Proceedings of the Federated Conference on Computer Science and Information Systems pp. 433–440
DOI: 10.15439/2019F59
ISSN 2300-5963 ACSIS, Vol. 18

Whose Fault is It? Correctly Attributing Outages in Cloud Services

Matteo Adriani
Dpt. of Civil Engineering and Computer Science
University of Rome Tor Vergata
Rome, Italy
matteo.adriani93@alice.it

Maurizio Naldi
Dpt. of Civil Engineering and Computer Science
University of Rome Tor Vergata
Rome, Italy
Dpt. of Law, Economics, Politics and Modern Languages
LUMSA University
maurizio.naldi@uniroma2.it
m.naldi@lumsa.it

***Abstract*—Cloud availability is a major performance parameter in cloud Service Level Agreements (SLA). Its correct evaluation is essential to SLA enforcement and possible litigation issues. Current methods fail to correctly identify the fault location, since they include the network contribution. We propose a procedure to identify the failures actually due to the cloud itself and provide a correct cloud availability measure. The procedure employs tools that are freely available, i.e. traceroute and whois, and arrives at the availability measure by first identifying the boundaries of the cloud. We evaluate our procedure by testing it on three major cloud providers: Google Cloud, Amazon AWS, and Rackspace. The results show that the procedure arrives at a correct identification in 95% of cases. The cloud availability obtained in the test after correct identification lies between 3 and 4 nines for the three platforms under test.**

I. Introduction

Availability is a major Quality of Service descriptor in cloud services, and an essential component of Service Level Agreements [1]–[3].

Many efforts have been devoted to understanding and improving the availability of cloud systems. The relevance of the issue has been re-stated very recently by Varghese and Buyya, which list it among the top research directions, mentioning the 49-minute outage suffered by Amazon, which cost the company more than $4 million in lost sales, as an indicator of the economic importance of achieving a high availability [4]. The same concept had been voiced in [5], where the authors even propose to consider a *Reliability as a Service*, where reliability is a parameter that users can specify and a service by itself, rather than the random state of a cloud-based service. Concerns for the legal implications that may arise due to a less-than-adequate cloud reliability have been recently expressed in [6].

An analysis of the main causes of cloud failures has been carried out in [7], where growth trends are also identified, and [8], where mechanisms are subsequently discussed to minimize the impact of outages. Some papers have focussed on the analysis of the cloud architecture to get a high availability by design [9]–[11]. A different approach has been taken in [12] and [13], where machine learning technique have been employed to predict cloud outages (and react accordingly).

If we switch from the perspective of a cloud designer to that of a cloud user, the main interest lies in understanding if the cloud is performing up to the expectations. Setting up, or employing the services of, a cloud monitoring platform is essential in this respect. Several architectures have been proposed for that purpose, e.g. in [14]–[16], and a recent review is contained in [17].

Unfortunately, very few attempts have been done to actually measure cloud availability from a third party vantage point. An early attempt based on users' reports has been reported in [18]. The shortcoming of that approach is that the starting time of the outage may not be reported correctly, since a time lag is always present between the time an outage occurs and the time a user first reports it. The ending time of the outage may be also reported wrongly, since most users do not take on themselves to report it, and we have to rely on the cloud provider announcing that the problem has been solved and the cloud is back to its fully operational state. Statistics of working periods and outages have been modelled in [19] with data coming from a small private cloud. Active measurement systems based on ICMP probing packets have been investigated in [20]–[22]. A major issue with all measurements campaigns conducted so far is that they do measure the quality of service experienced by the user, but in doing so they include the loss contribution provided by the network located between the cloud user and the cloud server. The availability that is measured in the end is an underestimation of the actual cloud availability.

In this paper, we propose a measurement method that allows to distinguish between the losses due to the network and those due to the cloud, returning the true cloud availability. After describing the intrusive network problem in Section II and recalling the definition of availability in Section III, our study provides the following original contributions:

- we propose a measurement procedure to measure true cloud availability (Section IV);
- we assess its success rate (Section V), showing that it outperforms previous methods usable for that purpose;
- we apply our procedure to three major cloud providers and contrast the results with concerns arisen in early measurement campaigns (Section V), showing that the

availability at IP level is close to four nines, and the the network contribution is so relatively large as to significantly alter the overall results in the absence of a correct failure attribution procedure.

II. The Long Road to the Cloud

Cloud availability measurements are a major tool in assessing a cloud provider's compliance with SLA targets and obligations. However, those measurements may lead to false conclusions if they are not carried out properly. In this section, we take a look at what is probably the most important reason for lack of accuracy.

Contents placed on a cloud are located among one or more data centers, whose location is, by definition, unknown to the user [23], [24]. Whatever the way by which we probe the cloud to measure availability, third-party measurements are conducted from outside the cloud, i.e. through the network. In probing the cloud, we can therefore mimic the experience of the user, traversing one or more Internet Service Providers (ISP) and several Autonomous Systems (AS), as shown in Fig. 1.

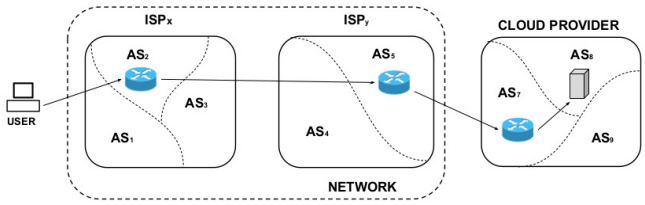

Fig. 1: The path from user to cloud provider.

It has been noted that false outages may be declared, since the lack of response to a user's request to a cloud may be due to packet losses in the network rather than the cloud itself [20]. This appears to be a major problem if we wish to get an accurate measurement of the actual outage rate for the cloud. When measurements are conducted through ICMP probing packets (pings), the Majority Voting rule to declare an outage has been analysed as an effective remedy in several contexts [21]. Under Majority Voting, an outage is declared if a majority of pings get no echoes. However, it cannot be considered as the definitive solution, since its accuracy depends on the specific combination of cloud and network performances.

We therefore need a more generally reliable approach to obtain an accurate measurement of cloud outage in the face of the losses of probing packets due to the network.

This is particularly relevant, since such measurements can be employed to enforce the contractual obligations contained in the SLA and the legal dispute that may arise, a danger that has been dreaded in [6]. Actually, the liability of the cloud provider in the case of obligations related to service malfunctioning has been mentioned as a major obstacle to the wide adoption of the cloud by banks [25]; the same has been reported for the semiconductor industry [26]. If we fail to recognize that service outages may be due to

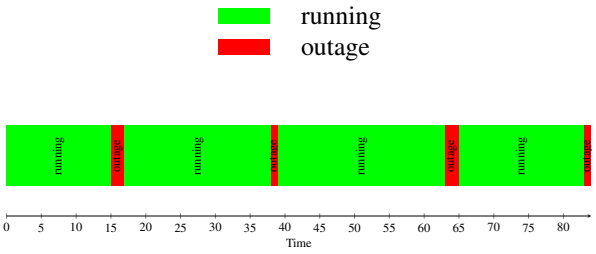

Fig. 2: Cloud state sequence

the network rather than the cloud, the cloud availability is actually underestimated, and the cloud provider may incur undue penalties. At any rate, the overall cost of data center outages is made of many components, which can build up to a very large amount, as reported in a study by the Ponemon Institute [27].

III. Availability Operational Definition

Before dealing with the contributions of the network and the cloud to the availability as seen from an external observer, we have to define how the observed availability is measured. In this section, we arrive at the operational definition of availability we have employed in this paper.

For our purposes, the state of the cloud is considered as a succession of working periods and outages, as shown in Fig. 2. If we describe the state of the service through the function $a(t): \quad t \to \{0,1\}$, the availability over an observation period T is then

$$A = \frac{1}{T}\int_0^T a(t)dt. \tag{1}$$

Within this paper, we do not consider the case of graceful degradation, where the cloud service is still running, but with a significantly worsened quality of service. Even though a service may experience a graceful degradation, we imagine that we can always classify the service as either being available or not. For example, if we tolerate a latency lower than a prescribed value, the service may degrade down to that value, while still being considered as available, but will be considered as unavailable when the latency exceeds that threshold.

If we indicate by W the overall sum of the durations of working periods and by F the overall sum of the durations of outages, we have the usual definition of availability as the fraction of ON periods over the observation window T

$$A = \frac{W}{W+F}. \tag{2}$$

However, the actual measurement process does not allow to recover the function $a(t)$, but rather its sampling version, obtained by probing the system at a discrete set of times. The discrete times are those at which discrete events take place, such as failed or successful service queries.

As a consequence, for cloud services, two general models have been defined in [28] to describe availability from discrete events:

- The dual state model;

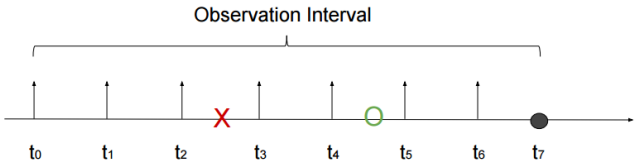

Fig. 3: Probing sequence example

- The success ratio model.

In the dual state model, the availability is computed as a function of the sum of the durations of all down states experienced during contracted service time. In the success ratio model, we refer instead to the event themselves, rather than their duration: the availability is computed as a function of the number of successful and failed resources requests during the contracted service period.

However, if we probe the cloud at periodic intervals (as opposed to random or irregular ones), the distinction between the two models blurs. Considering, e.g., the sequence of probing queries shown in Fig. 3, if we define the down duration as the time distance between the first failed probing query and the first subsequent successful probing query, the two models provide exactly the same availability output (5/7 in this case).

IV. Cloud availability measurement

The third-party measurements reported so far in the literature adopt a probing mechanism employing the ping command, which however does not allow to distinguish between outages due to the network and those due to the cloud. In this section, we propose a procedure that allows to obtain the availability of the cloud only. In Section IV-A, we first outline the problems affecting the measurement schemes employed so far, then provide an overview of our new procedure in Section IV-B, and finally describe its phases.

A. End-to-end availability

Current procedures to measure the availability of a cloud rely on the use of probing packets sent out from one or several vantage points mimicking the location of a real user. These packets are sent out periodically, as shown in Fig. 3. So far, the ping utility has been used for this purpose. Ping operates by sending Internet Control Message Protocol (ICMP) echo request packets to the target host and waiting for an ICMP echo reply, as shown in Fig. 4 (ping operations are described in Chapter 8.4 of [29]). Echoes from ping are counted as indicators of an operating cloud, while missed echoes are counted as indicators of a failing cloud. It is assumed that a cloud server returning the probing packets is also working correctly to provide services to its clients, i.e., we do not consider software problems related to service provisioning. The ratio of returned echoes to the overall number of sent probes gives us the availability of the cloud.

However, the use of this utility suffers from two main drawbacks:

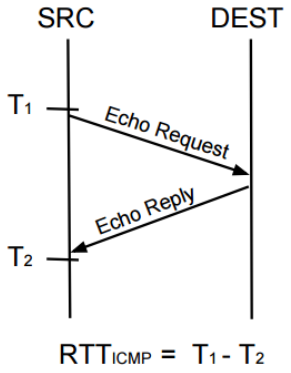

Fig. 4: Ping utility

- it returns an end-to-end measurement that incorporates all the failures taking place on the road to the cloud;
- it employs the ICMP protocol, which may be dealt with differently than TCP/UDP segments, and typically with a lower priority so that the reported availability may be lower than that actually experienced with the cloud-based service.

While the former problem cannot be solved by acting on the probing mechanism alone, the latter problem can be eliminated by employing nping probing packets instead. Nping is an open source tool for network packet generation, response analysis and response time measurement (see Chapter 18 of [30]); it can generate network packets for a wide range of protocols, allowing users full control over protocol headers. We can therefore employ it to generate TCP-like probing packets, which undergo the same priority treatment as the true packets we would employ when using the cloud service [1].

B. Overall procedure

For the time being we consider the reliability at the IP level only, meaning that we are interested in assessing if IP packets transporting the payload involved in the cloud service actually make it through the cloud once they reach it. Our procedure to measure the availability of the cloud, and the cloud only, goes through the following steps:

1) Probing the whole sequence of hops along the path from the measurement vantage point to the cloud;
2) Associating an ISP to each hop along that path;
3) Identifying the first hop belonging to the cloud provider, i.e. the hop marking the entry into the cloud providers domain;
4) Counting missing echoes from that first cloud hop and computing the corresponding cloud availability.

In the following, we take care of step 1 in Section IV-C, steps 2 and 3 in Section IV-D, and step 4 in Section IV-E.

C. Tracing probing packets

As just recalled, ping (or nping for that purpose) is an end-to-end tool, which does not reveal anything about what

[1] https://nmap.org/nping/

happens in between the probing source and the end host. We wish instead to get the sequence of IP addresses of routers that make the path from source (our probing vantage point) to destination (the cloud server).

In order to get a complete view of the path from source to destination, along which packets enter the cloud, we can employ the `traceroute` programme[2]. This programme uses limited Time-To-Live (TTL) ICMP probes to discover the IP addresses of IP router interfaces along the path from source to destination, using ICMP echo requests (see Fig. 5). Despite being the most used method to get information about Internet topology, `traceroute` suffers from the following major problems, which may lead to no return from the probed routers or to returned invalid IP addresses:

- ICMP packets may be filtered out by firewalls along the way.
- load-balancing routers may alter the path [31];
- successive TTL-limited packets do not necessarily follow the same forwarding path, so that we may get different chains of routers while we try to discover a single path to destination;
- some hops do not return ICMP replies;
- some routers may be anonymous, i.e., their existence is detected but their interface address is not returned [32];
- some routers may return the address of the interface from which the message came [33];
- some routers return a fixed IP address, regardless of the address of the actual interface on which the message has landed;
- some routers may return an IP address chosen randomly among those of the router's several interfaces;
- the connectivity between routers may be provided through chains of ATM (Asynchronous Transfer Mode) switches or MPLS (MultiProtocol Label Switching) tunnels (reported to account even for 30% of paths [34] [35]), which may make the path opaque to IP probes.

Though the original version of `traceroute` employs ICMP packets, other versions may employ UDP or TCP probing packets.

The UDP version employs limited TTL packets and large destination post numbers. When an intermediate router receives such a probing packet with a zero TTL, it returns an *ICMP time exceeded* message. The source can progressively increase the TTL discovering farther routers along the path, till it reaches the destination. However, the use of UDP messages to high ports shares the same problem with firewall as ICMP packets [36].

A version called `tcptraceroute` has been proposed[3]. The TCP version of `traceroute` bypasses firewalls by directing TCP packets to well-known ports (e.g. port 80), though some firewall may still block TCP packets when no host behind the firewall accepts the TCP connection.

[2]https://wiki.geant.org/display/public/EK/VanJacobsonTraceroute

[3]https://www.freebsd.org/cgi/man.cgi?query=tcptraceroute&manpath=FreeBSD+9.3-RELEASE+and+Ports

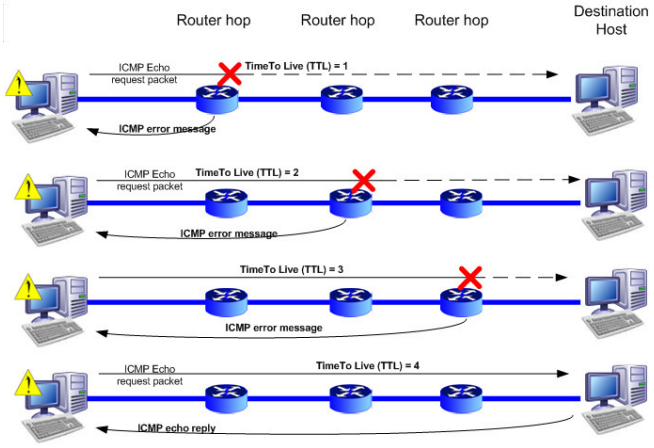

Fig. 5: Working of the `traceroute` programme

Despite the shortcomings of ICMP-based `traceroute`, it appears however to reach targets more successfully than its UDP and TCP counterparts [37]. In this paper, we therefore stick to the classical ICMP-based traceroute.

Other methods have been proposed to bypass the limitations of `traceroute`; a recent survey is contained in [38]. The most relevant category at the interface level (which is the one we adopt here) is based on the options of the IP packet header. However, in most cases, it relies either on the cooperation of intermediate routers or on the use of multiple vantage points, which excludes them from the horizon of third-party measurements. In addition, their use increases the chances of packets being discarded or triggering alarms on IDS (Intrusion Detection systems) [39].

D. Identifying cloud boundaries

After identifying the chain of routers that lead to the ultimate cloud destination, we wish to identify the AS (Autonomous System) to which each router belongs, and eventually the ISP administering that AS. This is essential to identify the router marking the cloud ingress. Our procedure will go through the following steps:

1) Get the ASN associated to each hop on the path to the cloud;
2) Get the ISP administering the AS along the path;
3) Extract the router marking the ingress into the cloud provider's domain.

Our procedure employs the following protocols and programs:

- `Traceroute`;
- `whois`;
- `RISwhois`.

While the first two are pretty standard programmes, the third one is actually a modified version of `whois`. While the standard version of that protocol queries the Internet Routing registries, the modified version `RISwhois` has been devised in the context

of the RIPE RIS (Routing Information System) project[4] and allows to get data directly from a network of BGP collectors, which collect data from the BGP tables of their peers. Such a use of BGP is envisaged and described in several papers concerned with the need to build the AS topology [38], [40]. Our aim in combining the response from the tools listed above is to arrive at a consensus about the correct ASN to attribute to each hop in the path to the cloud. Our procedure will return a positive result if we achieve either a 2/2 result or a 2/3 results, i.e. at least 2 of the tools agree.

It is to be noted that the `brdmap` programme is also available to identify domain borders, as described in [41], where it has been employed, however, through the use of 19 vantage points, whereas our procedure employs a single vantage point.

The first tool we employ to carry out the IP-to-AS attribution is `traceroute`. For each hop, the -a option allows to get the AS number. However, as pointed out in [40], the traceroute command alone does not give an accurate AS number in all cases, and it does not even return an AS number in roughly 10% of cases. We consider separately the case where Traceroute returns an ASN for each hop and that where it does not.

If Traceroute does not return an ASN number, we resort to whois and RISwhois in parallel. If they both return the same ASN, then we consider that to be the correct ASN; otherwise (i.e., if they either return different ASNs or do not return an ASN at all), the procedure is considered to fail.

If Traceroute does return an ASN, however, we do not stay content with that, since we strive for a higher reliability, aiming at least at two sources confirming the same ASN. Therefore, we first turn to RISwhois. If RISwhois gives us the very same ASN as Traceroute, we end the procedure and output that ASN as the correct one. If that's not the case we call the standard whois, which acts as the final referee. If it confirms one of the two ASNs previously obtained by Traceroute and RISwhois, then we obtain a 2/3 majority vote and declare that as the correct ASN. If, unfortunately, whois returns a third ASN, different from those obtained with Traceroute and RISwhois, the procedure is considered to fail. The whole procedure is reported as Algorithm 1.

At this point, we have the full list of ISPs administering the hops along the path to the end cloud server. We can then identify the hops belonging to the cloud provider through the algorithm described as Algorithm 2.

Algorithm 1: Identification of ASNs and ISPs

```
Input: Cloud Provider, AS-traceroute to Cloud Provider,
       selected hop
Output: ASN and ISP of selected hop
ASN ←null;
ISP ←null;
if all reports of selected hop are empty then
    return null;
else
    current_report ←select not empty report from hop;
    while ISP is null do
        ASN_1 ,IP_1 ←AS-traceroute (current_report);
        if ASN_1 is not null then
            ASN_2, ISP_1 ←RISwhois(IP_1);
            if ASN_2 is equal to ASN_1 then
                ASN ←ASN_1;
                ISP ←ISP_1;
            else
                ASN_2.1, ISP_1.1 ←whois(IP_1);
                if ASN_2.1 is equal to ASN_1 then
                    ASN ←ASN_1;
                    ISP ←ISP_1.1;
                else
                    if ASN_2.1 is equal to ASN_2 then
                        ASN ←ASN_2;
                        if ISP_1 is equal to ISP_1.1 then
                            ISP ←ISP_1 ;
                    else
                        Error: current_report ←select
                          another not empty report if
                          there's else break;
        else
            ASN_2, ISP_1 ←RISwhois(IP_1);
            ASN_2.1, ISP_1.1 ←whois(IP_1);
            if ASN_2 is equal to ASN_2.1 then
                ASN ←ASN_2;
                if ISP_1 is equal to ISP_1.1 then
                    ISP ←ISP_1 ;
            else
                Error: current_report ←select another not
                  empty report if there's else break;
    return ASN, ISP;
```

E. Cloud availability estimation

Now, we have hopefully identified where the probing packets actually enter the cloud. We have all the data needed to measure the actual cloud availability.

If we indicate by N_{in} the number of probing packets entering the cloud, i.e. making it to the first cloud hop, and by N_{out} the number of echoes actually returned from the cloud end server (i.e., the final hop in the sequence of hops obtained with traceroute), our measurement of the cloud availability is

$$A = \frac{N_{\text{out}}}{N_{\text{in}}}. \tag{3}$$

This approach allows not to factor in the losses due to the network on the way to the cloud, since they do not enter the N_{in} term. A remaining limitation of the approach is that echoes actually sent back by the cloud end server may get lost due to network problems on the return path.

[4]https://www.ripe.net/analyse/archived-projects/ris-tools-web-interfaces

Algorithm 2: Identification of the first cloud hop

Input: Cloud Provider, AS-traceroute to Cloud Provider
Output: Position of the first Cloud hop in AS-traceroute report

```
ISP ←null;
hopPosition ←1;
cloudStart ←null;
currentEntryPosition ←1;
Table ←empty key-value table;
while there's hop in hopPosition of AS-traceroute do
    selectedHop ←select hop in hopPosition;
    ISP ←Identification of ASNs and ISPs (selectedHop);
    newTableEntry ←append entry (hopPosition, ISP);
    hopPosition ←hopPosition + 1;
while there's table entry in currentEntryPosition do
    currentISP ←getValue (currentEntryPosition);
    if currentISP is equal to Cloud Provider then
        if cloudStart is equal to null then
        cloudStart ←currentEntryPosition ;
    else
        cloudStart ←null;
    currentEntryPosition ←currentEntryPosition +1;
return cloudStart;
```

V. Experimental results

We have applied the procedure described in Section IV to three major cloud providers. In this section, we report the results.

Our aim is to assess two different things: the dependability of our procedure and the availability of cloud providers. The latter is of course meaningful if our procedure possesses the former feature.

For both purposes we considered three major cloud providers: Google Cloud, Amazon AWS, and Rackspace (all of them are included in the survey reported in [42]). We performed 50 tests for each of them, for a total of 150 tests. Each test consisted in sending probing packets over a period of 8 hours, going through the procedure described in Section IV, and assessing whether the cloud has responded (i.e., it is working) or not. The overall duration of test was therefore 400 hours for each provider.

The dependability issue is crucial. We have to be sure that the procedure works under real conditions and may be employed routinely. We stress the fact that our procedure requires neither the use of special software nor restricted information.

In Table I, we report the test results. Reporting an ASN as outcome means that we were able to get an ASN for all the hops along the path from source to destination, excluding from the count those hops that did not respond (for which we have of course no elements at all to infer their ASN). Overall, we get the ASN for the whole path roughly in 95% of cases; this result represents a good advance over the 90% declared in the reference paper [40]. In the remaining 5% of cases there were some hops for which, though they did respond, we were not able to get a consensus over the ASN. For non-responding hops, a possible way to dispel the darkness is suggested again in [40]: if a non-responding hop is located along the path between two responding hops exhibiting the same ASN, then it is safe to assign that same ASN to the non-responding path. This solution would leave out just those non-responding hops located at the border between two ASes.

Outcome	Frequency [%]
ASN (2/2 confidence level)	90.66
ASN (2/3 confidence level)	4.00
No ASN	5.34

TABLE I: Test results for procedure dependability assessment

However, the final aim is to correctly assign outages, and we need to identify the first hop belonging to the cloud provider. In that case, non-responding hops may represent a problem, since they can actually be those belonging to the cloud provider. In our battery of tests, we were unable to identify the first cloud hop in 30% of cases, practically all due to Amazon (where the identification procedure failed in 45 out of 50 tests). Though this may appear as a disappointing performance, we must consider that a) it concerns a single provider; b) it is a matter of policy, which may be circumvented by arrangements between the cloud provider and the third-party organisation in charge of conducting the availability measurement (for example, if allowing for such measurements to be conducted on the basis of an agreement, Amazon could enable its routers to respond to probing packets sent by the authorised organisation, e.g. by recognising its IP addresses).

Once we have assessed that the procedure can be routinely carried out, we can employ it to assess the actual availability of the cloud. We consider the three major cloud providers that we have already mentioned: Google, Amazon, and Rackspace. We have carried out daily tests (lasting 8 hours) over 30 days, identifying the first hop belonging to the cloud provider and correctly assigning packet losses.

The results are shown in Table II, where we see that all three providers offer an availability better than 3 nines (actually quite close to 4 nines). There are two questions that naturally arise after these results:

- Is the contribution of the network relevant in availability assessment?
- Do these results confirm previous measurement campaigns?

The former question impacts the relevance of our measurement procedure. If the contribution of the network were negligible, there would be no interest in providing a measurement procedure capable of distinguishing between the failures taking place on the net and on the cloud. We see in Table II that in two out of three cases the network losses are at least twice as large as those due to the cloud. Even in the case of Rackspace they are all but negligible. In the absence of any loss attribution procedure the observed availability would

Measurement	Google Cloud	Amazon AWS	Rackspace
# probing packets	864 000	864 000	864 000
# packets reaching cloud	863 750	863 711	863 899
# lost packets (network)	250	289	101
# lost packets (cloud)	125	104	195
Availability	99.9855%	99.9879 %	99.9774%

TABLE II: Availability measurements

be 99.9566%, 99.9545%, and 99.9657% respectively. The difference between those values and those in Table II may look negligible, but we must not forget that we are talking about figures very close to 100% anyway and a difference as low as 0.01% is significant in this context (see an account of router availability issues in [43]). In the case of Google Cloud the actual difference appears to be 0.0289%, which would amount to 152 min (roughly 2 hours and a half) more downtime in a year, which is not negligible, given the quality-of-service expectations of customers. In addition, making a bundle of network and cloud losses would significantly alter the relative performances of the three cloud providers: Rackspace, ranking third in the correct measurement, would jump to the first place if we decided not to distinguish between the two sources of loss.

We can now turn to the latter question: how do these results compare with past measurement campaigns? We have reminded in the Introduction that there's not a host of measurement campaigns on cloud performance. However, a procedure like ours, which does not attribute to the cloud losses that are not its fault, naturally results in better performance figures for the cloud. Actually, though the results reported here are by no means exhaustive and conclusive, the availability look much better than was previously feared [18].

VI. Conclusions

Our procedure allows us to assign the cloud the outages that are actually due to it and excluding those due to the network. It increases the accuracy of existing availability measurement procedures. The procedure can bee conducted from any third-party vantage point and may be safely employed to assess the compliance of cloud providers with SLAs. The early results of its application show that the availability of cloud providers may be significantly underestimated.

Some limitations need to be addressed, though. A major limitation is that the non-response rate may be significant and must be reduced, since that prevents from obtaining a full view of the ISPs along the way. Though this can be achieved by way of agreements between measuring parties and cloud providers, it is too optimistic to hope for a 100% response rate. A second limitation is that network losses on the path back to the source may still cause the availability to be underestimated.

References

[1] M. M. Qiu, Y. Zhou, and C. Wang, "Systematic analysis of public cloud service level agreements and related business values," in *Services Computing (SCC), 2013 IEEE International Conference on*. Santa Clara, CA, USA: IEEE, 2013, pp. 729–736.

[2] S. A. Baset, "Cloud SLAs: present and future," *ACM SIGOPS Operating Systems Review*, vol. 46, no. 2, pp. 57–66, 2012.

[3] M. Alhamad, T. Dillon, and E. Chang, "Conceptual sla framework for cloud computing," in *Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on*. Dubai, United Arab Emirates: IEEE, 2010, pp. 606–610.

[4] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Generation Computer Systems*, vol. 79, pp. 849–861, 2018.

[5] R. Buyya, S. N. Srirama, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, L. M. Vaquero, M. A. Netto *et al.*, "A manifesto for future generation cloud computing: Research directions for the next decade," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, p. 105, 2018.

[6] M. Cinque, S. Russo, C. Esposito, K.-K. R. Choo, F. Free-Nelson, and C. A. Kamhoua, "Cloud reliability: Possible sources of security and legal issues?" *IEEE Cloud Computing*, vol. 5, no. 3, pp. 31–38, 2018.

[7] L. Fiondella, S. S. Gokhale, and V. B. Mendiratta, "Cloud incident data: An empirical analysis," in *Cloud Engineering (IC2E), 2013 IEEE International Conference on*. San Francisco, California, USA: IEEE, 2013, pp. 241–249.

[8] P. T. Endo, G. L. Santos, D. Rosendo, D. M. Gomes, A. Moreira, J. Kelner, D. Sadok, G. E. Gonçalves, and M. Mahloo, "Minimizing and managing cloud failures," *Computer*, vol. 50, no. 11, pp. 86–90, 2017.

[9] R. Nachiappan, B. Javadi, R. N. Calheiros, and K. M. Matawie, "Cloud storage reliability for big data applications: A state of the art survey," *Journal of Network and Computer Applications*, vol. 97, pp. 35–47, 2017.

[10] M. R. Mesbahi, A. M. Rahmani, and M. Hosseinzadeh, "Highly reliable architecture using the 80/20 rule in cloud computing datacenters," *Future Generation Computer Systems*, vol. 77, pp. 77–86, 2017.

[11] B. Liu, X. Chang, Z. Han, K. Trivedi, and R. J. Rodríguez, "Model-based sensitivity analysis of iaas cloud availability," *Future Generation Computer Systems*, vol. 83, pp. 1–13, 2018.

[12] H. Adamu, B. Mohammed, A. B. Maina, A. Cullen, H. Ugail, and I. Awan, "An approach to failure prediction in a cloud based environment," in *Future Internet of Things and Cloud (FiCloud), 2017 IEEE 5th International Conference on*. Prague, Czech Republic: IEEE, 2017, pp. 191–197.

[13] Q. Lin, K. Hsieh, Y. Dang, H. Zhang, K. Sui, Y. Xu, J.-G. Lou, C. Li, Y. Wu, R. Yao *et al.*, "Predicting node failure in cloud service systems," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Lake Buena Vista, Florida: ACM, 2018, pp. 480–490.

[14] T. Labidi, A. Mtibaa, W. Gaaloul, S. Tata, and F. Gargouri, "Cloud sla modeling and monitoring," in *Services Computing (SCC), 2017 IEEE International Conference on*. Honolulu, HI, USA: IEEE, 2017, pp. 338–345.

[15] F. Nawaz, O. K. Hussain, N. Janjua, and E. Chang, "A proactive event-driven approach for dynamic qos compliance in cloud of things," in *Proceedings of the International Conference on Web Intelligence*. Leipzig, Germany: ACM, 2017, pp. 971–975.

[16] S. Alboghdady, S. Winter, A. Taha, H. Zhang, and N. Suri, "C'mon: Monitoring the compliance of cloud services to contracted properties," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*. Reggio Calabria, Italy: ACM, 2017, p. 36.

[17] H. J. Syed, A. Gani, R. W. Ahmad, M. K. Khan, and A. I. A. Ahmed, "Cloud monitoring: A review, taxonomy, and open research issues," *Journal of Network and Computer Applications*, 2017.

[18] M. Naldi, "The availability of cloud-based services: Is it living up to its promise?" in *9th International Conference on the Design of Reliable Communication Networks, DRCN 2013, Budapest, Hungary*, 2013, pp. 282–289.

[19] J. Dunne and D. Malone, "Obscured by the cloud: A resource allocation framework to model cloud outage events," *Journal of Systems and Software*, vol. 131, pp. 218–229, 2017.

[20] M. Naldi, "Accuracy of third-party cloud availability estimation through ICMP," in *Telecommunications and Signal Processing (TSP), 2016 39th International Conference on*. Vienna, Austria: IEEE, 2016, pp. 40–43.

[21] ——, "ICMP-based third-party estimation of cloud availability," *International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems*, vol. 6, no. 1, pp. 11–18, 2017.

[22] Z. Hu, L. Zhu, C. Ardi, E. Katz-Bassett, H. V. Madhyastha, J. Heidemann, and M. Yu, "The need for end-to-end evaluation of cloud availability," in *International Conference on Passive and Active Network Measurement*. Springer, 2014, pp. 119–130.

[23] Y. Jadeja and K. Modi, "Cloud computing-concepts, architecture and challenges," in *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*. IEEE, 2012, pp. 877–880.

[24] M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud computing: Distributed internet computing for it and scientific research," *IEEE Internet computing*, vol. 13, no. 5, 2009.

[25] W. K. Hon and C. Millard, "Banking in the cloud: Part 3–contractual issues," *Computer Law & Security Review*, vol. 34, no. 3, pp. 595–614, 2018.

[26] S. B. Rahi, S. Bisui, and S. C. Misra, "Identifying critical challenges in the adoption of cloud-based services," *International Journal of Communication Systems*, vol. 30, no. 12, p. e3261, 2017.

[27] AA.VV., "Cost of data center outages," The Ponemon Institute, Tech. Rep., 2016.

[28] G. Hogben and A. Pannetrat, "Mutant apples: a critical examination of cloud SLA availability definitions," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1. Bristol, United Kingdom: IEEE, 2013, pp. 379–386.

[29] K. R. Fall and W. R. Stevens, *TCP/IP illustrated, volume 1: The protocols*. addison-Wesley, 2011.

[30] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009.

[31] F. Viger, B. Augustin, X. Cuvellier, C. Magnien, M. Latapy, T. Friedman, and R. Teixeira, "Detection, understanding, and prevention of traceroute measurement artifacts," *Computer networks*, vol. 52, no. 5, pp. 998–1018, 2008.

[32] B. Yao, R. Viswanathan, F. Chang, and D. Waddington, "Topology inference in the presence of anonymous routers," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 1. San Francisco California, USA: IEEE, 2003, pp. 353–363.

[33] R. Govindan and H. Tangmunarunkit, "Heuristics for internet map discovery," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. Tel Aviv, Israel: IEEE, 2000, pp. 1371–1380.

[34] J. Sommers, P. Barford, and B. Eriksson, "On the prevalence and characteristics of mpls deployments in the open internet," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. Berlin, Germany: ACM, 2011, pp. 445–462.

[35] B. Donnet, M. Luckie, P. Mérindol, and J.-J. Pansiot, "Revealing mpls tunnels obscured from traceroute," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 2, pp. 87–93, 2012.

[36] S. Savage *et al.*, "Sting: A TCP-based Network Measurement Tool." in *USENIX Symposium on Internet Technologies and Systems*, vol. 2, Boulder, Colorado, USA, 1999, pp. 7–7.

[37] M. Luckie, Y. Hyun, and B. Huffaker, "Traceroute probe method and forward ip path inference," in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*. Vouliagmeni, Greece: ACM, 2008, pp. 311–324.

[38] R. Motamedi, R. Rejaie, and W. Willinger, "A survey of techniques for internet topology discovery," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 1044–1065, 2015.

[39] W. De Donato, P. Marchetta, and A. Pescapé, "A hands-on look at active probing using the ip prespecified timestamp option," in *International Conference on Passive and Active Network Measurement*. Vienna, Austria: Springer, 2012, pp. 189–199.

[40] Z. M. Mao, J. Rexford, J. Wang, and R. H. Katz, "Towards an accurate as-level traceroute tool," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. Karlsruhe, Germany: ACM, 2003, pp. 365–378.

[41] M. Luckie, A. Dhamdhere, B. Huffaker, D. Clark *et al.*, "bdrmap: inference of borders between ip networks," in *Proceedings of the 2016 Internet Measurement Conference*. Santa Monica, California, USA: ACM, 2016, pp. 381–396.

[42] M. Naldi and L. Mastroeni, "Cloud storage pricing: A comparison of current practices," in *Proceedings of the 2013 International Workshop on Hot Topics in Cloud Services*, ser. HotTopiCS '13. New York, NY, USA: ACM, 2013, pp. 27–34.

[43] A. Agapi, K. Birman, R. M. Broberg, C. Cotton, T. Kielmann, M. Millnert, R. Payne, R. Surton, and R. Van Renesse, "Routers for the cloud: Can the internet achieve 5-nines availability?" *IEEE Internet Computing*, vol. 15, no. 5, p. 72, 2011.