IP addresses as a function of time in months (a) and in rounds (b)

conditions on the number of measurement rounds and the time elapsed between consecutive rounds.). Note that Figure 2.4(a) may seem to disprove the former statement since, during similar periods of time, we discover less IP addresses in the *ovh* dataset than in the *woolthorpe* one, although the former has a higher frequency. This is because it uses less destinations, which naturally induces a lower number of discovered IP addresses.

In order to study the impact of measurement frequency rigorously, we use the *woolthorpe* data set and simulate slower measurements by considering only one out of every two rounds. Figure 2.5 presents the number of distinct IP addresses observed with both these measurements, as a function of time elapsed since the beginning of the measurement, and the number of measurement rounds performed.

As expected, less IP addresses are observed over time with the slow measurements than with the faster ones. Figure 2.5(a) shows that in a given time interval, performing more measurement rounds therefore allows to discover more IP addresses. This confirms that several measurement rounds are needed to discover all existing routes. This is caused by factors such as load balancing. Conversely, Figure 2.5(b) shows that the slow measurements discover more IP addresses *at each round* than the faster ones. Therefore if more time elapses between two consecutive rounds, then each round discovers more IP addresses. This indicates that routes evolve with time.

In both cases, the gap between the plots for the slow and faster measurements are significant, which shows that both factors play an important role in the dynamics. This is why we propose a model that incorporates load balancing and route dynamics.

2.2.4.1 Model

Our purpose here is to propose relevant and simple mechanisms that reproduce the observations made above. We propose a simple model for these phenomena. Note that this model does not aim at being realistic, but rather at providing a first and significant step towards understanding the impact of simple mechanisms on the observed network dynamics.

Given a graph representing the topology, we assume that the route between the monitor and a destination is a shortest path. In order to simulate load-balancing, we implement a *random breadth-first search (BFS)*. It generates a tree from the monitor to the destinations by considering the neighbors of explored nodes in a random order. These routing trees will therefore be different from one random BFS to the next, even if the underlying graph does not change.

Second, we need to model changes in the routing topology. We use a simple approach based on link rewiring, or *swap*. It consists in choosing uniformly at random two links (u, v) and (x, y) (We choose them such that the four nodes are distinct.) and swap their extremities, *i.e.* replace them by (u, y) and (x, v).

Finally, our simulation setup consists in the following. First, we generate a random graph G_1 using the Erdös-Rényi model [30]. From G_1, we randomly select one node as the monitor and d nodes as the destinations. We then simulate r measurement rounds

by iterating the following steps:

- extract a routing tree T_i from G_i ($i \in [1..r]$) by performing a random BFS from the monitor towards the destinations;
- modify the graph G_i by performing s random swaps, which produces the graph G_{i+1}. s is a parameter of the model.

This process generates a series of r trees $T_1, T_2, ..., T_r$ that simulates periodic *tracetree* measurements, on which we can conduct similar analysis as those we performed on real data.

Notice that ER graphs are certainly not realistic models for the internet topology, as well as link swaps are certainly not accurate models of the topology dynamics. They however have the avantage, crucial here, of being extremely simple and well known models. In addition, link swaps turn any ER graph into another one, and so all graphs G_i are ER graphs. This ensure that we always deal with well known and formally tractable objects here. We will see that, although considering more realistic models is an important direction for future work, this simple model already capture key features of observed dynamics.

2.2.5 Results

We investigate here whether this model is relevant to explain the dynamics properties observed on the data. To that purpose, we perform several simulations with various parameters of the model: the numbers n of nodes, m of links, d of destinations, and s of swaps per round. Our goals are (1) to find whether the simulations reproduce the observations and (2) how the values of parameters impact the results and which relations between the parameters lead to invariant behaviors.

Reproducing the evolution of node discovery. We first study how the simulations behave regarding the evolution of the number of distinct nodes over time. Figure 2.6 presents such an analysis on a graph with $n = 500,000$, $m = 1,000,000$, $d = 3,000$ and various values of s. It shows a similar behavior to the one we observed in real data (see Figure 2.4). In particular, for a small number of swaps (less than 100), the curves present clearly a fast initial growth (this phase lasts more than 1 round, although it is difficult to visualize it on the plot.) and then a linear progression. Moreover, the slopes of the curves increase with the number of swaps. This is due to the fact that with a higher number of swaps, the paths to the destinations change even more quickly and thus more nodes are discovered at each step.

The extreme cases are also very informative. For instance, when the underlying graph does not evolve ($s = 0$), there is an initial growth in which all shortest paths are explored. Once all nodes on these paths have been discovered, the curve becomes flat. This confirms that the regular discovery of new *IP* addresses in real data may stem from route dynamics. At the opposite, when the route dynamics is very high (*e.g.*, $s = 10,000$), almost all the nodes are discovered (89.49% of the nodes after $2,000$ rounds). Indeed, the modifications of the underlying topology is so important – $20,000,000$ swaps over

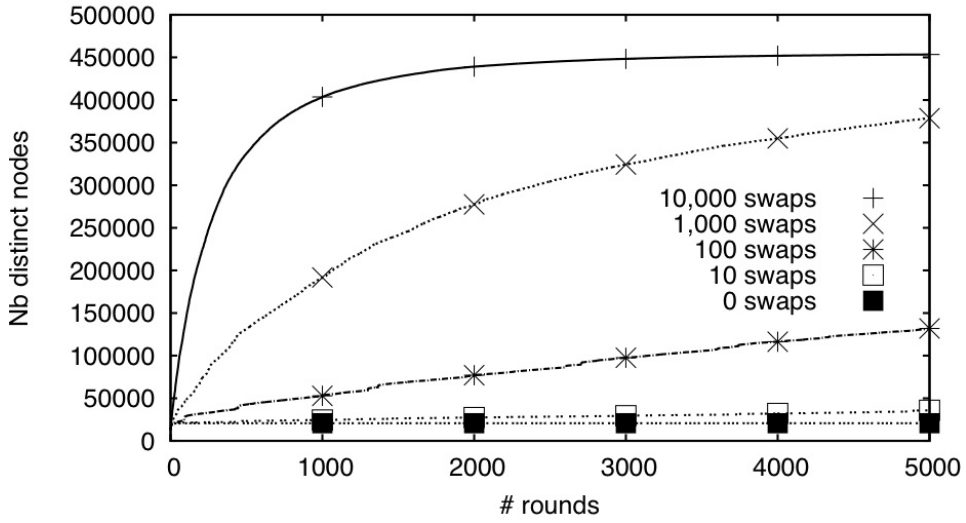


Figure 2.6: Impact of the number of swaps ($n = 500,000$, $m = 1,000,000$, $d = 3,000$)

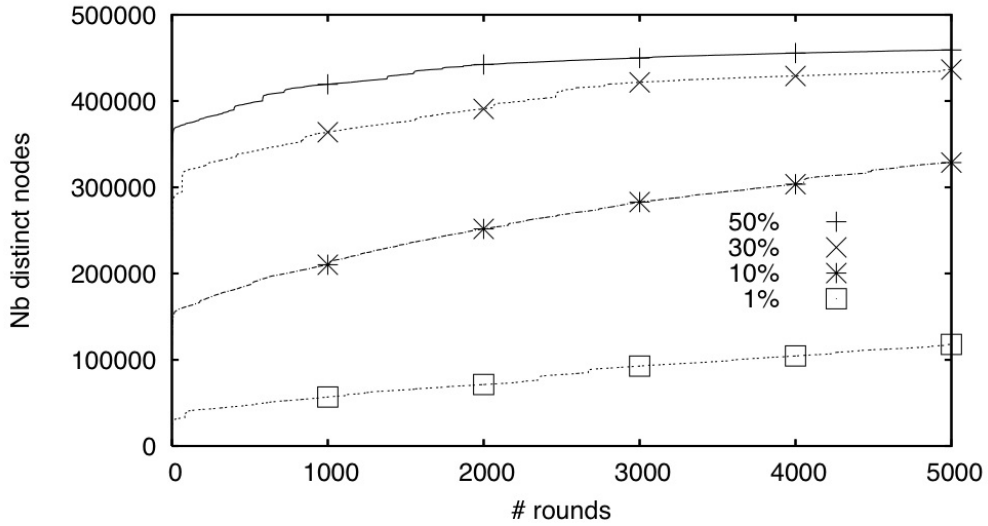


Figure 2.7: Impact of the number of destinations ($n = 500,000$, $m = 1,000,000$, $s = 50$)

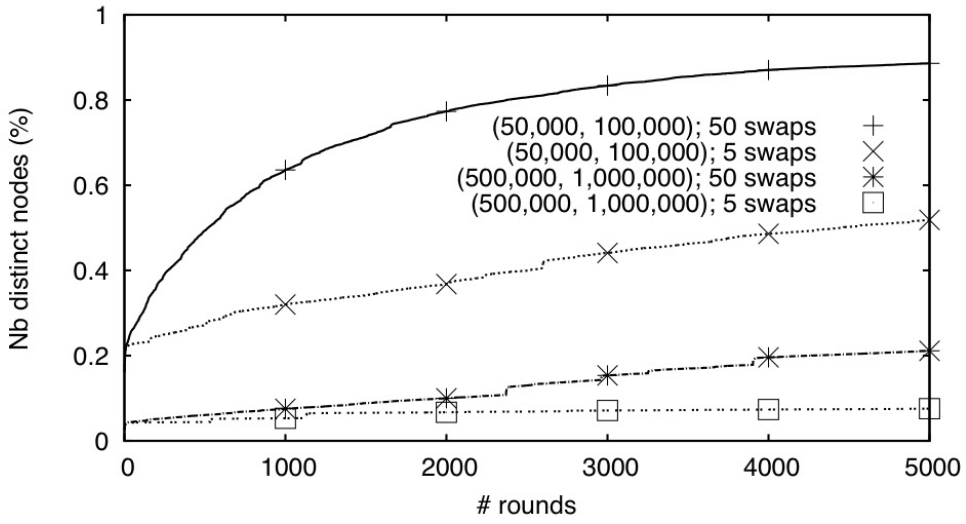


Figure 2.8: Relation between size and swaps ($m/n = 2$, $d = 3,000$)

the $1,000,000$ original links– that eventually all the nodes will appear at some point on a shortest path (note that we never succeed in discovering the entire graph as degree-0 nodes will remain unreachable since the swaps preserve node degrees.)

In order to confirm these results, we also study the impact of the number of destinations d on a graph with $n = 500,000$, $m = 1,000,000$ and $s = 50$. Intuitively, increasing the number of destinations causes the shortest paths to the destinations to represent a larger fraction of the graph. Indeed, we see in 2.7 that the initial growth phase reaches a higher value. As before, this phase is followed by a linear increase, except for extreme cases in which most of the nodes are discovered. Notice that increasing the number of destinations does not affect much the slope.

Invariant relations between parameters. In order to further study the parameter ranges which allow to reproduce the invariants, we now vary several parameters in parallel.

We first set $m/n = 2$ and $d = 3,000$ and then vary n and s. Figure 2.8 shows that for given values of n and m, the value of s does not affect the height of the initial growth. Indeed the number of nodes on shortest paths does not depend on the number of swaps. This result confirms that swaps affect mainly the linear part. In addition, it seems that the slope of the linear part of the curves are similar when the ratio of the number of swaps over the number of links remains the same. The two curves in the middle of Figure 2.8 seem to increase with same relative slope, although they have different heights.

We now turn to the relation between the number of swaps and the number of links in Figure 2.9. We set $n = 500,000$ and $d = 3,000$. We first observe that, for a given number of swaps, the larger the number of links, the smaller the slope of the corresponding curve. This comes from the fact that, when the number of links increases, a smaller fraction of them is affected by swaps. Second, different curves with a same ratio s/m will have the same slope. We can observe this in the two middle curves. Note that there

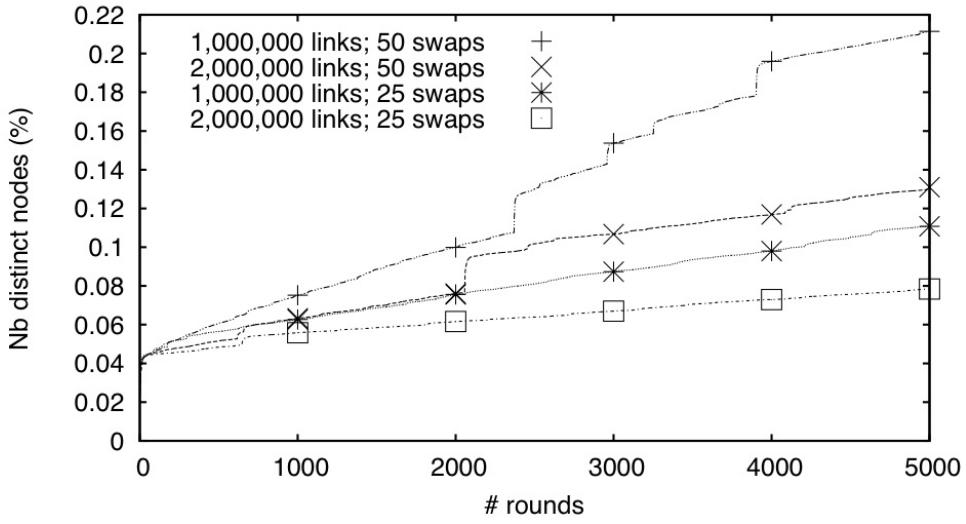


Figure 2.9: Relation between links and swaps ($n = 500,000$, $d = 3,000$)

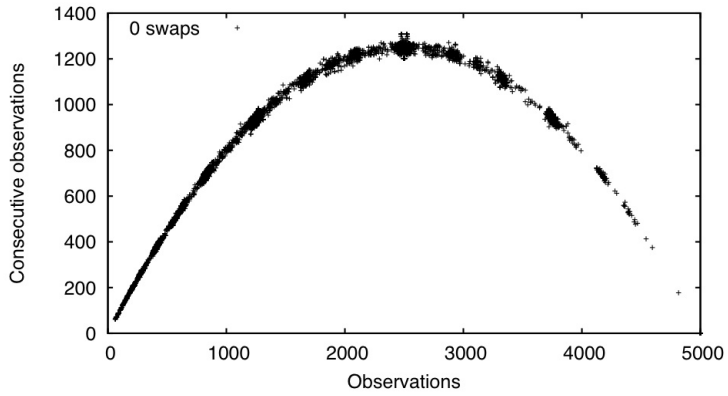
is a sharp increase in one of them close to $x = 2,000$. It is probably caused by a swap happening very close to the monitor, which therefore affects a larger part of the paths than usual. Yet, this does not affect the slope after such an event.

Observations vs. consecutive observations. We finally study in Figure 2.10 the correlations between the number of observations and consecutive observations. We fix $n = 500,000$, $m = 1,000,000$ and $d = 3,000$.

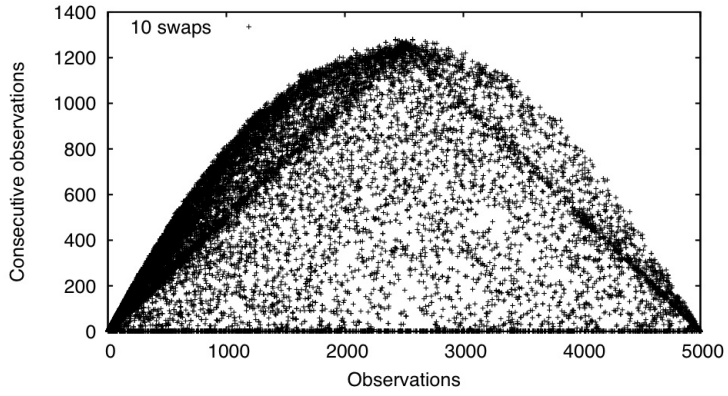
For $s = 10$ (Figure 2.10(b)), the main invariants we observed are reproduced: the parabola, the $y = x/2$ line and a dense strip close to the x-axis. Notice that, in this case and the others, no triangle appears above the parabola. As already explained, this behavior observed on real data comes from an artefact induced by the *Tracetree* tool itself in presence of per-destination load-balancing. It is therefore natural that the model does not reproduce it.

We also observe a high density of nodes on a line with equation $y = (r-x)/2$, r being the total number of rounds performed. As already explained before, the line $y = x/2$ corresponds to nodes that are observed with probability $p = 1/2$ for a given duration, and are not observed before or after. The line $y = (r-x)/2$ has a similar explanation: it corresponds to nodes which are observed with probability $p = 1/2$ for a given duration, and are observed *at all rounds* before and after that. Although this line is not present in Figure 2.4(b), it sometimes can be observed in other datasets, although not as clearly as here.

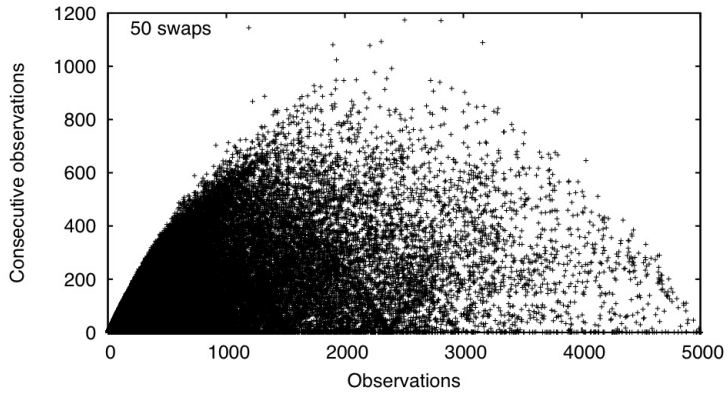
When no route dynamics is simulated ($s = 0$, Figure 2.10(a)), only the parabola is present, thus confirming that this phenomenon observed in real data is due to load-balancing mechanisms which is well captured by the random BFS model. At the opposite, when the number of swaps is too high (Figure 2.10(c)), route dynamics get the better of load-balancing phenomena and the parabola tends to vanish.



(a)



(b)



(c)

Figure 2.10: Observations vs. consecutive observations for $s = 0$ $n = 500,000$, $m = 1,000,000$, $d = 3,000$ in (a), $s = 10$ $n = 500,000$, $m = 1,000,000$, $d = 3,000$ in (b), and $s = 50$ $n = 500,000$, $m = 1,000,000$, $d = 3,000$ in (c)

We have shown in Figure 2.8 that the number of swaps and links have opposite effects. We studied the correlations between observations and consecutive observations for different number of links m. Indeed, when m is large, we observe the same behavior as in Figure 2.10(b). When it decreases, the shape is degraded, as in Figure 2.10(c).

As a conclusion, by exploring the impact of the parameters, we showed that a wide range of their values are relevant. Finally, under simple constraints on the parameter values, the model succeeds in capturing the invariant properties observed in real data.

2.2.6 Conclusion and perspectives

We have conducted periodic measurements of ego-centered views of the Internet topology and studied their dynamics. We identified key characteristics of these dynamics, and showed that load balancing and evolution of the routing topology are key factors in the observed properties. Based on these observations, we proposed a model for the dynamics of the topology, which integrates both load balancing and routing changes. Simulations show that this model captures the main characteristics of the dynamics of the ego-centered views.

Our model is based on simple mechanism for both the topology generation and the characteristics of route dynamics. It is therefore not suitable for generating *realistic* time-evolving topologies. However, the fact that it captures the main characteristics of the observed dynamics shows that the factors it mimics play a strong role in the Internet topology dynamics, which offers key insight on the understanding of these dynamics. We therefore consider this model as an important first step towards the realistic modeling of the Internet topology dynamics, as well as towards its understanding.

Future work lies in two main directions. First, the field of Internet topology modeling is very active, and models far more realistic than ER graphs are available. One should explore the combination of our routing mechanism principles with these topology models, to investigate the role played by topology structure on the observed dynamics. In particular, our model does not take into account the long term topology evolution, since it does not model node birth or death. Coupling the ingredients of our routing dynamics with, e.g., a growing model for the Internet topology which would reflect its long term dynamics would surely lead to insightful results.

Second, since it is based on random graphs and simple mechanics for load balancing and routing dynamics, our model lends itself well to formal analysis. Obtaining analytical results for the empirical observations we made would be an important step towards quantifying formally the role played by the different factors in the Internet topology dynamics.

2.3 Networks events

Automatically detecting events in the dynamics of a complex system like the Internet is crucial for both our understanding of the underlying object and for monitoring purposes. Ideally, such detection leads to alarms with a low rate of false positives (alarms when

there is no significant event) and false negatives (undetected events). In the case of the Internet, this is extremely challenging as no reference information exists: knowledge on the dynamics of the Internet topology is very poor, not to mention knowledge on abnormal events.

To tackle this issue, we propose here to use *radar* measurements of the Internet topology, which consist in periodic measurements of (a part of) the topology from one monitor with a traceroute-like approach. We then define some graph properties which describe the dynamics of the obtained views. We design two methods to automatically detect events using two notions of *statistically significant events*. Such events are *outliers* in the observed values (associated to the property under concern), and we present here methods to define and observe them.

Notice that the technique presented in this section is generically applicable: the graph properties we propose may be computed on any dynamic graph, and the event detection methods we propose may be applied to any time-evolving data. We will present them first and then apply the, in the context of Internet topology dynamics and show that, in this case, they succeed in detecting relevant events.

2.3.1 Objectives and Motivations

Our central objective is to design automatic methods for detecting events in graph dynamics and apply it to the case of Internet topology. More precisely, we want to address two challenges:

- We want to define dynamic graph properties which we are able to observe on real-world networks like the Internet topology and which succeed in exhibiting statistically significant events;
- We want to design and apply automatic methods able to detect outliers in the values taken by these properties.

Such outliers are nothing but statistically significant events, which will be our definition of events. When applying this approach to the Internet topology, we expect such events to be correlated to major breakdowns and congestions, removals of routers and wires, and other events significant from a networking point of view.

The motivations for such work are twofold. First and most obviously, it is crucial to be able to automatically detect events for monitoring purposes: one cannot manually inspect and monitor the network under concern and needs a tool to point out moments in time and places in the network at which events occur. The second motivation is to provide a statistically rigorous way to distinguish what should be considered as *normal* dynamics, in opposition to what should be considered as abnormal events. This should give much insight on the dynamics of the considered networks, and help in the design of appropriate models. For the Internet, both issues are crucial as the system lacks a centralized management authority and local events may influence a large portion of the whole. Also, in this case, modeling issues are crucial for formal approaches and simulations.

To reach our objectives, a natural way is to define time series corresponding to the evolution of graph properties during time, and then observe the distribution of obtained values. Depending on the types of obtained distributions, one may use the considered statistics to detect events or not. Going further, one may use graph properties which do not lead to time series (for instance because they produce several and a variable number of values per time step) and still observe the obtained distributions. Our goal is to define such statistics and ways to study the obtained distributions to obtain automatic event detection in the dynamics of Internet topology.

2.3.2 State of the art and contribution

Network event detection (also called anomaly detection, outlier detection, novelty detection, noise detection, deviation detection or exception mining) has received a large attention in the past [71] but is generally applied to some specific properties of the network under study [17]. Almost nothing has been done to detect events with no prior knowledge of the network, thus allowing a general methodology. However, we distinguish three fundamental approaches to the problem of outlier detection with or without prior knowledge:

- Type 1: Determine the outliers with no prior knowledge of the data. This is essentially a learning approach analogous to unsupervised clustering. The approach processes the data as a static distribution, pinpoints the most remote points, and flags them as potential outliers. Type 1 assumes that errors or faults are separated from the 'normal' data and will thus appear as outliers. The approach is predominantly retrospective and is analogous to a batch processing system. It requires that all data be available before processing and that the data is static. However, once the system possesses a sufficiently large database with good coverage, then it can compare new items with the existing data.

- Type 2: Model both normality and abnormality. This approach is analogous to supervised classification and requires pre-labeled data, tagged as normal or abnormal.

- Type 3: Model only normality or in a very few cases model abnormality. Authors generally name this technique novelty detection or novelty recognition. It is analogous to a semi-supervised recognition or detection task and can be considered semi-supervised as the normal class is taught but the algorithm learns to recognize abnormality [41]. The approach needs pre-classified data but only learns data marked normal. It is suitable for static or dynamic data as it only learns one class which provides the model of normality. It can learn the model incrementally as new data arrives, tuning the model to improve the fit as each new exemplar becomes available. It aims to define a boundary of normality.

As outlier detection encompasses aspects of a broad spectrum of techniques, we also point out these approaches not based on models:

- Information-theoretical analysis - Its detection strategy is to estimate the amount of information carried in a network stream, which is a flow of changes in the network (adding and removing of nodes and links). This approach takes samples from the stream and estimates the information content of these samples based on the entropy [29]. An event is determined if the entropy crosses a threshold.

- Statistical analysis - One study the distribution of values and detect outliers, values that deviate significantly from the others[9, 42]. Statistical tests can be used such as the Grubb's test or Dixon test.

Our contribution is twofold.

First, we propose a set of properties to capture the dynamics of an evolving graph. These properties span a wide range of a network properties, from the simplest like its number of nodes and links, to more subtle ones, based on distances and connectivity. These properties provide a first toolbox for quantifying the dynamics of the network.

Secondly, we provide two automatic methods for the detection of statistically significant events in the dynamics of observed properties.

- The first one consists in observing the distribution of values taken by the considered property. If this distribution is homogeneous (well fitted by a normal distribution) then there is no outlier: all values are close to a *normal* one. If the distribution is heterogeneous (well fitted by a power-law) then there is no outlier: there is no notion of *normal* value, and so no notion of *abnormal* value. If the distribution is homogeneous with outliers (tested by removing extremal values and then fitting to a normal distribution), then these outliers are the statistically significant events we detect. This method and its application to the Internet topology are described in [42]; we present it shortly below.
- The second one relies on the intuition that an event significantly disturbs the statistical equilibrium of the network properties. To estimate this, we propose a statistical test based on distribution skewness to decide whether the considered distribution is balanced (i.e., it is unimodal and symmetrical). This test relies on the iterative removal of extremal values in the distribution, until its skewness reaches certain predefined conditions and remains stable. If the test fails to reject the hypothesis that the distribution is balanced, we consider that the removed values indicate outliers in the distribution dynamics. This work is currently under submission and we detail it below.

Both methods have the advantage that they require no prior knowledge on the dynamics of the considered network and that they detect statistically significant events. To this regard, they fit our needs for detecting events in the dynamics of the Internet topology much better than previous approaches. Moreover, they are very general and may be applied to many other cases.

2.3.3 Methodology and tools/dataset used

Our approach first relies on a set of graph dynamic properties which we have to define. To do so, we proceeded in a simple-to-complex way, going from the most basic graph properties to much more subtle ones. We detail them below. Then we present our two methods for identifying outliers detected by these properties.

2.3.3.1 Dynamic graph properties

We assume that the considered dynamic graphs are described as series of snapshots of the underlying network. Then we may consider each snapshot independently, or consider

the union of several consecutive snapshots, which may lead to more interesting observations in some cases. For each time-step t we therefore consider the snapshot at t and we define the 'previous graph' as the union of the p previous snapshots for a given p and the *current* graph as the union of the c next snapshots for a given c.

The simplest graph property certainly is its number of nodes and links, which in the dynamic case translates to the time-evolution of these values. Starting from these basic statistics, we also considered:

- number of distinct nodes/links in each snapshot or in the current graph. Both may lead to the detection of different events as the union captures changes in the observed nodes/links (even if they are in the same number).
- number of new nodes/links that appear (nodes/links observed in the current graph but not in the previous one). It is an appropriate property to detect abnormal arrivals of previously unseen nodes/links, as one might expect that the number of appearing nodes/links is roughly stable when nothing special occurs.
- number, size and density of connected components in the sub-graph induced by appearing nodes and/or links. One may expect that the number of non connected groups in a graph is roughly stable, so a fast increase or decrease may indicate a significant event.
- distances between two nodes immediately before a link appears between them (a link which was previously not there). One may expect these distances to be small and so a large value may indicate a significant event.
- distances that change from the ends of one appearing link to the rest of the graph. This property helps to detect the situation when a new link reduces the distance from a sub-graph to one end of the new link.
- distances that change from the ends of all appearing links to the rest of the graph. One can expect that the changes of distance is stable in the whole graph, hence a brutal increase or decrease may indicate a significant event.

Most of these properties lead to one value for each time step (like for instance the number of nodes), but some properties lead to several values (like for instance the size of all connected components), thus providing much richer information. One strength of our methods below is to be able to handle both situations with no modification.

2.3.3.2 Homo- vs heterogeneity method

Let us assume that we try to detect events using a given property, like the ones above. This means that we observe one or several numerical values at each time step (for instance, the size of all connected components in the graph). We detect events by studying the distribution of these values, which we call below the *empirical distribution*.

Our first method consists in fitting the empirical distribution with two kinds of model distributions: homogeneous ones (typically normal distribution) and heterogeneous ones (typically power-law ones). In addition, we remove extremal values (possible outliers) and then fit the distribution of remaining values with a homogeneous model distribution.

We then compare the quality of these fits by computing the KS test (Kolmogorov-Smirnov test, i.e. the maximal difference between the empirical distribution and the result of the fit) and the MK distance (Monge-Kantorovitch distance, i.e. the average difference). These two statistics give complementary indications (like a worst case and an average case) on the quality of the fit. We use them to decide whether the empirical distribution is closer to an homogeneous one, an heterogeneous one, or an homogeneous one with outliers.

If the empirical distribution is best fitted by an homogeneous model, then we cannot detect any event: all values are close to a *normal* one and no event occurs.

If the empirical distribution is best fitted by an heterogeneous model, then we cannot detect any event: there is no notion of *normal* value in such distributions, and so no *abnormal* value.

Finally, if the empirical distribution is well fitted by an homogeneous model after a small number of extremal values were removed, then these values may be considered as outliers indicating events: there is indeed a notion of *normal* value (defined by the homogeneous fit) and values abnormally far from this normal one (the extremal values we removed).

2.3.3.3 Symmetry-based method

The method described above has the advantage that it reliably defines events in a statistically significant way. It however relies on automatic fit of empirical distributions, which may be difficult in practice, and needs a large amounts of values to be relevant.

We propose here another approach, based on the following observations. First notice that an homogeneous distribution is symmetric and that removing a few values from it will change its symmetry only a little. On the contrary, an heterogeneous distribution is not symmetric in general and cannot become symmetric after the removal of a few values. Instead, a homogeneous distribution with outliers will be symmetric once these outliers have been removed. It may however already be symmetric *before* these removals, but notice that if we remove the extremal valuex one by one the symmetry will be temporarily broken.

We therefore propose here to study how the symmetry of the considered distributions evolve when we remove extremal values one by one. If the symmetry changes much but we rapidly reach a situation where the remaining values form a symmetric distribution, then we are in a case where removed values were outliers.

There are many ways to quantify the symmetry of a distribution. We use here the notion of *skewness* of the distribution, which is defined as the third centered moment:

$$\frac{1}{Ns^3}\sum_{i=1}^{N}(x_i-\overline{x})^3$$

with N the sample size, s the empirical variance and $\overline{x}$ the empirical mean. The intuitive meaning of the skewness is represented in Figure 2.11.

The core of our test therefore consists in iteratively removing the extremal values in the considered distribution and computing the skewness of the distribution obtained after

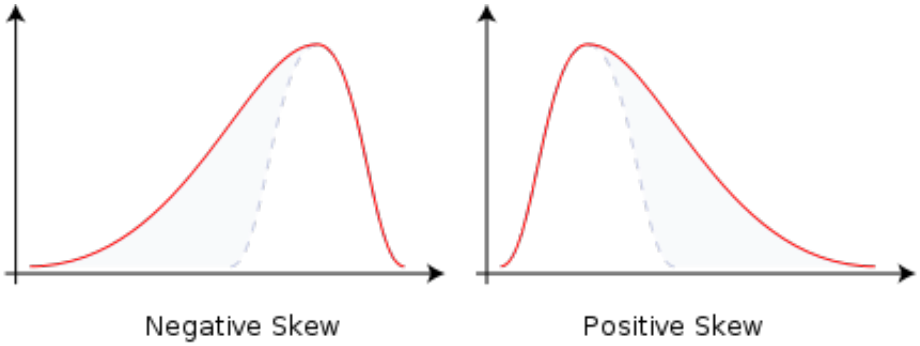


Figure 2.11: Representation of the meaning of the skewness sign

each removal. The obtained series of values of the skewness encodes much information on the distribution. We stop when 50% of all values have been removed (removing more values would have little sense, if any). If the last 10% of the values of the series are not above 0.1 (i.e. removing 40% of all values was not sufficient to ensure that the skewness remains afterwards below 0.1) then we conclude that the considered distribution does not allow event detection (with our method). If we reach this criteria, we compute the largest threshold t between 0 and 100 such that the t% last values of the series are smaller than $t/100$. Outliers are defined as the values we removed before reaching a skewness of $t/100$.

This procedure identifies possible outliers in *one* distribution. In order to extend it to more dynamic situations, we define a sliding time-window and apply the procedure to the set of values in each window.

At this stage, the method still relies on a number of parameters which are poorly understood. Current work focuses on the identification of such key parameters and features of the method in order to improve it and understand it better. First results below however show that it is very promising.

2.3.4 Results

We applied both methods above on a radar measurement of the Internet topology, i.e. iterated traceroute-like measurements from one monitor to a set of targets (see Section 3.1.2 and [53]) for details). Such measurements provide an ego-centered view of the network (similar to routing trees) every 15 minutes approximately and were designed for observing the dynamics of the Internet topology from the monitor. We are currently in the process of applying our methods to another dataset, namely the Internet traffic flow collected by the MAWI project [32], which consists in a stream of IP packets sent over a backbone between Japan and the USA, captured day by day.

Both methods succeeded in detecting appropriate events in the considered dynamics, as illustrated below. The advantage of the symmetry-based method is that it does not rely on automatic fits of empirical distribution, which is a challenge in itself and prone to errors in practice. Still, both methods have their advantages and weaknesses, and provide complementary results: although most events are detected by both methods, some events are detected by one method only.

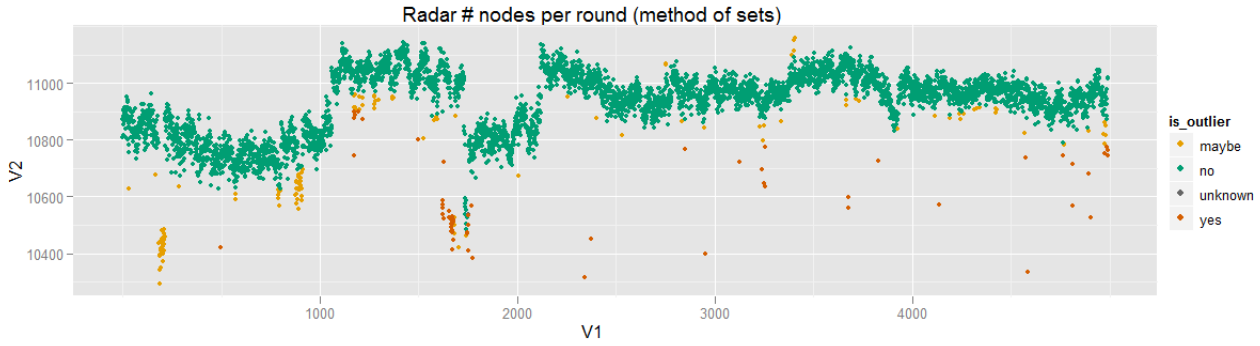


Figure 2.12: Time series of the number of nodes measured in a Radar graph, with values classified. Green points are classified as not outlier, orange points as possible outliers, red points as outliers, and gray points as having an unknown status.

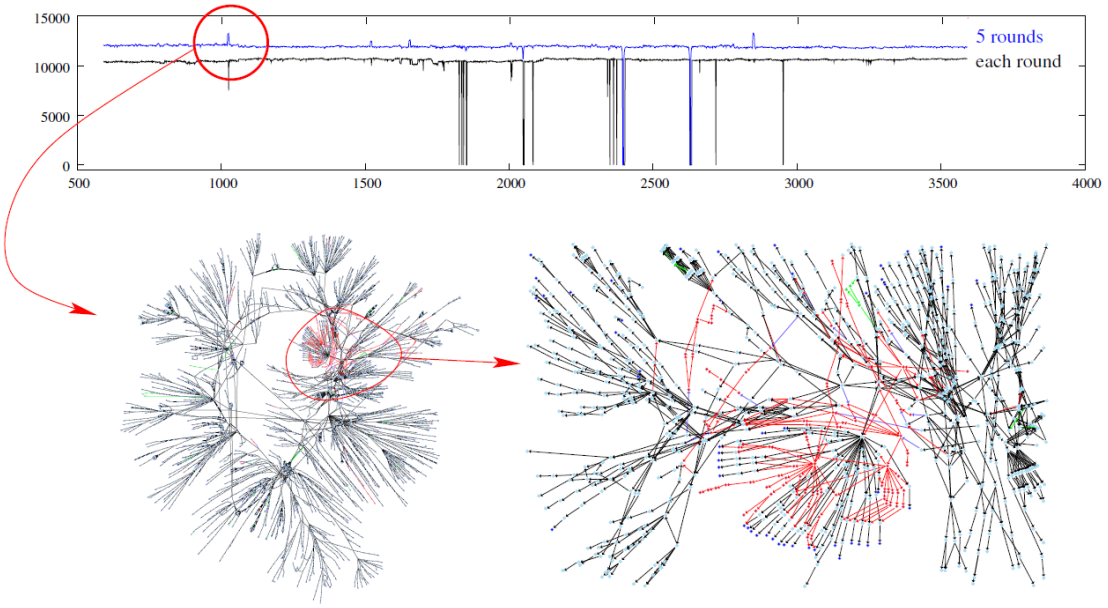


Figure 2.13: Events in a Radar network

The main output of these methods is a criterion to tag some values of observed properties as *abnormal*, thus pointing our *events* in the dynamics, as illustrated in Figure 2.12.

Once events are detected this way, one may manually inspect the graph dynamics *at that time and place* in order to gain more insight on the detected event. In the case represented in Figure 2.13, for instance, we drew the network during the event and displayed in red color the nodes and links which appear during the event but neither before nor after it. The picture clearly shows that these nodes and links are located in a specific region of the network, where the event occurred.

These results are very important in the context of EULER. First, we provide a library of dynamic graph properties which may be used to analyze the dynamics under concern. Moreover, we observe the behavior of these properties in the case of a measurement of the Internet topology, thus providing insight on what are *normal* dynamics and what

kind of *events* occur in them. This is essential for modeling issues as dynamic models of the Internet should be able to capture these behavior, and for protocol design as protocols must be robust to such events. Also, knowing the *normal* dynamics of the Internet topology is important for protocol optimization, which make take advantage of its real-world features.

2.4 Stability of Routing Paths

2.4.1 Motivation and Objectives

The stability of a routing system is characterized by its response (in terms of processing routing information) to inputs of finite amplitude. Inputs may be classified either as internal or external events to the routing system. *Internal events* (to the routing system) refer to routing protocol changes including re-configuration, BGP session failure, BGP route attribute changes for policy/administrative reasons (e.g. communities), changes to filtering policies, etc. whereas *external events* (to the routing system) to routing states changes (new routes appearing, changes to existing routes, etc.) resulting from topological changes. Both types of events result in routing information updates and that BGP does not differentiate its updates with respect to their root cause, their identification (origin), their scope, etc. BGP selection process behaves homogeneously. So, although current routing engines could potentially support up to $O(1M)$ routing table entries resulting instabilities may adversely affect the network's ability to remain in a usable state for extended periods of time.

BGP's instabilities can be categorized into policy-induced instabilities and protocol-induced instabilities. This classification accounts for the distinction between operation of the protocol (mainly resulting from its associated policies) and the inherent behavior of protocol (mainly resulting from the path-vector routing algorithm):

- *Policy-induced instabilities*: addressing routing stability consistently with planned BGP routing policy implies eliminating non-deterministic routing states resulting from policy interactions and in particular, non-deterministic and unintended but unstable states. Griffin et al.'s seminal work [36] modeled BGP as a distributed algorithm for solving the Stable Paths Problem, and derived a general sufficient condition for BGP stability, known as "No Dispute Wheel". This sufficient condition guarantees the existence of a stable solution to which BGP always converges. Informally, this sufficient condition allows nodes to have more expressive and realistic preferences than always preferring shorter routes to longer ones. The game theoretic approach introduced by Levin and Shapira [54] relies on the best-reply BGP dynamics: a convergence game model in which each Autonomous System (AS) is instructed to continuously execute the following actions: i) receive update messages from BGP peering nodes announcing their routes to the destination, ii) choose a single peering node whose route is most preferred to send traffic to, iii) announce the new route to peering nodes. However, as proved in the seminal work of Levin and Shapira [54], best-reply BGP dynamics is not incentive-compatible even if No Dispute Wheel condition holds: even if all but one AS are following the BGP rules, the remaining AS may not have the incentive to follow them. Interestingly, as demonstrated by the same authors, incentive compatibility of best-reply BGP dynamics requires combining an additional global condition (Route Verifica-

tion) together with the "No Dispute Wheels" to guarantee stability. As of today's understanding and current modeling of BGP routing system behavior, known conditions for global stability are sufficient but not necessary conditions. Moreover, checking them is an NP-hard problem and enforcing them requires a global deployment of an additional mechanism (route verification being one of them). On the other hand, local instability effects have yet to be characterized.

- *Protocol-induced instabilities*: BGP is an inter-AS path-vector routing protocol subject to Path Exploration phenomenon like any other path-vector algorithm: BGP routers may announce as valid, routes that are affected by a topological change and that will be withdrawn shortly after subsequent routing updates. This phenomenon is the main reasons for the large number of routing updates received by BGP routers which exacerbate both inter-domain routing system instability and processing overhead [44]. Both effects result in delaying BGP convergence time upon topology change/failure as observed in [49]. Several mitigation mechanisms exist to partially limit the effects of path exploration; however, none actually eliminate its effects. Hence, BGP is intrinsically subject to instability.

Henceforth, the **objectives** for the investigation of path-vector routing stability are to:

1. Develop a methodology to process and interpret the data part of BGP routing information bases in order to identify and characterize occurrences of Internet BGP routing system instability; such characterization can be used as a comparison point for the stability of the newly developed schemes (candidate to replace BGP) and characterize instability phenomena any routing system would have to cope with;

2. Determine a set of stability metrics and develop methods for using them in order to provide a better understanding of the BGP routing system's stability;

3. Investigate how path-vector routing protocol behavior and network dynamics mutually influence each other. The proposed approach aims to bring rigor and consistency to the study of routing stability. For example, it would allow for a unified approach to the cross-validation of techniques for looking at improving path exploration effects on the routing system.

2.4.2 State of the art

As outline in the Introduction Section, understanding the dynamics of the Internet routing system is fundamental to ensuring its stability and improving the mechanisms of the BGP routing protocol [RFC4271]. Numerous research work dedicated to understand this behavior have been conducted over last 20 years (i.e., once BGPv4 became the de-facto inter-domain routing protocol of the Internet).

To the knowledge of the authors BGP stability studies have already been conducted experimentally (empirically) by means of ad-hoc analysis of BGP data traces. However, these studies have been conducted without prior specification of a stability analytical method including the use of stability measurement metrics for systematic analysis of BGP traces and associated meta-processing for determining the actual state of the local routing table and its individual entries.

Note that BGP stability has also been reported in [RFC4984], outcome of the Routing and Addressing Workshop held by the Internet Architecture Board (IAB). In this context, it is important to underline that the dynamics of the Internet routing system determines the routing engine resource consumption, in particular, memory and CPU. System resource consumption depends on the size of the routing space but also on the number of adjacency and peering relationships between routers. Indeed, the greater the dynamics associated with the routing information updates exchanged between all these adjacencies and peerings, the higher the memory and CPU requirements for the operation of the routing protocol.

2.4.3 Methodology

2.4.3.1 Preliminaries

The AS topology of the routing system is described as a graph $G = (V, E)$, where the vertices (nodes) set V, $|V| = n$, represents the AS, and the edges set E, $|E| = m$, represents the links between AS. At each node $u \in V$, a route r per destination d ($d \in D$) is selected and stored as an entry in the local routing table (RT) whose total number of entries is denoted by N, i.e., $|RT| = N$. At node u, a route r_i to destination d (i.e., denoting the reachable destination network) at time t is defined by $r_i(t) = \{d, (v_k = u, v_{k-1}, .., v_0 = v), A\}$ with $k > 0 | \forall j, k \geq j > 0, (v_j, v_{j-1}) \in E$ and $i \in [1, N]$, where $(v_k = u, v_{k-1}, .., v_0 = v)$ represents the AS-path from node u to node v where v is the attachment point of the network prefix denoted by destination d, v_{j-1} the next hop of v_j along the AS-path from node u to node v, and A the attribute set of the AS-path. By attribute, we refer here to all attributes associated to the AS-path, e.g., Origin, Next-Hop, Multi_Exit_Discriminator, and Local_Preference (definition and usage of these attributes can be found in [RFC 4271]).

Let $P_{(u,v),d}$ denote the set of paths from node u to v towards destination d where each path $p(u, v)$ is of the form $\{(v_k = u, v_{k-1}, .., v_0 = v), A\}$. A routing update leads to a change of the AS-path $(v_k, v_{k-1}, .., v_0)$ or an element of its attribute set A. A withdrawal is denoted by an empty AS-path ϵ and $A = \emptyset$;, $\{d, \epsilon, \emptyset\}$. If there is more than one AS-path per destination d, according to the definition above, they will be considered as multiple distinct routes.

2.4.3.2 Routing Table Stability

The stability of a routing system is characterized by its response (in terms of processing of routing information) to inputs of finite amplitude. Routing system inputs may be classified as i) internal system events such as routing protocol configuration change or ii) external events such as those resulting from topological changes. Both types of events lead to the exchange of routing updates that may result in routing states changes. Indeed, BGP does not differentiate routing updates with respect to their root cause, their identification (origin), etc. during its selection process.

Definition 1: Let $RT(t)$ represent the routing table at some time t. At time $t + 1$, $RT(t+1) = RT_0(t)(+)\Delta RT(t+1)$ where, $RT_0(t)$ is the set of routes that experience no change between t and $t+1$, and $\Delta RT(t+1)$ accounts for all route variations (additions, deletions, and changes to previously existing routes) between t and $t + 1$.

The magnitude of the output of a stable routing system should be small whenever the input is small. That is, a single routing information update shall not result in output amplification. Equivalently, a stable system's output will always decrease to zero whenever the input events stop. A routing system, which remains in an unending condition of transition from one state to another when disturbed by an external or internal event, is considered to be unstable. Provide means for measuring the magnitude of the output is the main purpose of the metric referred to as "stability of the selected route".

For this purpose, we define the criteria for qualifying the effect of a perturbation on the local routing table so as to locally characterize the stability of the routing system. More precisely, let $|\Delta RT(t+1)|$ be the magnitude of the change to the routing table (RT) between time $t = t_0 + k$ to $t+1 = t_0 + (k+1)$, where t_0 is the starting time of the measurement sequence. We distinguish three different equilibrium states for the routing table:

Definition 2: when disturbed by an external and/or internal event, a routing table is considered to be stable if the following condition is met: $|\Delta RT(t+1)| \leq \alpha$, $t \to \infty$, where $\alpha > 0$ is small. That is, the magnitude of change of the routing table in response to a perturbation remains over time below a certain small threshold. In these conditions, if the routing system locally returns to its initial equilibrium state, it is considered to be (asymptotically) stable.

Definition 3: when disturbed by an external and/or internal event, a routing table is considered to be marginally stable if the following condition is met: $\alpha < |\Delta RT(t+1)| \leq \beta$, $t \to \infty$, where $\beta > 0$ is small, and $\alpha < \beta$. In these conditions, if the routing table locally transitions to a new equilibrium state, the routing system is considered to be marginally stable.

Definition 4: when disturbed by an external or internal event, a routing table is considered to be unstable if the following condition is met: $|\Delta RT(t+1)| > \beta$, $t \to \infty$. That is, the magnitude of change of the routing table in response to a perturbation never decreases below a certain threshold. In these conditions, the routing system locally remains in an unending condition of transition from one state to another and is considered to be locally unstable.

The values of parameters α and β shall be set based on operational criteria. Among other things, α and β depend on the observation sampling period that needs to be the MRAI time. This ensures at most one routing update per sampling period (that is determined by the MRAI time interval). A similar reasoning to the one applied for the Loc_RIB stability (that corresponds to the BGP routing table) can be applied to the Adj_RIB_In (which stores incoming routes from neighbors). It is also interesting to measure the instability induced by the BGP selection process.

2.4.3.3 Metric Set

To measure the degree of stability of the Loc_RIB, Adj_RIB_In, and determine how close the system is to being unstable the following stability metrics are defined:

1. Stability of selected routes r_i

```
When route r_i is created: φ_i(t) ← 0
if r_i experiences a path or an attribute change
(r_i(t+1) ≠ r_i(t)) then φ_i(t+1) ← φ_i(t) + 1
else /* r_i experiences no changes */
     if φ_i(t) = 0 then φ_i(t+1) ← 0
     else if φ_i(t) > 0 then φ_i(t+1) ← φ_i(t) - 1
          end if
     end if
end if
```

Figure 2.14: Route and RT Stability

The stability $\phi_i(t)$ of the selected routes $r_i(t)$ characterizes the stability of the routes stored at time t in the $Loc\_RIB(|Loc_RIB| = N)$. This metric quantifies the magnitude of change for these routes between time t_0+k to $t_0+(k+1)$, where t_0 is the starting time of the measurement sequence (time units are counted by default in terms of minimum routing advertisement interval (MRAI)), and the integer k accounts for the number of MRAI times that have elapsed since the starting time of the measure sequence. The latter determines the minimum amount of time that must elapse between a routing advertisement of route to a particular destination by a BGP peer. This metric quantifies thus the magnitude of change to route r_i, and a routing table with a periodicity determined by the $MRAI$ time.

Note that this metrics is equivalent to the metric defined in [60] also referred to as Loc_RIB Stability Metric. This metric can be directly computed by using the algorithm described in Figure 2.14.

The computation of the stability metric for an entire routing table (RT) can then be derived form the stability of its individual routes. Let $|\Delta r_i(t+1)|$ denote the change in stability metric associated with a single route r_i from time t and $t+1$. These values are used to compute $|\Delta RT(t+1)|$ defined as the change in stability metric for the entire routing table from time t and $t+1$. Moreover, $|\Delta RT(t+1)|$ is normalized so that $0 \leq |\Delta RT(t+1)| \leq 1$, where 0 implies perfect stability, and 1 indicates complete instability.

2. Metric to determine the Most Stable Route in the Adj_RIB_In

The *Most stable route in the Adj_RIB_In* ($|Adj\_RIB\_In| = M$) quantifies the relative stability between the routes to the same destination d, learned from different upstream BGP peers (i.e., downstream from the point of view of the AS-path towards destination d).

For this purpose, let $W_u \subset V$ denote the set of BGP peers of node u, $|W_u| = W \leq M$, and w one of its elements such that $(u, w) \in E$. Let $\phi_{i,j}(t)$ denote the stability of the route $r_i(t)$ towards destination d, as received by peering router j such that index $j \in [1, W]$. At node u, $r'_{i,stable}(t) = min\{\phi_{i,j}(t), \forall j \in [1, W] | \{(v_k = u, v_{k-1} = w, .., v_0 = v), A\} \in P_{(u,v);d}, \forall w \in W_u\}$ defines -independently of the BGP selection rules- the selectable route that is the most stable for destination d at time t.

In Figure 2.15, we define $\Delta\phi_i$ as the relative measure of route's r_i stability with respect to the most stable route for the same destination d, $\phi_{i,stable}$.

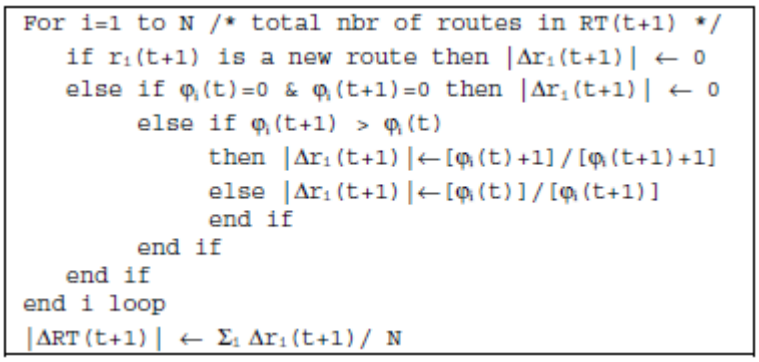


Figure 2.15: Most Stable Route in Adj_RIB_In

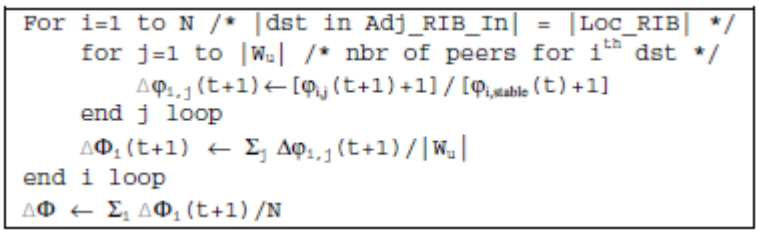


Figure 2.16: Best Selectable Route in Adj_RIB_In

3. Metric to determine the Best selectable route in the Adj_RIB_In

The *Best selectable route in the Adj_RIB_In*: quantifies the relative stability between routes to the same destination d as learned from all upstream peers and the one amongst them selected by BGP at time t as the best route (thus following the BGP route selection). The computational procedure, represented in Figure 2.16, is the same as the previous one if one replaces $\phi_{i,stable}$ by $\phi_{i,selected}$.

4. Differential Stability

Differential stability between the most stable route in the Adj_RIB_In and the selected route stored in the Loc_RIB for the same destination d: characterizes the stability of the currently selected routes for a given destination d against most stable routes as learned from upstream neighbors. This metric provides a measure of the stability of the learned routes compared to the stability of the currently selected route. A variant of this metric, denoted $\delta\phi_i(i \in [1, |D|])$, characterizes the stability of the newly selected path $p^*(u, v)$ at time t for destination d against the stability of the path $p(u, v)$ that is used at time t (i.e., stored in the Loc_RIB) for destination d and that would be replaced at time $t + 1$ by the path $p * (u, v) : \delta\phi_i(t) = \phi_i(t) - \phi_i^*(t)$. In turn, if $\delta\phi_i(t) > 0$, then the replacement of $r_i(t)$ by $r_i^*(t)$ increases stability of the route to destination d; otherwise, the safest decision is to keep the currently selected route $r_i(t)$ stored in the Loc_RIB.

Application of the metric $\delta\phi_i$ during the BGP selection process would prevent replacement of more stable routes by less stable ones but also enable selection of more stable routes than the currently selected routes. However, for this assumption to hold we must prove the consistency of the stability-based selection with the preferential-based selection model that relies on path ranking function. For each $u \in V$, there is a non-

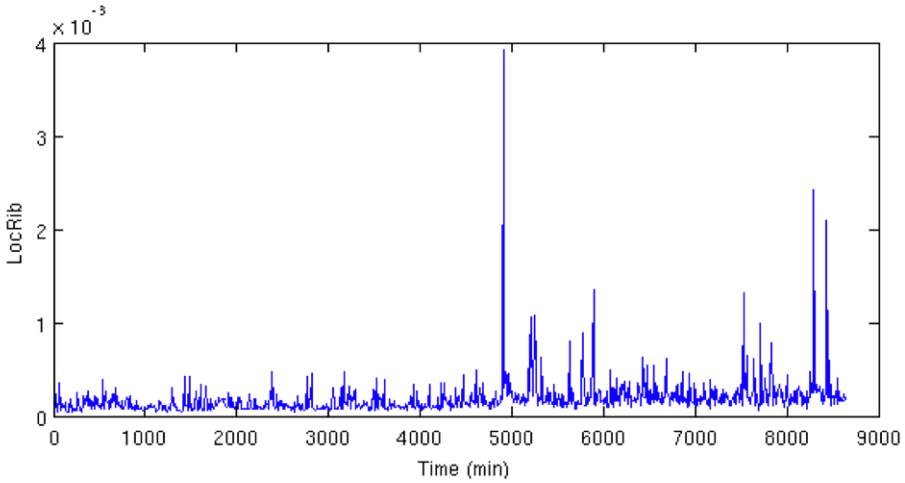


Figure 2.17: Stability of selected routes

negative, integer-value ranking function λ_u, defined over $P_{(u,v);d}$, which represents how each node u ranks its paths: if $p_1(u,v)$ and $p_2(u,v) \in P_{(u,v);d}$ and $\lambda_u(p_1) < \lambda_u(p_2)$ then path p_2 is said to be preferred over path p_1.

Definition 5: The route selection problem is consistent with the differential stability function $\delta\phi(t)$ if for each $u \in V$ and $p_1(u,v)$ and $p_2(u,v) \in P_{(u,v);d}$ (1) if $\lambda_u(p_1) < \lambda_u(p_2)$ then $\delta\phi(t) = \phi_1(t) - \phi_2(t) \geq 0$ and (2) if $\lambda_u(p_1) = \lambda_u(p_2)$ then $\delta\phi(t) = \phi_1(t) - \phi_2(t) = 0$.

2.4.4 Experimental Results

This section presents a set of experimental results obtained by applying the metrics defined in Section 2 to real-world BGP data. The dataset we used was obtained from the Route Views project [60] that comprises archives containing BGP feeds from a set of worldwide distributed Linux PCs running Zebra. As the only policy applied by Route Views sets the next hop to the peer IP address, only Adj_RIBs_In is accessible. As a consequence, the Loc_RIB was inferred from the Adj_RIBs_In by implementing a selection process used by Zebra routers.

2.4.4.1 Stability of Selected Routes

Considering that the number of selected routes is around 318k, Figure 2.17 shows that on average the Loc_RIB contains a few, between 60 and 120, unstable routes with minor contribution to the metric which can be interpreted as a sign of routing table stability. During the third day (5k minutes), 2 spikes separated by around two hours indicate large changes in stability. The first spike seems to suggest that 21 times more routes than the average experienced instabilities. However, BGP quickly converges to a new state that is disturbed by the second (smaller) spike since part of the affected routes return to the state before the occurrence of the first event. Overall, the plot shows a constantly changing but bounded churn within the table.

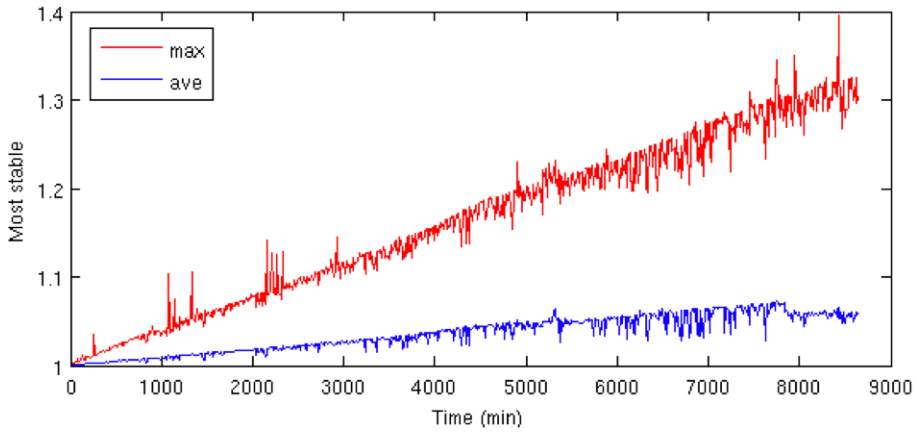


Figure 2.18: Stability of most stable route

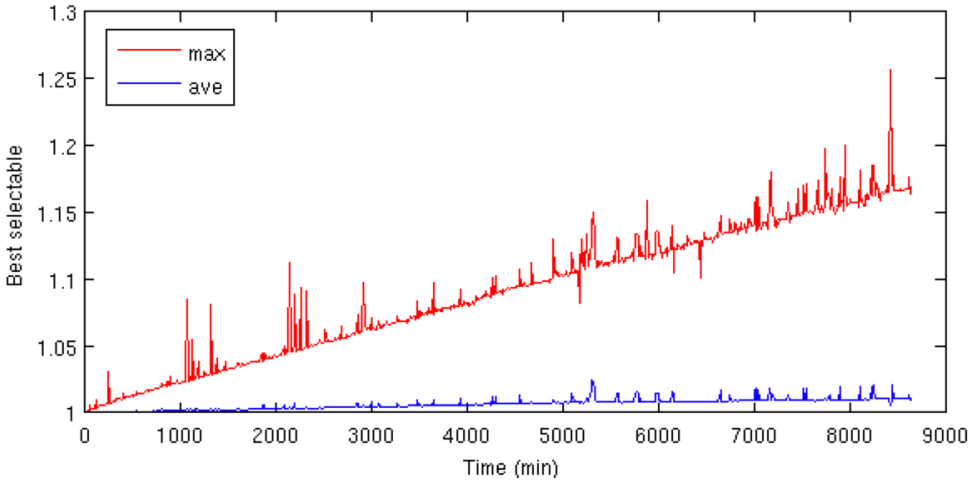


Figure 2.19: Stability of best selectable route

2.4.4.2 Most Stable Route

Figure 2.18 shows that on average the routes have slowly decreasing stability when compared to the most stable route. As a result, the plot has a small but positive slope. It does sometime present local minima which mark the points in time when the most stable route experiences changes. The average of the maximum metric value per destination d shows a positive but larger slope: the most unstable routes have a faster paced increasing instability. Further, during the entire observation duration (6 days), a subset of routes continuously presented instabilities leading to a monotonic increase of the metric.

2.4.4.3 Best Selectable Route

It can be seen from Figure 2.19 that the BGP selected route has, on average, a better stability than the other routes out of which it is selected. However, comparison between Fig.2 and Fig.3 reveals that the selected route exhibits slightly more changes than the most stable route (a lower metric value indicates a higher instability).

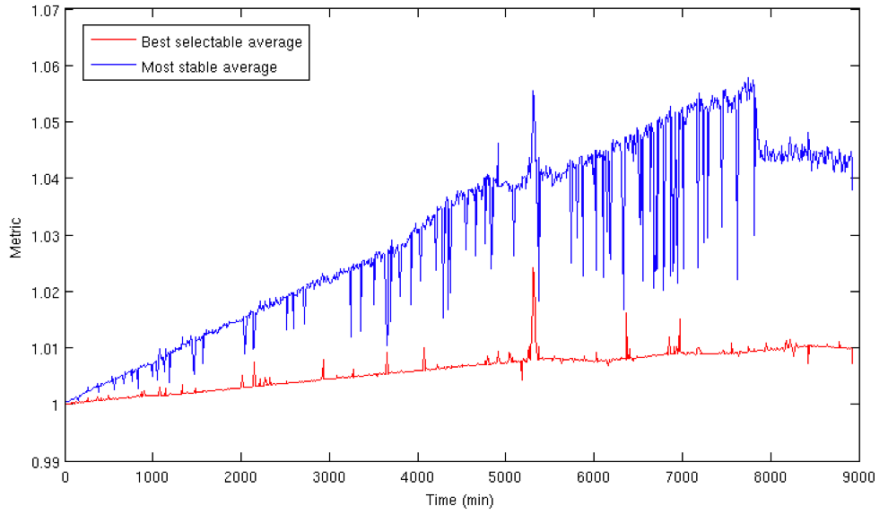


Figure 2.20: Avg curve - correlation between local maxima (of best selectable) and local minima (of most stable)

Additionally, for the *avg* curve (see Figure 2.20), the local maxima are correlated with the local minima of the previous metric and are likewise due to a diminishing stability of the most stable route. One can also observe the same monotonously increasing trend of the metric for both the average and the maximum, due to routes with sustained instability.

2.4.5 Conclusions

By means of the proposed metrics, detection and identification of routing paths subject to instability event can be achieved. We have also determined a stability decision criteria that can be taken into account as part of the BGP routing path selection process. Further, correlation between routing and forwarding path stability becomes now possible by correlating events.

Ongoing work includes verifying if the route selection problem is consistent with the stability function $\delta\phi$, and determining the general trade-offs between stability-based route selection and the resulting stretch increase/decrease factor on the selected routing paths. Moreover, the relationship between local and global stability will be further elaborated so as to provide analytical model enabling to determine the effects of selecting of a route that is more stable locally onto the global stability of the routing system. The idea here is to determine the necessary but sufficient conditions for preventing oscillations (the act of selecting a more stable route locally induces a perturbation leading to unwanted instabilities).

2.5 Internet Traffic

In order to develop a true measurement-based modeling of the Internet, it is crucial to not only conduct such work for the infrastructure, but also for what occurs over it, i.e. what it is used for. We investigate here how files are exchanged between users in a large P2P network, based on large-scale real-world measurements. Indeed, such

traffic represents a significant part of all Internet traffic [3, 33, 40], and thus modeling it in an appropriate way would improve significantly current state-of-the-art of Internet modeling, in particular regarding modeling the traffic on the physical network of the Internet.

Our key objective is to develop the first measurement-based modeling of such phenomena. To do so, we need appropriate data capturing exchanges in a large population of P2P users, and we need to confront these exchanges to currently common assumptions made in most models. We therefore aim at developing methods to observe key parameters of such models on real-world data obtained from measurement. Our goal is to gain significant insight on which models are most relevant, and which parameters should be used for them.

Specifically, we propose an analytical framework in order to model the popularity distribution of the files and activity of the peer nodes in the system. In addition, we are interested in modeling the request pattern of an individual file throughout the observation period. We use the dataset/trace observed in the e-Donkey server during two days of time to validate our model. Our priority will of course be to implement the model and then to compare the obtained simulated data with the real ones on a number of properties that were observed on the latter. Depending on the diffusion of files, we also study whether influent communities exist among the peer nodes in the network. Here the apriori assumption is that, the bunch of nodes interested (requesting/providing) for the similar file belongs to the same community. This may enable us to test how predictions can be made to assess the behavior of peers which may help us to design a recommender system. In a similar way, we study the diffusion of files in the network as a rising and then dying process: therefore, we could for instance study how the popularity of a given file increases and eventually falls. Finally, we can follow the diffusion of files and then systematically investigate their spreading in the P2P network. The objective of this investigation is to identify the specific events which trigger the spreading of files among the peer nodes in the network. Eventually, we will obtain a measurement-based modeling of exchanges in a large P2P system. The next step will be to design a traffic model for the physical network of Internet, including burstiness, rate, etc., from the model of traffic and diffusion of files on the overlay P2P network (see Fig 2.21).

2.5.1 State of the art and contribution

Diffusion phenomena in complex networks – such as the spread of virus on a contact network, gossip on social networks and files in peer-to-peer (P2P) networks – have spawned an increasing interest in recent years. Among other reasons, the boost of computer networks and online social network platforms offers data and new insights on these phenomena in large scale networks; the possibility to validate and refine current models might lead to breakthroughs in the field. Even though large scale diffusion phenomena have always known considerable interest, it has been consistently challenging to obtain large-scale and reliable real-world data at this level [21].

Indeed, a comprehensive diffusion trace consists of the information on the network on which the process takes place and the chronological information of who transmitted the target – e.g., an infection – to whom. In large epidemic bursts the data from health centers often provide the evolution of the number of infected individuals, but rarely uncover the local trail of the epidemic – i.e., how (by whom) a given individual got infected. Conversely, other empirical studies feature transmission events, but no a priori knowledge of the network on which the diffusion takes place [7, 45].

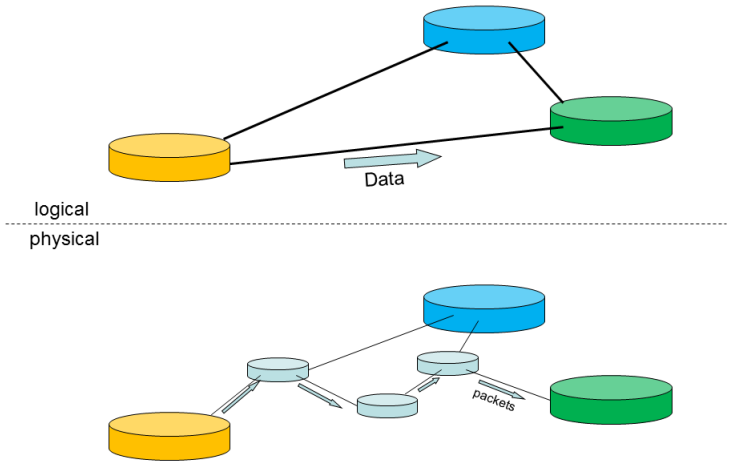


Figure 2.21: Illustration of a physical and a logical network

Despite this historical challenge of obtaining such empirical data, diffusion models blossomed. Many models have been proposed to study such phenomena, mainly coming from the epidemiology community. In the SI model for instance, each node is susceptible (S) or infected (I) and at each time-step an infected node turns a number (or a fraction) of its neighbors into the infected state. The standard SIR model [46] also includes the case of infected nodes recovering (R) and thus being unable to transmit the infection. Most of current models are more complex variations of these two basic ones (see [11] for a survey).

[43] investigated the diffusion behavior of a given file, shared by the multiple source in an eDonkey-like environment. This article proposed simple analytical expressions showing the relationship between the propagation of the file, i.e. the number of peers sharing the file, and the main influencing factors: e.g. sharing probability, access probability, or arrival rate. The article concluded that the popularity of a specific file depends primarily on the age of the information the file contains. New files are of special interest and this is reflected in the arrivals of requests for this file, whereas older files have a much less popularity and thus, the request rates for these files show a reduced burstiness. However, this simplified model fails for the files of very large sizes. In [33], Ge et al. introduced a flexible mathematical abstraction of a peer-to-peer file sharing system that is general enough to capture the essence of different architectures and different peer behaviors. They presented an approximate solution method for the system throughput which can be used to analyze the system performance. By defining specific model parameters, they focused on three different architectures: centralized indexing, distributed indexing with flooded queries, distributed indexing with routed queries) and two classes of peers (non-freeloaders and freeloaders). Their concluded that p2p file sharing systems can tolerate a significant number of freeloaders without suffering much performance degradation. In many cases, freeloaders can benefit from the available spare capacity of peer-peer systems and increase overall system throughput. [40] analyzed a 200-day trace (over 20 terabytes of traffic during May 2002 and December 2002) of Kazaa p2p file sharing traffic collected at the University of Washington in order to explore the nature

of the file sharing workload. They proposed a model based on the dataset to explain the root causes behind many of the trends shown in KaZaA and to predict how trends may change as the workload evolves. The results demonstrated that the "fetch-at-most-once" behavior of clients causes the aggregate popularity distribution of objects in Kazaa to deviate substantially from Zifs law which we typically see for the Web network. [12] presented a scalable p2p network model, particularly focused on the popularity of the files. In p2p network modeling, the query generation process is considered as the one of the most significant elements. Its correct representation strongly depends on the precise description of file popularity dynamics, i.e. basically the evolution of the user interest with respect to different files. Measurements performed by Reza et al. [75] report a particular and interesting behaviour of the file popularity, which has many similarities with the product life cycle (PLC) behaviour reported in marketing literature [14]. For this reason, the product life cycle in marketing environment has been used to model the level of interest for the files. The number of queries performed by every peer and the peer uptime are considered in order to validate the model, comparing simulation output with the measurement results in Gnutella network. The properties of such a model are studied considering the resulting file query distribution, by running multiple independent replications for every simulation. The model can represent huge overlay networks, up to 500000 peers.

Overall, there are two main approaches that describe the dynamics of the process: the spreading point of view and the adoption point of view. In the former category many widespread models evolved from the SIR model [11], which in its minimal setting is characterized by the probability for agents to transmit to its neighbors some information (or file, infection, etc). On the other hand, adoption models rely on a threshold: as soon as the number of infected neighbors of a node becomes larger than this threshold the node becomes infected itself [21].

Here we confront these assumptions to a real-world large-scale spreading phenomena, namely diffusion of files in a large P2P network. By doing so, we assess the two large classes of models cited above (spreading and adoption) and we compute which parameters should be used with them. In particular, we show that the spreading ratio and the adoption threshold have heterogeneous values, while current models generally assume that they are homogeneous. We eventually obtain measurement-based models of file diffusion in P2P systems by using the observed values for these parameters as entry of models (for example specified user arrival rate, user login-logout distribution, network size, file popularity, etc.).

This model is useful for internet simulations as it captures a significant amount of traffic over the internet. It becomes helpful to understand the different activities of the users (such as community formation) centered to the different shared files in the network and eventually simulate P2P traffic demand in a measurement-based approach, much more accurately than previous methods.

2.5.2 Dataset and framework

The first step in this study is to obtain a comprehensive dataset of diffusions in a large scale real world P2P network. We consider here one of the largest running P2P networks, eDonkey, which has the advantage (for this study) of being semi-centralized: it relies on a set of servers which index available files and their providers (but these servers do not store the files themselves) and answer to queries of users (peers). A peer interested in a file F sends a query for this file to one or several server(s) and receive in response a set

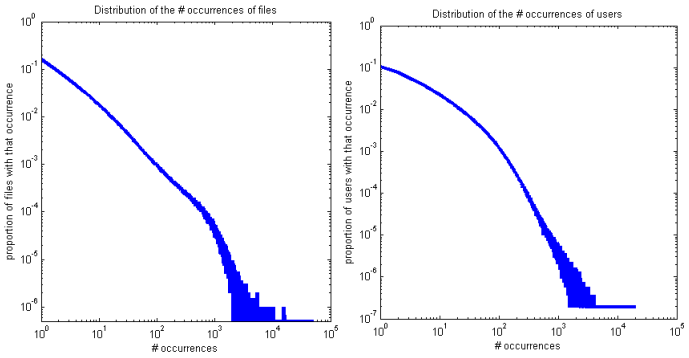


Figure 2.22: Left : the distribution of the popularity of files. It seems to follow a power law : there are many unpopular files but few very popular ones. Right : the distribution of the activity of users.

of possible providers for this file (other peers). The file exchange then occurs between peers, with no longer involvement of the server. See [8] for details.

We use here a trace of answers to file requests managed by one of the largest e-Donkey servers, obtained from a measurement of two days of activity [8]. This dataset consists in a series of 212086691 answers to user request, each indicating to the user a set of providers for a file he/she seeks.

More precisely, this dataset $\mathcal{D}$ is a collection of lines of the following type (capital letters represent unique integer ids):

$$t, C, F, P_1, P_2, P_3, ...$$

where each line represents a request made at time t of the file F by the peer C, potentially satisfied by the peers $P_1, P_2, ..., P_k$ — that is, $P_1, P_2, ..., P_k$ will potentially provide the file F to C at time t. Let $\mathcal{P}$ be the set of all peers (the union of $C, P_1, ..., P_k$ of all the lines) and $\mathcal{F}$ the set of all files (the union of F of all the lines). In this dataset we have $|\mathcal{P}| = 5380616$ peers, $|\mathcal{F}| = 1986588$ files and 212086691 requests (lines in $\mathcal{D}$).

2.5.3 Measured statistics

The measured statistics is based on the dataset $\mathcal{D}$. We measure the popularity of the files based on the request they receive from different users. Similarly, we measure the activity of the users based on the requests made by the users for the files. Finally, we focus on some particular file and observe the cumulative requests they receive from the different clients over time.

Formally, we define the popularity x_F of a file F as the number of times it appears in the whole data set. We define the activity y_P of a client peer P as the number of requests that he made during the two days of observation. Let $Prob_F$ be the popularity distribution of files, such that $Prob_F(x)$ is the probability for a file F to have a popularity $x_F = x$. Similarly, let $Prob_P$ be the activity distribution of clients, such that $Prob_P(y)$ is the probability for a client P to have an activity $y_P = y$. The actual distribution of the popularity of files and of the activity of users can be observed in Fig. 2.22. We find that the frequency of requests for an observed file is correlated with the number of available providers. While focusing on a specific file and observing their cumulative requests over time, we obtain curves similar to a step function relative to file F by normalized

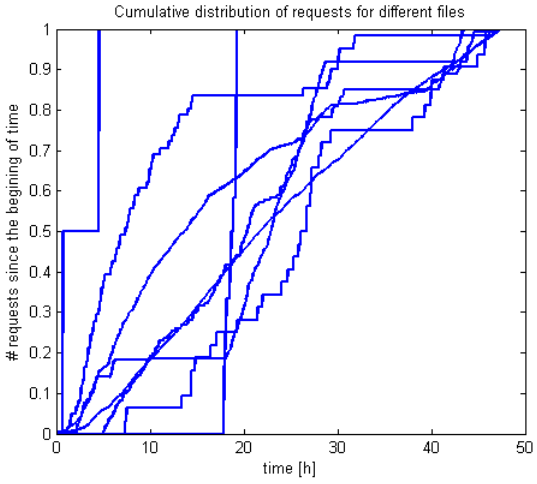


Figure 2.23: Cumulative number of requests over the whole time interval, normalized to one. Each curve corresponds to a different file

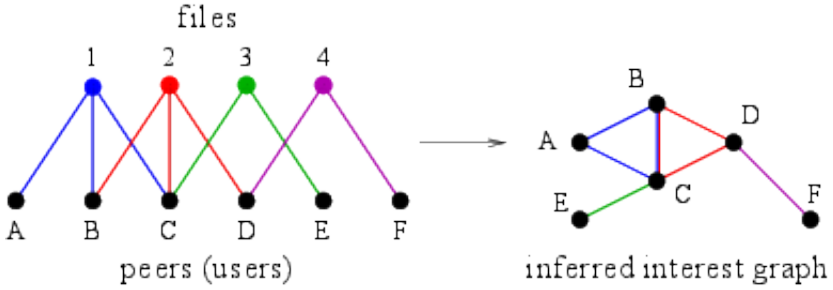


Figure 2.24: Interest graph as bipartite projection

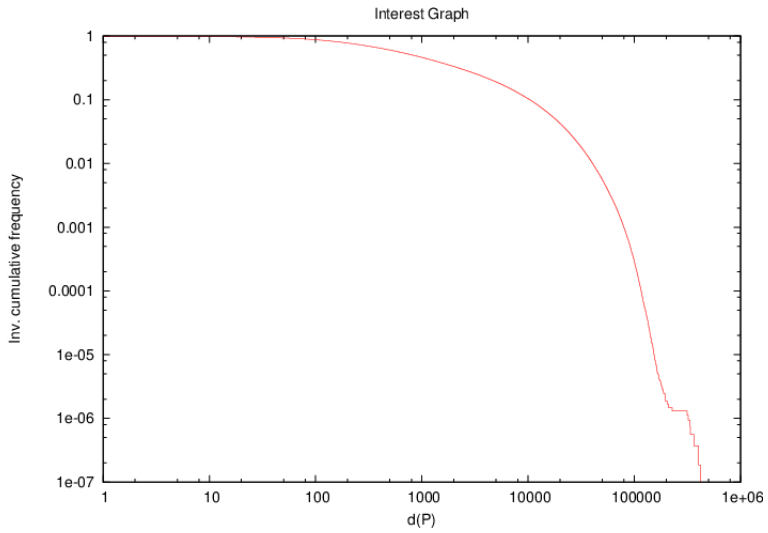


Figure 2.25: Inverse cumulative frequency plot of node degree

cumulative requests $r_F(t)$ (Fig. 2.23). Interestingly, for some curves, we observe plateaus that correspond to time intervals where no providers are present in the network. We also observe that some files are more requested than others.

2.5.3.1 Interest network between peers

One may intuitively assume that users of a P2P network, namely the peers, have some interests in common and thus are roughly structured by an interest graph where the nodes are the peers and two peers are linked together if they have some interest in common. For instance, two peers interested in Linux distributions are linked together, and two peers interested in Madonna song are linked together too, but they are not necessarily linked to one another.

We consider here that the diffusion process of files takes place on the interest graph: users exchange files with users with whom they have interests in common. Of course, knowing precisely the interest graph for a large set of users is in general out of reach. Nonetheless, it is possible to approximate this graph using the data in $\mathcal{D}$: two peers share an interest (and are thus connected) if each one has requested or provided a common file at least once. More precisely, a bipartite graph with the peers $\mathcal{P}$ on bottom, the files $\mathcal{F}$ on top is constructed (see Fig. 2.24) from the request trace in the following way: for each line in $\mathcal{D}$ the peers $C, P_1, ..., P_k$ are connected to the file F. The projection of the bipartite graph on the set of peers is precisely this inferred graph: in the projection the peers who belong to the neighborhood of a common file in the bipartite graph are connected. By sake of simplicity this inferred interest graph (the projection) will be called henceforth interest graph (Fig. 2.24). Notice that if a peer P provides a file F to another peer P' then there is link between them in the interest graph, since both are interested in the same content, namely F. Therefore, the diffusion of files among peers takes place on the interest graph and occurs from neighbor to neighbor: this fundamental property grounds our study, as it implies that the diffusion phenomenon under concern takes place over the network we have.

2.5.3.2 Topology of the interest graph

The interest graph constructed from the P2P trace $\mathcal{D}$ has a giant component that spans all peers but 39, roughly 1.01×10^{10} edges and density $\delta = 7.00 \times 10^{-04}$. The degree $d(P)$ of each node P was also calculated: the distribution of its values is significantly heterogeneous. Indeed the median value 827 contrasts the mean value of 3770.36 and the standard deviation estimated is 8146.85 (Fig. 2.25).

2.5.4 Modeling approach

The proposed model will enable us to recreate the data set that matches the measured distributions for files' popularity and for clients' activity (Fig. 2.22). The model is expected to correctly approximate the cumulative request patterns for the individual files observed in a given time window (Fig. 2.23). The input parameters of the model are a) time window $\{0, T\}$, b) number of files N_F, c) number of client peers N_P and d) number of file requests N_r. The aim of this model is to synthetically generate a sequence of timestamped requests for the available files in the network that should quantitatively resemble with the empirical (original) one. This will enable us to synthetically regenerate the realistic P2P traffic in a controlled environment which in the subsequent stages may quicken the computations with reasonable accuracy.

We present now a first version of the model, then we discuss how to improve it in the future. We assume that the distribution of the file requests or popularity (as measured in Fig. 2.22) can be approximated with the help of hypothetical distribution. For example, we can model the popularity distribution of files using a power-law $Prob(x_F = X) = aX^b$, where a and b are parameters that should be determined from the measured dataset (Fig. 2.22). Similarly, we can model the activity distribution (as measured in Fig. 2.22) of clients using a quadratic power-law of the kind $Prob(y_P = Y) = aY^b logY + c$, where a, b and c are parameters that should be determined from the measured dataset. These are early assumptions, and we will need further study of the most adequate model for the degree distribution, e.g., power law, stretched exponential, lognormal, etc.

As specified in the interest graph (provided by LIP6), we create a bipartite graph with the top set of file-nodes $\mathcal{F}$ and the bottom set of peer-nodes $\mathcal{P}$ (see Fig 2.24). We aim to connect the nodes of set $\mathcal{F}$ and set $\mathcal{P}$ such that the degree distribution of the nodes in the individual sets $\mathcal{F}$ and set $\mathcal{P}$ follow the measured distribution of file popularity and client activity respectively (Fig. 2.22). However, we need to ensure that the total number of links originating from the top set $\mathcal{F}$ and the bottom set $\mathcal{P}$ should be identical (or close to that). In order to do that, we introduce the concept of 'hypothetical degree' which is the degree assigned to the individual nodes of the set $\mathcal{F}$ and $\mathcal{P}$ according to the measured distribution of the file popularity and client activity. Next, we place N_r number of 'timestamped' links connecting the nodes of the sets $\mathcal{F}$ and $\mathcal{P}$. Here each timestamped link l_t represents a client's request for a file at some timestamp t. The links are created at timestamp t such that the following two conditions are satisfied

- The 'original' degrees associated with the nodes in set $\mathcal{F}$ and $\mathcal{P}$, as a result of the placement of the timestamped links l_t, follow the similar distributions as the 'hypothetical' distribution. Essentially the top and bottom nodes of the link l_t are selected based on the hypothetical degree of the nodes in the set $\mathcal{F}$ and the set $\mathcal{P}$ respectively.

- Both the files and the peer clients are available at time t. The availability of the files in the network is based on the presence of the provider nodes in the network.

On the other hand, the (file request) activity of the client nodes depends on the presence of the clients in the network. Hence, the timestamp associated to the links between the set $\mathcal{P}$ and $\mathcal{F}$ should be consistent with availability of the files and the client peers. The detail illustration follows.

Modeling file availability: The availability of the files highly depends on the number of providers that are present in the system. For each file, we compute the number of available providers using a queuing system based on the arrival rate λ and service rate μ: the idea is that if the queue is empty, that implies no providers are present in the system for the observed file. The queuing process may enable us to determine the intervals on which the providers of an observed file are present in the network. The arrival rate of nodes (λ) in this queuing system models the joining of the provider peers of the requested file in the network, which we assume follows Poisson distribution. On the other hand, service rate (μ) models the rate at which the provider nodes leave the network. This event primarily depends on the logout rate of the peers and can be approximately modeled with the help of exponential distribution (see 'Modeling client activity' next). Here we assume that there exists an infinite number of fictive servers so that every connected provider is being served at all times and leaves the queue whenever he disconnects. This situation corresponds to an $M/M/\infty$ queue which can be used to compute the number of providers available for a requested file F at some time t. Precisely the $M/M/\infty$ queue relies on its parameters λ_F and μ_F that respectively represent the arrival and departure rate of providers of file F in the network. (These parameters may also depend on the popularity x_F of the file F). Using this queuing system, we can model the availability of the observed file during the time window ($0 < t < T$). The assignment of the model parameters in the queuing system can be done in two steps. First we determine the number of providers that share a file depending on its measured popularity (Fig. 2.22). Then we determine the rate at which a single provider logs in and logs out (illustrated below as the client activity). The product of these rate by the number of providers that share the file gives us the desired rates.

Modeling client activity: We model the login and logout activities of peer clients with the help of two distinct Poisson distributions. At any time, if a client is logged in, the time before he logs out (with respect to login) is distributed exponentially (since the distribution of the time between two Poisson events is distributed exponentially). More precisely, the time between these two events follows an Erlang distribution since it is the sum of two exponential distributions. Briefly, this is the model we use to mimic the online availability of a given peer during the observed period $0 < t < T$.

Request generation: We generate N_r number of 'timestamped' links in between the top set $\mathcal{F}$ and bottom set $\mathcal{P}$ (Fig 2.24). A file F in set $\mathcal{F}$ and a node P in set $\mathcal{P}$ are chosen such that

- The selection of node P and F gives preference to the 'hypothetical' degree distribution
- The timestamp t assigned with the link (P, F) is consistent with the availability time window of the node P and F.

Our goal is to estimate the parameters corresponding to the data set, and generate some artificial data set, and check whether the model and the data agree. In order to do so, we can apply the statistical methods mentioned in Section 4 (Topological modeling), which can be adapted to the present case. More specifically, we can check if the model

recovers some statistics of interest as mentioned and pictured above in Fig. 2.22 (time interval between two requests of a file, distribution of popularities, etc.).

We believe that this model is the simplest possible, and the comparison with the data will undoubtedly reveal its shortcomings, thus reveal interesting features of the data. We may for instance find that the users can be separated into different communities with similar likings, identify those communities from the interest graph, and find that some parameters should be chosen differently for every community. A similar analysis can be done for files. We may also find that the diffusion implies (or not) a burstiness of the traffic between peers: the appearance of a much anticipated file generate a burst of interest in some communities. We also expect to find that a Poisson model is inadequate for modeling the activity of peers, and should be replaced for instance with a Gamma law.

2.5.5 Results and expected outcome

2.5.5.1 Diffusion of files

As mentioned previously, in fundamental spreading models such as the SIR model infected nodes spread the infection (or file in this case) to some of its neighbors, which spread in turn. The key parameter describing such model is the probability for each node to spread to each of its neighbors the file F. This can be estimated by the proportion of infected neighbors to the total number of neighbors—the latter is the degree $d(P)$ of P. Formally, let $s(P, F)$ be the number of peers to whom P provide F. This ratio estimation is given by

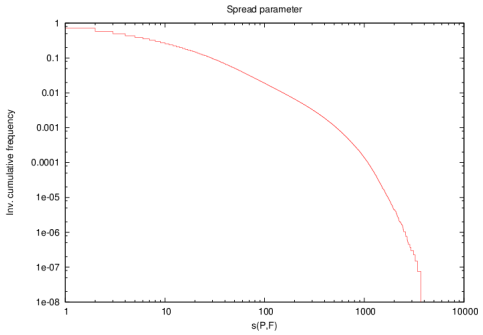
$$\sigma(P, F) = \frac{s(P, F)}{d(P)}.$$

The distribution of the values of $s(P, F)$ and $\sigma(P, F)$ are plotted below (Fig. 2.26(a) and 2.26(b)). The range of values for the former is 1 to 3709 with median 3, the mean value is 13.22 and the corresponding standard deviation is 39.48. The latter estimator ranges from 4.40×10^{-6} to 1 with median 2.36×10^{-4}, the mean value is 1.62×10^{-3} and the corresponding standard deviation is 1.01×10^{-2}.

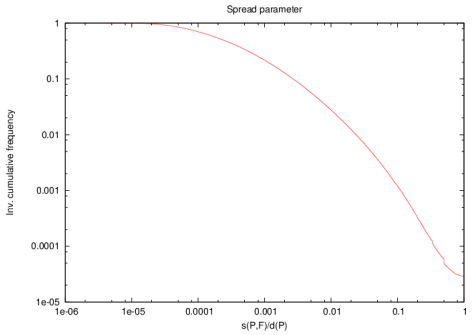
Of course, many other models than the simple spreading model may be considered: one may consider that the spreading probability depends on the edge, for instance. Our method makes it possible to study such variants, but we focus on the simple spreading case here, which we consider as a first step towards more intricate models.

The estimation for the parameters in the spreading models feature significantly skewed value distributions, both when considering the absolute number of infected neighbors $s(P, F)$, for a given peer P and file F, or correlating this number with the network topology (node degrees)—in the form of the ratio $\sigma(P, F)$. This study suggests the use of heterogeneous parameters and provides related statistics. Given that this local transmission mechanism forms the core hypothesis of the SIR model and derivatives, this is a valuable empirical information in the subsequent validation of improved models.

Finally, we give evidence for the fact that the classical hypothesis which consists in assuming that the spreading probability is homogeneous in the network is false. More importantly, we compute a distribution of this probability in the case of file diffusion in a large-scale P2P systems; this distribution helps in improving significantly modeling of such diffusions, as it may be used as a parameter for diffusion models. As a consequence, our modeling of file diffusion in a large P2P system, and thus of internet traffic, is improved by this result.



(a) Inverse cumulative plots of parameter estimations.



(b) Inverse cumulative frequency plots of spread parameter estimations.

Figure 2.26: Parameter estimation in spreading model

2.5.5.2 Towards a traffic model for the Internet

The implementation of the model will provide us a synthetically generated cumulative request for an observed file. We expect that the insights gained on the activity patterns of the peers, combined with a diffusion model, will be the key to formalizing a model for traffic on the overlay networks of peers, then on the physical network. For example, the burstiness of activity created in a community of peers by the diffusion of popular files may result in certain burstiness patterns for the traffic on the physical network of the Internet, assuming of course that this traffic is mainly generated from P2P networks. The resulting model will certainly be rather crude, but more realistic however than assuming a uniform all-to-all traffic, as is often done in routing simulations.

Chapter 3

Tools and Datasets

3.1 Tools

This section provides a description of the tools developed in order to measure the quantities defined in Section 2.

3.1.1 Neighbourhood flooding tools

3.1.1.1 UDP Ping

UDP Ping is a measurement tool developed in the project to discover interfaces of machines in the Internet. It is run on a machine M called the *monitor*. It takes the following parameters:

- Source interface (I_S). An 4-byte integer corresponding to an interface (IP address) of M on which UDP Ping is executed. When M is an end-host, it often has only one interface, called *the* IP address of M.
- Target interface (I_T). A 4-byte integer corresponding to an IP address.
- UDP Destination Port (P). A 2-byte integer in the 49152 - 65535 range. UDP Ports in this range are assumed to be usually unallocated.

The goal of UDP Ping is to detect if there is an active host T (the *target*) that owns the address I_T and, if such T exists, to obtain an interface $I_{T'}$ of T that is used by T to send packets towards I_S (belonging to M). To do so, UDP Ping forges an IP packet, carrying an UDP packet. The destination address of the IP packet is I_T. The destination port of the UDP packet is P. I_T is split in two 2-byte parts. The first 2-byte part is stored in the ID row of the IP packet header. The second 2-byte part is stored in the UDP Source Port row of the UDP message. By doing so, all the 4 bytes of I_T are stored in the packet/message headers. The data field of the UDP packet contains a signature (typically a contact e-mail address) to allow network administrators to contact the sender upon receiving the message.

The IP/UDP packet is then sent by UDP Ping through I_S. Once the packet is sent, UDP Ping listens to all incoming ICMP messages. If I_T corresponds to an active host T, then after receiving the packet, it detects that P is unreachable. T generates a type 3 ICMP message (Destination Unreachable), with error code 3 (Port Unreachable). This ICMP message also contains the headers of the incriminating packet, and in particular, the original IP ID row and UDP Source Port row. T then chooses one of its interfaces,

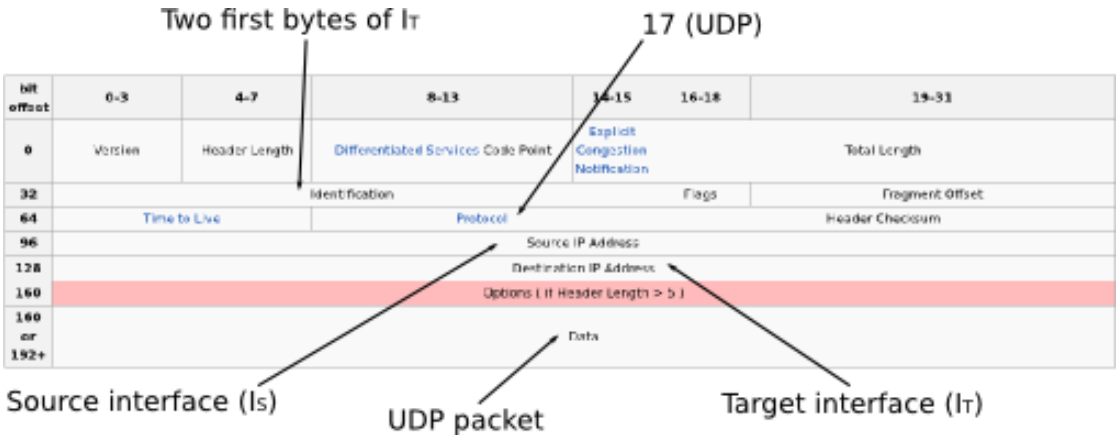


Figure 3.1: Ping UDP IP Packet structure

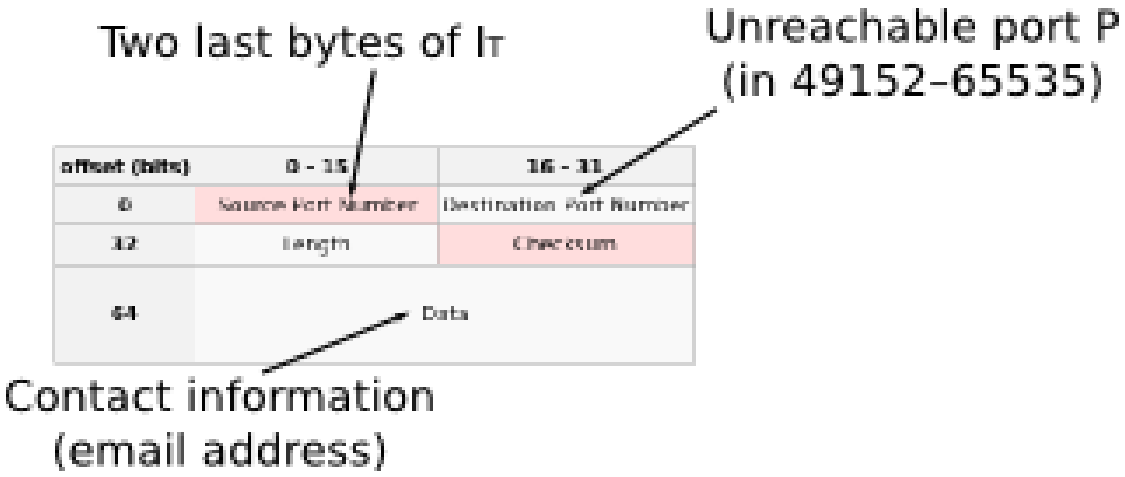


Figure 3.2: Ping UDP UDP Packet structure

$I_{T'}$, according to its routing policy, to send the ICMP message back to its sender, I_S. Note that if T has more than one interface, $I_{T'}$ can be different from I_T. Eventually, UDP Ping catches this ICMP message. It extracts the following information:

- ICMP attached information, including the original IP ID row and UDP Source Port row. This allows UDP Ping to reconstruct the 4-byte integer corresponding to I_T, and to identify this ICMP message as the expected one.
- Source address of the ICMP message - that is $I_{T'}$.

UDP Ping then exits with a success code and returns $I_{T'}$. After a set amount of time, if UDP Ping has not caught any ICMP message properly identified as the expected one (using the 4-byte integer reconstruction), it exits with a failure code. This can happen for multiple reasons, the most common being :

- I_T does not belong to any connected machine (target offline)
- I_T does belong to an existing, active machine T, but T discards UDP errors without sending ICMP error messages back
- I_T correspond to an active machine T that generates an ICMP error message but this message is filtered on its way back to M (usually near T).
- I_T is located beyond a firewall that silently discards unsolicited UDP traffic.
- P is used by T, making it effectively reachable (and then no error is generated at the network level).

3.1.1.2 Distributed UDP Ping

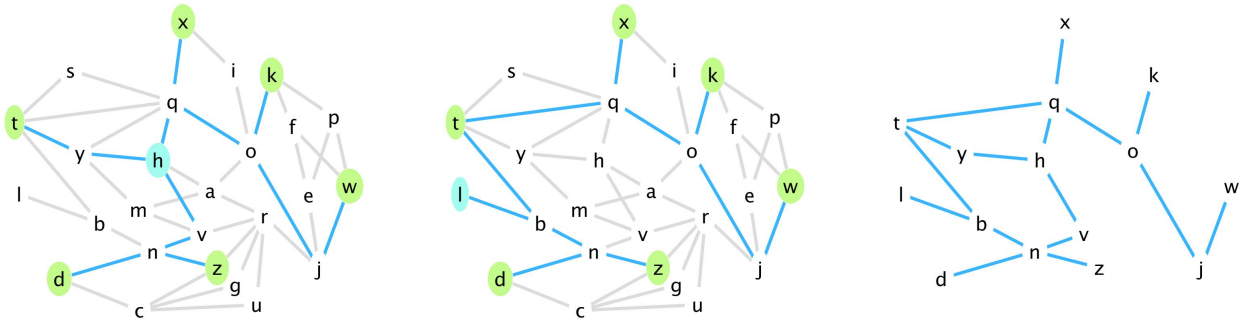
Since $I_{T'}$ can vary depending on I_S (and therefore depending on M), UDP Ping may be used from multiple monitors to get a list of distinct interfaces belonging to the same target. Using UDP Ping from a set of monitors towards the same target I_T is called Distributed UDP Ping, or Neighborhood Flooding. If the set of monitors is large enough and well distributed in the Internet, one may then obtain several (and even *all*) interfaces of the target.

Note that UDP Ping requires privileges not only to execute binary code on the monitor, but also to forge and send packets at a very low level, and to listen and decode all the incoming ICMP messages. On most UNIX system, UDP Ping therefore requires root privileges, which can be restrictive. However, if one disposes of a set of monitors and is granted such privileges, UDP Ping can be effectively distributed using shell scripts and ssh tools to launch remote execution of the tool an retrieve the answers.

Since receiving a large amount of UDP messages from distributed machines can look like a distributed attack by the target host, or even unintentionally disable it, extreme care must be taken not to send all the UDP messages at once. A delay can (and should) be set between the sending of successive UDP Ping probes from the different monitors. Likewise, all monitors should use the same target port, or else the measurement may look like a UDP port scanning. Once all these precautions are taken, the tool has no specific parameters to tune and is rather simple to use.

3.1.2 Tracetree tool

We propose here a novel approach which enables to observe the dynamics of the Internet topology at the scale of a few minutes. It consists in focusing on the part of the Internet topology viewed from a single monitor, which we call an *ego-centered view*, see Figure 3.3. Such views are far from representative maps of the Internet, but they have several key advantages. In particular, they can be obtained very rapidly with low network load, and thus may be repeated at a high frequency. This makes it possible to study their dynamics, and then to gain insight on the dynamics of the Internet topology itself.



ego-centered view from h ego-centered view from l obtained map

Figure 3.3: Measurement of an internet-like topology. The classical approach consists in running traceroute from a number of monitors, here h and l, towards some destinations, here t, x, k, w, z and d. We obtain here the following paths from h: h–y–t; h–q–x; h–q–o–k; h–q–o–j–w; h–v–n–z; h–v–n–d. We obtain the following paths from l: l–b–t; l–b–t–q–x; l–b–t–q–o–k; l–b–t–q–o–j–w; l–b–n–z; l–b–n–d. The final map of the network is obtained by merging all these paths. Instead, one may focus on the set of paths obtained from a *single* monitor, which we call its *ego-centered* view of the network.

3.1.2.1 Rationale and Motivations

In principle, one may use the traceroute tool [1] to obtain ego-centered views: it suffices to run it from a monitor towards a given set of destinations, and then merge the obtained paths, as in Figure 3.3.

However, this approach has severe drawbacks. In particular, it is highly redundant and it induces a very heterogeneous load on links: since traceroute sends one probe for each link on the path to discover, the links close to the monitor are overloaded. For instance, the traceroute ego-centered measurement from l described in Figure 3.3 discovers link l–b six times, using six probes. The situation is even worse in practice as we will detail below.

3.1.2.2 Tracetree

In order to perform fast ego-centered measurements with low and balanced network load, we therefore had to design a dedicated measurement tool, called *tracetree*. We provide a solution by designing the tracetree algorithm (detailed hered below).

[1] traceroute discovers a path from a monitor to a destination by sending one probe packet per link on the path.

Algorithm 1: tracetree algorithm.

```
Input: set D of destinations, with d ∈ D at distance ttl_d.
to_probe ← empty queue, to_receive ← ∅, seen ← ∅
foreach d ∈ D do add (d, ttl_d) to to_probe
while to_probe not empty or to_receive ≠ ∅ do
α   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
    // here necessarily to_receive ≠ ∅
β   if answer p to a probe to (d, ttl) received then
        // p sent by p.source, reply to a probe to (d, ttl)
        if (d, ttl, _) ∈ to_receive then
            // else timeout
            remove (d, ttl, _) from to_receive;
            print p.source ttl d
            if (p.source, ttl) ∉ seen then
                add (p.source, ttl) to seen
                push (d, ttl − 1) in to_probe if ttl > 1
    for (d, ttl, t) ∈ to_receive if timeout exceeded do
        remove (d, ttl, t) from to_receive
        print * ttl d
        if ttl > 1 then
            push (d, ttl − 1) in to_probe
```

The traceroute tool discovers a path by sending a series of probes towards the destination in a forward manner: the first probe discovers the first link, the second probe discovers the second link, and so on. Instead, the tracetree tool discovers a tree by sending probes towards all destinations in parallel in a backward manner and avoids redundancy by stopping probing towards some destinations when paths collapse: given a set of destinations, it first discovers the last link on the path to each of them, then the previous link on each of these paths, and so on; when two (or more) paths reach the same node then it stops probing towards all corresponding destinations except one. See Figure 3.4 for an illustration and details below.

The tracetree tool performs ego-centered measurements very efficiently, both regarding time and network load. It sends exactly *one* probe for each link to discover, and thus induces a perfectly balanced load. Radar measurements then consist in iterating ego-centered measurements with tracetree, from a monitor to a given set of destinations.

3.1.2.3 Unbalanced load induced by traceroute ego-centered measurements

One may use traceroute directly to collect ego-centered views by probing a set of destinations. This approach however has serious drawbacks. First, as detailed in [27], the measurement load is highly unbalanced between nodes and there is much redundancy in the obtained data (intuitively, one probes links close to the monitor much more than others). Even worse, this implies that the obtained information is not homogeneous, and thus much more difficult to analyze rigorously.

Figure 3.5 illustrates this imbalance. The distribution of the link load (left) shows that some links are probed a very high number of times with traceroute, up to 3 000

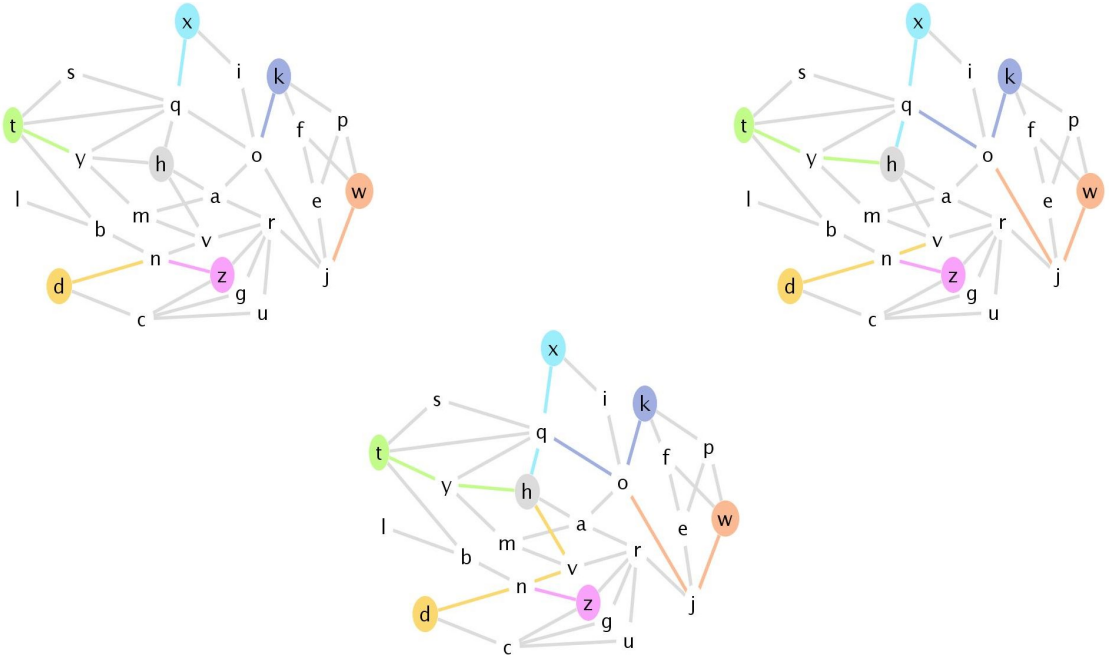


Figure 3.4: Illustration of the tracetree measurement method. Top left: the first series of probes discovers the last link before each destination; tracetree stops probing towards z because the paths to d and z collapse (at n). Top right: the second series of probes discovers the links just before, and tracetree stops probing towards most destinations: towards t and x because the full path towards them has been discovered; towards k and w because the corresponding paths collapse with previously discovered nodes. Bottom: the third (and last) series of probes contains only one probe, sent towards d which ends the tracetree measurement. Finally, tracetree sent exactly one probe for each link, thus avoiding redundancy and reducing the network load significantly.

times. This load is placed on links close to the monitor (center, black plot): a very large number of probes are sent a small distance from the monitor. Comparatively, our tracetree tool imposes a much lighter load: it sends a much smaller number of probes, in particular to links close to the monitor (Figure 3.5, center, blue plot). Finally, repeated ego-centered measurements with tracetree discover more IP addresses (and therefore more interesting information) with fewer probes sent than comparable traceroute measurements (Figure 3.5, right).

3.1.3 Tool for BGP routing stability

In order to provide another perspective on the dynamics of routing, we have implement a tool that provides relevant information with respect to the stability of prefix reachability. The tool uses the Routing System Stability metric initial specified in [60], further formalized and extended in Section 2.4.

3.1.3.1 Overview

The idea was to obtain a tool to process real observed BGP updates, extract the information they contain and follow the evolution of the stability of the routes across time. In order to compute the stability of the routes we used the metrics defined in Section

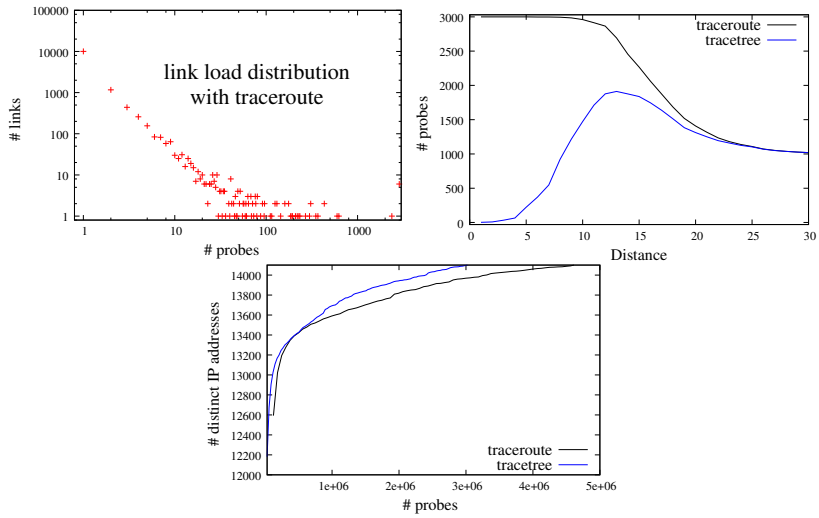


Figure 3.5: Top left: typical link load distribution with a traceroute ego-centered measurement. For each value x on the horizontal axis, we give the number of links which are discovered x times during a traceroute ego-centered measurement with 3 000 destinations (base value). In an equivalent tracetree measurement, exactly one probe is sent for each link. Top right: number of probes sent, as a function of the distance from the monitor, with traceroute and tracetree ego-centered measurements with 3 000 destinations. Bottom: number of distinct IP addresses discovered with several successive traceroute and tracetree ego-centered measurement rounds, as a function of the number of probes sent.

2.4 to determine the most stable route.

The tool uses the BGP updates provided by the Route-Views project. Route Views is a project founded by the University of Oregon which consists in a set of routers, mainly running Cisco software (or Cisco equivalent, like Zebra or Quagga), distributed across the world. The BGP routing information of these routers can be accessed by anyone, anywhere in the world. The primary objective of this project was to provide information to the Internet Service Providers about how their network prefixes were viewed by other across the Internet. Nowadays this project provides useful information to anyone interesting or involved in the field of Internet research. See Section 3.3.2.1 for more details about this dataset and the definition of the variables/concepts used below. The Route-Views updates contain the BGP information a router receives from its neighbours. That is basically each neighbour route (with its attributes) to each prefix the neighbour has knowledge about. With that, the router we are monitoring can find a route for each IP prefix it needs to send packets to.

Our tool tracks the changes and evolution over the time of these routes that the neighbours are sending to "our" router and generate some stability metrics associated to each of these routes. With this stability metrics computed for each route, our tool also take care about storing in given instants of time the most stable route for a prefix, and also the BGP selected route for that prefix.

Given the fact that the current Route-Views file format does not provide us the Loc_Rib information of their routers (that is, basically and for our interests, the information about which route BGP selects for each prefix). We must manually derive it from the Adj_RIB_In table (provided by Route-Views). The table we can use is the Adj-

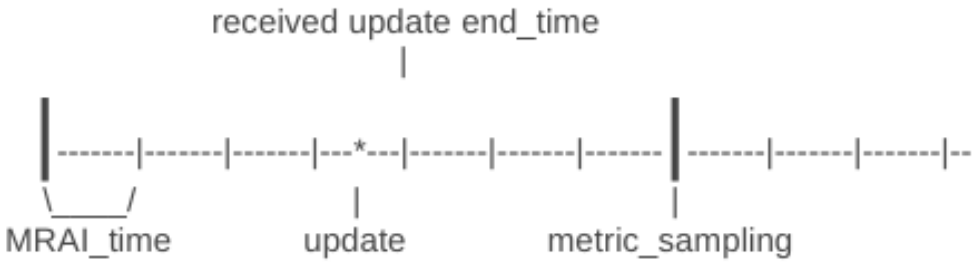


Figure 3.6: Tool time model

_RIBs_In. We have decided to obtain the Loc_RIB tables from the Adj_RIBs_In ourselves. This will be done by implementing a selection process based on the algorithm used in Quagga/Zebra routers [2], which is a simplification of the selection process implemented by Cisco. The path selection process of Quagga routers consist of the following decision steps:

1. Prefer the path with highest Weight attribute
2. Prefer the path with highest Local Preference value
3. Prefer the path with the shortest AS-path length
4. Prefer the path with the lowest origin value (IGP ¡ EGP ¡ Incomplete or ?)
5. Prefer the path with the lowest MED

The Weight is a local attribute of the router and it is Cisco and Quagga specific. As this attribute is not present in the data sets obtained from Route Views, the selection process to be implemented will consider just the points from 2 to 5.

The main operation of the program is the following:

1. Read a Route-Views BGP update file.
2. Process a given amount of time of BGP updates.
3. Update the state of the route table in memory
4. Based in this update, update the stability metrics
5. Periodically store the actual state of the route table and its correspondent stabilities in a database

Specifically, the time model used by the tool is on Fig. 3.6, with the following meaning:

- MRAI_time: Time in which we increment the time counter in one unit
- received update end_time: When we process the received update
- metric_sampling: When we compute the stability metrics for the whole table and store it in the database

Each time we receive an update we do the following: we look at the arrival time of the update and calculate the correspondent end_time, if end_time is greater than the last_metric_sampling, we update the database until last_metric_sampling, if not, we store in memory the stabilities for the prefix which generated the update.

3.1.3.2 Design of the tool

We have designed and implemented the tool using an object oriented programming language: C++. This language also provides us fast-processing, this is important since the amount of information that the tool has to process is very large.

i Description of the classes

Here is a description of the most important classes implemented in our tool.

Entry class

This class represents a possible route for a given prefix. It contains the actual stability metric for this route which is updated over the time. It also stores the number of AS hops required to reach the prefix using that route, and the rest of the BGP attributes related to a route, like local-pref, MED, community, etc.

Route class

This class represents the aggregation of all the routes for a given prefix. It stores this set of entries associated with a prefix using an unordered map, which is a data structure that provides fast access and arbitrary amount of data.

Moreover it maintains the data about the actual, and previous BGP selected route, and the information about the most stable route in the previous iteration.

RouteTable class

This class, stores all the prefixes of the system. Each of this prefix is a Route object. It uses again an unordered map to keep the information. Also, this class contains the data of the database object to use, and keeps the track of the last metric sampling time.

The collaboration diagrams for these classes can be found on Section 3.1.3.4, and the call graph for the program appears in Section 3.1.3.5.

ii Description of the database tables

Here we provide a description of the database tables that our tool uses. There are three of them: *route*, *selected* and *taula* tables.

***route* table:**

The *route* table stores one entry for each prefix and for each metric sampling time. It is updated in each metric sampling time. The columns of this table are explained here:

- time = Metric sampling time for this entry
- IP1 = First part of the prefix IP address
- IP2 = Second part of the prefix IP address
- IP3 = Third part of the prefix IP address
- IP4 = Fourth part of the prefix IP address
- MASK = Prefix IP mask
- selectable_vs_selected_min = min (fi/fselected)
- selectable_vs_selected_max = max (fi/fselected)
- selectable_vs_selected_ave = add(fi/fselected) / prefix entries

- selectable_vs_selected_var = variance of fi/fselected
- selectable_vs_stablest_entry_min = min (fi/fstablest_entry)
- selectable_vs_stablest_entry_max = max (fi/fstablest_entry)
- selectable_vs_stablest_entry_ave = add(fi/fstablest_entry) / prefix entries
- selectable_vs_stablest_entry_var = variance of fi/fstablest_entry
- fn_locRib_selected = BGP selected route stability at instant n-1
- fn1_locRib_selected = BGP selected route stability at instant n
- hops_selected = Number of ASs in the ASPath of the BGP selected route
- fn_locRib_stablest = Stability of the most stable RIB route for the given prefix at instant n-1
- fn1_locRib_stablest = Stability of the most stable RIB route for the given prefix at instant n
- hops_stablest = Number of ASs in the ASPath of the most stable RIB route for the given prefix
- hops_difference = The difference in number of ASs in the ASPath of the most stable RIB route for the given * prefix and the selected route for that prefix.
- removed = Boolean that indicates if the prefix has been removed or not

***taula* table:**

The *taula* table stores the averaged values of the *route* table. Which are much more suitable to deal with and to plot.

***selected* table:**

The *selected* table stores the following data each time BGP selects a new route for a prefix. The data stored is independent of the temporal model previously showed.

- time = Metric sampling time for this entry
- IP1 = First part of the prefix IP address
- IP2 = Second part of the prefix IP address
- IP3 = Third part of the prefix IP address
- IP4 = Fourth part of the prefix IP address
- MASK = Prefix IP mask
- f_prev_selected : previous selected route stability before the new route is selected.
- f_selected: stability of the new selected route.
- f_stablest: stability of the most stable route for that prefix.
- h_prev_selected : Number of ASs in the ASPath of the previously selected route.
- h_selected: Number of ASs in the ASPath of the new selected route.
- h_stablest: Number of ASs in the ASPath of the most stable route.

3.1.3.3 Database tables

This section shows the diagrams for the database tables. Fig. 3.7 shows the *route* table, Fig. 3.8 show the *selected* table, and finally the table *taula* is showed in Fig. 3.9.

3.1.3.4 Class collaboration diagrams

This section shows the class collaboration diagrams for the most important classes of our tool. Fig. 3.10 shows the diagram for the class Route, in which the class collaboration of the class Entry is also shown. Fig. 3.11 corresponds to the diagram of class RouteTable. Another quite important class, Metric_Result, can be found in that figure.

3.1.3.5 Call graph

In this section is shown a call graph for the program. Figure 3.12 represents the functions calls relationships starting from the main function.

3.1.3.6 Conclusion

In this section, we have provided the description of a tool to compute BGP routing stability metrics. Its basic operation consists in process real BGP updates obtained from the real Internet, extract and store over the time the route parameters that these updates contain and apply some metrics to obtain the stability of the routes. Also the tool provides us aggregated information of these routes and its stability.

3.2 Measurements

3.2.1 Tracetree measurements

As already discussed in various contexts [28, 27, 26, 58, 56, 67], one may avoid the issues described above by performing tree-like measurements in a backward way: given a set of destinations to probe, one first discovers the last link on the path to each of them, then the previous link on each of these paths, and so on; when two (or more) paths reach the same node then the probing towards all corresponding destinations, except one, stops.

Note that this requires knowing the distance towards each destination, which is not trivial [56]. This plays a key role here, since over-estimated distances lead to several packets hitting destinations. Under-estimated distances, instead, miss the last links towards the destinations. Each tracetree measurement therefore relies on an *estimation* of the distance towards each destination (we detail in the next section how we obtain this estimation). If the distance is over-estimated, then more than one packet reaches the corresponding destination; we only output the information corresponding to the last packet to which the destination replies, which corresponds to the accurate distance. If the distance happens to be under-estimated (we do not see the destination at this distance), then we set it to a default maximal value (generally equal to 30) and start the measurement from there.

However, as illustrated in Figure 3.13, such naive measurements encounter serious problems because of routing changes and other events. Instead, when using the proposed tracetree algorithm, the tree nodes are not IP addresses anymore, but pairs composed of

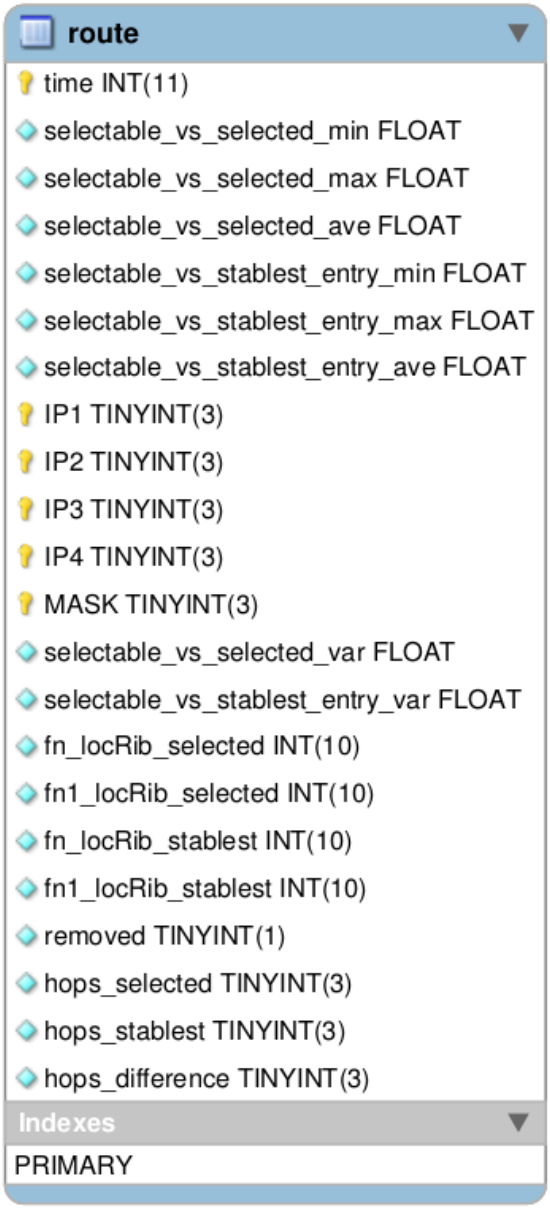


Figure 3.7: Route table

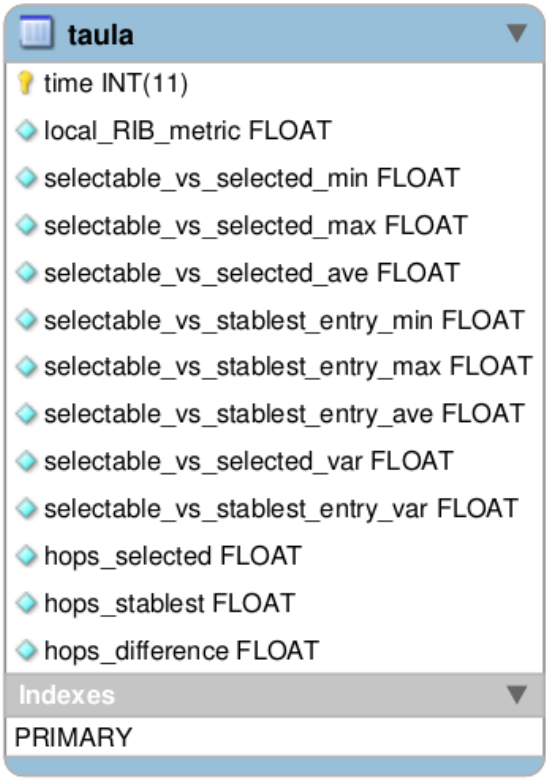


Figure 3.8: Taula table

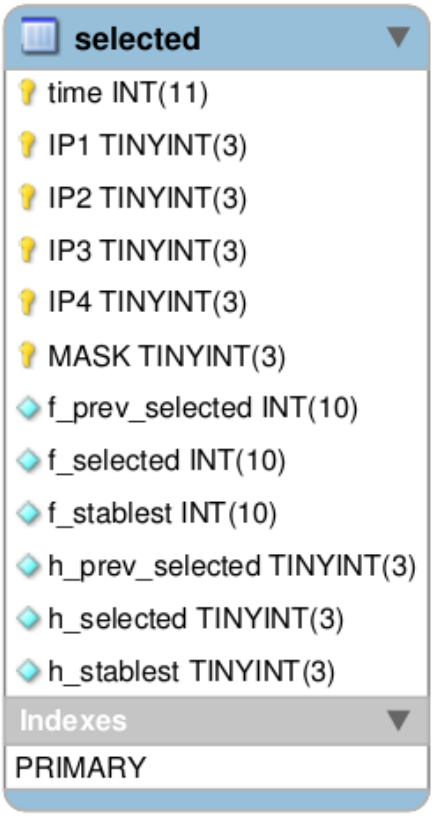


Figure 3.9: Selected table

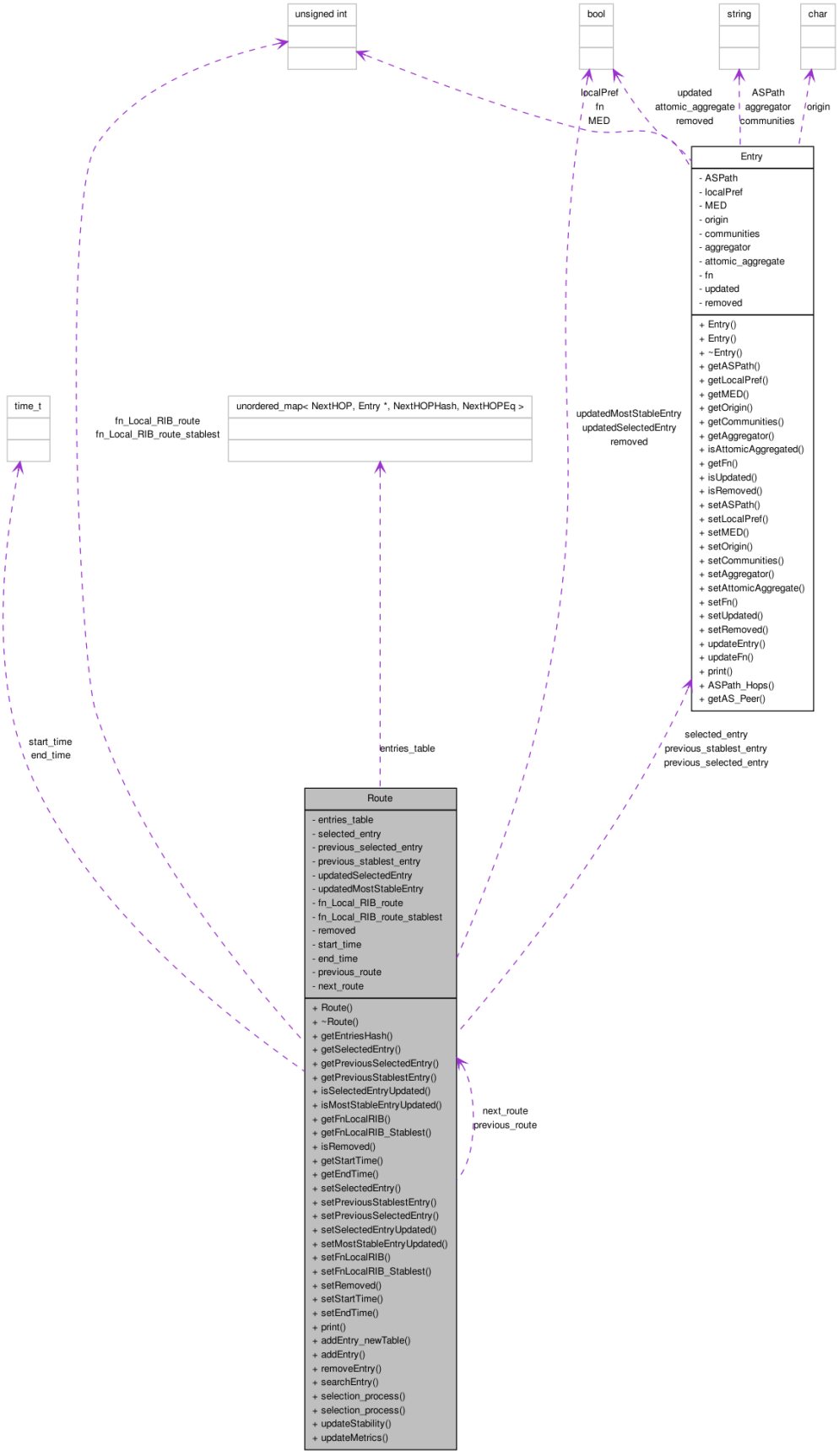


Figure 3.10: Route class collaboration diagram

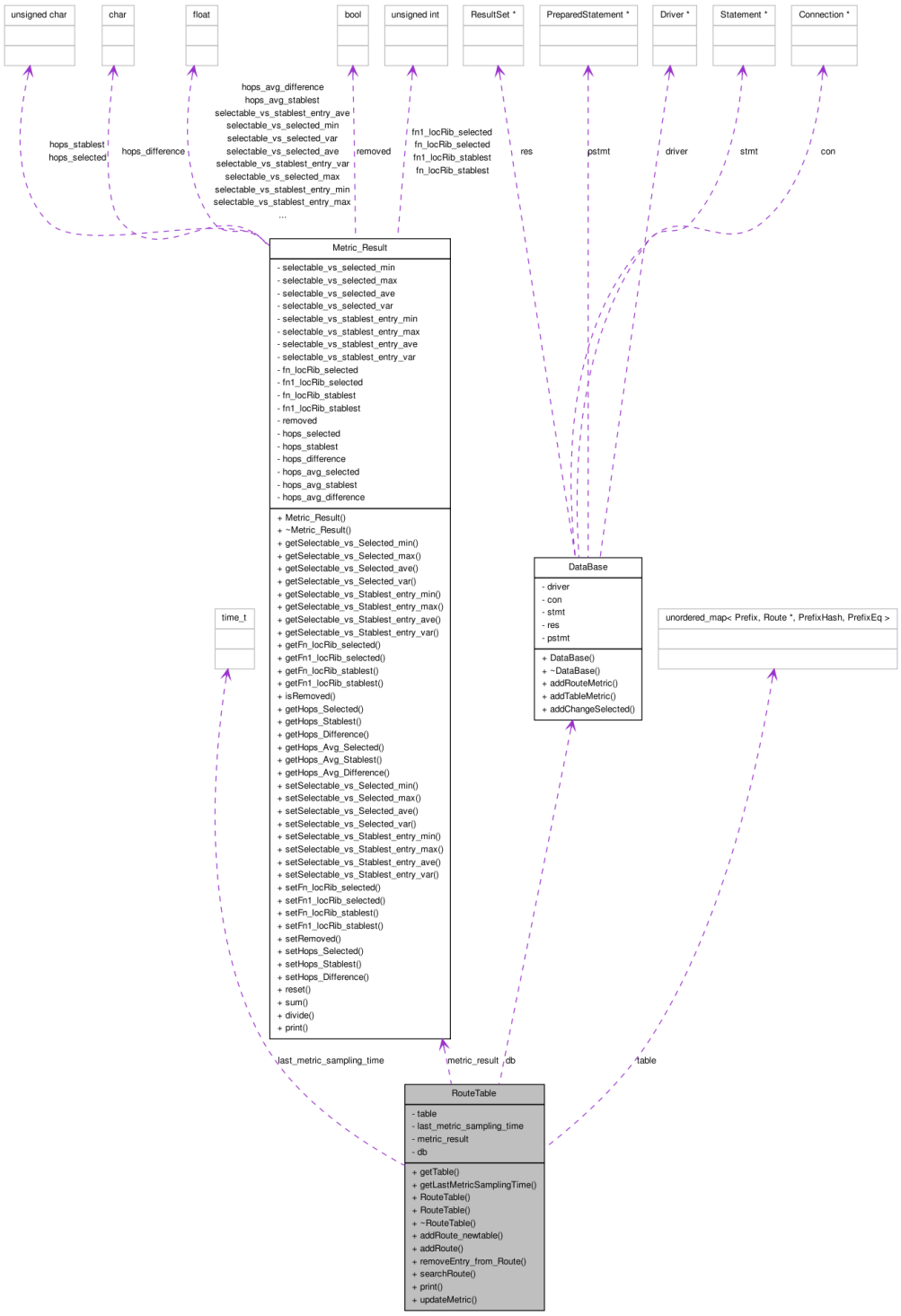


Figure 3.11: RouteTable class collaboration

Figure 3.12: Call graph

an IP address (or a star if a timeout occurred) and the TTL [2] at which it was observed (see Figure 3.13 for an illustration). This is sufficient to ensure that the obtained view is a tree, while keeping the algorithm very simple. It sends only one packet for each link, and thus is optimal. Moreover, each link is discovered exactly once, which gives an homogeneous view of the topology and balances the measurement load.

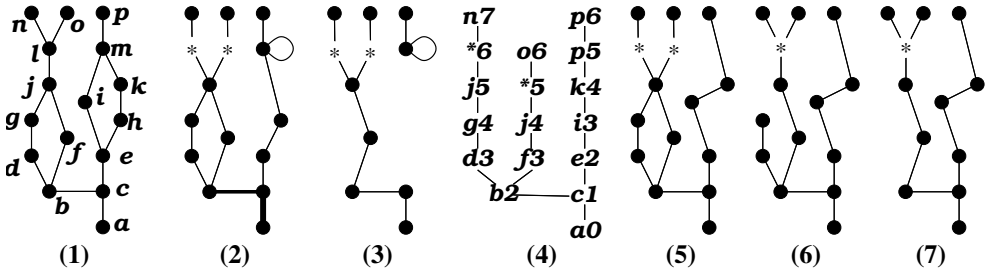


Figure 3.13: Typical outputs of various measurements schemes. (1) – Real topology. a is the monitor, n, o, and p are the destinations. We suppose that l does not answer to probes, that b is a per-destination load balancer, forwarding traffic for n to d, and traffic for o to f, and that e is a per-packet 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 thicker lines here). (3) – Naive tree measurement. Because of a route change due to per-packet load balancer e, one obtains a disconnected part. (4) – Measurement with tracetree. Nodes are pairs of IP addresses and TTL, with redundancy in the addresses; one necessarily obtains a tree. (5–7) – Main steps of the filtering process. (5) – Pairs with same IP address are merged and loops are removed; (6) – Appropriate stars are merged and a BFS tree is computed; (7) – Leaves which are not the last node on a path towards a destination are iteratively removed in order to ensure that the leaves of the obtained trees are destinations (and thus these trees are similar to routing trees). This is the final output of the filter.

From such trees with (IP,TTL) nodes, one obtains a tree on IP addresses by applying the following filter (illustrated in Figure 3.13) [3]: first merge all nodes of the tree which correspond to the same IP; remove loops (links from an IP to itself); iteratively remove the stars with no successor; merge all the stars which are successor of a same node into a unique star; construct a BFS tree of the obtained graph which leads to a tree on IP addresses; iteratively remove the leaves which are not the last nodes encountered when probing any destination.

The key point is that the obtained tree is a possible IP routing tree from the monitor to the destinations (similar to a broadcast tree). The obtained tree contains almost as much information as the original tracetree output and has the advantage of being much more simple to analyze. We empirically evaluated the impact of this filtering on our observations, and found that it was negligible (a very small fraction of the nodes or links were removed).

Many non-trivial points would deserve more discussion. For instance, one may apply a greedy sending or receiving strategy (by replacing line α or β in Algorithm 1 by a

[2] *Time to Live*: it corresponds to the number of hops the packet can perform before considered as having failed to reach the destination

[3] The measurement would be slightly more efficient if the filter was included directly in tracetree; however, to keep things simple and modular, we preferred to separate the two.

`while`, respectively); identifying reply packets is non-trivial, as well as extracting the relevant information from the read packets; introducing a delay may be necessary to stay below the maximal ICMP sending rate of the monitor; one may consider answers received after the timeout but before the end of the measurement (whereas we ignore them); one may use other protocols than ICMP (the classical traceroute uses UDP or ICMP packets); the initial order of the destinations may have an impact on the measurement; there may be many choices for the BFS tree in the filter; etc. However, entering in such details is far beyond the scope of this paper, and we refer to the open-source code and its documentation [4] for full details.

3.2.2 Radar measurements

With the tracetree tool and its filtered version, 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. The measurement frequency must be high enough to capture interesting dynamics, but low enough to keep the network load reasonable. We will discuss this in the next section.

We also use each tracetree measurement to estimate the distances towards all destinations for the next round of measurement: one may indeed suppose that the distance between the monitor and any destination generally is stable between consecutive rounds. Then, the distances at the start of a given round are the ones observed during the previous round (or the maximal value if the destination was not seen).

We performed massive radar measurements. For this purpose, we have used a wide set of more than one hundred monitors scattered around the world, provided by PlanetLab [64] and other structures (small companies and individual DSL links) [4]. In order to remain as general as possible, and to simplify the destination setup, we used destinations chosen by sampling random valid IP addresses and keeping those answering to ping *at the time of the list construction*. Other selection procedures are not precluded.

3.2.3 Influence of parameters

All our measurements were conducted with variations of the base parameters (see Table 3.1); wherever it is not explicitly specified, the parameters were the base ones. We ran measurements continuously during several weeks, with some interruptions due to monitors and/or local network shutdowns. The obtained data is available at [4].

First notice that many parameters (including the monitor and destination set) may have a deep impact on the obtained data. Estimating this impact is a challenging task since testing all combinations of parameters is totally out of reach. In addition, the continuous evolution of the measured object makes it difficult to compare several measurements: the observed changes may be due to parameter modifications or to actual changes in the topology.

To bypass these issues while keeping the study rigorous, we propose the following approach. We first choose a set of seemingly reasonable parameters, which we call *base parameters*. Those parameters are described in Table 3.1. 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 (generally one of) these parameters.

Control monitors make it possible to check that the changes observed from test

Parameter	Base value
# destinations	3 000
choice of destinations	random, answering to ping
choice of monitor	PlanetLab
maximal TTL	30
timeout	2s
delay between rounds	10 minutes

Table 3.1: Parameters of our radar measurements, and base values of these parameters.

monitors are due to changes of parameters, not to events on the network. The alternation of periods with base parameters and modified ones also makes it possible to confirm this, and to observe the induced changes in the observations. In many cases, it is also possible to simulate what one would have seen in principle if the parameters had stayed unchanged, which gives further insight (we will illustrate this below).

We focus on a few representative parameters only, the key conclusion being that the base parameters fit our needs very well.

Figure 3.14 (left) shows the impact of the inter-round delay: on the rightmost part the delay was significantly reduced, leading to an increase in the observation's time resolution (*i.e.* more points per unit of time). It is clear from the figure that this has no significant impact on the observed behavior. In particular, the variations in the number of IP addresses seen, though they have a higher resolution after the speed-up, 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 higher dynamics.

Figure 3.14 (middle) shows the impact of the number of destinations. As expected, increasing this number leads to an increase in the number of observed IP addresses. It is to be noted however that the two increases are not proportional. The key point however is that increasing the number of destinations may lead to a relative loss of efficiency: simulations of what we would have seen with 3 000 or 1 000 destinations display a smaller number of IP addresses than direct measurements with these numbers of destinations (the control monitor proves that this is not due to a simultaneous topology change). This is due to the fact that probing towards 10 000 destinations induces too high a network load: since some routers answer to ICMP packets with a limited rate only [35], overloading them makes them invisible to our measurements. 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 3.14 (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 and $2s$). The base value of the timeout (2s) seems therefore appropriate, because it is rather large and does not lead to a long round duration.

We also considered other observables (like the number of stars seen at each round, and the number of packets received after the timeout), for measurements obtained from various monitors and towards various destinations; in all cases, the conclusion was the same: the base parameters meet our requirements.

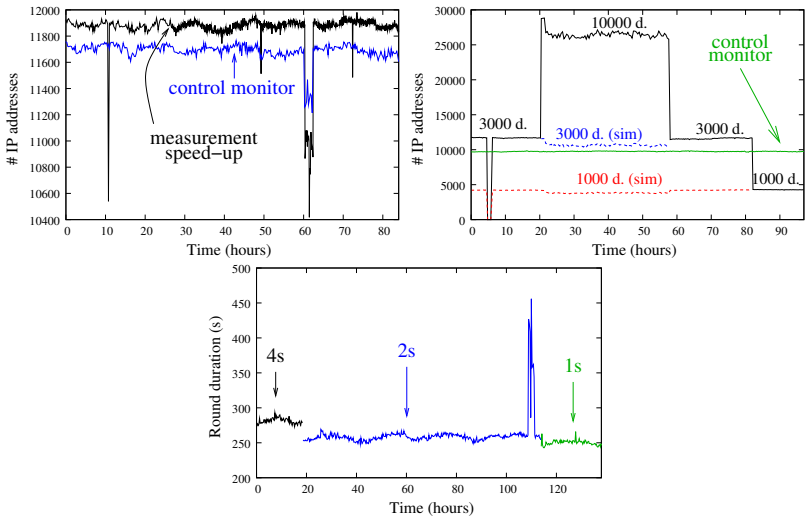


Figure 3.14: **Impact of measurement parameters. Top left: impact of inter-round delay.** Number of distinct IP addresses viewed at each round. The bottom plot corresponds to a control monitor with the base parameters; the other monitor starts with the base parameters, and about 27 hours later we reduce the inter-round delay from 10 minutes to 1 (each ego-centered measurement takes around 4 minutes). **Top right: impact of the number of destinations.** Number of distinct IP addresses viewed at each round. 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 the base parameters, thus with a destination set D of size 3 000, changes to a set D' of 10 000 destinations containing D, goes back to D, and finally turns to a subset D'' of size 1 000 of D. In addition, the dotted plots are 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). **Bottom: impact of timeout value.** Round duration (in seconds). The monitor starts with a timeout value of 4 s, then we change it to 2 s, and finally to 1 s.

3.3 Datasets

3.3.1 Peer to peer dataset

3.3.1.1 The eDonkey protocol briefly

EDonkey is a semi-distributed peer-to-peer file exchange system based on directory servers. These servers index files and users, and their main role is to answer to searches for files (based on metadata like filename, size or filetype for instance), and searches for providers (called *sources*) of given files.

Files are indexed using a MD4 hash code, the fileID, and are characterized by at least two metadata: name and size. Sources are identified by a clientID, which is their IP address if they are directly reachable or a 24 bits number otherwise.

EDonkey messages basically fit into four families: management (for instance queries asking a server for the list of other servers it is aware of); file searches based on metadata, and the server's answers consisting of a list of fileIDs with the corresponding names, sizes and other metadata; source searches based on fileID, and the server's answers consisting of a list of sources (providers) for the corresponding files; and announcements from clients which give to the server the list of files they provide. Those different exchanges between peers and a server are sketched in Fig. 3.15.

An unofficial documentation of the protocol is available [48], as well as source code of clients; we do not give more details here and refer to this document for further information.

3.3.1.2 Obtaining data

Before starting any traffic capture, one has to obtain the agreement of a server administrator. The following guarantees made it possible to reach such an agreement: negligible impact of the capture on the system; use of collected data for scientific research; and high level of anonymisation (higher than requested by law).

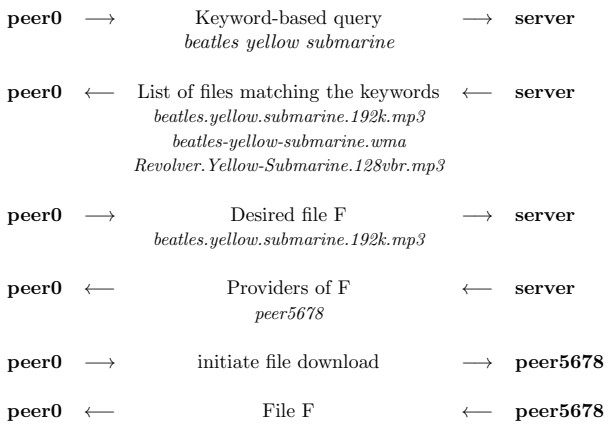
The ideal solution is to patch the server source code to add a traffic recording layer. However, as this source code is not open-source, we had to obtain from the code author a specialized version transmitted directly to the administrator. The administrator ran this patched version during 2 days on a server located in a *datacenter* to which we had no access. This server was the biggest server running in the eDonkey network at that time.

3.3.1.3 Final dataset

The final dataset we obtained consists in a series of 8 867 052 380 eDonkey messages (queries from clients and answers to these queries from the server) in text format. It contains very rich information on users at 89 884 526 distinct IP addresses dealing with 275 461 212 distinct fileIDs, while preserving the privacy of users, as it was anonymized on the fly during the capture.

Here is a sampling of the first lines extracted from the data:

```
@@0 LOGIN 16777297@@\\
@@1 LOGIN 16777336@@\\
@@1 LOGIN 16777353@@\\
@@1 UDP_SEARCH2 16777379 1223301efeef19068df06a589b9a15a8 16777265 16777275@@\\
@@1 LOGIN 16777382@@\\
```



1

Figure 3.15: Exchanges between peers and servers in eDonkey.

```
@@1 LOGOUT 16777336@@\\
@@2 UDP_SEARCH2 16777419 2852c276ad052ea6727d4bf8344705a2 16777242@@\\
@@2 TCP_SEARCH2 16777290 f561b54ca122ff8bfb94a31dad0ef1b2 16777331@@\\
@@2 LOGIN 16777469@@\\
@@2 UDP_SEARCH2 16777495 6902a315e52a6c9a0ea9d127ade1cdca 16777358@@\\
@@2 UDP_SEARCH2 16777444 b199f09bf25f3382ee123c8983a431b5 16777245 16777375@@\\
@@3 LOGIN 16777640@@\\
@@4 LOGIN 16777671@@\\
@@4 UDP_SEARCH2 16777697 060aa143a0e5547e16e8b6a061b2a407 16777225@@\\
@@4 LOGIN 16777767@@\\
@@5 TCP_SEARCH2 16777457 539d5607334da526896aa17ee18c6393 16777384@@\\
@@5 LOGIN 16777819@@\\
@@5 UDP_SEARCH2 16777842 32af7b7945f05078aa588bd250fe4b48 16777266@@\\
```

From this format, we can extract the relevant information that is used in EULER, which relies on the time stamp, a unique identifier for each user and a unique identifier for each file, following the following syntax: $t\ C\ F\ P_1\ P_2\ ...\ P_k$

Thus, from the raw data, we filter the relevant lines and map each user and file into a unique integer. This process applied on the example above results in the following lines:

```
@@1 0 0 1 2@@\\
@@2 3 1 4@@\\
@@2 5 2 6@@\\
@@2 7 3 8@@\\
@@2 9 4 10 11@@\\
@@4 12 5 13@@\\
@@5 14 6 15@@\\
@@5 16 7 17@@\\
```

3.3.2 BGP dataset

We have to distinguish two different datasets here. First the dataset we have used to feed our tool (see Section 3.1.3 for details about the tool) and second the dataset generated by our tool processing the first one.

The first dataset corresponds to real world BGP updates obtained from a router of the Route Views project. These updates are the messages that this router receives from its neighbours that provides it information about how to reach the different IP prefix over the Internet. In other way, they provide it the routes to reach each prefix. This updates information consists on BGP protocol defined attributes for each route, like next hop, AS path, etc. From this information the BGP router decides (as described in Section 3.1.3.1) which route is the optimal for each prefix. As also described in Section 3.1.3.1 we have no direct access to the BGP selected route for each prefix, so we have to compute ourselves the one that BGP, following its decision process, will chose.

The second dataset corresponds to the information we have computed from the first dataset. Our tool observed the changes in the routes over the time and process them to assign a stability metric to each route in each instant. Given the fact that this amount of stability metrics is so huge, our tool summarized the data given an average metric for each prefix, comparing the stability of all the possible routes for that prefix with the stability of the selected route by BGP for that prefix, but also the comparing the most stable route with all the possible routes.

These stability metrics are processed for all the BGP routes we can track from the Route Views router that we are monitoring.

3.3.2.1 Description of the used dataset

The Routing Information Base (RIB) within a BGP speaker (a router which runs the BGP protocol, thus is capable of *speaking the BGP language* with their BGP neighbors) consist of three different tables:

- a) Adj-RIBs-In: The Adj-RIBs-In store routing information that has been learned from inbound UPDATE messages. Their contents represent routes that are available as an input to the Decision Process.
- b) Loc-RIB: The Loc-RIB contains the local routing information that the BGP speaker has selected by applying its local policies to the routing information contained in its Adj-RIBs-In.
- c) Adj-RIBs-Out: The Adj-RIBs-Out store the information that the local BGP speaker has selected for advertisement to its peers. The routing information is stored in the Adj-RIBs-Out to be carried in the local BGP speaker's UPDATE messages and advertised to its peers

The dataset that can be used to feed our tool is obtained from the Route Views project. Route Views has an archive which contains BGP feeds from a set of routers: Cisco and Linux boxes running Zebra. The dataset gathered from Cisco routers contains their BGP routing tables (Adj_RIBs_In) and it is updated each two hours. Zebra boxes also provide the Adj_RIBs_In files but additionally, once at 15 minutes, a file with the received BGP Updates. Unfortunately, as we mention in a previous section although the archive files of the Cisco routers published before March of 2008 contain the preferred routes (Loc_RIB), the current format used by Route Views doesn't provide this information for any of their routers. As a result, considering that the only policy applied by Route-Views sets the next hop to the peer IP address, the table we can use is the Adj¬_RIBs_In.

We have selected a Zebra box's data from Route Views as input for our tool as its archives also contains BGP updates information necessary in determining the changed routes but also because it provides more flexibility by allowing the computation of other BGP updates dependent metrics. We have specifically chosen the router route-views2.oregon-ix.net. The archive from this router contains data since October of 2001. Specifically, the dataset is obtained in the following way. Zebra has a built-in mechanism to dump BGP data. It can dump its BGP RIB, the full BGP data stream of each peer and various state information. These traces are in the MRT file format. We use zebra-dump-parser to process MRT files.

The Route-Views BGP update files we have been working with corresponds each one for 15 minutes of BGP updates, except the first one which is the complete state of the route table. Each of this update files names follows the pattern: updates.[year][month][day].[hour][minutes].bz for example, updates.20100113.1930.bz2

3.3.2.2 Generated dataset

Using the tool that we have developed (which is described in Section 3.1.3), we have generated a dataset containing data about the stability of the Internet routes. It consists

of a database which main table contains, for each instant of time and prefix, the data about the stability of the BGP selected route for that prefix, the stability of the most stable one, and the number of AS hops for these routes. There is also another table containing the averaged values of the main table, and another one keeping an historic of the BGP selected routes. The name of these tables, and proper description of them can be found in Section ii.

To generate this dataset, we have processed 124 Route-Views files which correspond to the period of time from 00:00 hours of January the first to the 07:00 hours of January the second of 2010. The database is stored in a MySQL server, version MySQL 5.0.51a-24+lenny5 and its size is 68.44 GB, which corresponds to:

- Table route: 68.44 GB
- Table taula: 3.72 MB
- Table selected: 192.85 KB

The tool have been running for 8 days in a machine with the following configuration:

- Processor: Intel(R) Xeon(R) CPU E3110 (3.00GHz)
- Memory: 8GB
- Operating system: Debian GNU/Linux 5.0
- Kernel version: 2.6.26-2-amd64

The table route consists of 1185M entries, which can be grouped into sets according to its time indexes. Each of these sets contains approximately 318K routes.

In order to exemplify the functionality we present a set of plots generated using the data of our dataset. Fig. 3.16 and 3.17 represent the stability difference of the selectable routes versus the selected ones and its variance over the time. Fig. 3.18 and 3.19 do the same but for the selectable routes versus the stablest ones. Finally Fig. 3.20 shows the average difference in number of AS hops of the selected route versus the most stable route.

3.3.2.3 Conclusion

In this section we have described two data-sets. The first one was used to compute the second one. The first one is from the Route-Views project and consists in BGP updates extracted from routers connected to the real Internet. The second one was generated by ourselves and consists in the stability metrics of the routes computed from the Route-Views data-set. This second data-set is stored in a database, split over some different tables. It contains all the routes data and their stabilities and also aggregated versions of this data. Whit this data-set we can see the evolution of the routes stability over the time.

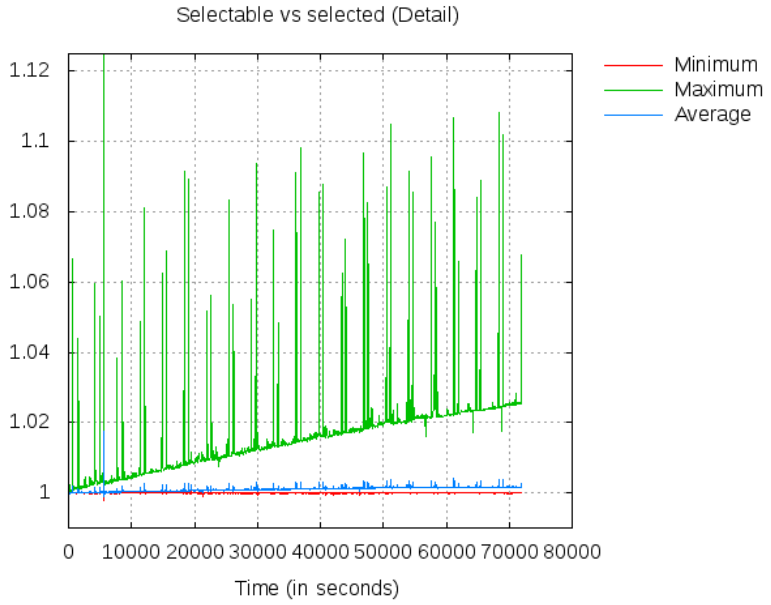


Figure 3.16: Selectable route stability vs selected route stability

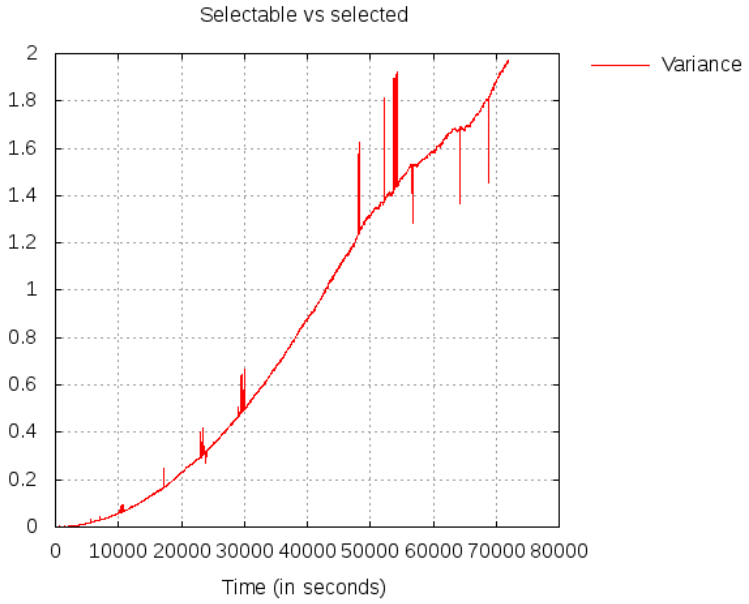


Figure 3.17: Selectable route stability vs selected route stability variance

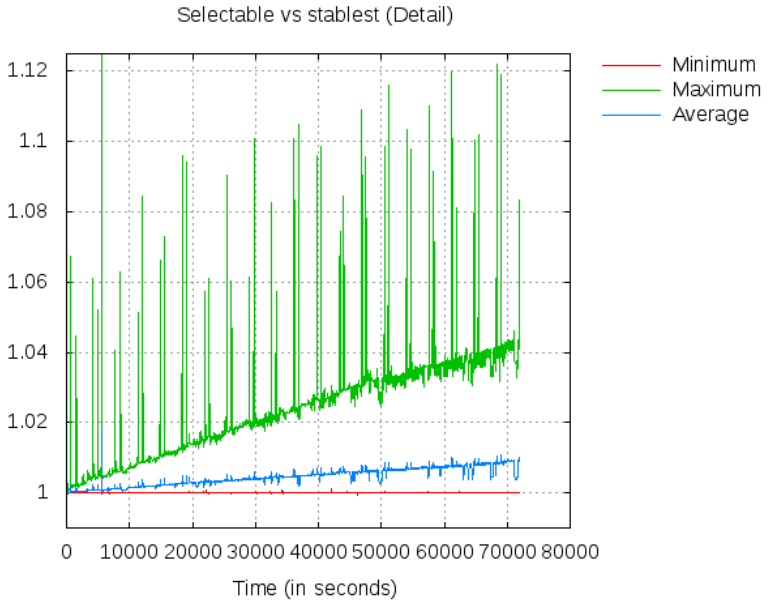


Figure 3.18: Selectable route stability vs most stable route stability

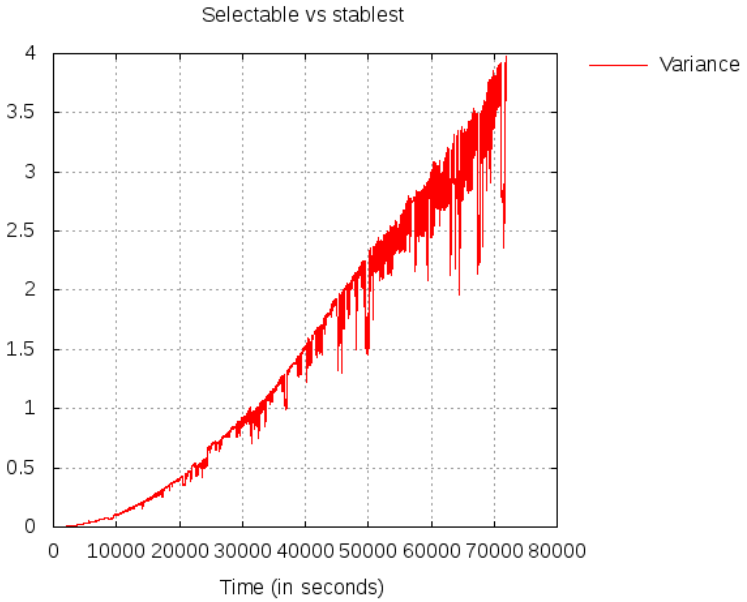


Figure 3.19: Selectable route stability vs selected route stability variance

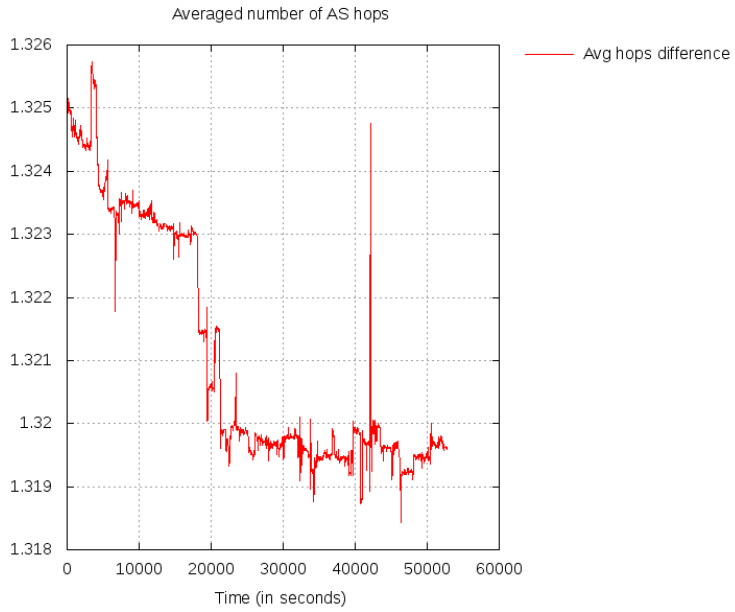


Figure 3.20: Averaged difference in number of AS hops of the stablest route vs the selected one

Chapter 4

Topology and Modeling

4.1 Introduction

As described in Section 2.1 we have developed in this project a method to rigorously measure the node degree of the Internet at the router level. The knowledge of the degree distribution will be helpful by itself to evaluate the likely performances of different routing algorithms, and will also allow us to discriminate between various random graph models that have been, could be or will be proposed to model the Internet graph (e.g., Erdos-Renyi model, configuration model, preferential attachment, etc. as described in deliverable D3.1). Indeed, one expects that a valid model of the Internet topology should at least approximately reproduce its degree distribution.

While we take the degree distribution as the main example for the discussion below, in reason of its availability, importance and its conceptual simplicity, we may want in the future to extend the analysis to other crucial characteristics of networks such as clustering coefficient and degree-degree correlation, possibly the strength-degree correlation, in order to confront the various random graph models of interest with the measured data.

In Section 4.2. We first describe the important example of testing scale-freeness (power law distribution) from an empirical distribution of degrees. Then we suggest methodologies for other properties such as clustering coefficients, etc.

In Section 4.3 we go one step further in suggesting a tentative general theory of modeling networks from partial data, based on statistical learning theory.

4.2 Hypothesis testing

Suppose that we have a random graph model for the Internet, which produces power law degree distributions, and we want to check the validity of this model. One way to proceed is to check whether the actual Internet has a power law degree distribution. In case not, this allows to discard the model, in case of a positive answer, that only validates the model on this specific property of course.

Due to finite size effects of our data sample and the fact that our measurements were made on a finite subset of the Internet nodes, and in view of the complexity of the Internet, we should of course not expect a perfect matching between any theoretical distribution corresponding to a random graph model and the measured distribution.

We intend thus to discriminate different random graph models using hypothesis testing. Concretely, a given class of random graphs parameterized by a vector α, and suppose we want to check whether such a random graph for one value of the vector α

could represent the Internet. We would first compute the vector α of parameters for which the corresponding theoretical degree distribution D_α matches best our measured degree distribution $D_{measured}$, and then test the hypothesis could $D_{measured}$ have been obtained by randomly selecting degrees according to D_α.

We plan thus to design a generic statistical test or method that would work for the theoretical distribution of any given class of random graphs.

4.2.1 Methodology

We intend to test different sorts of statistical tests based on what has been developed in pure statistical theory, on what has been already applied to test hypothesis relative to large networks, and possibly on tests that have been developed and applied in other experimental fields.

For example, one standard statistical method to test the hypothesis that a list of i.i.d. samples were produced by a certain random distribution is the χ^2 test. The data are binned into finitely many categories according to their values (e.g., degrees from 1 to 5, degrees from 6 to 10 and degrees above 10, to take a simplistic example). Then one measures the distance (defined in a certain way) between the predicted probability distribution and observed frequency distribution for the bins. This distance is distributed as a χ^2 if the hypothesis is true, and a p-value assesses the confidence we may have in the truth of the hypothesis. There also exists more sophisticated tests, such as Fisher's exact test.

Another solution, in case of degree distribution, is to follow a specific test that has been designed to test that a given graph follows a power law degree distribution [18]. This test is based on a Monte Carlo method, where many sample distributions are generated from the same theoretical power law distribution, and their distance is compared to the closest ideal power law distribution. Comparing this distribution of distances to the same distance between the actual measured sample and an ideal power law distribution allows to compute an empirical p-value for the power law hypothesis.

Both approaches present challenges, in their theoretical justification and practical usability for various properties beyond degree distribution and various graph models. Moreover, almost any method that we select will need to be adapted or re-evaluated due to specific features of the data that we treat, such as the absence of strict independence between the degrees of the nodes (see below).

4.2.2 Independence of the degrees

Most hypothesis testing methods consider a realization $d_1, \ldots, d_n$ of a sequence of independent identically distributed random variables, and test the hypothesis the (common) distribution is D. However, we are not interested in a sequence of independent variables, but a sequence of degrees in a single realization of a random graph. One can prove that the independence assumption never formally holds in that case, for example because the sum of the degrees is always an even number. This particular dependence is unlikely to be problematic, but one can easily construct cooked-up random graph models for which the independence assumption is strongly violated.

For this purpose, we will thus need to

1. determine if random graph models that reasonably model the Internet may strongly violate the independence assumption;

2. evaluate or approximate how such violations may affect the hypothesis tests;

3. design alternative tests taking into account the possible dependencies if needed.

Let us delve a bit deeper into what formal concept can replace the concept of strict independence. Since we cannot hope that the degrees of a same instance are perfectly independent, we can however hope that the average degree across one sample of the graph matches the expected degree from the ensemble, if exists (almost surely when the number of nodes increases to infinity). Similarly, we may require that other observables computed from the degree (such as the square of the degree) may be deduced by averaging from one sample. In other words, we need to have an 'ergodic' random graph model, where ensemble averages match the average over a single sample, with high probability.

4.2.3 Number of nodes

Almost all statistical tests rely on the number of samples that have been measured. However measuring the degree distribution was a nontrivial process (see Section 2.1) that required many post-processing steps, so that the measured distribution does not really correspond to a given number of independent samples. We will thus need to find the best equivalent number of samples, and to verify if the post-processing and measurement method create any dependence problem.

4.3 Statistical Learning Theory

It would be desirable to have a general theory on how to exploit the empirical data from graph. In the above, we had only one sample graph from which we had to deduce (e.g. through counting the frequencies of degrees on that sample) an information about the random model that generated that sample (e.g., degree distribution). This is a very common situation in complex network theory, yet a very uncommon situation for the general theory, which usually assumes many samples drawn in a i.i.d. manner.

Another example is the following. Network events (such as studied in Section 2.3) may be seen as a temporal sequence of random events occurring on a fixed or evolving networks, from which we want to infer properties for the underlying random process generating those events and this graph. The general field stating how an algorithm can learn properties of the hidden theoretical distribution from samples is statistical learning theory. Adopting the core of statistical learning theory to the constraints of data collection is an interesting prospective line of research.

We present an overview of statistical learning theory together with the rationale and motivations in the context of our study interests. Statistical learning theory (SLT) is a theoretical branch of machine learning which attempts to explain the theory of (empirical) learning process from a statistical point of view. This theory provides the theoretical basis for many of today's machine learning algorithms. The fundamental questions asked by SLT are:

- Which learning tasks can be performed by computers in general (positive and negative results)?

- What kind of assumptions do we have to make such that machine learning can be successful?

- What are the key properties a learning algorithm needs to satisfy in order to be successful?
- Which performance guarantees can we give on the results of certain learning algorithms?

As explained in The Nature of Statistical Learning Theory [**?**], learning theory has to address the following four questions.

- What are (necessary and sufficient) conditions for consistency of a learning process based on the empirical risk minimization (ERM) principle?
- How fast does the sequence of smallest empirical risk values converge to the smallest actual risk? In other words what is the rate of generalization of a learning machine that implements the empirical risk minimization principle?
- How can one control the rate of convergence (the rate of generalization) of the learning machine?
- How can one construct algorithms that can control the rate of generalization?

The answers to these questions form the **four parts** of learning theory:

Part.1: The theory of consistency (asymptotic theory) of learning processes: describes the necessary and sufficient conditions for convergence of the solutions obtained using the proposed method to the best possible as the number of observations is increased. The following question arises: *Why do we need a theory of consistency if our goal is to construct algorithms for a small (finite) sample size?*. The answer is: *We need a theory of consistency because it provides not only sufficient but necessary conditions for convergence of the empirical risk minimization inductive principle.* Therefore any theory of the empirical risk minimization principle must satisfy the necessary and sufficient conditions. This part of the theory which relies on *Key Theorem of the Learning Theory concerning the Empirical Risk Minimization (ERM)-based learning processes* introduces the main capacity concept –the so-called Vapnik-Chervonenkis (VC) entropy– which defines the generalization ability of the ERM principle. It leads to the so-called *Three Milestones in Learning Theory* which form a foundation for constructing both distribution independent bounds and rigorous distribution dependent bounds for the rate of convergence of learning machines. The *Three Milestones in Learning Theory* consists of a set of equations describing:

1. The necessary and sufficient condition for consistency of the ERM principle (any machine minimizing empirical risk should satisfy it).
2. The sufficient condition for fast convergence that guarantees a fast asymptotic rate of convergence.
3. The necessary and sufficient conditions under which the learning machine implementing ERM principle has an asymptotic high rate of convergence independently of the problem to be solved (independently of the probability measure).

Part.2: The non-asymptotic theory of the rate of convergence of learning processes: In order to estimate the quality of the ERM method for a given sample size it is necessary

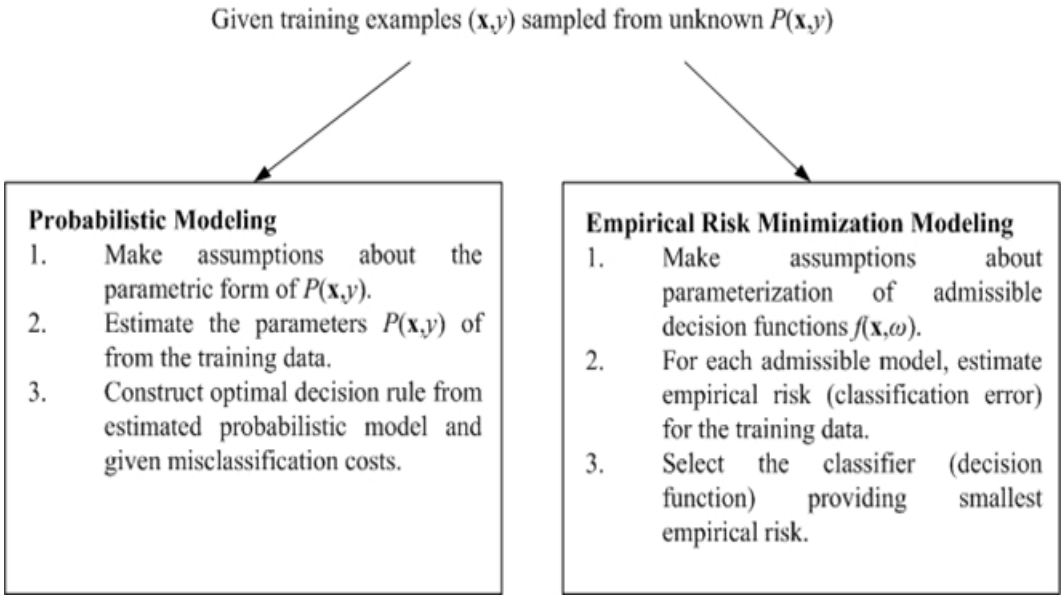


Figure 4.1: Probabilistic vs Empirical Risk Minimization (ERM)-based Modeling

to obtain non-asymptotic bounds on the rate of uniform convergence. A non-asymptotic bound of the rate of convergence can be obtained using a capacity concept, called the VC-dimension, which allows us to obtain a constructive bound for the growth function. The finiteness of the VC-dimension of the set of indicator functions (functions which take on only two values zero and one) implemented by the learning machine forms the necessary and sufficient condition for consistency of the ERM method independent of probability measure. Finiteness of VC-dimension also implies fast convergence.

Part.3: The theory of controlling the generalization of learning processes: The theory for controlling the generalization of a learning machine is devoted to constructing an induction principle for minimizing the risk functional which takes into account the size of the training set (an induction principle for a small sample size). The goal is to specify methods which are appropriate for a given sample size. Structural Risk Minimization (SRM) Induction Principle: is intended to minimize the risk functional with respect to both empirical risk and VC-dimension of the set of functions. The SRM principle actually suggests a tradeoff between the quality of the approximation and the complexity of the approximating function (formal mechanism for controlling model complexity). The main result of the SRM theory which takes both factors into account is the following theorem *For any distribution function the SRM method provides convergence to the best possible solution with probability one.* In other words SRM method is universally strongly consistent. To implement the SRM induction principle in learning algorithms one has to control two factors: 1) the value of empirical risk, and 2) the capacity factor.

Part.4: The theory of constructing learning algorithms: constructive methods (learning algorithms) for implementing (Part.3).

Note on VC Dimension: Vapnik-Chervonenkis dimension, or VC dimension, is a measure of capacity of a statistical classification hypothesis. The VC-dimension of a

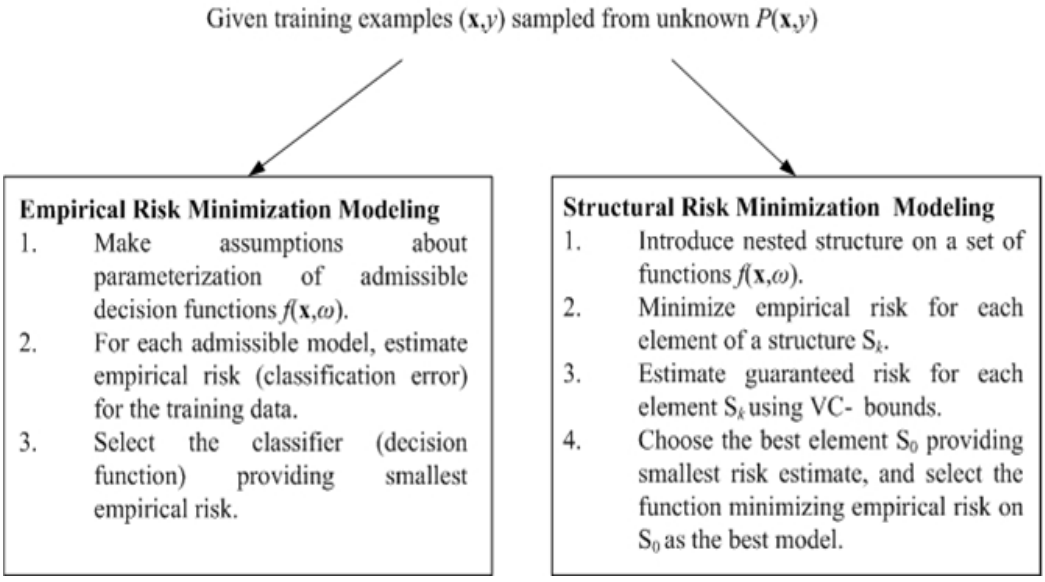


Figure 4.2: Empirical Risk Minimization (ERM)- vs Structural Risk Minimization (SRM)-based modeling.

system is thus defined as the cardinality of the largest set of points that the hypothesis can shatter. A classification model f with some parameter vector θ is said to shatter a set of data points $(x_1, x_2, \ldots, x_n)$ if, for all assignments of labels to those points, there exists a θ such that the model f makes no errors when evaluating that set of data points. The VC dimension of a model f is h', where h' is the maximum value of h such that some data point set of cardinality h can be shattered by f.

Chapter 5

Conclusion

In this deliverable, we have detailed the important breakthroughs performed within the EULER project about measurement-based modeling of the Internet topology, including its dynamical properties:

1. We have provided the first unbiased experimentally measured distribution of degrees of the core Internet at the physical level.
2. We have developed a first model of the Internet dynamics that explains key non-trivial experimental observations.
3. We have provided a first analytical method to analyze the stability of BGP routing paths using BGP traces.
4. We have developed a general method for event detection in graph dynamics, and applied it to the Internet via ego-centered measures.
5. We are developing a first (albeit partial since only based on peer-to-peer traffic) model of Internet traffic based on experimental measurements.

We are now in the process of developing statistical tools to formally test the conclusion that we can derive from these measurements and methods developed.

Our results do not provide yet a complete realistic model of the Internet and its dynamics, which would be the ultimate goal of the project, but already constitute an important progress over the previous state-of-the-art. Moreover, for several key properties, they constitute for the moment the most accurate available description of the real Internet. As such, they provide readily available answers and data for everyone wishing to perform realistic simulations of the Internet, or analyze its properties.

In particular, in the context of the EULER project, the data gathered will have direct applications in Task 4.2 for the design of realistic scenarios on which the different routing schemes will be tested. Our contribution will also allow selecting appropriate models of the Internet among those studied in the Task 3.1 by eliminating those models whose properties do not correspond to our experimental measurements. Finally, the observations made here will also help us designing efficiently new routing schemes in the task 2.2, as they provides a new light on the evolving topology on which these schemes will be deployed.

Perspectives

We have developed dedicated methods to measure several properties of the Internet. Methods could also be developed to measure many other properties in the future. On the long run, and given the cost of developing these experiments, this will lead to the question of knowing which are the relevant or most useful properties to measure. In other words, what are the experiments that would provide the maximal amount of information?

This problem can be approached under the angle of parameter identification: Suppose that we had a generic model of Internet-like networks $G(n, \alpha)$ for some vector of parameters α. Selecting the measurements to make would then become an experimental design issue, in which one would need to design optimal, or at least sufficient, experiments in order to identify the parameters α. Obtaining such a generic model containing networks "close" to the real Internet would therefore represent a major theoretical breakthrough, providing at the same time intuition and explanation about the Internet, and way of choosing the best future experiments.

Finding suitable generic models remains however a very challenging issues, and will requiring comparing the results of many experiments. In this context, it could be particularly interesting to analyze the correlations between the different properties that have been measured. We are for example currently studying the possibility of measuring the link between the dynamics of the Internet topology at the IP level and the stability of the routing paths.

Bibliography

[1] Caida - skitter project. *http://www.caida.org/tools/measurement/skitter/*.

[2] Quagga routing software. gpl licensed ipv4/ipv6 routing software. http://www.quagga.net.

[3] Sandvine global Internet phenomena report spring 2011. http://www.sandvine.com/news/pr_detail.asp.

[4] Supplementary material, including programs and data. `http://www-rp.lip6.fr/~latapy/Radar/`.

[5] macroscopic topology measurement project. *http://www.caida.org/projects/macroscopic/*, 2010.

[6] D. Achlioptas, A. Clauset, D. Kempe, and C. Moore. On the bias of traceroute sampling: or, power-law degree distributions in regular graphs. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (STOC)*, pages 694–703. ACM, 2005.

[7] E. Adar, L. Zhang, L. A. Adamic, and R. M. Lukose. Implicit structure and the dynamics of blogspace. In *World Wide Web Conference Series*, New York, NY, 2004.

[8] F. Aidouni, M. Latapy, and C. Magnien. Ten weeks in the life of an edonkey server. In *IEEE IPDPS*, pages 1–5, Rome, Italy, 2009.

[9] L. Akoglu and C. Faloutsos. Event detection in time series of mobile communication graphs. In *27th Army Science Conference, December 2010a*, volume 2, page 18.

[10] R. Albert, H. Jeong, and A.L. Barabasi. Error and attack tolerance of complex networks. *Nature*, 406(6794):378–382, 2000.

[11] A. Barrat, M. Barthelemy, and A. Vespignani. Dynamical processes on complex networks. *Proc. Roy. Soc. Lond.*, A 115:700–721, 2008.

[12] R. Bolla, M. Eickhoff, K. Pawlikowski, and M. Sciuto. Modeling file popularity in peer-to-peer file sharing systems. In *Proceedings of the* 14^{th} *ASMTA 2007*, pages 1–5, Prague, Czech Republic, 4-6 June 2007.

[13] D.S. Callaway, M.E.J. Newman, S.H. Strogatz, and D.J. Watts. Network robustness and fragility: Percolation on random graphs. *Physical Review Letters*, 85(25):5468–5471, 2000.

[14] E. W. Candiff and R. R. Still. Basic marketing, concepts. *Decisions and Strategies. Second edition, Prentice Hall, Inc., Englewood Cliffs*, 1971.

[15] H. Chang, S. Jamin, and W. Willinger. To peer or not to peer: Modeling the evolution of the Internet?s as-level topology. *Ann Arbor*, 1001:48109–2122, 2006.

[16] Q. Chen, H. Chang, R. Govindan, and S. Jamin. The origin of power laws in Internet topologies revisited. In *Proc. IEEE INFOCOM*, volume 2, pages 608–617. IEEE, 2002.

[17] D.R. Choffnes. *Service-Level Network Event Detection from Edge Systems.* PhD thesis, Northwestern University, 2010.

[18] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51:661–703, 2009.

[19] R. Cohen, K. Erez, D. Ben-Avraham, and S. Havlin. Resilience of the Internet to random breakdowns. *Physical Review Letters*, 85(21):4626–4628, 2000.

[20] R. Cohen, K. Erez, D. Ben-Avraham, and S. Havlin. Breakdown of the Internet under intentional attack. *Physical Review Letters*, 86(16):3682–3685, 2001.

[21] J.-P. Cointet and C. Roth. How realistic should knowledge diffusion models be? *Journal of Artificial Societies and Social Simulation*, 10(3):3, 2007.

[22] C. Crespelle and F. Tarissan. Evaluation of a new method for measuring the Internet degree distribution: Simulation results. *Computer Communications*, 2010.

[23] I. Cunha, R. Teixeira, and C. Diot. Measuring and characterizing end-to-end route dynamics in the presence of load balancing. In *Passive and Active Measurement*, pages 235–244. Springer, 2011.

[24] L. Dall'Asta, I. Alvarez-Hamelin, A. Barrat, A. Vázquez, and A. Vespignani. A statistical approach to the traceroute-like exploration of networks: theory and simulations. *Arxiv preprint cond-mat/0406404*, 2004.

[25] L. Dall'Asta, I. Alvarez-Hamelin, A. Barrat, A. Vázquez, and A. Vespignani. Exploring networks with traceroute-like probes: Theory and simulations. *Theoretical Computer Science*, 355(1):6–24, 2006.

[26] B. Donnet, P. Raoult, and T. Friedman. Efficient route tracing from a single source. cs.NI 0605133, arXiv, May 2006.

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

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

[29] R.E.A. Eimann. *Network event detection with entropy measures.* PhD thesis, The University of Auckland, 2008.

[30] P. Erdos and A. Renyi. On random graphs, i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[31] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. In *ACM SIGCOMM Computer Communication Review*, volume 29, pages 251–262. ACM, 1999.

[32] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda. Mawilab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proceedings of the 6th International COnference*, page 8. ACM, 2010.

[33] Z. Ge, D. R. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley. Modeling peer-peer file sharing systems. In *IEEE INFOCOM 2003*, San Francisco, California, 2003.

[34] R. Govindan and A. Reddy. An analysis of Internet inter-domain topology and route stability. In *INFOCOM'97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 850–857. IEEE, 1997.

[35] Ramesh Govindan and Vern Paxson. Estimating router ICMP generation delays. In *Proc. of Passive and Active Measurement Workshop*, March 2002.

[36] T.G. Griffin, F.B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking (ToN)*, 10(2):232–243, 2002.

[37] J.L. Guillaume and M. Latapy. Relevance of massively distributed explorations of the Internet topology: Simulation results. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 1084–1094. IEEE, 2005.

[38] J.L. Guillaume, M. Latapy, and D. Magoni. A radar for the Internet, publicly available datasets. *http://data.complexnetworks.fr/Radar/*.

[39] J.L. Guillaume, M. Latapy, and D. Magoni. Relevance of massively distributed explorations of the Internet topology: Qualitative results. *Computer networks*, 50(16):3197–3224, 2006.

[40] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP-2003)*, Bolton Landing, NY, October 2003.

[41] J. Gupchup, A. Terzis, R. Burns, and A. Szalay. Model-based event detection in wireless sensor networks. *Arxiv preprint arXiv:0901.3923*, 2009.

[42] A. Hamzaoui, M. Latapy, and C. Magnien. Detecting events in the dynamics of ego-centered measurements of the Internet topology. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2010 Proceedings of the 8th International Symposium on*, pages 505–512. IEEE, 2010.

[43] T. Hossfeld, K. Leibnitz, R. Pries, K. Tutschku, P. Tran-Gia, and K. Pawlikowski. Information diffusion in edonkey filesharing networks. In *ATNAC 2004*, pages 1–5, Sydney, Australia, 2004.

[44] G. Huston. Damping BGP. *RIPE 55, Routing Working Group*, 2007.

[45] J. L. Iribarren and E. Moro. Impact of human activity patterns on the dynamics of information diffusion. *Physical Review Letters*, 103(3):038702, 2009.

[46] W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proc. Roy. Soc. Lond.*, A 115:700–721, 1927.

[47] A.M. Kuatse, R. Teixeira, and M. Meulle. Characterizing network events and their impact on routing. In *Proceedings of the 2007 ACM CoNEXT conference*, page 59. ACM, 2007.

[48] Y. Kulbak and D. Bickson. The emule protocol specification. *eMule project, http://sourceforge. net*, 2005.

[49] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet routing convergence. In *ACM SIGCOMM Computer Communication Review*, volume 30, pages 175–187. ACM, 2000.

[50] C. Labovitz, G.R. Malan, and F. Jahanian. Origins of Internet routing instability. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 218–226. IEEE, 1999.

[51] A. Lakhina, J.W. Byers, M. Crovella, and P. Xie. Sampling biases in ip topology measurements. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 1, pages 332–341. IEEE, 2003.

[52] M. Latapy and C. Magnien. Complex network measurements: Estimating the relevance of observed properties. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1660–1668. IEEE, 2008.

[53] M. Latapy, C. Magnien, and F. Ouédraogo. A radar for the Internet. In *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*, pages 901–908. IEEE, 2008.

[54] H. Levin, M. Schapira, and A. Zohar. Interdomain routing and games. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 57–66. ACM, 2008.

[55] C. Magnien, F. Ouédraogo, G. Valadon, and M. Latapy. Fast dynamics in Internet topology: Observations and first explanations. In *Internet Monitoring and Protection, 2009. ICIMP'09. Fourth International Conference on*, pages 137–142. IEEE, 2009.

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

[57] R.V. Oliveira, B. Zhang, and L. Zhang. Observing the evolution of Internet as topology. *ACM SIGCOMM Computer Communication Review*, 37(4):313–324, 2007.

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

[59] J.J. Pansiot and D. Grad. On routes and multicast trees in the Internet. *ACM SIGCOMM Computer Communication Review*, 28(1):41–50, 1998.

[60] P. Papadimitriou and D. Lowe. Routing system stability. http://tools.ietf.org/html/draft-ietf-grow-rss-00, 2009.

[61] S.T. Park, A. Khrabrov, D.M. Pennock, S. Lawrence, C.L. Giles, and L.H. Ungar. Static and dynamic analysis of the Internet's susceptibility to faults and attacks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 2144–2154. IEEE, 2003.

[62] S.T. Park, D.M. Pennock, and C.L. Giles. Comparing static and dynamic measurements and models of the Internet's as topology. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1616–1627. IEEE, 2004.

[63] V. Paxson. End-to-end routing behavior in the Internet. In *ACM SIGCOMM Computer Communication Review*, volume 26, pages 25–38. ACM, 1996.

[64] PlanetLab Consortium. PlanetLab project, 2002. See `http://www.planet-lab.org`.

[65] Y. Schwartz, Y. Shavitt, and U. Weinsberg. On the diversity, stability and symmetry of end-to-end Internet routes. In *INFOCOM IEEE Conference on Computer Communications Workshops, 2010*, pages 1–6. IEEE, 2010.

[66] Y. Shavitt and E. Shir. DIMES: Let the Internet measure itself. *ACM SIGCOMM Computer Communication Review*, 35(5):71 – 74, October 2005.

[67] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A public Internet measurement facility. In *Proc. 4th USENIX Symposium on Internet Technologies and Systems*, 2002. see also `http://www.cs.washington.edu/research/networking/scriptroute/`.

[68] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger. Sampling techniques for large, dynamic graphs. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–6. IEEE, 2006.

[69] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger. On unbiased sampling for unstructured peer-to-peer networks. *IEEE/ACM Transactions on Networking (TON)*, 17(2):377–390, 2009.

[70] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network topology generators: Degree-based vs. structural. *ACM SIGCOMM Computer Communication Review*, 32(4):147–159, 2002.

[71] M. Thottan and C. Ji. Anomaly detection in ip networks. *Signal Processing, IEEE Transactions on*, 51(8):2191–2204, 2003.

[72] F. Viger, B. Augustin, X. Cuvellier, C. Magnien, M. Latapy, T. Friedman, and R. Teixeira. Detection, understanding, and prevention of traceroute measurement artifacts. *Computer Networks*, 52(5):998–1018, 2008.

[73] X. Wang and D. Loguinov. Wealth-based evolution model for the Internet as-level topology. In *Proceedings of IEEE Infocom*, 2006.

[74] W. Willinger, D. Alderson, and J.C. Doyle. *Mathematics and the Internet: A source of enormous confusion and great potential.* Defense Technical Information Center, 2009.

[75] S. Zhao, D. Stutzbach, and R. Rejaie. Characterizing files in the modern gnutella network: A measurement study. In *Proceedings of SPIE/ACM Multimedia Computing and Networking (MMCN 2006)*, San Jose, CA, October 2003.