IBTrack: An ICMP Black holes Tracker

Ludovic Jacquin, Vincent Roca, Mohamed Ali Kaafar, Fabrice Schuler
Inria, France
{ludovic.jacquin, vincent.roca, mohamed-ali.kaafar, fabrice.schuler}@inria.fr

Jean-Louis Roch
LIG, France
jean-louis.roch@imag.fr

***Abstract*—`ICMP` is a key protocol to exchange control and error messages over the Internet. An appropriate `ICMP`'s processing throughout a path is therefore a key requirement both for troubleshooting operations (e.g. debugging routing problems) and for several functionnalities (e.g. Path Maximum Transmission Unit Discovery, PMTUD). Unfortunately it is common to see `ICMP` malfunctions, thereby causing various levels of problems. The contributions of this paper are threefold. We first introduce a taxonomy of the way routers process `ICMP`, which is of great help to understand for instance certain `traceroute` outputs. Secondly we introduce IBTrack, a tool that any user can use to automatically characterize `ICMP` issues within the Internet, without requiring any additional in-network assistance (e.g. there is no vantage point). Finally we validate our IBTrack tool with large scale experiments and we take advantage of this opportunity to provide some statistics on how `ICMP` is managed by Internet routers.**

I. Introduction

The Internet Control Message Protocol (`ICMP`) is one of the main protocols used on Internet and more generally on `IPv4` and `IPv6` networks. It is responsible for exchanging control and error messages over the network, like routing control or packet processing error notifications.

`ICMP` is also implied in several functionalities, like the Path Maximum Transmission Unit Discovery (PMTUD) mechanism [1], [2], [3], which has been developed to avoid `IP` fragmentation. For instance, in `IPv4`, the sender sets the *don't fragment* bit[1]. If a router cannot transmit the packet because of its size, it must send back to the source an `ICMP` "Too Big" (type 3, code 4 with `IPv4` and type 2, code 0 with `IPv6`) packet. Iteratively, the source will lower the packet size it uses in order to match the lowest MTU on a path. The importance of a well-chosen PMTU has been discussed in several previous works, and one key aspect to consider from a router perspective is the number of packets per second to handle[4], [5]. In essence, using the highest possible PMTU value can result in a significant bandwidth improvements [6] since the packets treatment overhead remains the same regardless of the packet's size. This requires having a fully functional `ICMP` service.

Our study aims at analyzing issues related to `ICMP` packet processing by routers, and in particular so-called "ICMP black holes". The tool presented in this paper, IBTrack, provides users with a thorough analysis of the routers behavior that lie along the path from a source host controlled by the user and any given destination (that is not assumed to be under control). In particular, IBTrack characterizes the routers forging of `ICMP` error packets and their transport the way back to the source host. It is important to note that IBTrack is designed to be a "lightweight" tool that only relies on measures performed by the end user, at the source, without any additional assumption: there is no external monitoring or vantage point, nor any collaboration with the destination. We further require that IBTrack does not involve any long-term measurement, so it could be used in a timely basis, when a user needs to establish a diagnosis and locate the possible problem origins. It is similar in spirit to the `ping` tool.

Let us now consider some related works. In [7], authors describe Reverse traceroute a tool that can be used to perform measurement (using `ICMP`) at the level of Internet routers back to a specified host. The tool is capable of examining the router-level topology and does not look into the behaviors of routers to forge different types of `ICMP` packets when needed. Authors of Reverse Traceroute also assume users do have access to multiple vantage points distributed across the Internet. Our approach is similar to the concept explored in [8]. The authors propose a system, called Hubble, which detects IP-level black holes at a global level of IP connectivity. Besides the fact that Hubble needs periodic probing campaigns of the Internet, something that IBTrack avoids, our tool differs in the sense it tries to characterize the reasons for each `ICMP`-connectivity issue and it provides a fine-grained analysis.

The rest of this paper is organized as follows. Section II presents the model used to describe routers and the assumptions made in our work. section III describes the IBTrack algorithms. Section IV focuses on the methodology and tools used for our measurements. Section V presents the results from both coarse-grained Internet and fine-grained ISP levels tests. Finally section VI concludes the paper.

II. Assumptions and Taxonomy

In this section, we define the terminology used, we detail the restrictions implied by IBTrack's main goals, and we introduce a router property taxonomy.

A. Path definitions

Let us consider a path from a source S to a destination D, and a router R along this path (there are typically several routers and R is one of them). We assume that the packet flow generated by S and destined to D uses the probing protocol PP. More specifically in this work we consider

[1]This is useless in IPv6 since fragmentation is not supported any more.

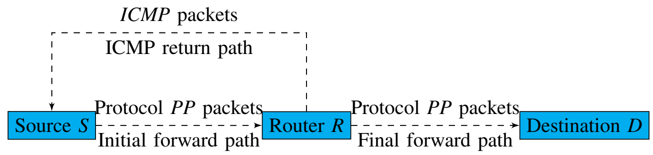

Fig. 1: The various path definitions

either `ICMP/IP`, `UDP/IP`, or `TCP/IP`, where `IP` denotes either `IPv4` or `IPv6`. Figure 1 illustrates the following definitions:

- the *initial forward path* is the path between *S* and *R*;
- the *final forward path* is the path between *R* and *D*;
- and the *ICMP return path* is the path taken by `ICMP` packets destined to *S* and either generated by *R* or by a router on the final forward path and that go through *R*;

It is important to note that the initial forward paths and the return `ICMP` paths are not necessarily identical: routers often use different forwarding strategies depending of routing policies. The forward initial or final paths also potentially depend on the nature of *PP*, since routers often use protocol-dependant forwarding rules. Finally, there are potentially as many `ICMP` return paths as the number of routers on the forwarding paths.

B. Assumptions

Our algorithms only consider information gathered by the source. We deliberately choose not to rely on external information that may come from vantage point or from the destination.

We assume that the routing only depends on the packet's protocol and destination. This is a strong assumption that is however reasonable in the current Internet.

Furthermore, the property used to characterize a router is considered as *global*, independently from the router's IP interface that has been used during the measurement inferring this property. In other words, we assume that all IP interfaces of that router behave identically.

Finally, if we ignore `ICMP Echo Response` messages, the only `ICMP` return packet type we can trigger from any router[2] is the `Time Exceeded` type. During our measurements, we will examine `ICMP` return paths only by using `ICMP Time Exceeded` packets.

C. Router properties taxonomy

Let us now consider a certain "Probing Protocol" *PP* used by the source (e.g. `traceroute` uses `UDP` probes by default). For this protocol *PP*, each router along the given path is characterized by the following three key properties:

[2] A packet whose IP field "Time To Live" or "Hop Limit" turns to zero on a router should trigger an ICMP "Time Exceeded" message.

1) Property P1: "*R* forwards all packets of type *PP* towards *D*".
Said differently, at *R*, each incoming packet on the initial forward path is forwarded on the final forward path. This property, of course, does not imply that these packets reach destination *D*.

2) Property P2: "*R* is cooperative for packets of type *PP*". In case *R* should send an `ICMP` packet back to the source, either because of an error (e.g. a packet that exceeds the forwarding link MTU) or because the packet asks for a reply (e.g. in case of an `ICMP Echo Request`), then the `ICMP` packet is correctly initialized and sent by *R* on the `ICMP` return path.

3) Property P3: "`ICMP` packets are not filtered by *R* nor by any router on the `ICMP` return path from *R*".
When *R* emits an `ICMP` packet on the return path, this `ICMP` packet is routed all the way back to *S*. This property implies that this `ICMP` packet arrive to *S*, i.e. they are not filtered by any router on the `ICMP` return path from *R*, which is a strong property. In the particular case where the return path contains one or several routers that are also part of the forward path, this property on R implies the same property on these routers.

The router behavior can now be expressed as three logical expressions, one for each property. For instance, for a packet type *P*: `P1.(P2.!P3 + !P2)` means that router *R* forwards these packets and is either cooperative (i.e. sends `ICMP` packets to *S*) but these `ICMP` packets are filtered on the `ICMP` return path, or is non cooperative (i.e. does not generate `ICMP` packets back to *S*).

III. The ICMP black holes tracking (IBTrack) tool

In this section we describe the algorithms used by our IBTrack tool. Our goal is obtain the best possible knowledge of the routers behavior by considering only the initial traffic we generate and the backward `ICMP` traffic we observe. In a first step, we only consider a given Probing Protocol *PP* to determine the router's behavior. In a second step, we explain how one can sometimes refine this analysis by crossing the results achieved independently with several probing protocols.

A. IBTrack base algorithm

The base algorithm is described in figure 2. Let us consider a packet we generate at *S*, destinated to *D*, and crossing one or several routers. Let *R* be one of the router(s). Initially we have no knowledge at all for router *R*, which corresponds to the state at the top of the graph. In that case each property is "undecided" (i.e. it can be true or false), which is expressed by: `P + !P`.

The algorithm is composed of two main steps:

- test the forwarding path through *R*;
- test the `ICMP` return path from *R*;

We now detail these two steps.

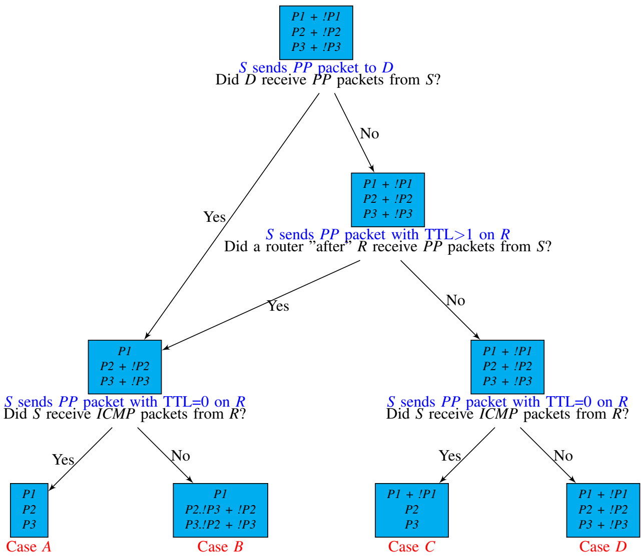

Fig. 2: Base algorithm for router's behavior characterization

1) Forwarding path through R: We first want to test the `P1` property (forwarding property towards *D*). It can be verified in two ways: either the destination replied, which implies that it has received an incoming packet, or we can assert that a router on the final forward path received an incoming packet (i.e. source *S* received an `ICMP` packet from a router after *R*).

2) ICMP return path from R: We now want to check the `P2` and `P3` properties. For the `P2` property we must verify that the router forges the `ICMP` error packet and emits it on its `ICMP` return path. This could be tested by monitoring the `ICMP` return path. Nevertheless, as we do not control any router of the `ICMP` return path, we only rely on the reception by the source of the `ICMP` error packet. For the `P3` property, since it concerns the `ICMP` return path itself, it can only be evaluated with the reception of `ICMP` error packets by the source.

3) Classification details: We now present the result of this classification, which consists in four cases, depending on the properties that could be verified or not.

a) Case A: `P1.P2.P3`
This is the "ideal" router (at least for protocol *PP*) since it forwards the initial packets and forges `ICMP` error packets if need be that all arrive to the source.

b) Case B: `P1.(P2.!P3 + !P2).(P3.!P2 + !P3)` (or more simply `P1.!(P2.P3)`)
`P1` is verified, but the interesting part of this case comes from the `P2` and `P3` properties. Upon receiving a packet whose TTL (or "Hop Count" with `IPv6`) is decremented to 0 by *R*, either *R* does not generate any `ICMP` error packet, or this packet is generated but does not arrive to *S*. Said differently:

- `P2` ⇒ `!P3`: the `ICMP` error packet is forged but is blocked along the `ICMP` return path.
- `P3` ⇒ `!P2`: the `ICMP` return path can forward `ICMP` packets correctly, however the router *R* did not not forge

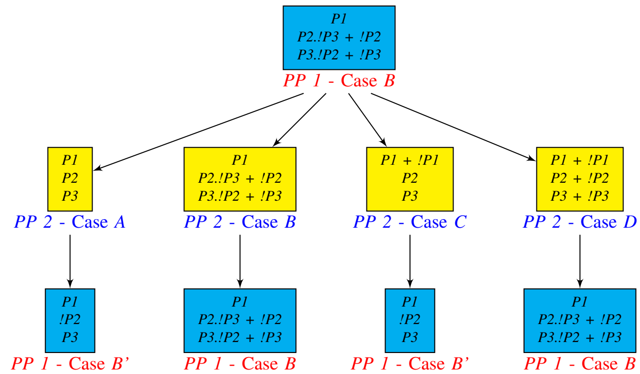

Fig. 3: Refining case *B* with another probing protocol

any such packet.

c) Case C: `(P1 + !P1).P2.P3`
The source gets the `ICMP` error from the router, but we have no clue that a router on the final forward path received any initial packet.

d) Case D: `(P1 + !P1).(P2 + !P2).(P3 + !P3)`
For these routers, we cannot infer anything because the source did not get back any data.

This base algorithm is applied to all the router in a path (defined by its source and destination) for protocol *PP*, by increasing the TTL value It results in a collection of cases (*A*, *B*, *C* or *D*) for each router along the forward path.

B. IBTrack refining algorithm

Let us now refine this base algorithm, that naturally depends on the probing protocol, *PP*. First of all we notice that the `P3` property concerns exclusively the `ICMP` return path, which is the same independently from the probing protocol *PP* [3]. Thus, for a given router *R*, we can use the deductions made on the `P3` property using a probing protocol *PP1* to refine the deductions made using another probing protocol *PP2*. This is the key idea of our refining algorithm. However, in order to be able to apply this refinement, we must first identify the same router for both probing protocols, whereas we do not necessarily have its identity (it can be a wildcard "*" in a `traceroute` trace). This aspect is addressed in section IV-C, and for the moment we assume that this assertion is true.

The combination of the results of the algorithm for two distinct probing protocols is detailed in figures 3 and 4, for router *R*. Since we can only use deductions for the `P3` property, cases *A* or *C* cannot be refined (property `P3` is already known in that case). Therefore we focus on cases *B* and *D*.

1) Initial case B for protocol PP1 (Fig.3): This is the most interesting case. When property `P3` is verified with the second protocol *PP2*, from the second conditional we can

[3] This is a direct consequence of our assumption that routing only depends on the protocol along the return path, here ICMP, and the destination, here *S*.

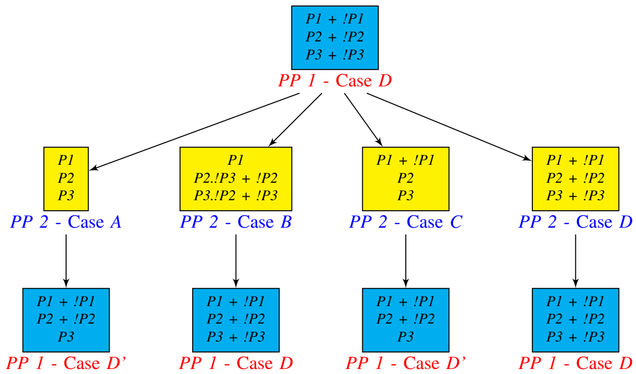

Fig. 4: Refining case *D* with another probing protocol

ICMP	UDP
15 IP_R15 16 * 17 IP_R17	15 IP_R15 16 * 17 IP_R17

(a) two *B* cases

ICMP	UDP
6 IP_R6 7 IP_R7 8 IP_R8	6 IP_R6 7 * 8 IP_R8

(b) *A* case and *B* case

Fig. 5: Two examples of multiple probing protocol results

deduce that `P3` $\Rightarrow$ `!P2`. Thus, if probing with protocol *PP2* ends in cases *A* or *C*, we can conclude that the `P3` property is verified for protocol *PP1*, which implies that `P2` is not. This creates a subcase for *B* that we call *B'*: `P1.!P2.P3`.

2) Initial case D for protocol PP1 (Fig.4): This case can only be refined on the `P3` property itself since there is no material implication or link with another property. This only happens when we end in cases *A* or *C* with protocol *PP2*.

3) Refining using more than two probing protocols: It is of course possible to use more than two probing protocols to refine the results obtained with *PP1*. Here also, the constraint is that the paths must be the same for every combination of probing protocols in order to consider that the `P3` property is common.

C. Examples

Figure 5 illustrates the refining algorithm through two examples extracted from our experiments.

In the `traceroute` outputs of figure 5a, the 16^{th} router is a *B* case for both protocols since: (i) it forwards the initial protocol (the 17^{th} router replied which implies that it has received at least one initial packet); but (ii) no `ICMP` return packet is received by the source. There is no refining possible in this case and: `P1.!(P2.P3)`

In the `traceroutes` outputs of figure 5b), the 7^{th} router is a *B* case for `UDP` probing whereas it is an *A* case for `ICMP` probing (the `ICMP` return packet is received by the source). As the `ICMP` return path is the same (source, protocol and destination are identical), we deduce that for `UDP` packet the router does not forge the `ICMP` return packets. The categorization for `UDP` probing is therefore refined to the *B'* case: `P1.!P2.P3`

The formal model and algorithms being described, the next section will focus on the methodology and tools used for Internet measurements.

IV. Measurement Methodology

Before describing the tools and methodology used during the measurements, we detail how we chose the `IP` addresses to probe.

A. Selection of destination's IP

To probe a meaningful set of the Internet, we used a CAIDA[9] snapshot of the routed `/24 IPv4`. This snapshot is composed of a part of the routed `/24` (approximately 10,000) and gives for each a random IP in the `/24`. It is important to notice that there is not necessarily a running host behind each `IP` in the snapshot.

B. Tools

To perform probing, we used `scamper`[10] that implements, among others, the `paris-traceroute`[11] functionality over the Planet-Lab[12] nodes.

1) Paris-traceroute: This is an improved version of the classical `traceroute` tool. In particular this tool reveals more routers and links over the path explored and removes some false links inferred by the usual `traceroute`. This is therefore highly beneficial to our needs.

2) Planet-Lab: This is a world wide network aiming to help academic and industrial researchers focusing on new network services. In our test, it provides multiple servers across the world that we use to probe the same destination set of `IP` addresses. Therefore it helps us improve the coverage of the possible paths to the selected set of `IP` addresses.

3) Scamper: This is a program that implements most of the classical Internet measurement tools (like `ping` or `paris-traceroute`) that can be launched to run in parallel. In particular it is developed to run on Planet-Lab nodes.

One of the features of `scamper` we use is that after five unidentified routers (i.e. not responding, thus appearing as "*" in the `traceroute` output) it stops. Figure 6c shows a `traceroute` terminating like this. This situation can result from different causes, like for instance a single router filtering packets, or multiple routers not replying. Nevertheless, we will deliberately take into account only the first "*" during our tests.

C. Identifying common path

One of the main challenges before applying the refining algorithm which uses multiple probing protocols is to be able to match paths that seem to be different. In principle the

Protocol 1	Protocol 2	Protocol 1	Protocol 2
1 R1	1 R1	1 R1	1 R1
2 *	2 R2	2 *	2 R2
3 *	3 R3	3 *	3 R3
4 *	4 R4	4 R4	4 *
5 *	5 R5	5 R5	5 *
6 R6	6 R6	6 R6	6 R6

(a) Two pairs of traceroute outputs considered as the same path

Protocol 1	Protocol 2
1 R1	1 R1
2 R21	2 R22
3 *	3 R3
4 R41	4 R42
5 *	5 R5
6 R6	6 R6

(b) Two traceroute outputs considered as the same path when allowing two different routers

ICMP
12 198.108.23.21
13 *
14 *
15 *
16 *
17 *

(c) Example of a traceroute finishing with five unidentified routers

Fig. 6: Examples of the path matching algorithm and scamper output

presence of an unidentified router on a path (i.e. a "*" in the `traceroute` output) prevents path matching for the final forwarding path. But this situation must be improved. Indeed, if the router has to be identified to allow the application of our refining algorithm, then we would only perform refining between *A* and *C* cases, which is of little interesting because these cases already share the same state for the property `P3`.

1) Handling unknow routers: The first and straightforward assumption we make consists in ignoring unknown routers from a path when matching two paths with the same source and destination but with different probing protocols.

Formally, there is no proof for this assumption, but we are convinced that given the maximum number of consecutive unknown routers we allowed (5 before `traceroute` halts) for a protocol, if two paths share the same number of routers and if the identified routers are the same, then the two paths are most likely the same.

For instance, figure 6a exhibits two situations considered as belonging to the same path by the path matching algorithm.

2) Handling tunnels: We also wanted to add to the path matching algorithm the capability to identify tunnels. More particularly, we want to isolate path divergences due to *per-flow* routing (for example through `MPLS` tunnel[13]). Therefore, we improved our algorithm by allowing some consecutive identified routers to be different on the same path. This feature adds itself to the previous one; thus "consecutive identified routers" do not involve potential unknown routers. Nevertheless, the number of routers and their position on the path must remain the same.

Figure 6b illustrates two paths being considered as identical by the path matching algorithm when allowing two different consecutive identified routers.

The following section presents large scale measurements

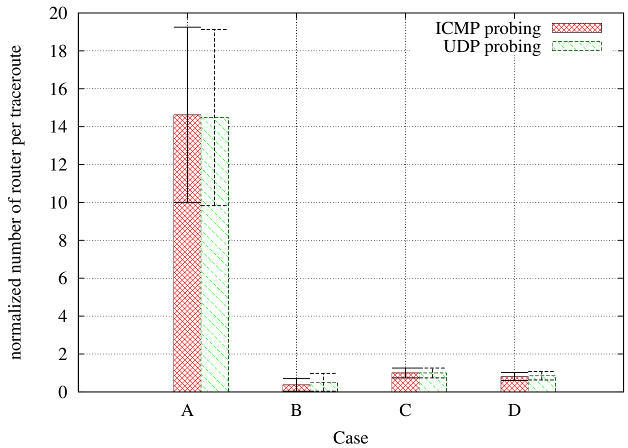

Fig. 7: Normalized (per traceroute) number of routers per case (with standard deviation)

gathered from Planet-Lab nodes and a fine-grained analysis of the algorithm results.

V. Measurement results

We first detail the results obtained from 30 Planet-Lab nodes to a set of CAIDA's routed `/24` composed of more than 10,000 `IP` addresses. Then we detail the results obtained from a typical ISP user probing the same set of `IP` addresses, using `ICMP`, `UDP`, but also `TCP`.

A. Coarse-grain Internet results

1) Paths characterization: figure 7 represents the normalized number of router per case. On average, a `traceroute` is composed of 16.78 routers distributed as followed:

- *A* case - 14.5 routers
- *B* case - 0.45 router
- *C* case - 1 router
- *D* case - 0.83 router

Figure 8 shows the distribution of cases depending on the distance between the router and the initial source. Note that when handling a "premature end" of a `traceroute` (i.e. when finishing with five "*"), the *D* cases are only counted once (it corresponds to the closest router from the source) as explained in section IV-B3.

This figure shows that the closer a router is to the source, `S`, the mostly it will be of case *A*. On the opposite, the further a router is, the highest the probability to fall in *C* and *D* cases, which in turn comes at the expense of a lower probability of the router would be in case *A*. Perhaps not surprisingly, this shows that the longer the path is the higher the chances the routers would behave according to the *D* unexplicative case.

2) Routers characterization: figure 9 summarizes the classification of routers encountered during the probing of every `IP` of our destination set, from each Planet-Lab nodes, using the `ICMP` and `UDP` protocols. We remind that for the *D* case, results are divided by 5 due to the "premature end" of a traceroute (section IV-B3). In opposition to the two previous

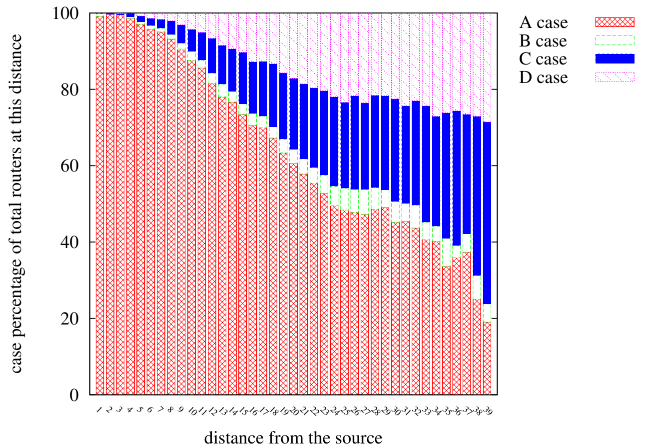

Fig. 8: Case percentage with regards to the distance to the source S

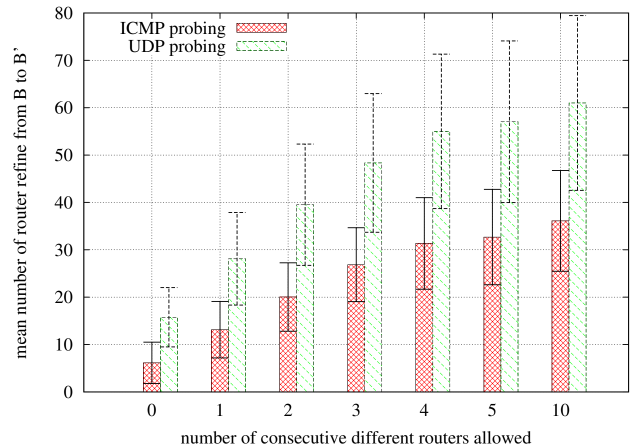

Fig. 10: Mean number of router refined from *B* to *B'* case depending of the path matching algorithm configuration (with standard deviation)

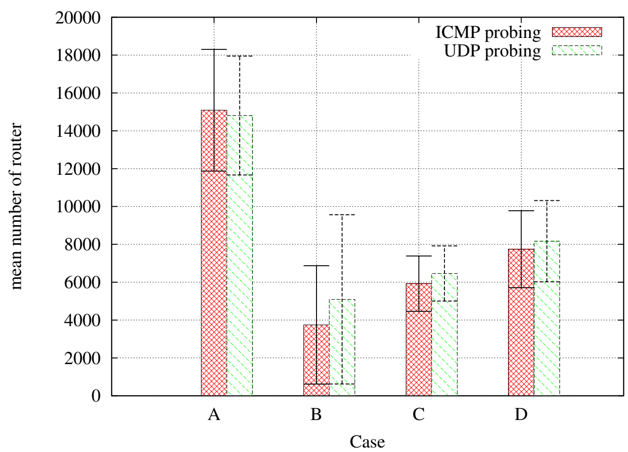

Fig. 9: Mean number of routers per case (with standard deviation)

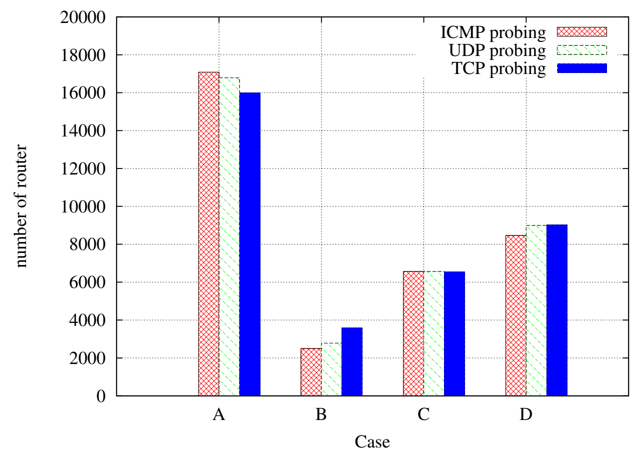

Fig. 11: Number of routers per case with 3 probing protocols

figures, we add that the router seen are counted only once as we want to survey the Internet's average router behavior (previously we were focusing on the path instead).

Figure 10 details the results of the refining algorithm applied to the Planet-Lab outputs. We first configured the path comparison algorithm to perform an exact path matching and then changed it to allow a certain number of different routers along the path. With a strict path matching configuration, we refine 0.16% (`ICMP`) and 0.31% (`UDP`) of the *B* case. This ratio grows up to 0.87% (`ICMP`) and 1.12% (`UDP`) when allowing up to five different consecutive identified routers. This measure (particularly the small difference when allowing five or ten different routers) can be associated to the fact that more than 90% of `MPLS` tunnels (which are present in at least 30% of Internet's traceroute) are at most five hops long[13].

B. Fine-grain results

Let us now consider the results obtained by a single user using a single host, connected through a given ISP. The set of `IP` addresses is the same, and we use the `ICMP`, `UDP`, but also `TCP` probing protocols.

Figure 11 presents the results of the classification algorithm applied on the three probing protocols`ICMP`, `UDP` and `TCP`. Although the general distribution is globally the same, there are more *A* cases and fewer *B* cases than with Planet-Lab results (figure 9).

Figure 12 details the refining algorithm used for `TCP`, which refines `TCP` results either using `ICMP` solely, or `UDP`, or both probing protocols. It illustrates well the fact that, when using multiple probing protocols to refine a single other protocol-based properties output, the results are lower than what could be expected from the sum of the refining using single protocols outputs, as explained in section III-B3. Indeed, we observe that the improvement of the refining algorithm while using two protocols to refine a single one is small. For instance, when allowing five different consecutive identified routers, the improvement is: 5.2% when comparing `ICMP` to `ICMP+UDP`, and 14.5% between `UDP`

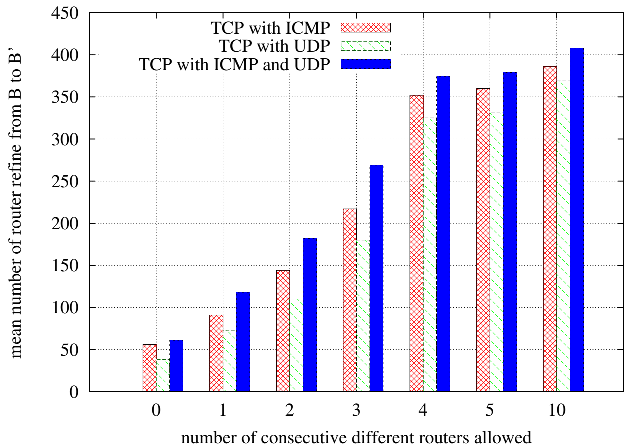

Fig. 12: Refining of TCP probing protocol *B* case

and `ICMP+UDP`. We interpret this at a router level by bearing that there is an explicit policy on these routers that forbids it to forge `ICMP Time Exceeded` packets when the initial protocol is `TCP`.
Nevertheless, when using both `ICMP` and `UDP` to refine the *B* case of `TCP`, we see that we are able to completely specify the behavior of 10% of the routers encountered. Globally, we manage to refine the classification of 1% of the routers seen.

VI. Conclusion

This work has investigated the problem of `ICMP` black holes tracking. Our contributions are threefold: we first introduced a taxonomy for router properties, regarding `ICMP`. We have shown that three fundamental properties are sufficient to classify Internet routers with respect to the `ICMP` protocol processing.

Secondly we have designed the IBTrack tool to automatically characterize `ICMP` issues within the Internet. This tool classifies the routers of a path, using a given probing protocol, into four cases. These cases represent the possible logical relationships between the three fundamental properties. The tool also introduces a refinement procedure, by using two or more probing protocols, that enables to better classify the routers. Our proposal does not require any control within the Internet or at destination, which means it can be immediately used by anybody.

Third, we performed several large scale experiments in order to validate our IBTrack tool and methodology in real conditions, but also in order to have insights on the way Internet routers currently process `ICMP`.

We therefore believe that this work is a significant step forward. In particular it will help many network administrators to better understand `traceroute` outputs and to debug complex `ICMP` related problems within Internet routers.

This work deliberately makes very minimalist assumptions on what users might control while processing and diagnosing network failures messages (only the source is controlled). Therefore the proposal can easily be extended by relaxing some constraints (e.g. if we assume the user also controls the destination) or by revisiting some of our assumptions on network functionalities (e.g. if we add vantage points).

References

[1] "Rfc 1191: Path mtu discovery," http://datatracker.ietf.org/doc/rfc1191/.

[2] M. Luckie, K. Cho, and B. Owens, "Inferring and debugging path mtu discovery failures," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, ser. IMC '05. Berkeley, CA, USA: USENIX Association, 2005, pp. 17–17. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251086.1251103

[3] M. Luckie and B. Stasiewicz, "Measuring path mtu discovery behaviour," in *Proceedings of the 10th annual conference on Internet measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 102–108. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879155

[4] N. Egi, M. Dobrescu, J. Du, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, L. Mathy, and S. Ratnasamy, "Understanding the packet processing capabilities of multi-core servers," 2009. [Online]. Available: http://infoscience.epfl.ch/record/134539

[5] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "Routebricks: exploiting parallelism to scale software routers," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, ser. SOSP '09. New York, NY, USA: ACM, 2009, pp. 15–28. [Online]. Available: http://doi.acm.org/10.1145/1629575.1629578

[6] "Myricom 10-gigabit ethernet performance measurements," http://www.myricom.com/scs/performance/Myri10GE/.

[7] E. Katz-Bassett, H. V. Madhyastha, V. K. Adhikari, C. Scott, J. Sherry, P. van Wesep, T. E. Anderson, and A. Krishnamurthy, "Reverse traceroute," in *NSDI*, 2010, pp. 219–234.

[8] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, D. Wetherall, and T. Anderson, "Studying black holes in the internet with hubble," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 247–262. [Online]. Available: http://dl.acm.org/citation.cfm?id=1387589.1387607

[9] Y. Hyun, B. Huffaker, D. Andersen, E. Aben, C. Shannon, M. Luckie, and kc claffy, "The caida ipv4 routed /24 topology dataset," http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml, november 2011.

[10] M. J. Luckie, "Scamper: a scalable and extensible packet prober for active measurement of the internet," in *Internet Measurement Conference*, 2010, pp. 239–245.

[11] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding traceroute anomalies with paris traceroute," in *Internet Measurement Conference*, 2006.

[12] "Planet-lab," http://www.planet-lab.org.

[13] B. Donnet, M. Luckie, P. Mérindol, and J.-J. Pansiot, "Revealing mpls tunnels obscured from traceroute," in *SIGCOMM Computer Communication Review*, vol. 42, no. 2, Apr. 2012.