Université Nice SOPHIA ANTIPOLIS

MASTER II IFI (Informatique : Fondements et Ingenierie)

Parcours UBINET - Ubiquitous Networking and Computing

Masters Program Internship Report, 2011

Monitoring the Quality of Internet Access by Active Probing

Student : **JOST, Mauricio**

Reviewer : **LIQUORI, Luigi**

Advisor : **BARAKAT, Chadi**

Team : **PLANETE**

Period: **March 1st, 2011 - August 31st, 2011**

Abstract

In the present work we aim to improve and study the tool ACQUA[1], that reveals to us the Impact Factor Estimation (IFE), where the Impact Factor is an index that tells how important is the impact of a network anomaly in the connectivity perceived from the point of view of a user. We define a new anomaly detection method used to get the IFE, we then give to ACQUA a new pipeline based software architecture, and we go deeply into experimentation inside and outside Planetlab. Then we show what the properties and usage of the algorithm are, focusing also on how this tool can help us to get information about the network anomalies detected. Later we extend the idea of IFE by using what we call Inverse IFE from Planetlab, where the computer of the user whose connectivity is tested has a completely passive role in the measurements procedure. We study its strong and weak points, and we show conditions under which Inverse IFE from Planetlab gives similar results to traditional IFE.

[1] http://planete.inria.fr/acqua/

Contents

Chapter 1

Introduction

In the present work we aim to improve the overall behavior of the network anomaly detection algorithm that the tool ACQUA uses to give an Impact Factor Estimation (IFE), where Impact Factor is an index that tells how important is the impact of a network anomaly, from the point of view of a user.

We change deeply the software architecture of the tool by using as a pattern a pipeline based design. Thanks to this we can easily connect and disconnect or replace pipeline elements to the ACQUA core, which allows to have high flexibility and test different algorithms in every stage of the detection without affecting the whole software project. We then implement a measurements element that lets us generate screen-shots of the network much faster than before by using concurrency techniques. This allows us to technically do better experiments, with more probing capacity and small period of measurements.

Then we study one of the stages of such IFE calculation algorithm, which involves simple Path Anomaly Detection (not involving the whole network). We reconsider this part and its original implementation, we show some weak points of it, and we do a deep analysis of two new different Path Anomaly Detection algorithms to use in ACQUA, in order to give a better IFE result. We show the results of several simulations and we make a decision about the next Path Anomaly Detection algorithm to be embedded in ACQUA.

Once we make a decision, we implement this new algorithm as a pipeline element.

Then we run the tool in several nodes of Planetlab, we show some properties of the tool that match with the theory described in [1], and we show some cases where the information provided can help us to easily have an idea of the source of the network anomaly.

Later we extend the idea of IFE by using what we call Inverse IFE, where the computer of the user whose connectivity is tested (Monitored Point) has a completely passive role in the measurements procedure. We discuss about its strong and weak points, and we show some results obtained thanks to this new feature added to ACQUA.

1.1 Objectives

The main objectives of this work are the following:

- Give a new pipeline based software architecture to ACQUA, to let break down the IFE calculation and do experiments replacing elements in the pipeline and comparing results.
- Re-implement the measurements module to allow faster network data acquisition.
- Study the current Path Anomaly Detection algorithm, determine whether this is suitable and if it is not, propose a new alternative, study it and implement it in the new core of

ACQUA.

- Test the current implementation, see how well it behaves when detecting anomalies, determine how sensible are the IFE results to the amount of landmarks used, and to the landmarks themselves being used in comparison with other landmarks.
- Study feasibility of Inverse IFE from Planetlab.

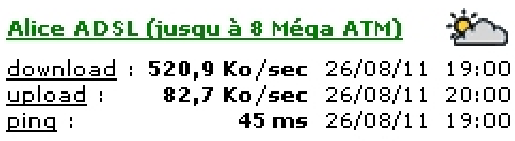
Chapter 2

State of the Art and ACQUA

2.1 Context and Motivation

The main purpose of this whole project is to contribute with the Grenouille [19], an already existent project that measures the *weather* of the Internet, classifying Internet Service Providers (ISP) as if they were geographical zones. We contribute by inserting a new metric to this *weather* notion. Currently they have what we can see in Figure 2.1.

The performance of each ISP has a big impact on the general system. An ISP might be the intermediate Autonomous System (AS) to reach certain destination, or even more, it might be network in which the destination or the source is. ISPs are the blocks that form Internet. That is why in Grenouille project they focus so much on them.



```
Alice ADSL (jusqu à 8 Méga ATM)

download :  520,9 Ko/sec  26/08/11  19:00
upload :     82,7 Ko/sec  26/08/11  20:00
ping :              45 ms  26/08/11  19:00
```

Figure 2.1: *Grenouille project, example of the information given.*

Even though the provided information is good enough to give us an idea of how good is performing one ISP currently, it would be interesting to introduce a metric that describes what the impact of an anomaly (in certain ISP) from the user's point of view is. This clearly involves the topology of the network, so it is more complex, but for sure it adds information that is not negligible when measuring the network performance from a particular users' point of view.

This chapter is mainly related to ACQUA, the tool that performs Impact Factor Estimation, the indicator we are looking for. Before explaining any other topic, we will start addressing what is the *Impact Factor* [1].

2.2 Impact Factor

In the current work we improve the Impact Factor Estimation algorithm given in ACQUA. But, *what is it?* The Impact Factor (defined formally in [1]) represents the seriousness of an anomaly in the network. The Impact Factor (or IF) depends on the Monitored Point that is analyzed, two different users may clearly have a different impact in their access to Internet even for the same network anomaly (think of a failure in the link that directly connects a user with Internet). More precisely, the Impact Factor is defined in terms of the percentage of all possible network destinations that will manifest a service degradation due to this anomaly. We refer to these network destinations as *landmarks*, and we refer to the end-user node (for which we monitor the Impact Factor) as *Monitored Point*.

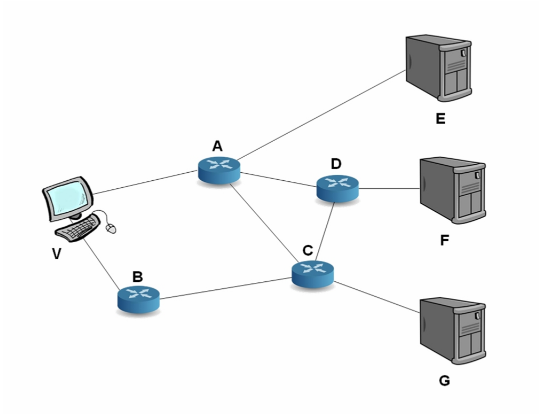


Figure 2.2: *Impact Factor Definition.*

In the Figure 2.2 the Monitored Point is V, and the landmarks are nodes E, F and G. One failure in link A-D might impact 1/3 of all the landmarks E, F and G, thus leading to an Impact Factor in V of 1/3 (this case happens if the link A-D is used by the Monitored Point V to reach only F). A failure in A-D might also lead to an Impact Factor of 2/3, if A-D is used to reach both F and G. It also might result in Impact Factor equals to 0, when A-D is not being used by any of the current routes of V towards the landmarks. The Impact Factor depends on V clearly, and it ranges between 0 and 1, where 0 means that no destination is affected by the anomaly, and 1 means that 100% of all the possible destinations will suffer a performance degradation due to this anomaly.

The definition of *anomaly* in the path towards certain landmark will be seen in next chapters. But to give an idea, we ping each landmarks periodically to test it, so an anomaly might represent a non-reachability situation towards the given landmark, or more frequently a big variation of the RTT towards it (which may happen because a router in the route failed and hence a different path was assigned in the network to reach the landmark).

In [1] it is shown the minimum amount of probing destinations to analyze in order to have a good estimator of the Impact Factor as a function of the confidence interval and the significance level that are required for the estimator. Theoretically, when the Impact Factor value is completely unpredictable and it might fall in any part of the range 0 to 1, the amount of landmarks required (probe destinations) is quite large. But in fact, in the paper it is mentioned that in practice generally two things may happen: some close link may fail resulting in a close to 1 Impact Factor value (suppose V-A is used to reach the three destinations in Figure 2.2). Otherwise a far/medium distance link may fail, and this is likely to have an impact in fewer than half of the landmarks. So, in most of the cases for the real topology of Internet we will be facing Impact Factors close either to 0 or to 1, but generally not in the middle. This implies that the amount of landmarks required to have a good estimator is reduced to just a few, which makes this approach feasible in practice.

The original tool is named ACQUA (we will call it also IFE tool, IFE stands for Impact Factor Estimator). This obtains an estimator of the Impact Factor by probing actively several landmarks by means of ICMP echo packets. Then for each landmark, we get a sequence in time of Round Trip Times, which is analyzed in real time to detect anomalies. From the anomalies of all paths we estimate how many general destinations will be affected by the general anomaly. To estimate the percentage of destinations affected we take advantage of the fact that few paths to the destinations are way more significant than others when it comes to general reachability, so that they give information enough to have a good approximation to the Impact Factor.

2.3 Current Software Architecture

As we said we work with a tool named ACQUA. It stands for Application for Collaborative estimation of QUality of internet Access. This is the module in charge of performing measurements and process their results to compute the Impact Factor Estimation that the Monitored Point perceives. The core of ACQUA was originally written in pure C. In order to let non-research users use the application it was necessary to provide some more features.

The next step performed during the PFE was to build a Graphical User Interface to provide an easier way to address the usage of the tool. This was done by adding to the original core a front-end implemented in Java. The interaction between both layers was carried out using the output files of the core of ACQUA as input for the new additional layer.

However, there were some issues with this architecture:

- Since the core was implemented in C, once compiled this binary was not portable, even though the front-end Java layer was. Originally this module worked by parsing the output of the *ping* tool from Unix based systems. It means that systems not having the same *ping* application could not run the tool even if the core's binary worked.
- As it will be detailed in the Section 3.1, the *Path Anomaly Detection* method originally used was not convenient. Since the ACQUA's core was implemented in a monolithic way, modifying the method cleanly would have not been an easy work.
- The input for the core is taken from *ping* processes, executed all with different parameters. Originally each *ping* process was executed back-to-back. Considering that normally the amount of landmarks to ping is at least 20, and the number of pings to do towards each is around 3, we have to wait for at least 60 *ping* processes sequentially executed, and considering that each ping might take up to 3 seconds (timeout case) every campaign would last for at most 180 seconds. This clearly represents a serious limitation in the frequency of the campaigns, it would be good to complete a campaign every minute for instance. An easy way of addressing this problem is by executing parallelly the *ping* processes. However, the architecture of the core was not thought to do so.

For these reasons it was considered convenient to port the complete core to Java, resulting the whole ACQUA implemented in the same language. But more than benefits like portability, or Object Oriented Programming approach, we pursued a pipeline based architecture, which will be explained now.

2.4 New Pipeline based Architecture

While porting the application, a new suitable design took place. It can be seen in the Figure 2.3. The main idea of such design is to let new modules (or pipeline elements) be plugged with no strict need of changing other modules because of only software issues. The only point to take care about when connecting a new module, should be related to dependencies inherent to the logic of the module itself with regards to the other modules already forming part of the pipeline.

The list of some important elements that were implemented to behave as the core of ACQUA are the following:

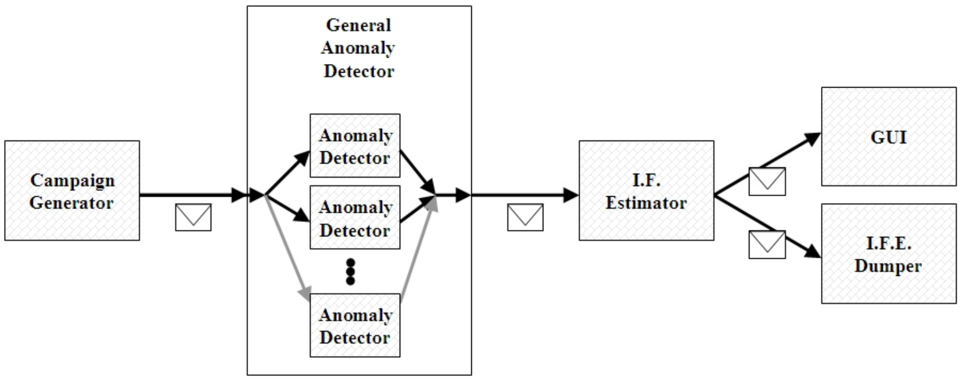


Figure 2.3: *Pipeline based architecture.*

- **Campaign Generator element.** It generates campaigns of measurements, it means that it pings different landmarks sequentially and creates a campaign, a set of RTT values towards all the landmarks in a specific moment.
- **Campaign Generator 2G element.** Same as previous, but it pings landmarks concurrently.
- **Dump Reader element.** Reads dump files generated by Campaign Generator elements. Useful to work offline with the measurements.
- **Inverse Dump Reader element.** Reads dump files generated by Campaign Generator elements in Inverse mode. Inverse IFE will be explained later.
- **Raw Histogram Anomaly Detector element.** Detects anomalies towards one landmark using the original Anomaly Detection method, except for the fact that it uses a histogram to estimate the real distribution of the RTTs.
- **Bernoulli Anomaly Detector element.** Detects anomalies towards one landmark using Bernoulli based method.
- **Likelihood Anomaly Detector element.** Detects anomalies towards one landmark using Likelyhood based method.
- **General Anomaly Detector element.** Wraps all the Anomaly Detector elements, and gathers their results to give it to the next element.
- **Impact Factor Estimator element.** It uses all the results of the elements Anomaly Detector, and calculates the IFE.
- **Impact Factor Estimator Dumper element.** It dumps the results of the IFE element to a file for later analysis.

Pings are done to several landmarks using the Campaign Generator, then the information about the RTT obtained flows towards the General Anomaly Detector module, that scatters this information to several Anomaly Detector modules, whose main responsibility is to detect anomalies in the path to one particular landmarks. The General Anomaly Detector gathers all these results, and sends them to the next element in the pipeline, the Impact Factor Estimator.

It will process the results and send them to either an element to show results to the user, or to an element to dump all the results to a file for later usage.

Note that if we want to use a dumped file produced by the Campaign Generator, we just replace at the beginning of the pipeline the element Campaign Generator for the element Dump Reader and all the pipeline will process the information obtained before.

Immediately after having this new architecture, we found fantastic advantages. The original IFE algorithm had a campaign generator module performing ping measurements towards several landmarks using back-to-back pings, i.e. sequential pings. It means that for a case of only 10 landmarks, 3 pings to each, and 3 seconds of timeout for each ping, we have to wait 90 seconds, not to generate measurements at a faster frequency. However we would like to use around 30 landmarks, 4 pings to each and 3 seconds of timeout (worst case of 6 minutes per campaign, too slow). So we implemented an improved module (or pipeline element) named Campaign Generator 2G element, that pings concurrently all the landmarks. It uses random generated times to trigger each ping, so that their measurements will not be affected by a burst of ICMP packets. With no problem we executed 68 landmarks with 4 pings per campaign, with 1 minute period. Mainly the amount of landmarks improves a lot the accuracy of the overall results, as described in [1].

Chapter 3

Path Anomaly Detection Method

The Impact Factor Estimation works using as input anomaly detections performed on paths towards several landmarks. The input of this method is a sequence of RTT to one landmark, and our aim is to determine when an anomaly is going on, whatever is the definition of anomaly we are interested in. There are several methods to perform single path anomaly detection, the one used at the beginning of the implementation of ACQUA is an example.

3.1 Need of a new Method

Once the new architecture/implementation was finished, some points about the original Path Anomaly Detection method were discussed. Before introducing this points we will explain a little bit about it.

3.1.1 Original Method

The original path anomaly detection method used by ACQUA cumulated a set of RTT measurements (obtained from last N campaigns). Once these N RTT values are available, the mean of all of them was calculated, and also a threshold value based on two inputs: the N RTT values and a significance level value. This significance level gives an idea related to how sure we want to be that what is going on now in the measurements is not normal.

A set of experiments where launched to check how the original Anomaly Detection algorithm was performing. There were some uncertain points on top of which this it was built:

1. RTT variable for one path follows a T-student distribution.

2. A timeout ping event is replaced by the timeout time, so we never know certainly when a landmark is reachable or not at all.

We went through each of them to analyze their impact and correctness.

The RTT of pings follows a T-Student distribution. It is actually an assumption. With just a simple set of tests for both wired and wireless networks we found out that this is not quite close to the reality. An example of this can be seen in the Figure 3.1.

A timeout event (caused by an echo ICMP packet whose reply did not arrive back before reaching the timeout time) is considered as a normal ping whose RTT is equal to the timeout time. This consideration has some problems. We assume that every timed-out ping will effectively arrive at the moment given by the timeout time. It is not true. Usually the timeout is set up to a value such that there is a high probability that this packet will never arrive because of many reasons, the link is down, the landmark is unresponsive, etc. It

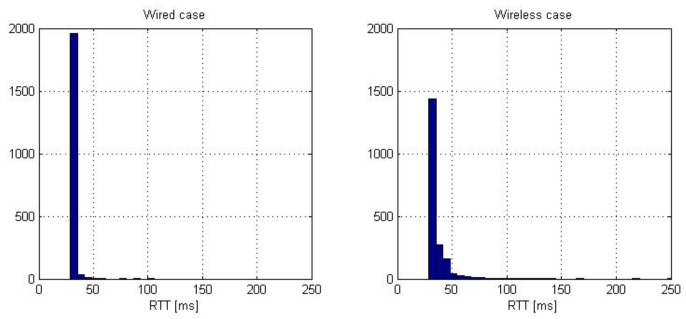


Figure 3.1: *General behavior of RTT for pings towards a commercial landmark.*

is necessary to give a special treatment to those timed-out ping measurements, and show them properly to the next processing elements of the pipeline.

Mainly these two problems lead us to the search of a new Path Anomaly Detection algorithm.

3.1.2 New Alternatives for the Path Anomaly Detection Method

When it comes to Anomaly Detection related to networking, there are plenty of methods that were already studied. Some examples of these methods are here.

- Statistical Approaches. In this approaches the procedure is usually the following: first it is necessary to get some input data coming from the phenomenon measured, then preprocess and filter this data, then perform a statistical analysis and a transformation, then perform a threshold determination and finally the anomaly detection. One typical example is the GLR method (General Likelyhood Ratio).

- Wavelet analysis. It consists on both temporal and frequency domains analysis. In [6] Soeule showed that this is not best than GLR with regards to anomaly detections in networking.

- Principal Component Analysis. It maps a set of points into new coordinates. In [5] Ringberg et al. point out the practical difficulty in tuning the parameters of the PCA based network anomaly detector.

- Kalman Filter. This is a method that uses samples of a phenomenon, then filters noise and inaccuracies from them and gives as result estimations for the real values.

It is important to remark that our main focus is not to have the exact anomaly detection results for one particular path. We actually want to estimate the Impact Factor using many anomaly detection results, obtained from many landmarks. It means that in some way we do not depend in the accuracy of each anomaly detection performed, but we depend only on the overall accuracy. In other words, the impact of one anomaly wrongly detected is not that big since there are other anomaly detection results (for other landmarks) that will alleviate the impact of the error.

Keeping this in mind, we can relax a little the accuracy of the algorithm and make it light-weight, so we can use it for several paths without loading unnecessarily the system on which the IFE analysis takes place.

Based on the good results of DTA (Decision-Theoretic Approach) in comparison with Kalman Filter's results given in [7], we first chose Statistical Approaches, expecting to get results good enough to skip more complex methods. Two methods (and its variants) were developed based on [7]. Their analysis will be presented below.

3.2 Path Anomaly Detection

First of all we answer the question *what is an anomaly?*. We define anomaly as a substantial variation of the RTT to landmark i. By substantial we mean a variation of a predefined ϵ value on the RTT. Our aim is to detect an anomaly given a sequence of RTT measurements.

Having a landmark to measure, a basic Anomaly Detection approach would be to take sample by sample and judge it according to the historical value of the measurements (for instance its mean). However this does not work at all if the variations of the RTT measurements exceed the value of ϵ for the normal case.

Thus we apply an iterative testing approach. We will introduce the algorithms that pursue this Anomaly Detection objective, and we will evaluate them by simulating a realistic set of RTT measurements with controlled variations of RTT.

3.2.1 Bernoulli based Iterative Testing Method

We define *normal profile* as a sequence of RTT values measured in a period of time where the behavior of the path is considered normal by the user. Our aim now is to have a policy to detect anomalies using both, the incoming samples and the just defined *normal profile*.

We establish an *anomaly threshold* value ϵ for the RTT to landmark i. We define a *rare sample* as a measurement X_m that falls outside the interval with radio ϵ centered in the mean of the *normal profile*.

All the samples that fall outside these boundaries are hence **suspected** to be caused by an *abnormal* phenomenon. We use the histogram of the *normal profile* to get an estimation of the probability that a measurement X_m generated by a *normal profile* falls outside these limits. We define this probability as α.

Therefore we have that

$$Pr(E[X_n] - \epsilon \leq x \leq E[X_n] + \epsilon) = 1 - \alpha \tag{3.1}$$

where x is a RTT sample.

A *rare sample* can fall outside these boundaries because of two reasons, this sample either:

- still belongs to the *normal profile* but in the rare area with probability α (the histogram obtained from the *normal profile* tells that the probability that a sample falls outside the boundaries is α).
- belongs to a different profile whose nature does not correspond with the *normal profile* and therefore is **abnormal**.

To realize whether an abnormal situation is taking place, we see the detection of sample rareness as a Bernoulli trial, with probability of success $1 - \alpha$ and probability of failure α as we defined. Note that success is only the probability that, given normal case, the sample falls inside the just defined boundaries, which corresponds to the fact that a sample is not affected by a significant short term phenomenon.

We define an observation in a window N as the amount of successes obtained in the last N measurements[1]. Now, knowing that α is small (in the *normal profile* histogram only a few samples should fall outside the specified limits), in N trials the probability of having n successes is

[1]Note that we do not care about the order of the samples, only about the amount of failures, i.e. *rare samples*.

$$Pr(n\ successes) = \binom{N}{n}(1-\alpha)^n(\alpha)^{N-n} \tag{3.2}$$

If we increase N, the probability of having relatively big amount of *rare samples* (failures) decreases.

Note that this algorithm is based on the frequency of the rare samples. If a shift in the *normal profile* happens, it will affect this frequency even in the case when the variation is smaller than ϵ, so we can predict that this algorithm will be in some way not very respectful of the definition of anomaly (i.e. it will declare anomalies in case of deviations from the *normal profile* smaller than ϵ). However, as we will see later, the usual distribution of the RTT measurements (very skewed) makes this wrong detection not to happen frequently at all.

3.2.2 Likelihood based Iterative Testing Method

Now we will present the second Path Anomaly Detection method.

Let X_m be the RTT samples to certain landmark i in certain period of time. The set of measurements X_m is influenced by a random variable X_n that represents the behavior of the path in the normal case. The p.d.f. of X_n is unknown.

The measurements X_m is also influenced by a shift component X_a.

We define **anomaly** as the presence of a shift X_a on the measurements X_m. So the samples X_m are:

$$X_m = X_n \tag{3.3}$$

with a no anomaly scenario, and

$$X_m = X_n + X_a \tag{3.4}$$

in case of anomaly.

Since X_m has its own probability distribution, an hypothesis of anomaly (existence of X_a) cannot be so easily accepted or rejected.

Being more strictly, we define

$$Anomaly\ A : |X_a| > \epsilon \tag{3.5}$$

were ϵ defines the shift that the RTT should present in order to report it as an anomaly.

We define the observation M as the last set of J observations X_m, but binned (i.e. M has the amount of samples X_m that fell in predefined ranges, for example $[0ms; 100ms)$, $[100ms; 200ms)$...). Having certain M, we want to tell the probability of an anomaly in this case.

So the probability of anomaly A given certain observation M is:

$$Pr(A|M) = \frac{Pr(A \bigcap M)}{Pr(M)} = \frac{Pr(A)Pr(M|A)}{Pr(A)Pr(M|A) + Pr(\bar{A})Pr(M|\bar{A})} \tag{3.6}$$

If we assume that $Pr(\bar{A}) = Pr(A)$ we have:

$$Pr(A|M) = \frac{Pr(M|A)}{Pr(M|A) + Pr(M|\bar{A})} = \frac{1}{1 + \frac{Pr(M|\bar{A})}{Pr(M|A)}} \tag{3.7}$$

Now we are interested in the ratio $\frac{Pr(M|\bar{A})}{Pr(M|A)}$. We need to calculate both numerator and denominator.

The probability $Pr(M|\bar{A})$ in the numerator can be calculated from a *normal profile* provided by the user. For every binned element in M we do a counting of its occurrences in the *normal profile* and divide it by the total amount of samples of the *normal profile*. Once the probability for all elements is obtained, and assuming no correlation between them, we multiply all these probabilities and we get $Pr(M|\bar{A})$. It means:

$$Pr(M|\bar{A}) = \prod_{j=1}^{W} Pr(X_m^j|\bar{A}) = \prod_{j=1}^{W} \frac{occurrences\ of\ binned\ X_m^j\ in\ normal\ profile}{number\ of\ elements\ of\ normal\ profile} \tag{3.8}$$

The probability $Pr(M|A)$ can be calculated from the fact that if $|X_s| > \epsilon$ then $|X_m - X_n| > \epsilon$ (since $X_m = X_n + X_s$). So we calculate:

$$Pr(M|A) = Pr(|X_m - X_n| > \epsilon) \tag{3.9}$$

which means

$$Pr(X_n < X_m - \epsilon) \tag{3.10}$$

and

$$Pr(X_n > X_m + \epsilon) \tag{3.11}$$

Iteration

Once we have a sequence M of binned X_m, we do the following:

- For every binned X_m we compute the probabilities $Pr(X_n < X_m - \epsilon)$ and $Pr(X_n > X_m + \epsilon)$ and we sum them. Once we have this probability for every X_m, we multiply all of them and we get $Pr(M|A)$.
- Now we compute $Pr(M|\bar{A})$ by multiplying the probability of occurrence of each binned X_m^j (we use the history of binned samples obtained while having the *normal profile*).
- We calculate finally

$$Pr(A|M) = \frac{1}{1 + \frac{Pr(M|\bar{A})}{Pr(M|A)}} \tag{3.12}$$

and using a threshold we declare the current situation as **anomaly** or not.

3.2.3 RTT Simulation and Analysis of both Methods

We have implemented the algorithms to check their performance. We use as base for our scenarios a set of real measurements obtained to different landmarks in different circumstances.

The way we use these samples is as following. For every RTT sequence of samples obtained we extract a segment of samples that are clearly not affected by any anomaly. In these segments our assumption is that we have *baseline behavior*, which means clean measurements affected only by short term phenomenons (delay caused by retransmissions, short term router queueing delays, short term load in end-host). Then we build a histogram with these samples (it is an estimator for the RV X_n) and we use it to simulate a normal path behavior.

So far we have no anomalies in the generated sequence of RTT. So we introduce them: we add to the sequence a second component with a rectangular signal in which the amplitude of each pulse and its width were tuned. This can be seen in the Figure 3.2. In these circumstances we study the performance of each algorithm given one ϵ. If the amplitude of the pulse added (i.e. simulated anomaly) is bigger than ϵ, the algorithm should declare anomaly.

Since we have observed that some anomalies are presented with both shift and gaussian noise, we also include in some pulses some gaussian noise of different standard deviations.

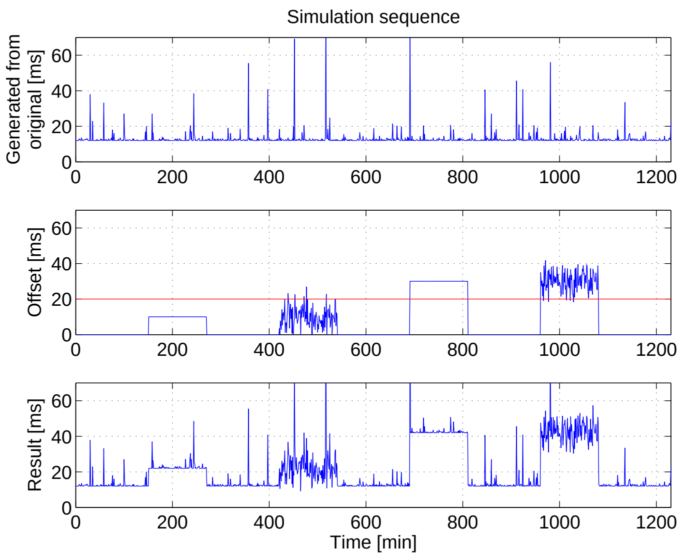


Figure 3.2: *Simulation of RTT Sequences.*

Both algorithms already presented above are pure ones. To complete the study we also added some modifications to them and analyzed their results. The following correspond to the set of algorithms (and its variants) that had the best results.

1. Bernoulli based algorithm
2. Likelihood based algorithm
3. Likelihood based algorithm with median filter

The Algorithm 3 only differs from the Algorithm 2 in that a median filter is applied to the RTT sequence before injecting it into the algorithm.

The Algorithm 3 performed much better than Algorithm 2, therefore we will show ROC curves only for Algorithms 1 and 3. The ROC curves were traced for both clean (more predictable) environments and extremely dynamic environments (were a WIFI connection is present). These scenarios are the most representative ones for all the observed cases (in both WIFI and wired connections). As a remark, less than 10% of the observed samples are as dirty as the most extreme dynamic environment presented here, the remaining 90% of the cases are generally much closer to the cleaner case (even for a WIFI connection).

In Figures 3.3, 3.4,3.5 and 3.6 we show the results of the analysis for Algorithms 1 and 3. The left side curve in each figure is a standard ROC curve. Each point in there represents the behavior of the algorithms for several simulations. Ideally all points should be in the top right corner of the area, which means that the algorithm correctly told anomaly when a real anomaly was going on, and correctly told normal situation when a normal situation was going on. The closer to the right axis of the curve, the more normal cases are detected correctly. The closer to the upper part of the curve, the more abnormal cases are told correctly.

First we present the results using as base RTT obtained from a WIFI connection, i.e. high variance RTT sequence.

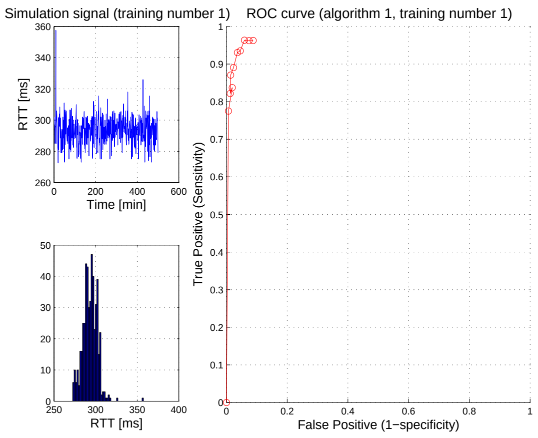


Figure 3.3: *Algorithm 1 (Bernoulli's based), high variance RTT sequence.*

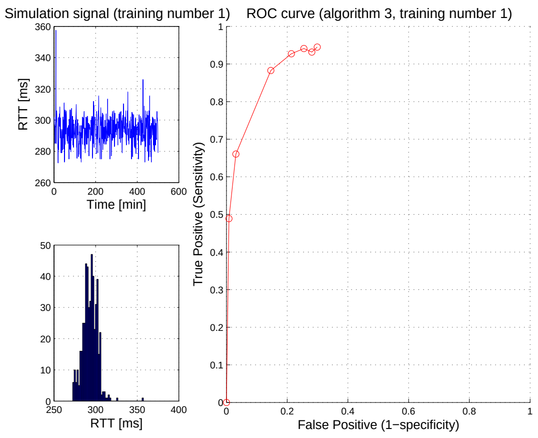


Figure 3.4: *Algorithm 3 (likelihood with median filter), high RTT variance sequence.*

Now we present the results using as base RTT obtained from a wired connection, i.e. low variance RTT sequence. Note that in this case both algorithms are way more accurate as we could expect.

Methods Comparison Summary

We present here more precisely and briefly the results in both qualitative and quantitative terms. The criteria took into account includes many points, we explain the meaning of each point below.

- The *parameters sensibility* represent how much the detection results vary when the input parameters of the algorithm are tuned in the whole valid range.
- The *accuracy* is shown in the ROC curves above, and the values presented next to the qualitative assessment describe the worst detection ratio between detected abnormal and real abnormal, and detected normal and real normal. The higher the better (for example

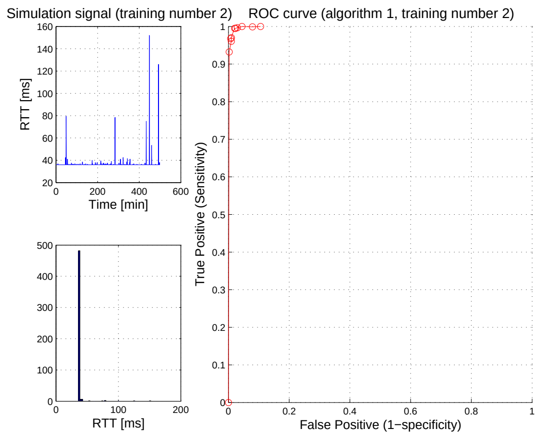


Figure 3.5: *Algorithm 1 (Bernoulli's based), low variance sequence.*

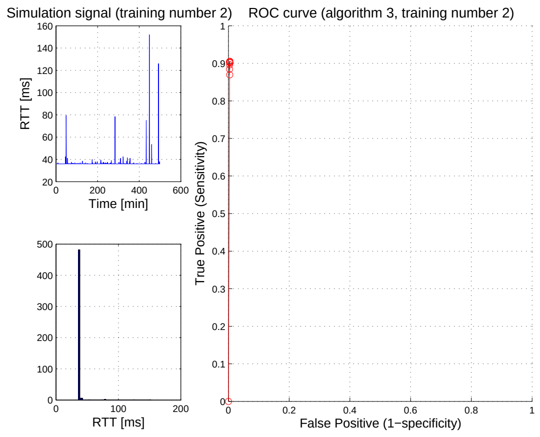


Figure 3.6: *Algorithm 3 (likelihood with median filter), low variance sequence.*

a value of 0.8 means that at least 80% of predictions are correct, either they are normal or abnormal).

- The *delay* is calculated by calculating the best matching between detected and real anomalies. The number given next to the assessment in the table tells the offset that gave the best matching between detected and real anomaly.

-	Bernoulli's based algorithm (1)	Likelihood based algorithm (2)	Likelihood based with median filter algorithm (3)
Parameters sensitivity	Wide range of input parameters which guarantees at least 90% for all the observed samples.	Too sensitive to binning method and number of bins.	-
Complexity	LOW. Calculation of mean, moving window, binomial p.d.f. valuation, thresholds.	LOW. Binning technique, histogram building, histogram smoothing, p.d.f. (histogram based) valuation	LOW. Same as 2, including median filter.
Memory usage	LOW	MEDIUM. Histogram bins' edges and values.	-
Accuracy (low dynamism)	HIGH. Reached 98%.	MEDIUM. Reached 80%.	HIGH. Reached 90%.)
Accuracy (high dynamism)	HIGH. Reached 95%.	LOW. Reached 60%.	MEDIUM. Reached 80%.
Delay (low dynamism)	HIGH. Average 4 samples.	LOW. Average 1 sample.	MEDIUM. Average 3 samples.
Delay (high dynamism)	HIGH. Average 7 samples.	LOW. Average 1 sample.	MEDIUM. Average 3 samples.
Additional remarks	-	-	-

Table 3.1: *Summary of the simulation analysis.*

Our intent is to apply this algorithm to RTT measurements obtained from different landmarks and to obtain several anomaly detection sequences. They will be used as input for the Impact Factor Estimator theory, in order to get a meaningful idea about the status of the network from the point of view of the monitored point.

The more accurate the anomaly detections are, the more accurate the Impact Factor estimation will be. As described in [1], the way errors in the anomaly detection affect the IF is given by

$$cI_f = I_f.TP + (1 - I_f).(1 - TN) \tag{3.13}$$

where cI_f represents the corrected Impact Factor, TP the true positive rate, and TN the true negative rate. As described there, the errors' effect can be alleviated by increasing the number of landmarks to

$$n_L \geq (\frac{z_{1-\alpha}}{\epsilon(TN + TP - 1)})^2.cI_f.(1 - cI_f) \tag{3.14}$$

Compared to the case where errors are not considered, with $\alpha = 0.05$, $z_{1-\alpha} = 1.6445$, $\epsilon = 0.15$, $TN = 0.95$, $TP = 0.95$, $I_f = 0.95$ we have for n_L (the minimum amount of landmarks) the following

$$n_L \approx 6 \tag{3.15}$$

and corrected (taking into account errors) the minimum amount of landmarks is

$$cn_L \approx 27 \tag{3.16}$$

If we set $TN = 0.90$, $TP = 0.90$ we have a much bigger difference

$$cn_L \approx 266 \tag{3.17}$$

Needing 266 landmarks to alleviate the bad impact of not accurate estimations lets us know that accuracy in singular anomaly detections is very important in terms of how feasible the calculation of I_f is. Due to this point, and the fact that Likelihood based algorithms are heavy in terms of memory usage (we think in the case when there are several dummy clients that send RTT measurements to a unique server for it to do the processing) we will go for the Bernoulli's based algorithm. We consider that even though its detection delay is relatively high compared with the other algorithms, its value is still negligible when it comes to our purpose, the calculation of I_f.

3.3 Analysis of Bernoulli based Path Anomaly Detection Method

Once the algorithm has been selected, we aim to study what its properties are. From now on we will only refer to the Path Anomaly Detection algorithm selected.

3.3.1 Sensitivity of the Algorithm

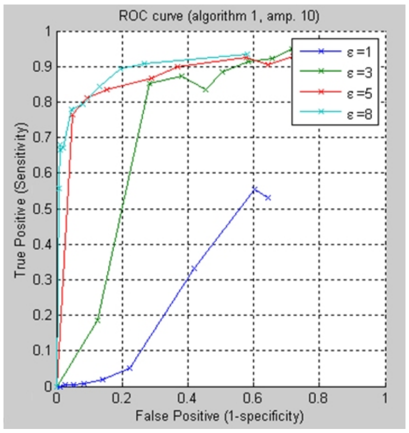
In Figure 3.7(a) and Figure 3.7(b) we can see the behavior of the method when facing WIFI connections in the path. We generate a simulation sequence of RTT exactly as described before, using shifts of width equal to 2 hours. We want to see how the detection varies in function of ϵ and the amplitude of the anomaly, so we trace several curves in the ROC diagram. In each case we generate RTT shifts of a fixed amplitude. Depending on the amplitude, this shift should be considered as an anomaly or not. Then we fix an ϵ value for the whole curve, and we generate a ROC curve with this parameter fixed. We do the same for many values of ϵ.

We remark that our main goal is to detect big anomalies, as big as needed for the end-user to perceive them. However we will explain what happens for small values of ϵ since we consider it is interesting to understand the algorithm. For values of ϵ of 1 millisecond (dark blue curves), it does not perform well even with big anomalies. This is because the algorithm simply judges each new incoming sample as *rare* or *not rare* according to the threshold ϵ. If ϵ is smaller than the dynamism of the measurements (what happens typically for wireless connections where the standard deviation of the RTT measurements is sometimes bigger than 5 milliseconds) then the training reveals that having many *rare* samples is very probable, so when many *rare* samples come they are assumed to be normal, so no anomaly is told even though an anomaly is going on. That is why the curves for small ϵ start with so many points on the lower left corner of the figure.

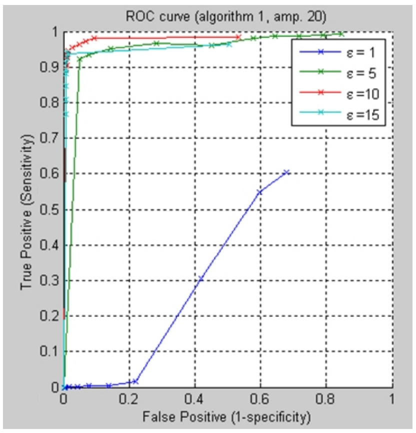
Note that for more reasonable values of ϵ, the curves show that there are configurations by means of which the accuracy of the algorithm is very good.

For the case of wired connection, the results are completely different. Even in the case when we have very small shifts, and very small values for ϵ, the algorithm manages to distinguish correctly the anomalies (see Figure 3.8(a) and Figure 3.8(b)).

With big anomalies the algorithm also performs very well, even for small values of ϵ (see Figure 3.9(a) and Figure 3.9(b)).

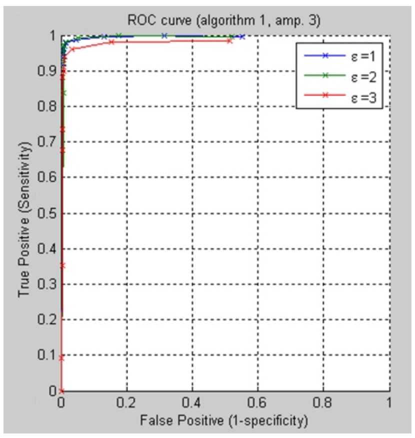


(a) RTT shift of 10 ms.

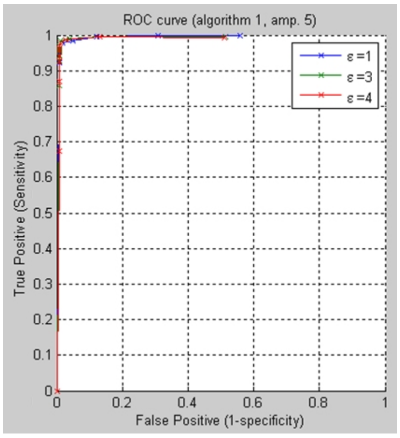


(b) RTT shift of 20 ms.

Figure 3.7: *ROC curves for different RTT shifts versus ϵ, for WIFI. Anomaly width equal to 2 hours.*

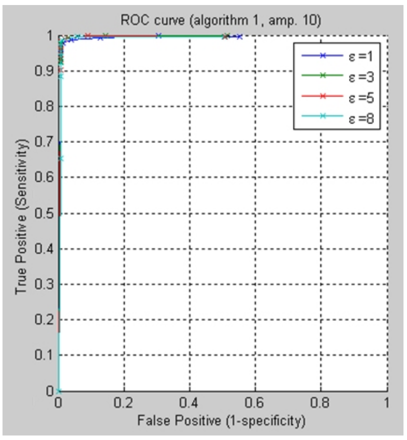


(a) RTT shift of 3 ms.

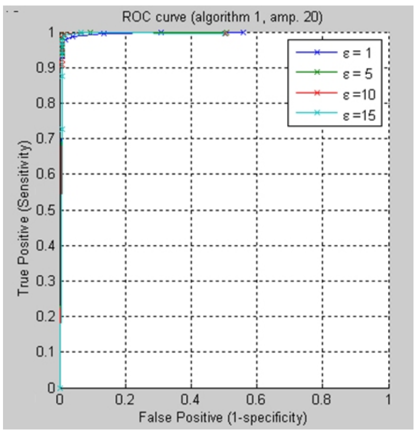


(b) RTT shift of 5 ms.

Figure 3.8: *ROC curves for different RTT shifts versus ϵ, for wired. Anomaly width equal to 2 hours.*



(a) RTT shift of 10 ms.



(b) RTT shift of 20 ms.

Figure 3.9: *ROC curves for different RTT shifts versus ϵ, for wired. Anomaly width equal to 2 hours.*

3.3.2 Treatment of Timed-out Pings

As we remember, one of the weak points about the original anomaly detection algorithm was the way it treats timed-out pings. So far, for our new algorithm we did not say anything about it. The way timed-out pings is treated here is simple. From one campaign we ping many landmarks. For each of these landmarks, we do the following in the given order:

- We ping once the landmark in order to refresh ARP caches.
- We do as many pings as given by parameters to the application, sequentially.
- From these pings, we discard timed-out pings and we do the average between the remaining. If all the pings were timed-out (more persistent status) then we judge this sample directly as *rare sample.*

Why do we discard timed-out pings if possible? Considering the case when we have n pings requested from the application parameters (which means that $n + 1$ pings are executed, considering the ARP refreshing ping), in case the landmark is actually not reachable, all of them will be timed-out and the campaign for this landmark will be declared *rare sample* as expected. If some of them were timed-out, but not all, there is a high chance that the remaining pings will be also affected by the phenomenon that caused the timeouts, thus they will be at the end declared as *rare case* anyway. If only some of the pings were timed-out, it means that the phenomenon is really a short-term event, so we keep on working with the averaged values of the remaining pings. Once again we remark that we are more concerned about long-term anomalies.

Chapter 4

Direct IFE

Direct IFE is what we originally mentioned simply like Impact Factor Estimation. We add this prefix *Direct* now since we came up with a different alternative of IFE that was not present at the beginning of our work. Direct IFE means sending probe packets from the Monitored Point itself towards several landmarks. However, in Section 5, we will see that it is also possible to do the opposite, i.e. to send probe packets from a particular set of landmarks towards the Monitored Point, and then get an IFE calculated somewhere.

In this section we focus on Direct IFE, the traditional approach. We will show which anomaly conclusions can be obtained using ACQUA.

4.1 Experiments Inside Planetlab

As we have just seen, during the simulations for unique paths we obtained satisfactory results for the path anomaly detection method. We had similar and more predictable results when we applied the algorithm to existent real samples. We now wonder how good our overall tool behaves once this new singular *path anomaly detection algorithm* is integrated and used to detect *network anomalies* in each of the tens of landmarks tested to calculate the IFE.

In this section we do experiments and we try to get some information about the properties and behavior of the tool. We run the application in 28 Planetlab [17] nodes distributed around the world. As landmarks we use this 28 Planetlab nodes themselves. We start the application approximately at the same time on each of them.

At the beginning we started with more than 28 nodes (they were around 40) but Planetlab maintenance and policies require some of the nodes (and its virtual machines) to be switched down from time to time. Since our aim with this experiment is to have the measurements of all the nodes synchronized in time, we discard those nodes that did not reach a reasonable amount of samples until turned off. So there are remaining only 28 nodes, the ones we used in the experiment.

4.1.1 IFE Sensitivity to the Amount of Landmarks

We wonder what is the effect in the Impact Factor Estimation if we take not the whole set of nodes, but only a subset.

We take one Planetlab node. We execute there our tool, against other 27 Planetlab nodes (landmarks) during around 2 weeks. We show here the results of a 4 days period, which is representative enough of the whole period. We have then one sequence of values IFE for all 27 landmarks. We wonder how much the IFE in one Monitored Point changes if we do not use all the landmarks (the 27) but only a percentage of them. Let's suppose we use half of the landmarks, we can have 2 extremely different situations: we might still have the same IFE shape along the time (revealing that fewer landmarks could have been used to still get the main information about the original IFE), or we might have a completely different shape (situation that tells that in the subset the main information about the IF was not completely captured because of the lack of landmarks).

To compare the original IFE (all the landmarks) and the one obtained from a percentage of the total, we define a way of comparing two IFE signals, the **IFE Difference Index**. To calculate it we proceed as following. We do the substraction of both signals (element to element) getting a difference vector, and we take the absolute value of it. We call this vector V. We count the amount of elements in V that are bigger than a threshold γ (if it is bigger, this sample is then interpreted as a sample *different enough* from the real IFE) and we divide this count by the total number of samples in V. We call this last value **IFE Difference Index**. This is then 1 if both signals are completely different, and 0 if each pair of samples is close to each other, as close as the threshold γ predefined.

About the way curves are generated, once a percentage has been selected to work, we generate 50 random subsets with this percentage, we calculate the IFE for the subsets and compare them with the real IFE, obtaining 50 IFE Difference Indexes. Then we use the average of all of them and plot this on the curve of the following figures.

In the Figure 4.1 the result can be seen for many situations. We detect anomalies of 5.0 milliseconds. To calculate the IFE Difference Index we use a threshold γ equal to 0.1, which matches with the confidence interval for IFE in [1] for around 27 landmarks (IFE values close to 0 or close to 1).

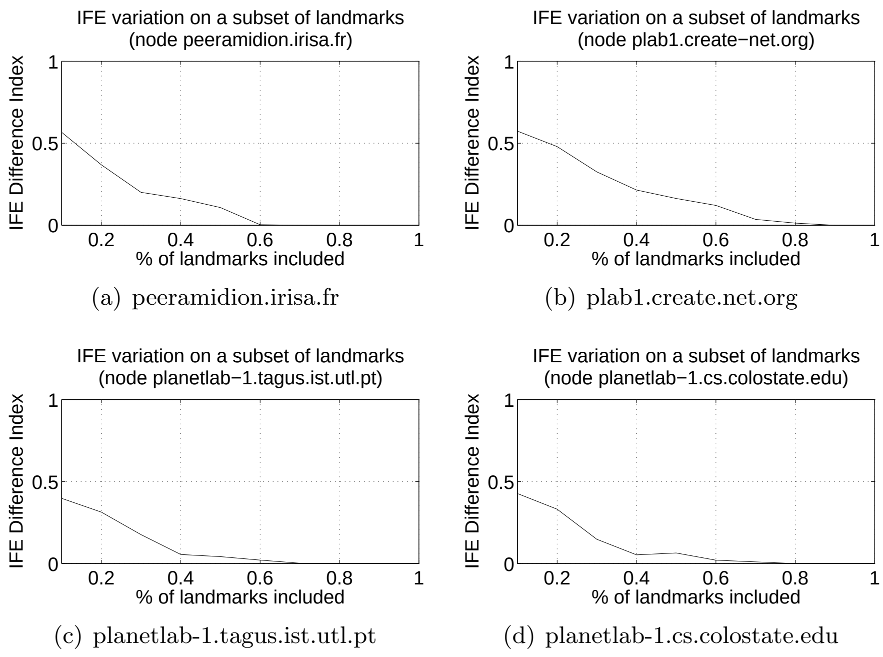


Figure 4.1: *Effect in the IFE when decreasing the number of landmarks. In total, 27 landmarks.*

As we can observe for the node *peeramidion.irisa.fr* (acting as Monitored Point), only 60% of the landmarks were needed to represent the IFE calculated. This means that *for the given set of anomalies observed during the experiment* only 60% of the 27 landmarks would be enough to obtain an IFE *similar* to the original IFE (for 27 landmarks). The concept of *similar* means that element to element the difference between them is smaller than γ. However for other Monitored

Points, we might need more. It depends on the access tree that each of them faces.

In Figure 4.2 we show the average of all the 28 cases. In this curve we can clearly see that the main IFE information was completely captured having 80% of the total number of landmarks used in the experiment.

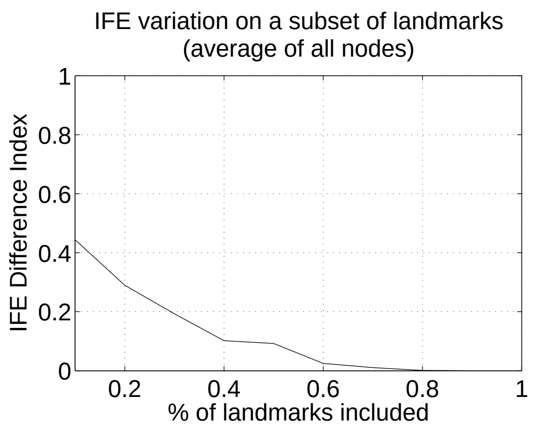


Figure 4.2: *Effect in the IFE when decreasing the number of landmarks. In total, 27 landmarks. Average among all cases.*

Given this case, we can expect that with only $0.8\dot{2}7 \approx 22$ landmarks we can monitor every Planetlab node's connectivity. Considering that Planetlab is currently composed of around 1080 nodes, using only 22 of them as inverse landmarks to monitor any other Planetlab node seems to be an interesting idea. It is important to remark that accordingly to what these last experiments show, this result holds only inside Planetlab. However, in Section 5.1.4 we include in our studies other non-Planetlab nodes.

These results agree with [1], since in most of the cases the IFE were either close to 0 (in the range 0 to 0.2) or close to 1 (range 0.8 to 1.0), meaning that with a IFE confidence interval equals to 0.1, and values of IFE in these areas, the amount of landmarks required for an IFE is less than 40.

4.1.2 Sensitivity to the Chosen Landmarks

We aim to show now what is the effect of changing the set of landmarks used to calculate the IFE. Now we take a subset of landmarks from the original set, and we compare its IFE with the IFE of another subset obtained in the same way. Then we obtain a IFE Difference Index. This is done 50 times with both fresh random generated sets, and all the IFE Difference Indexes are averaged and later shown in the curves.

In Figure 4.4 we show the average of all the above cases.

It is important to note that when we generate 2 subsets starting from the original subset, we take a percentage of the original one decreasing the amount of landmarks finally used. This means that the quality of the experiment decreases somehow.

In this last figure we can clearly see that generally for two subsets covering randomly 50% of the original set, in average the similitude between both IFE is given by a IFE Difference Index of around 8%. This index means that in 92% of time, the IFE measured in one set is *similar enough* to the IFE measured in the other set.

4.1.3 Anomalies Detected

In this section we show a couple of examples about the utility of the tool.

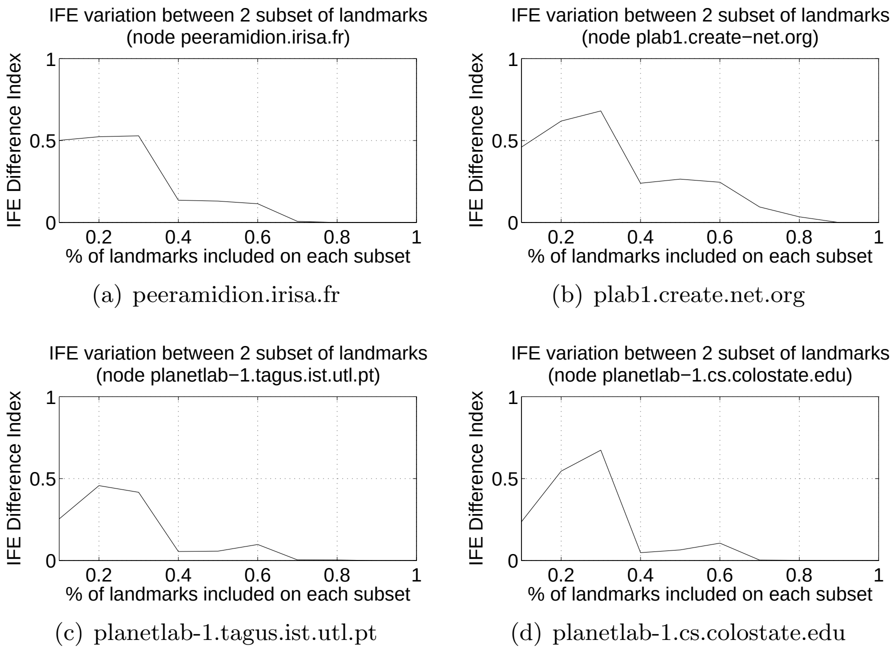


Figure 4.3: *Effect in the IFE when changing partially the set of landmarks. In total, 27 landmarks.*

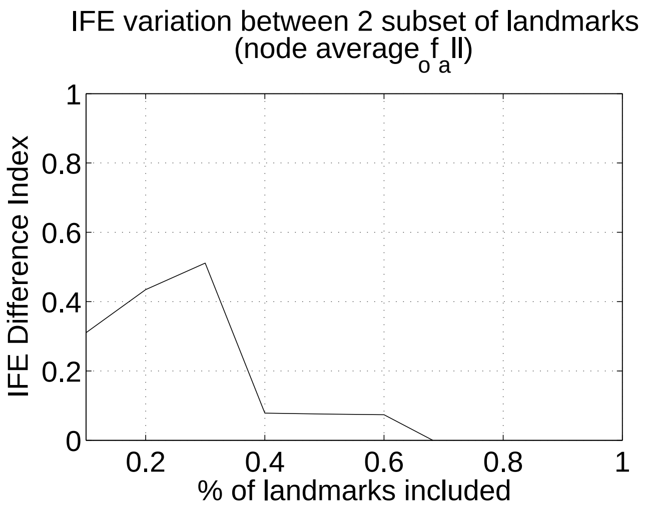


Figure 4.4: *Effect in the IFE between 2 random sets generated by means of the general set. In total, 27 landmarks. Average among all cases.*

Local Anomaly

In the Figure 4.5 we see a case where the network at the site of *planetlab-1.pdl.nudt.edu.cn* suffered an anomaly with IFE of around 100% (starting approximately at minute 1500 in the figure). We know that this anomaly was not caused by a heavy load in this machine. This is simply to find out because we add usually to the set of landmarks to probe a special *gateway landmark*. If the load is causing ping packets to be delayed in the OS queue, then this gateway landmark should also suffer an anomaly, which was not the case here. In fact this is the reason why the IFE is not exactly 100% but a little bit less. At that time **any** packet traversing the local network of this node suffered the impact of the local anomaly. In this case, we can suspect that the local network was overloaded, one or more of their routers went out of service for some reason and a under-dimensioned backup router took its place, etc. During the anomaly almost no unreachabilities occurred.

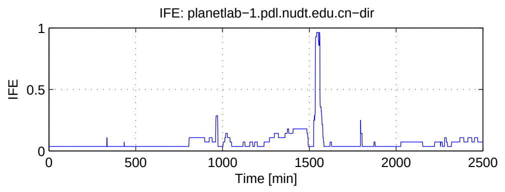


Figure 4.5: *Local anomaly detected.*

Non-local Anomaly

In the node *kostis.di.uoa.gr* we detected an anomaly on August 2nd, 2011 at 19:29:43 (GMT+0200). This is shown in Figure 4.6. This is the one that starts at minute 1540 until minute 1584, having an average IFE of around 0.4. This seems to be a non-local anomaly since a few (less than half in this case) landmarks are affected, which is not usually the case with a local anomaly.

Note that the mentioned anomaly is preceded for another anomaly, which seems to be a local one starting at minute 1503 until 1510. We could argue that this is not local since it is not covering all the landmarks, however, the reason why we tell this anomaly is local is that some paths will hardly be declared as abnormal, they are so noisy that even when there is an anomaly in the local segment of the path, the anomaly detection algorithm will behave in a conservative way not judging the situation for the path as abnormal. However these noisy paths represent usually just a small percentage over all possible random landmarks.

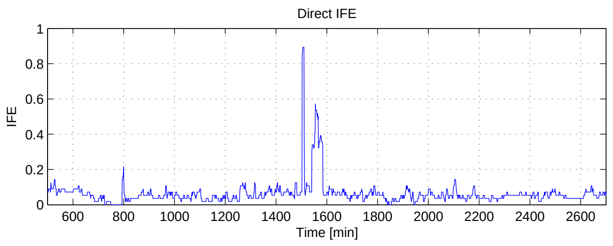


Figure 4.6: *Non-local anomaly detected.*

Global Anomaly

There is a particular case that drew our attention. It is about an anomaly that lasted for about 8 hours on July 2nd, 2011 at around 10:00:46 (GMT+0200). In the Figure 4.7 we show some of the Monitored Points (Planetlab nodes) that realized about it. This is clearly a non-local anomaly from the point of view of each Monitored Point since the Impact Factor Estimation obtained is relatively big, but not close to 1. Not being close to 1 means a non-local anomaly (a local anomaly affects all the paths, not half of them for instance). Being the IFE close to 0.5 means that only a set of paths suffered the problem. If we add to this the fact that all of them suffered a problem that starts and finishes at the same time, we are very likely to be facing a link anomaly case.

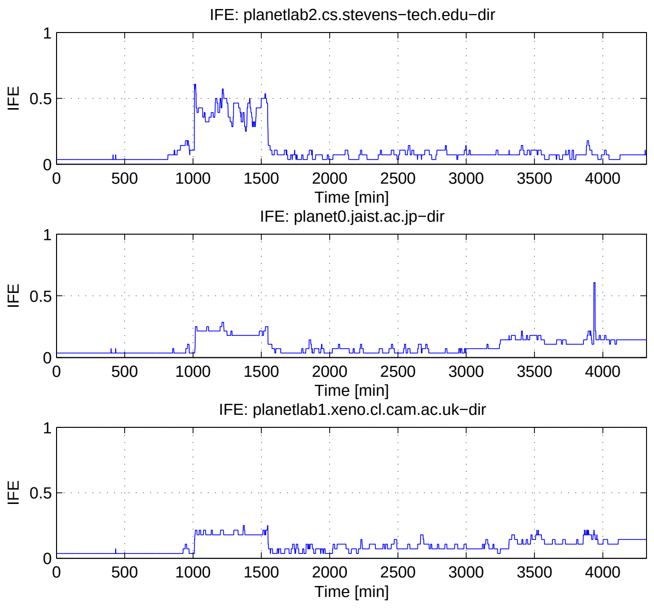


Figure 4.7: *Global anomaly detected.*

Going further, what is more interesting in this case is that Planetlab nodes from outside Europe that shown the anomaly (like *planetlab-1.cs.colostate.edu*, *planet0.jaist.ac.jp*, *planetlab1.cs.stevens-tech.edu*, *planetlab2.cs.stevens-tech.edu*) detected it because of anomalies in path heading to certain European nodes (like *planetlab1.xeno.cl.cam.ac.uk*, *planetlab4.inf.ethz.ch*, *planetlab3.inf.ethz.ch*, *planetlab4.inf.ethz.ch*, *ple02.fc.univie.ac.at*, *planetlab1.informatik.uni-wuerzburg.de*). On the other side, European nodes that shown the big anomaly (like *planetlab1.xeno.cl.cam.ac.uk*, *planetlab2lannion.elibel.tm.fr*, *peeramidion.irisa.fr*) detected it because of anomalies on paths towards non-European nodes (like *planetlab-1.cs.colostate.edu*, *planetlab1.cs.stevens-tech.edu*, *planetlab2.cs.stevens-tech.edu*, *planet0.jaist.ac.jp*). As it is shown in [12], in the research network GREN there is a subnetwork named GEANT, that is the biggest research network of Europe. Most of the packets coming from/going to European Planetlab nodes will traverse somehow this network. We can clearly suspect that something wrong happened there.

Although our tool does not specify exactly where the problem is, the information it gives in conjunction with other IFE measurements around the world let us:

1. Know that something serious happened.
2. Identify what set of nodes failed.
3. Have a guess about what went wrong, and where.

Chapter 5

Inverse IFE

5.1 Inverse Impact Factor Estimation

If we check the Figure 5.1 and we think about the Direct IFE (what we have been explaining so far), pinging is done from the Monitored Point. We have just shown that this approach works well to detect anomalies, and try to locate them. However there are some good reasons why to use as the input of the IFE method the pings done by the landmarks towards the Monitored Point (inverse pings) also, apart from using pings done by the Monitored Point towards the landmarks (direct pings).

- Pings that go towards the Monitored Point usually reach up to the public IP address. It means that its measurement is not highly influenced by local activity, which is interesting to discard the fact that a problem is happening locally. In other words, if a user uses Inverse IFE, and starts having problems in the network, in case the Inverse IFE tells everything is normal then for sure the problem is locally.
- If pings are not done in the Monitored Point, we can put aside the need of installing a program there. In practice, this is one of the strongest points for doing Inverse IFE. While doing the set of experimentations, we realized that not many people feel like collaborating with measurements by installing an application, they dislike the idea. Even if people are curious about the information they might get, they will hardly download the application, install it, and start using it for a long period of time.
- It could allow users to test the behavior of their connections even though their computers are switched off. A user would create an account in a centralized server and then a regular probing process would take place to start measuring the status of the connectivity of the user.
- Having a known centralized database with measurements of many users, some further ISP conclusions could be obtained. In the case when only Inverse IFE takes place, and some Monitored Points close to each other show a high IFE value with similar pattern, then the probability that the ISP is to blame is higher.

Naturally the problem with Inverse IFE (and inverse pings) lies in making landmarks ping the Monitored Points we want and giving us these results. However there is a way to solve this.

In the Figure 5.1 we can see that the Monitored Point sends probe packets to the landmarks. After getting the information, the Monitored Point simply calculates anomalies for each landmarks and once they are all calculated, the Impact Factor Estimation is processed. The theory described in [1] mentions that the IFE must be obtained by taking randomly distributed landmarks, it guarantees that we cover an amount of links enough to obtain a meaningful IFE for the user.

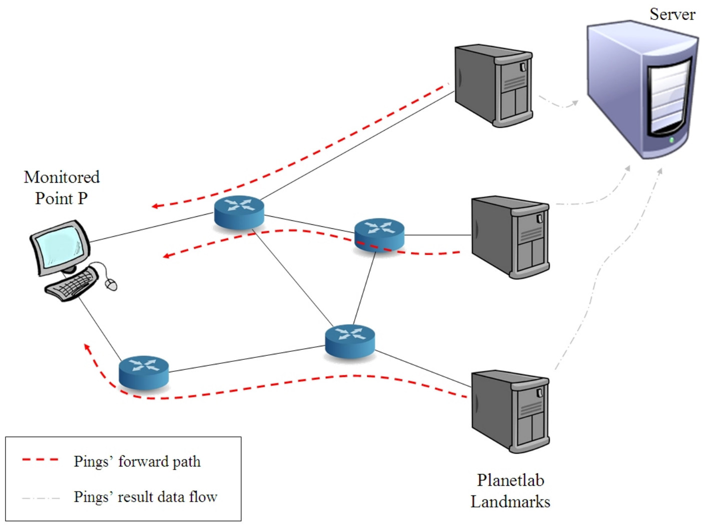


Figure 5.1: *Inverse Impact Factor Estimation.*

Once randomly nodes were taken (we can assume that for instance commercial landmarks that belong to Google or Yahoo could be considered), we should ask them to ping the Monitored Point and to give us the measurements. This is clearly not feasible. However, we can profit from **Planetlab nodes**, and make them do this job. It means that they would ping the Monitored Point, and give to a Server this information to calculate the IFE and show it later. Until this point we do not implement this architecture, we first study some offline results to see whether it is worth the effort or not.

5.1.1 Comparison between Direct and Inverse IFE from Planetlab

There are some differences between this approaches that should be kept in mind.

ICMP Packet Path and Inversion of the Ping

The probe packet we are using is an *ICMP echo request* packet, whose reply is an *ICMP echo reply*. They are both generally not treated in a different way by a router when it comes to routing, unless explicitly told in the routing tables which is a very rare case. So we can assume without making a mistake that when a router routes an *ICMP echo request* packet to the route A, an *ICMP echo reply* packet will go also though the route A.

Given this, since a ping is a round-trip of an ICMP packet (both *ICMP echo reply* and *request*), the path it goes through should not be different in the cases when it starts from the Monitored Point, or when it starts from the landmark (where we only swap the place of each kind of message, *ICMP echo reply* and *ICMP echo request*).

Public IP

Pinging from the Monitored Point the only limitation we have are landmarks that might not reply, and we solve it easily picking up another random set such that all the landmarks are responsive to pings.

However when we ping from the landmarks, we can only reach the public IP of the Monitored Point. If this public IP corresponds to a set-top box, then the local activity will be in most of the cases hidden from the pings coming from outside (unless the rare case where there is a routing policy for ICMP packets).

Software Layers

The ping not only traverses routers, but also layers of software in the end-nodes. And in that case there is a difference between times for both cases. If we see the Figure 5.2 we can see the difference clearly.

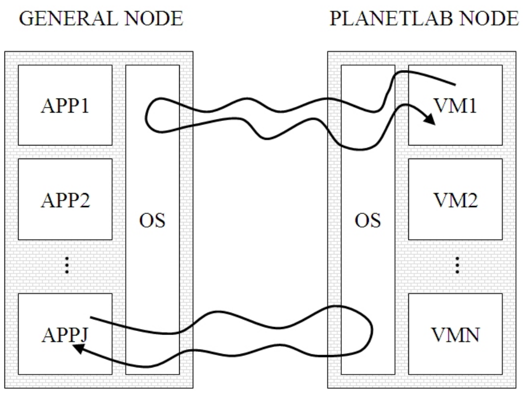


Figure 5.2: *Differences between Inverse and Direct IFE.*

When a general node requests an echo (sending an *ICMP echo request* packet), it leaves the application APPJ, traverses some OS layers, and it arrives to OS layers of the Planetlab node without going inside any *virtual machine*. Since *virtual machines* in Planetlab nodes have strictly limited resources, the fact that one or more of them is seriously loaded will not have a significative impact in the *ICMP echo* replying procedure. The OS is always kept alive and responsive.

However, in the case when the *ICMP echo request* is sent by the Planetlab node the packet does start in the *virtual machine* and goes through the *virtualization layers* towards the OS layers. This means higher dependance on the overall load of the whole node (many slices have allocated shared resources on the node).

Well, a similar thing happens in a Monitored Point (general node), the difference with a strict Planetlab node is that a high load in this node would have a clear impact in both cases (inverse and direct) since applications can use the whole resources of the node (and produce delays in OS tasks even though OS tasks have more priority).

Very few examples were observed with this behavior. Generally there is a strong correlation between pings from a node A to a Planetlab node B.

5.1.2 Access Tree

We present this potential difference between Direct IFE and Inverse IFE from Planetlab separately from other differences because it is one of the most challenging points that might make our Inverse IFE approach fail when using Planetlab nodes as landmarks. We know that if we take the landmarks randomly from all Internet, then the results about the representativeness of the IFE for the status of the network given in [1] hold. Otherwise it is necessary to check how much we modify this indicator.

In our case we want to use only Planetlab nodes as landmarks, not any random set of landmarks. This might represent a problem because Planetlab nodes are generally connected to GREN [12] (The Global Research and Education Network), a world wide **research** network and this might make of Planetlab a not representative scenario for the whole Internet, although still useful to understand the global Internet network [13]. Nodes generally connected to GREEN means that there might be policies that make packets coming/going to the Planetlab network follow predefined paths instead of the paths that we could expect. To explain this better, we present the following example.

In the Figure 5.3 we have both Planetlab nodes (in white) and non-Planetlab nodes (in black), they represent landmarks. To make the explanation very clear, we show a really pessimistic scenario where all the Planetlab nodes are clustered in a unique sub-network. Note that this clustering might not be only physical, but also might be logical in the sense that routing tables allow only the paths shown in the figure, even though there are other possibilities.

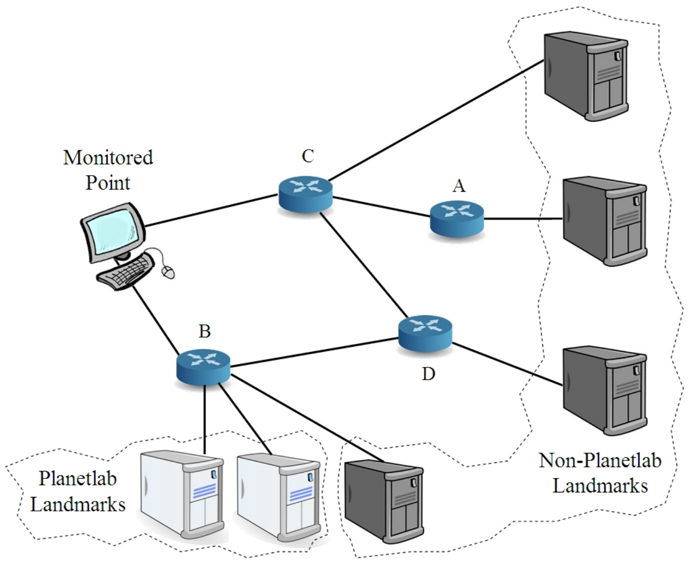


Figure 5.3: *Simple example of the importance of the Direct IFE access tree routers coverage in the Inverse IFE from Planetlab.*

As we can see here, by picking up random landmarks (from both sets) we increase the set of routers (and hence links) that are traversed. We want to traverse mainly routers that are closer to the Monitored Point, since their failure will have a much bigger impact in our network experience than a failure in routers far away.

We have two scenarios: in one of them we use random landmarks, and another one where we use only Planetlab nodes (a subset of all the possible random landmarks). Assuming we use completely random landmarks, the failures in routers C and B (1 hop of distance from the Monitored Point) are really likely to be detected and shown in our measurements by means of a high value of IFE. But if we use only Planetlab nodes as landmarks, the router C will not be traversed by the probes and hence its failure will not be detected, even though a failure in the router C represents a problem in the connectivity to a high percentage of general landmarks. This is why it is important to check the similitude between access trees for both cases (random landmarks, and Planetlab landmarks), and how well representative are Planetlab landmarks of the overall Internet landmarks.

Experiment about the Access Tree

We show in an experiment that Planetlab nodes are satisfactorily valid in most of the cases for our Inverse IFE studies, even though they are generally connected directly to GREN and not directly to commercial networks.

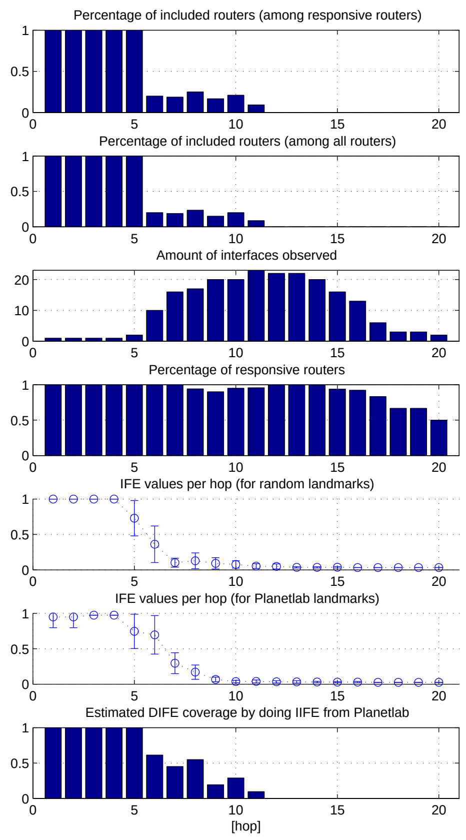
When we send probes to random landmarks (Direct IFE) they go and come back traversing routers on the way. If we send probes to several landmarks, those routers closer to the Monitored Point will be probably included in the path more times than routers that are further (for example, in Figure 5.3 the router C is used at least to reach 2 landmarks, while A is useful only to reach 1 of them). We define D_i as the set of routers used in the Direct IFE that are at i hops far away from the Monitored Point. Note that D_i involves both ways, forward and round, but starting from the Monitored Point. If we now send probes from Planetlab nodes to the Monitored Point, we would like to see that the set I_i (set of routers traversed in Inverse IFE being at i hops away from the Monitored Point) is included in D_i when i is small. This is equivalent to say that all the routers that we traverse in Direct IFE are also traversed during Inverse IFE. The reason why we use routers at a distance of i hops, is that we want to limit our focus on routers closer to the Monitored Point. We do not mind about routers far away from the Monitored Point because they have a smaller impact when trying to reach a random landmark.

In the following experiment we took 10 nodes from volunteers, they are Monitored Points. For each of them, we do trace-routes towards around 30 random landmarks (most of them are commercial) and towards 30 Planetlab nodes. Since the trace-route command only traces routers on the way forward (not on the way round) we would like to ask these landmarks to ping the Monitored Points and give us the measurements. Well this not possible for commercial landmarks (there are trace-route servers though, but by choosing them we stop having strictly random landmarks). For Planetlab nodes, we make them trace-route the Monitored Point. All this means that the route towards and from Planetlab nodes is totally traced, while the route towards and from random landmarks is generally only traced on the way forward (from Monitored Point to the landmark, but not in the opposite way). We can still make projections with this information to have an idea about how included is I_i in D_i for routers as far as i hops from the Monitored Point.

In the Figure 5.4 we show a real case as an example. We use it to explain how to interpret the curves.

In the x-axis we have the hop number. Each Figure represents the situation obtained in one particular Monitored Point. In the Figure 5.4 we show the case for a non-real Monitored Point. The third graph (*Amount of interfaces observed*) in this Figure shows how many different interfaces were traversed in each hop to reach the random landmark (as it would happen a Direct IFE approach). If an interface is not responsive, we consider it unique, i.e. different than any other interface. The fourth curve (from top to bottom) shows what is the percentage of routers that were responsive, in this lucky case most of the routers of each hope were responsive, except in hop 9 where about 10% of them were not. Both graphs, one and two, in Figure 5.4, show how many of the routers used while pinging random landmarks were traversed when pinging Planetlab landmarks (percentage taken among only responsive routers and all routers including non-responsive routers, respectively). We would like to have both curves with 100% at every hop (until we reach the destination).

In graphs one and two (that are similar since most of routers were responsive) we see that until hop 5, all routers are included. These routers in the first 5 hops are traversed all in both cases (towards Planetlab and random landmarks). We might interpret this as a very good result. However this might not be a good signal as we expect. If we see the case A in Figure 5.5, we might imagine that this is what is happening, shown in graphs one and two. In this case the topology diverges rapidly, and as we go further away from the Monitored Point (in terms of



(a) Case 1, guest INRIA Sophia Antipolis network

Figure 5.4: *Importance of the Access Tree Routers Coverage in the Inverse IFE for some particular cases.*

hops), the impact of a failure in each router becomes less and less important in the accessibility of a random landmark. For instance, in hop 4 the failure of router R would cause an Impact Factor influence of 50% (since 50% of random landmarks suffer the anomaly). In hop 5 a failure in router S would cause an Impact Factor influence of 25%. We are worried about conserving the elements that give most significative variations to IFE. This means being worried about traversing the same routers until a hop such that the IFE variations caused by its routers is not significative enough, this is our threshold of IFE. For this particular case, if our threshold is 25%, we can stop worrying at hop 3. But this interpretation might be wrong. If the reality is given by the case B in 5.6 things change completely. Note that the routers in the 5 first hops are all extremely important because of the network topology. Their failure would cause an Impact Factor influence of 100% in all the cases. They do not become less important as we go far away from the Monitored Point. So our question now is *up to which hop we should have a high routers coverage?* As we said, in Case A we would be happy to cover routers up to 5th hop, included, since we cover then routers with Impact Factor influence bigger than our threshold 25%, but in Case B to get this Impact Factor influence threshold we need to go up to hop number 6.

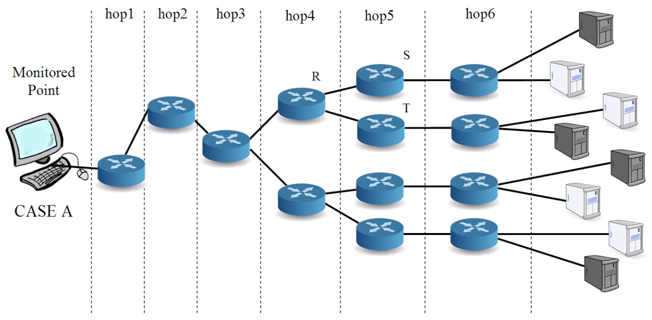


Figure 5.5: *Access Tree, Case A.*

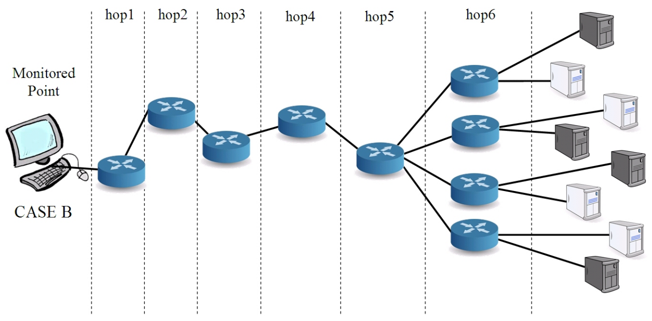
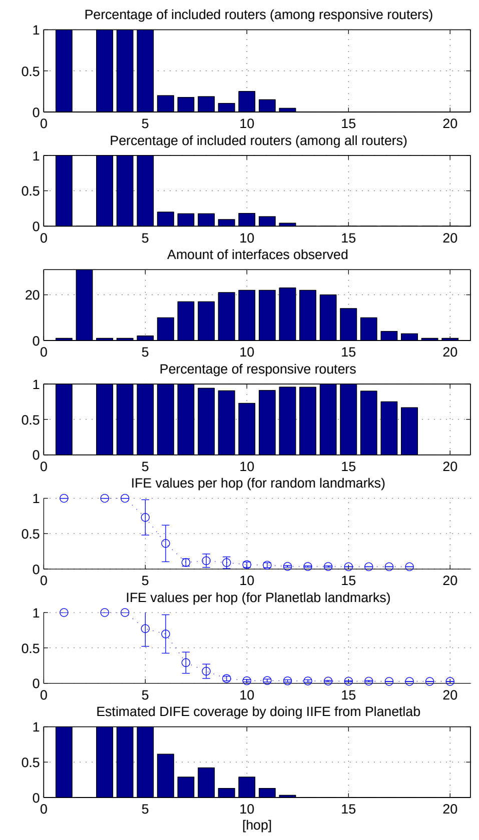


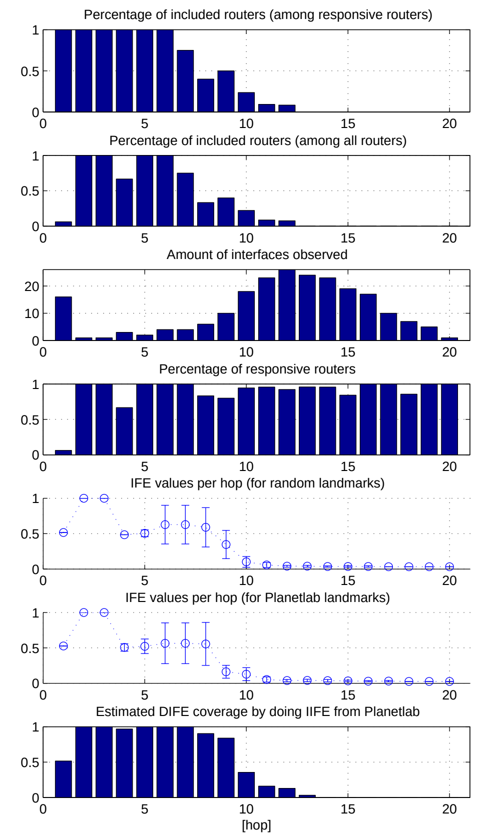
Figure 5.6: *Access Tree, Case B.*

In graphs fifth and sixth of Figure 5.4 we can see what is the mean IFE influence for routers of each hop and the standard deviation of their IFE influence (there are many routers per hop with different Impact Factor influences each). For this case, an important divergence starts only at hop 7 (for random landmarks), where the mean of the IFE influence for each router is 15% (small enough), but where the coverage of the routers is low (around 20%). However it might not represent a serious problem. Let's consider the case when a few routers are traversed to go toward most of the landmarks, even though our coverage is about 20%, in this 20% we may have routers that are very meaningful (having high IFE influence). This is what happened in hop 7, and that is why we see an IFE influence covered of around 60% (bottom graph in Figure 5.4).

The following cases were obtained in the same way as Figure 5.4. We discuss now these results.



(a) Case 2, moon.inria.fr, Sophia Antipolis



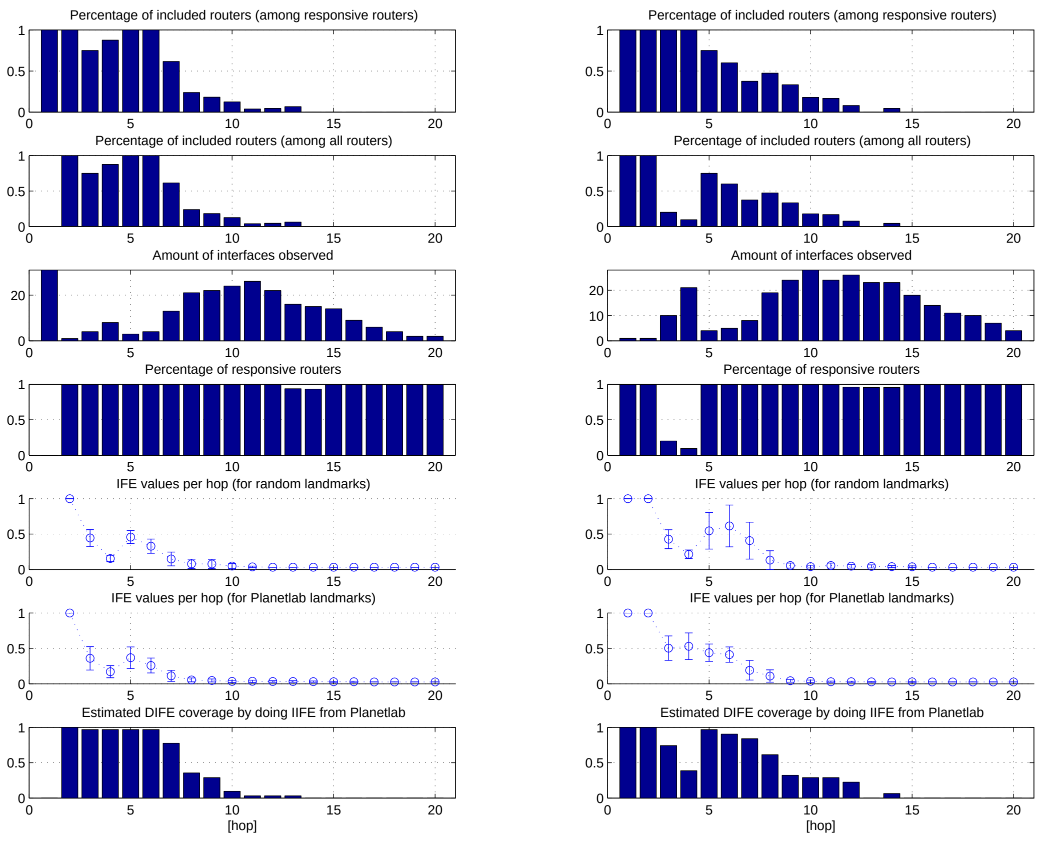
(b) Case 3, Set-top box, Nice

Figure 5.7: *Importance of the Access Tree Routers Coverage in the Inverse IFE for some particular cases (contd.).*

The Case 2 in Figure 5.7 is similar to the Case 1. Both Monitored Points were located in a big company site, INRIA Sophia Antipolis. Their 5 initial routers are apparently the same routers for any outgoing packet. These sites have direct Academic Network connection, so after leaving the site network, they might take a different path towards Planetlab landmarks, instead of following the expected route. We can see in the IFE values towards random landmarks that each router has small impact, while going to Planetlab landmarks the Impact of each is much bigger. This gives us a clue, the routers used to access Planetlab nodes differ from those used in general.

The Case 3 was taken from a home ISP connection. Up to 3rd hop it seems that the same 3 routers are used (although not sure, hop 1 presents a non very responsive router, giving less priority to the ICMP protocol maybe). In hop 9 around 10 routers are traversed, they have a mean Impact Factor influence of 10%. The IFE coverage at hop 9 is around 80%. This is a nice case (compare with cases 1 and 2), where we can expect that anomalies in these routers will be captured for both, DIFE and IIFE from Planetlab.

The Case 4 in Figure 5.8 show another good case. DIFE coverage from IIFE Planetlab of around 80% at hop 7, where the mean IFE influence per router is around 15%. We can also



(a) Case 4, Set-top box, Cordoba, Argentina (b) Case 5, big local network, CIV, Valbonne France

Figure 5.8: *Importance of the Access Tree Routers Coverage in the Inverse IFE for some particular cases (contd.).*

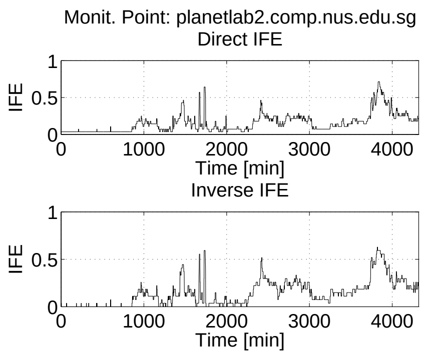
expect a good IFE approximation by means of DIFE from Planetlab here.

The Case 5 in Figure 5.8 shows a regular case, where good IFE coverage is high until hop 7, point in which the mean IFE influence per router is around 20%, with high standard deviation.

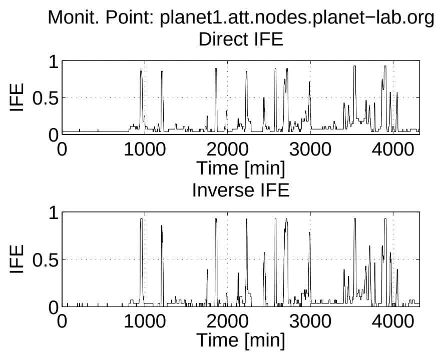
All other cases are represented by the ones that were shown. With this experiment we show that Inverse IFE from Planetlab is possible in most of the cases where the Monitored Point is a home ISP, although we sacrifice some accuracy in the Impact Factor Estimation.

5.1.3 Experiments on Direct and Inverse IFE inside Planetlab

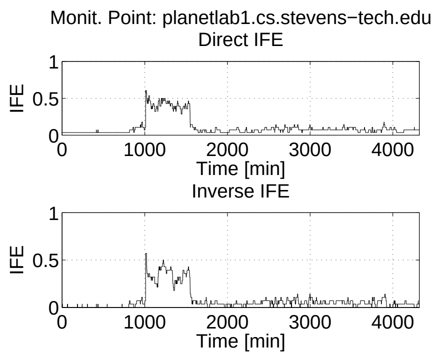
In this experiment we run ACQUA in 30 Planetlab nodes, and we use as landmarks this same set of nodes. After approximately 1 week of measurements, we take from there a representative period of 4 days of measurements, we compute for each node both the Direct IFE and the Inverse IFE.



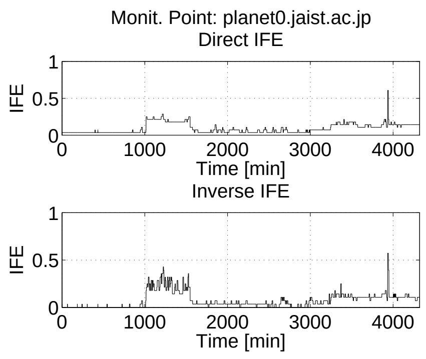
(a) planetlab2.comp.nus.edu.sg



(b) planet1.att.nodes.planet-lab.org



(c) planetlab1.cs.stevens-tech.edu



(d) planet0.jaist.ac.jp

Figure 5.9: *Direct versus Inverse IFE in Planetlab.*

In Figure 5.9 we can see that being in Planetlab, Inverse IFE is as representative as Direct IFE.

Empirical Differences Found

There was a strange case, shown in the Figure 5.10.

For the case of the node *planetlab2.arizona-gigapop.net*, it seems that it is configured in such a way that it does not reply to *ICMP echo request* packets. However there were a few nodes that managed to ping it (for instance planetlab3.inf.ethz.ch and planetlab4.inf.ethz.ch). We suspect that since this is the only node that we have seen not replying to echo probes, some user modified its properties in a greedy way.

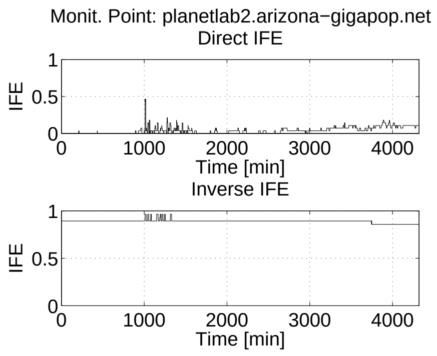


Figure 5.10: *Empirical difference found between DIFE and IIFE in Planetlab.*

5.1.4 Experiments Inside and Outside Planetlab

In the previous experiments we showed how well our tool behaves in the world of Planetlab. Now we want to involve also non-Planetlab nodes.

As we told, since Inverse IFE cannot make probe packets go inside the local network, its measurements are generally less contaminated with local activity. This represents a good advantage if what we are looking for is objective anomalies in our connectivity to Internet.

Huge local network

In this experiment we use a Monitored Point that is behind a huge local network, which contains a Planetlab site directly connected to it. We show this experiment in Figure 5.11.

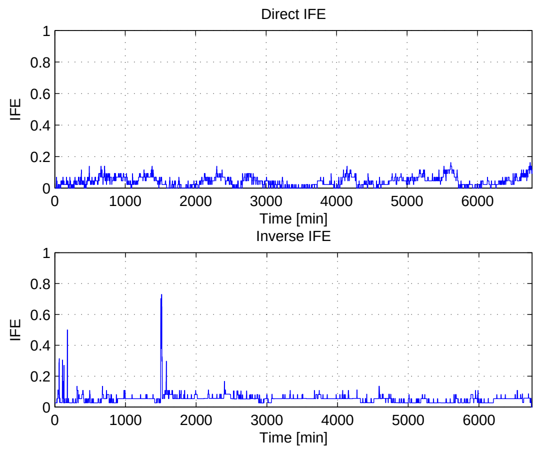


Figure 5.11: *Direct versus Inverse IFE in a big local network.*

Even though there were no anomalies observed that let us relate in a better way both IFE, we can see there that there significative differences. In this experiment we use random landmarks for DIFE and we do IIFE from Planetlab. *Why are both IFE different?* The explanation probably has its answer in Figure 5.7, Case 2 for *moon.inria.fr* Monitored Point. If we remember, the access tree towards random landmarks from this Monitored Point was poorly represented by the access tree towards Planetlab nodes. We ascribe these differences to the access tree issue.

Small local network

In this experiment we use a Monitored Point in Argentina whose access tree is described in Figure 5.8, Case 4. As we might expect from the access tree study, this node should behave somehow similarly when performing DIFE and IIFE from Planetlab.

In first place we show in Figure 5.12 both IIFE from Planetlab and DIFE using as landmarks for the last one the same set of Planetlab nodes of DIFE.

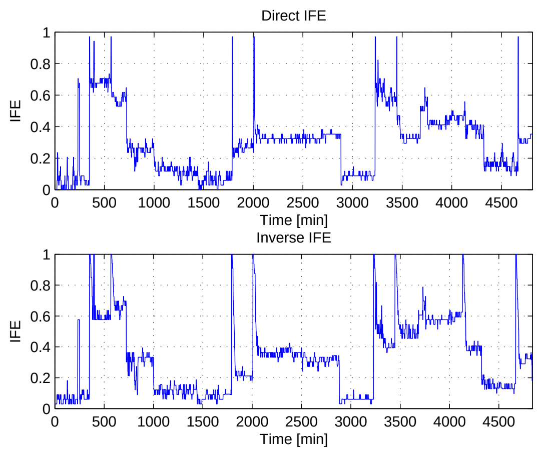


Figure 5.12: *Importance of the Access Tree Routers Coverage in the Inverse IFE.*

We see that the results are pretty similar.

In the case of the Figure 5.13 we used the same node in Argentina as Monitored Point. This time we use random landmarks in the Monitored Point (to perform Direct IFE) and we use Planetlab nodes in Inverse IFE mode.

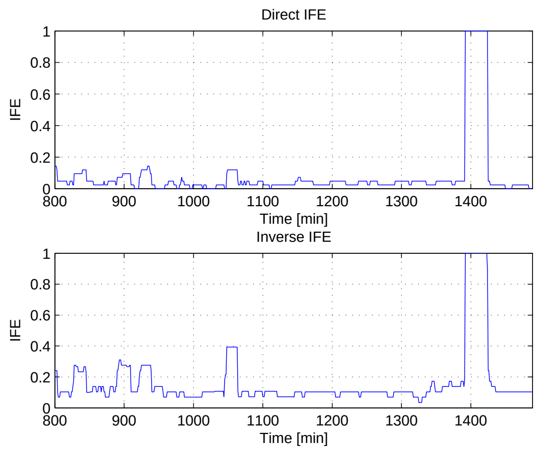


Figure 5.13: *Direct versus Inverse IFE, in Argentina. Direct IFE is performed with random landmarks, Inverse IFE is performed from Planetlab nodes.*

The huge anomaly at 1400 was a denial of service from inside (using huge and several ping packets). The reason why it is perceived from outside is simple, the router that has the tested public IP got overwhelmed and its processing capacity is provided by the same processor. It is to show that not all the local activity will be hidden from DIFE from Planetlab. However cases like bad WIFI connectivity causes irregular RTT, but only locally without loading the router. Other local circumstance not perceived from outside happens when more than one local router connect the Monitored Point to Internet. In this case local traffic is more isolated for IIFE, but clearly has an impact in DIFE.

However, the most important cases come here. Note that anomalies starting at minutes 830, 890, 920, 1050 are registered for both IFE calculations. This means that the abnormal links were traversed for both Inverse and Direct IFE probes. Note also that the IFE amplitudes differ a bit, this difference is due to several things:

- Inverse and Direct IFE here have not an exactly similar access tree (check it in Figure 5.8, Case 4 in Argentina).
- Inverse IFE does not go through the local network, which means that its measurements might be clearer (think about local WIFI contaminating DIFE measurements). The fact of having clearer measurements gives to the Inverse IFE algorithm a stronger fundament to tell that a particular situation is abnormal or not.

We can clearly see that the results are very correlated, and we understand why these differences are there. All these experiments show that the better the results of the access tree experiment, the more likely we are to get similar DIFE and IIFE from Planetlab.

5.2 Inverse IFE On The Fly

The Figure 5.1 represents a possible layout of the IIFE tool from Planetlab. Since we saw that this approach (given a good access tree comparison result) can be really representative of the original DIFE, we implemented that architecture, but extending it to make it work **on the fly**. It means that there are currently more than 30 Planetlab nodes probing a set of Monitored Points, set that is specified in a common server with another parameters.

Chapter 6

Conclusions

As conclusion for this work we enumerate a set of points.

- The new pipeline based software was tested and effectively let us easily change element by element without messing existing pipeline elements. Thanks to this we implemented innovative elements such that:
  - Campaign Generator 2G element, whose benefits' explanation is below.
  - Campaign dumper element (embedded in Campaign Generator 2G element) and Campaign dump reader, both pipeline elements allowed us to work offline and process once and again the results obtained from Planetlab without need of running the real experiment again on such platform.
  - Inverse dump reader, element that reads several dump files from Planetlab nodes behaving as landmarks. It delivers to its next pipeline element exactly the information that a Campaign Generator would deliver when doing the pings in real-time.
- We re-implemented the measurements module as the new Campaign Generator 2G element. It allowed us to execute concurrently several ping measurements for each campaign, reaching an accurate IFE based on 40 or more nodes, using 4 pings for each landmark and 3 seconds of timeout, and a **campaigns' periodicity of 1 minute**. We remind that we improved the original situation, where we were having an scenario that allowed us to generate the same kind of campaigns with at most 1 campaign every 8 minutes of periodicity.
- We studied the current Path Anomaly Detection algorithm, and since we determined that it was not considering important points we designed two new algorithms, showed the comparison study between them, and made a decision picking up one. Our simulations show that the selected algorithm performs well, reaching in both wired and WIFI simulations an accuracy of at least 95% detecting normal an abnormal cases.
- We tested the current implementation, and we showed cases in which our tool helps us to understand better where the anomaly has its source. We performed studies in Planetlab showing how sensible the IFE results are to the amount of landmarks used, and also its sensibility to the particular set of landmarks chosen.
- We completed a study of feasibility of Inverse IFE from Planetlab, showing that the access tree inclusion (routers in the path towards random landmarks included in routers in the path towards Planetlab landmarks) on the way Monitored Point-landmarks has a strong influence in the similitude between DIFE and IIFE from Planetlab. In the future, before

starting running the Inverse IFE towards certain Monitored Point, an access tree study could be done before, in order to tell the user how much representative will IIFE from Planetlab be with regards to the original DIFE.

- We implemented the Inverse IFE architecture **on the fly**. This drastically decreased the experimentation time (no need to collect data from Planetlab nodes), and its lack of need of installation in the Monitored Points represents a starting point for new users to become curious and more interested about this, the measurement of their Internet connectivity.

Bibliography

[1] Cascella R., Barakat C. *Estimating the access link quality by active measurements.* ITC 22 (2010).

[2] Barford P., Duffield N., Ron A., Sommers J. *Network performance anomaly detection and localization.* IEEE INFOCOM (2009).

[3] Lakhina A., Crovella M., Diot C. *Characterization of networkwide anomalies in traffic flows.* ACM IMC (2004).

[4] Dhamdhere A., Teixeira R., Dovrolis C., and Diot C. *Netdiagnoser: troubleshooting network unreachabilities using end-to-end probes and routing data.* ACM CoNEXT conference, USA (2007).

[5] Ringberg H., Soule A., Rexford J., Diot C. *Sensitivity of PCA for Traffic Anomaly Detection.* Proc. of ACM SIGMETRICS (2007).

[6] Soule A., Salamatian K., Taft N. *Combining Filtering and Statistical Methods for Anomaly Detection.* Proc. of IMC Workshop (2005).

[7] Hussain F., Kalim U., Latif N., Khayam S. *Using End-to-End Bandwidth Estimates for Anomaly Detection beyond Enterprise Boundaries.* (n.d.)

[8] DiCioccio L., Teixeira R., Rosenberg C. *Impact of Home Networks on End-to-End Performance: Controlled Experiments.* (n.d.)

[9] Kreibich C., Weaver N. *Netalyzr: Illuminating The Edge Network.* IMC'10 (2010).

[10] Cunha ., Teixeira R., Feamster N., Diot C. *Measurement Methods for Fast and Accurate Blackhole Identification with Binary Tomography.* IMC'09 (2009).

[11] Thottan M., Liu G., Ji C. *Anomaly Detection Approaches for Communication Networks*

[12] Banerjee S., Griffin T., Pias M. *The Interdomain Connectivity of PlanetLab Nodes.* (2004).

[13] Spring N., Peterson L., Bavier A., Pai V. *Using PlanetLab for Network Research: Myths, Realities, and Best Practices.* (n.d.) URL: http://nsg.cs.princeton.edu/publication/myths_worlds_05.pdf

[14] Banerjee S., Griffin T., Pias M. *The interdomain connectivity of PlanetLab nodes.* PAM (2004).

[15] Park K., Pai V. *CoMon: A monitoring infrastructure for PlanetLab.* http://comon.cs.princeton.edu.

[16] Floyd S., Paxson V. *Difficulties in simulating the Internet.* IEEE/ACM Transactions on Networking (2001).

[17] http://www.planet-lab.org/ . Jun 20th, 2011.

[18] http://svn.planet-lab.org/wiki/HelloWorldTutorial . Jun 22nd, 2011.

[19] http://www.grenouille.com/

Acknowledgements

I thank my Dear Advisor Prof. Chadi Barakat who was willing to help me in every moment I had problems in the understanding of the addressed topics, and more than that, he let me feel at work as if I were at home. Thank you, it was really a pleasure working with you.

Thanks also to all the Professors that taught me the subjects that let me write this report, and learn all the things I have learned here. To my colleagues, Ubinetters 2010, I will never forget the experience I shared with you all.

To my Argentinean family, ”Papi y Mami”, my Argentinean friends, and my dear friend Michael.

Finally, Fernando, why did you leave us... This is for you my Dear Friend.