Università degli Studi di Napoli Federico II

Scuola Politecnica e delle Scienze di Base

Corso di Dottorato di Ricerca in Ingegneria delle Telecomunicazioni
XXV Ciclo
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

Monitoring Internet Censorship: the case of UBICA

Giuseppe Aceto

Ph.D. Thesis

Tutor
Prof. Giorgio Ventre

Cotutor
Prof. Antonio Pescapè

Coordinator
Prof. Niccolò Rinaldi

March 2014

Possiamo impedire di leggere:
ma nel decreto che proibisce la lettura
si leggerà pur qualcosa della verità
che non vorremmo venisse mai letta...

We can forbid to read:
but in the decree that forbids reading
still it will be read something of the truth
that we wouldn't want ever to be read...

Italo Calvino
Se una notte d'inverno un viaggiatore
If on a winter's night a traveler

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet in the developed countries (and to a lesser extent, also in developing ones) is a mainstream medium leveraged by private companies for advertising and selling goods and services, by governments to provide services to the citizens, including information and news delivery, and by individuals to perform a wide spectrum of activities as: work, establish and keep social bonds, share information and opinions, participate in communities of interest and organize and coordinate actions. While the impact of Internet accessibility on national economies is easily acknowledged [Zwillenberg et al., 2014], its political significance has recently emerged with political campaigns leveraging the Web and specifically the Online Social Networks (see Alexandrova [2010] for an analysis of the use of these media on United States president Barack Obama's campaign and previous examples).

With the use of the Internet to promote information and political discussion and activities, the struggle of governments to control this information channel has become evident: the change over time of Internet Censorship has been analyzed in [Bambauer, 2013], where such worrisome evolution and its last stage (dubbed Censorship v3.1) are the basis to a call for transparency and public control on censorship application. The author makes the point that, regardless of the different motivations (legitimate or rhetorical) and the specific technical means, the intentional limitation of access to information is always a form of censorship, and as such should be clearly named, highlighting the sensitive and dangerous nature of its enforcement and requiring strict watch from the citizens.

Although Bambauer [2013] is focused on political and legal aspects more than on technical details, it gives both an embraceable frame and a strong motivation for censorship monitoring: an *independent*, *publicly available* and *global* watch on Internet censorship activities is a fundamental goal to be pursued, specially now that regulators are actively pushing it forward.

A chronology of the evolution of Internet Censorship, but with a more technical viewpoint, has been previously presented in [Deibert, 2010, chap. 1], in which the authors leverage the studies carried on in the *OpenNet Initiative* [ONI] research project to characterize the evolution of the normative and technical control of access to information on the Internet. Four phases are identified for “Cyberspace Regulation”, summarized as

- Open Commons (1990s-2000) - Internet is considered as an information sharing media it is named “cyberspace”, a world somehow separated and different from “real life”, not subjected to the same rules. Unbounded, cheap-as-free access to all available online information is the norm. Governments mostly ignore the Internet or lightly regulate it.
- Access Denied (2000-2005) - States and other entities start considering online activities as demanding management: filtering activities are enacted at country or organization level, often to block specific content categories (pornography, content deemed harmful to underages); the freedom of access to information is actively challenged.
- Access Controlled (2005-2010) - Regulatory approaches are emphasized, a wider range of means and points of control (non necessarily technical in nature) are exploited by authorities to shape and control the access to information. Active methods such as computer network attacks, espionage and public opinion forming (ad-hoc creation of content by fake “common citizen” or grassroots movements) are enacted. Surveillance and the implied self-censorship are linked with national conditions and regulations, as the obligation of registering with an identification document when accessing the Internet. From the technical point of view, ad-hoc filtering is the new challenge for censorship assessment: time limited blocking of specific online resources is hard to detect or can be mistaken by a failure, or an attack whose instigator is hard to provably traceback.
- Access Contested (2010-current) - The fight and the debate about regulation of access to online information has come in the foreground, with governments pushing their prerogative of controlling online information and companies providing filtering services and technologies on one side, and individuals, organizations and private companies arguing these prerogatives in order to defend their rights and interests, on the other side.

Besides the aforementioned ones, other definitions of Internet Censorship can be found in academic literature and in technical reports and news, often implicit in the description, or in the detection and analysis techniques adopted for its study. Even if censorship can also be enforced by limiting the use of other Internet applications, such as Voice-Over-IP, Virtual Private Networks, Tor, most literature and news refer to web content, i.e. information accessible through web browsers and employing the HTTP and HTTPS protocols. The targets of this type of censorship, i.e.whole websites or specific resources that are made unreachable to the user, have been described by [ONI] as belonging to the following categories:

- Political: the focus is on websites that express views opposing governments. In most cases the content is related to human rights, freedom of expression, minority rights and religious movements.
- Social: the focus is on content related to sexuality, gambling, illegal drugs and alcohol and any other issue considered illicit.
- Conflict/security: focuses on content related to armed conflicts, border disputes, and militant groups.
- Internet tools: websites that provide email, Internet hosting, search, translation, voice-over Internet Protocol, and telephone service, as well as circumvention methods.

This classification has been referenced in subsequent works on Internet Censorship detection [Bitso et al., 2012, Filastò and Appelbaum, 2012].

A few research efforts, NGOs and associations such as *Reporters Without Borders* [RWB], *the OpenNet Initiative* [ONI], *Human Right Watch* [HRW], provide reports on the state of Internet Censorship in different countries, often including information collected by local contacts on the network infrastructure and experimental data.

While providing an invaluable (and possible risky) service by giving insight in the objectives of censorship and the adopted techniques, the original information is collected, analyzed and released on varying timescales or in spot efforts, and the exposed results - often only in aggregated form - are not suitable for independent analysis and validation.

Collecting the base data in order to asses the near-real-time evolution of censorship is made difficult by the different possible censorship techniques, by the number of online resources and services that are potentially subject to blocking, and by the global scale of phenomenon. Moreover, there are strong ethical issues in involving in measurements the volunteers in countries not under the rule of law [RWB], as it makes impossible to define the legitimacy of the measurement activities and thus possibly exposes the volunteer to legal or personal risks. A process of informed consent for volunteers' participation, assistance from field activists and the exclusion from the research of the most dangerous countries has been the approach chosen by researchers in this field[1].

Crowd-sourced platforms, such as *Herdict* her, appear as the most viable approach to cover these needs. To this aim a limited number of tools and platforms have been proposed, but from the analysis of the state of the art the lack of a sustainable censorship monitoring platform has arisen. More specifically we have noticed the lack of a publicly available tool performing censorship detection that is: (i) user friendly, with minimal need for user intervention; (ii) directly useful to the user, fostering the adoption of the tool; (iii) integrated in a **monitoring** platform, to leverage multi-viewpoint and large time-scale analysis.

In turn, the monitoring platform backing the tool has to provide: (i) automatically updated censorship tests; (ii) automatically updated censorship targets to be verified; (iii) reports of detected censorship tailored for the user's environment (iv) support for different types of probing clients.

Available tools and platforms present only a limited subset of these properties, thus the candidate has proposed a design comprising all of them, also adding a novel feature, namely **the scheduling of evidence collection activities**, that was missing from the available literature to the best of the candidate's knowledge. A prototype of the proposed design has been implemented and experimentally validated, also including novel censorship detection algorithms based on the collected data.

This Thesis work is structured as follows.

In Chapter 2 an analysis of Internet Censorship techniques is provided based on literature. Documented techniques are characterized and systematically classified with reference to the communication phases they affect and disrupt, the definition of the protocols fields that elicit the reaction of the censoring system and the effect experienced by the user.

In Chapter 3 the detection and monitoring of Internet Censorship is presented, with an analysis of the state of the art listing the main censorship monitoring platforms and

[1]source: debate in the workshop on Monitoring Internet Openness and Rights [Aceto et al., 2013]

tools with their limitations and shortcomings.

The UBICA platform is then presented in Chapter 4 as the candidate proposal for a Censorship Monitoring platform addressing some of the shortcomings highlighted in the previous chapter.

The experimental validation of the proposed platform implementation is reported in Chapter 5 with the analysis of the results of the activity of the platform over different measurement campaigns.

Chapter 6 concludes the work, summarizing the thesis and its experimental validation, and proposing future research.

Chapter 2

Internet Censorship

Internet Censorship is a phenomenon that crosses several study fields, from computer networking and computer security to social sciences; several definitions can be found in academic literature and in technical reports and news, but often they are implicit and described by means of specific detection or analysis technique. Most literature refers to censorship of web content, i.e. information accessible through web browsers (adopting an *application level* view) and employing the HTTP and HTTPS protocols. We note that web browsing (and more generally applications adopting HTTP or HTTPS protocols) is only one kind of network application that use the Internet, other examples being Voice-Over-IP, Peer-to-peer file sharing, e-mail, multi-user video games, Instant Messaging; censorship has been found to be enforced also on some of these applications.

2.1 Overview

From a network topology point of view, a coarse-grain classification can be proposed with regards to the components of the communication system that are employed for censorship: client-based, and server-based, if censorship is applied at the ends of the communication path, network-based, if it happens in between. While this Thesis work is focused on network-based censorship detection, in the following an overview of client-based and server-based censorship techniques is given both for completeness and to clarify the scope of the work.

2.1.1 Client-based Censorship

We consider *client-based censorship* as the blocking of access to online content by means of applications running on the same system of the network application client. It can be implemented by different means, such as: an independent application, akin to a *keylogger*[1], that terminates the applications whose keyboard input matches a blacklisted keyword - such apparently is the technology employed in Cuba, [Voeux and Pain, 2006, anecdotal report from journalists]. Another form for this kind of censorship is a network filter like parental control or company policy control enforcement filters, running as a personal firewall. Finally, it can be enforced as a modified version of the network application client itself, added with surveillance "features", as the case of TOM-Skype in China [Villeneuve].

2.1.2 Server-based Censorship

The final node of the communication path, the server, is the component where server-based censorship is enforced, with no disruption of the communication mechanics: the censor selectively removes, hides, or impairs access to specific content directly in the server, by means of management facilities provided by the service itself. The censoring action can be enforced commanding the server manager to comply with the request.

This form of censorship is specifically hard to be analyzed, as its mechanics are internal to the service and not exposed to the users; a recent quantitative analysis of it has been performed in [Zhu et al., 2013], that reported several censoring evidences of different type (divided as proactive or retroactive mechanisms), and proposed hypotheses on how these mechanisms are actually enacted.

The existence of this kind of censorship is sometimes acknowledged by the Online Service Providers themselves. One such case is Google Transparency Report - Removal Requests[2] by which Google discloses a summary of requests from governments or from copyright owners to block access to specific content. While the actual removed targets are not disclosed, a categorization of removal requests is done according to the reason and the type of requester and the related statistics are provided. This kind of censorship and its consequences are analyzed under the term "intermediary censorship" in Deibert [2010, chap. 5].

[1]an application that covertly intercepts and processes key strokes directed to other applications running on the same system

[2]http://www.google.com/transparencyreport/removals/government

2.1.3 Network-based Censorship

In between the host running the application client and the host running the respective server part is where *network-based censorship* is enforced.

With respect to client-based censorship it provides the censor a much wider coverage of the network and with more efficiency, allowing the control of high number of communications through the management of a relatively few gateways or hubs (instead of one installation for each user system). On the other hand, client-based censorship implies the control of the user terminal or the compliance of the user herself, for each controlled user.

Similar considerations can be done with respect to *server-based* censorship, that in turn requires control or compliance of server host managers. The relatively small number of popular services helps the censor that wants to control them, but there is the possibility that such servers are located abroad or otherwise outside of the influence of the censor, thus nor direct control nor compliance can be forced.

These comparisons highlight the pivotal importance of network-based censorship, that constitutes the central phenomenon considered in the present Thesis work, and will be analyzed in more detail in the following section. Unless explicitly stated differently, hereafter by "censorship" will be intended "*network-based* Internet censorship", and similarly by "detection" will be intended "detection of *network-based* Internet censorship".

2.2 Definitions

Terminology in the matter of Internet Censorship is not well defined, being often ambiguous and inconsistent across different papers. The phenomenon under analysis also lacks a common definition, being named filtering, blocking, interference, tampering, surveillance, referring possibly to different aspects of it, with little or no formalisms. This does not help scientific discussion on the topic and the sharing of technical advancement in detection and circumvention. Valuable contributions in the direction of clearly defining the concepts related with Internet Censorship have been provided by Verkamp and Gupta [2012] and Filastò and Appelbaum [2012], even though they were not driven by this goal, their approaches aiming either at reporting the state of art or providing the researchers with detection tools. One of the first technical characterization of Internet Censorship can be found in [Dornseif, 2003], that defines it as " [...] *refusing users access to certain web pages without the cooperation of the content provider, the hosting provide and the* *owner of the client machine being used to access these pages.".* A somehow more generic definition is proposed in [Elahi and Goldberg, 2012]: *"Internet censorship is the intentional suppression of information originating, flowing or stored on systems connected to the Internet where that information is relevant for decision making to some entity.".* This definition, while having merits in highlighting the *motivation* behind the censorship and its intentionality, goes beyond the scope of the present Thesis work, that is focused on the detection of censorship and has little practical use for the modeling of the censor's decision making processes.

Therefore, extending the definition provided in [Dornseif, 2003] for web pages also to general Internet applications and to what we will describe as *soft censorship* techniques, we define "internet censorship" the intentional impairing of a client application in reaching a requested resource or service, enforced by a third party (neither the user, nor the server operator), named hereafter "censor". The intentionality differentiates censorship from outages and the selective censor behavior (affecting the censored resources or services and not others) is the condition necessary to detect it. We note that the adoption of a client-server terminology does not restrict the definition to applications implementing exclusively this communication paradigm, as also in peer-to-peer applications each communication sees at least one node initiating the exchange thus qualifying itself as "client" for that exchange.

In the following we define terms and concepts related to Internet censorship that will be useful in describing censoring techniques and censorship detection techniques, elaborating on definitions (explicit or implicit) from the related literature.

target an online resource or service; it is characterized by information needed by a client application to access it: e.g. for a document its URL, or for a service its application protocol (described by means of signatures potentially triggering kewword-based censorship) or a list of servers (characterized by hostname, IP addresses, transport port).

trigger the element or group of elements, in the client request to access the target, that cause the censoring system to act; if the trigger is absent (where possible) or substituted with a non-trigger then censorship does not happen i.e. the requested target is reachable; it is characterized by the phase of communication it affects, the levels of the network protocol stack that are involved and possibly specific protocols; implies the presence of a *surveillance device.*

surveillance device the device that analyzes the user traffic looking for *triggers*; it is characterized by the phase of communication it affects, the (topological) position in the network path, the levels of the network protocol stack that are inspected;

action the action performed by the censor to block, mangle or impair the access to the target; it is characterized, like the *trigger*, by the phase of communication it affects and the levels of the network protocol stack that are involved; implies the presence of a *censoring device* performing it.

censoring device a device that applies the censoring *action* by tampering with the communication between the user and the *target* so that it is impaired or altogether prevented; the surveillance and censoring devices can be co-located and coincide;

censoring system the *surveillance* and *censoring* devices;

symptom what the user experiences as result of the censor *action*; it can range from the access to a modified *target*, to an annoying worsening of quality of experience, to complete unreachability of the *target* possibly accompanied with an error; in case an error is provided, it characterizes the symptom with the phase of communication and the level of the network stack that appears to issue the error.

circumvention the process of nullifying the censoring *action*, i.e. accessing the unmodified *target* despite the presence of a censoring system; this can be done by preventing the *trigger* from being seen by the *surveillance device* or by countering the effects of the *action*.

2.2.1 Communication Phases

The *action* performed by the censor will involve (either interfering directly with-, or showing the effects on-) some specific phases of communication. In the following a basic knowledge of the TCP/IP suite of protocols and related terminology is assumed[3].

In order to easily refer to these phases we adopt a simplified model of a communication: the message exchange a web browser would establish to access a resource my means of the `HTTPS` protocol [Dierks and Rescorla, 2008]; the chain of events is shown in Figure 2.1 in an abstract form and as it would happen if not altered by the censor. For the purpose of the following description and analysis of the phenomenon, the considered application-level protocol (HTTP) can be substituted with a generic client-server application-level

[3]for a comprehensive coverage of the related definitions and concepts we refer to [Ross and Kurose, 1999]

protocol.

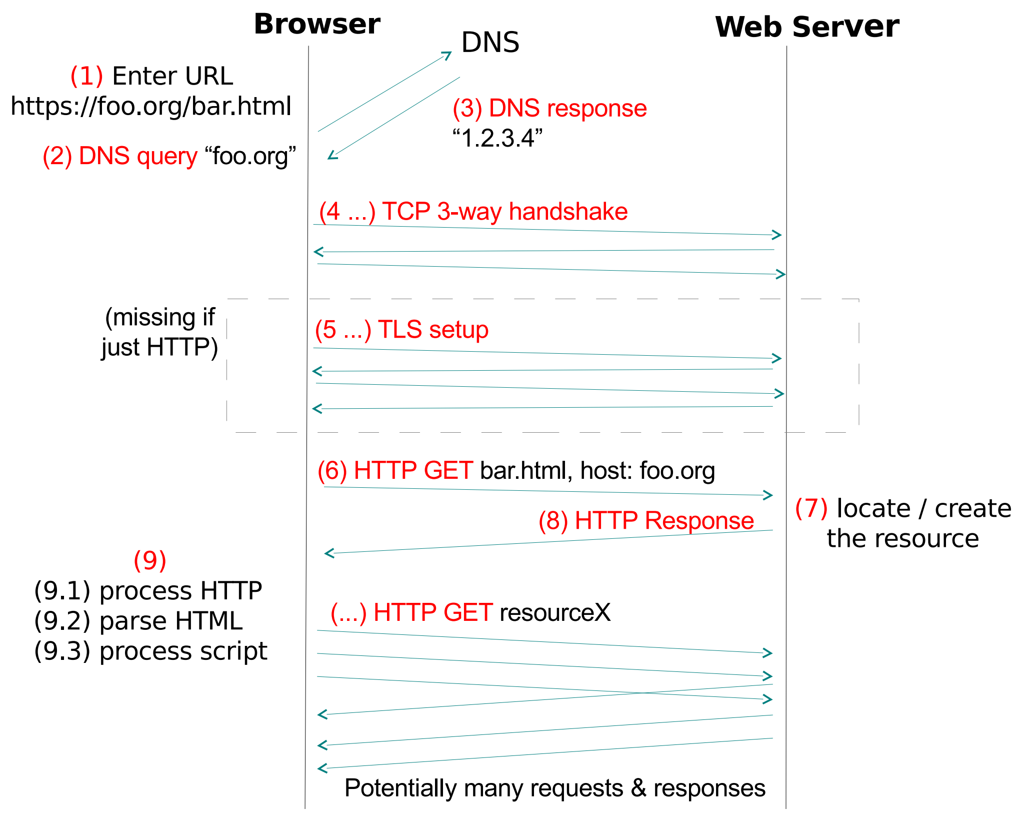


Figure 2.1: Generic HTTPS sequence diagram

In the considered scenario the request of the user (event (1)) starts the chain of events described hereafter.

The browser issues a DNS[4]query to the default recursive resolver asking for the IP address corresponding to the given URL (event (2));

The DNS server (more precisely, "recursive resolver") will search for the requested information in the DNS database by querying other Name Servers (not shown), finally obtaining the answer, that will be sent back to the client (event (3)). Censorship techniques acting on this phase of communication are described in Section 2.3.2.

Knowing the IP address of the desired *target* server, the browser sets up a reliable communication with it: the TCP protocol performs a three-way handshake (event (4)). Censorship techniques acting on this phase of communication are described in Section 2.3.4.

[4]Domain Name System, the hierarchical distributed database that translates between resources and their Internet addresses [Mockapetris, 1987a,b].

In order to authenticate the server (thus verifying that the other end of the communication is really the intended *target* server) and to prevent third parties from reading the content of the communication, a TLS session is established (event (5)). Censorship techniques acting on this phase of communication are described in Section 2.3.8. When the URL requested by the user requires the HTTP protocol instead of HTTP this phase is skipped.

Once the reliable, authenticated and private channel is established, the browser issues an `HTTP GET` query to the *target* server (event (6)). This request is subject to censorship of the kind described in Section 2.3.5, and in principle also to techniques in Section 2.3.4, as it is encapsulated in TCP packets.

The *target* server processes the requests (event 7), finding or dynamically generating the resource corresponding to the URL. This is the point where server-based censorship is enforced. Besides the overview in Section 2.1.2 we will not cover this phase in more detail.

The response containing the requested resource is then sent back to the browser (event 8), possibly implying several TCP packets according to the size of the returned resource. This event is in principle subject to the same censorship techniques of event 6.

The browser evaluates the response first evaluating the HTTP level protocol information (9.1), and accordingly parses the (optional) body of the response (9.2),(9.3), then it is possibly induced to request other resources.

Events 6 to 8 are repeated for each request of resources from the browser to the same *target*, until the session is closed; for resources requested to other *target* new communications are established, restarting from message (2).

2.3 Censorship techniques

The techniques employed by the censors can be characterized in terms of the *trigger* that initiates the censoring process (and thus implicitly the phase of communication in which the trigger is sent), the *action* itself, and the *symptom* experienced by the user. A general distinction is between *stateless* and *stateful* censoring technique or system: in the first kind the censoring *action* is performed deciding on a per-packet basis (presence of the *trigger*); in the latter the decision depends on both the information on past packets (status) and the presence of the related *trigger*: what constitutes a *trigger* changes over time according to the sequence of packets that are seen by the *surveillance device*.

The overall censoring system can operate in a single step, or may be designed as *multi-stage*, involving a cascade of two or more devices processing the inspected traffic.

A characterization of censorship techniques along these properties is presented in the following and summarized graphically in Figure 2.4, where the defining properties of one of the techniques are highlighted (two-stage DNS hijacking and HTTP injection, Section 2.3.10).

In the following the techniques are described, grouped according to the type of *action* (and the possible setups) adopted by the censor, also discussing the remaining elements.

2.3.1 BGP tampering

Packet forwarding - the basic functionality of packet-switched networks - is performed according to criteria set by a routing algorithm. In the Internet the routing algorithm that is used by routers to coordinate across different administrative boundaries (Autonomous Systems, ”AS”) is the Border Gateway Protocol (BGP) Rekhter et al. [2006]. Interfering with the intended functionality of the BGP protocol has potential impact to whole sub-networks up to the scale of whole countries. Such has been the case documented during the so-called “Arab Spring”, the events of social unrest manifestation and civil protests occurred during the first part of 2011 in Libya and Egypt [Dainotti et al., 2011], where these countries were made unreachable from outside the country ASes by withdrawing their presence from the BGP network view. When these techniques escape the control of the censor, dramatic side effects can verify, as happened in the 2009 incident that made *YouTube* unreachable also from *outside* the controlled network of Pakistan or a similar event caused by China in 2010 [Mueller, 2012, chap. 9]. The mechanics of such accidental misuse of BGP - that can also be intentional tampering [Feng and Guo, 2012] - have been studied by Ballani et al. [2007], that have estimated that a significant part of the Internet was potentially subject to *prefix hijacking*: the condition in which packets that an AS Y should forward to an AS $C1$ are erroneously diverted to another AS X because X advertised to Y a fake shorter path towards $C1$ (see Figure 2.2).

In the characterization of censorship that we have adopted, this technique presents as *trigger* the destination *or* source IP addresses; in fact by diverting to a black hole one direction of traffic consequently makes bidirectional exchange impossible: TCP connection are surely affected and only one-way traffic (notably related with malicious activities

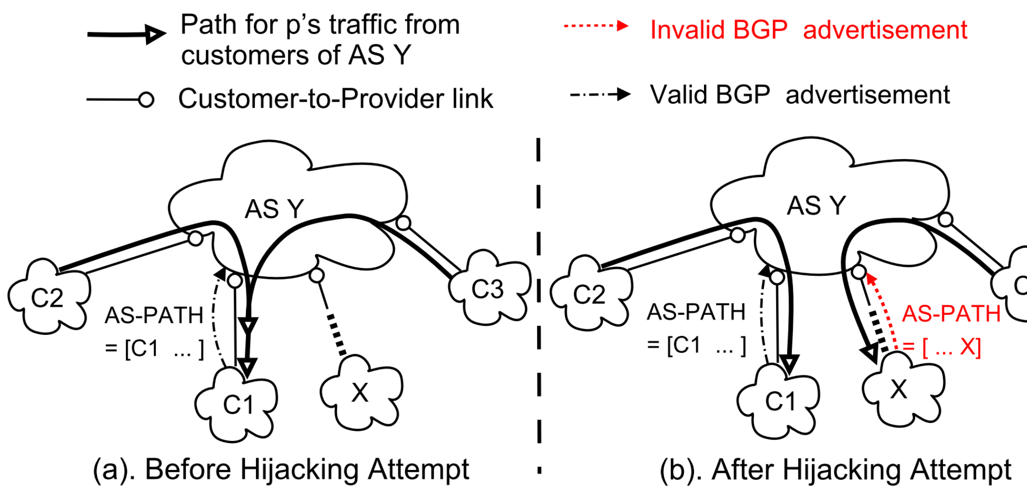

Path for p's traffic from customers of AS Y
Invalid BGP advertisement
Customer-to-Provider link
Valid BGP advertisement
AS Y
C2
C3
AS-PATH = [C1 ...]
C1
X
AS-PATH = [... X]
(a). Before Hijacking Attempt
(b). After Hijacking Attempt

Figure 2.2: BGP tampering - *prefix hijacking*: traffic from AS *C3* to AS *C1* is erroneously diverted to AS *X*. (from Ballani et al. [2007, fig. 2])

- scans, backscatter [Dainotti et al., 2011]) is allowed through. The *symptom* an user would experience is a `network unreachable` error in case of *prefix withdrawal*, and `time exceeded` TTL expiration in case of *prefix hijacking* that leads to loops or too longer paths.

2.3.2 DNS tampering

The access to a *target* usually implies the translation from the symbolic name of the server hosting the resource to its IP address (see exchanges 2 and 3 in Figure 2.1). The communications related with this phase can be subject to different censorship techniques, first analyzed in detail in [Dornseif, 2003]. The sequence diagram of a DNS request is shown in Figure 2.3: the software module on the client (referred to as "stub resolver") issues a `type A` query to the *recursive resolver* [5], that is in charge of performing the resolution asking the authoritative servers (or providing a previously cached response).

We have adopted the umbrella term "DNS tampering" to avoid confusion with the term "DNS redirection" found in literature [Gill et al., 2013] to specify one of the possible effects (thus a *symptom* in the lexicon defined in this Thesis) of these censoring techniques, while there are different variants involved with this process, according to (i) the presence

[5]The recursive resolver IP address is provided by the network administrator or obtained by automatic host configuration, and usually corresponds to a server of the Internet Service Provider that gives the client host connectivity to the Internet.

of *surveillance devices* on the path between the client (stub resolver) and the recursive resolver and (ii) the kind of response that is provided back. These variants are described in the following.

DNS hijacking

According to the protocol definition [Mockapetris, 1987b], when a DNS recursive server is queried for a resource record it should fetch it from the authoritative servers (if a cached entry is not found): censoring servers instead reply with a forged response, not corresponding to the legitimate DNS database entry. Having administrative control on the DNS server allows to alter its behavior diverting it from the standard; the following possible responses are given:

NXDOMAIN : an error response of type "no such domain" - the domain name referenced in the query does not exist [Mockapetris, 1987a]

forged Resource Record : a Resource Record of `type A, Class IN` with an IP falling in one of the following cases:

- *surveillance IP*: the returned IP address is assigned to a surveillance device that inspects higher layer protocols;
- *Block Page*: the returned IP address hosts a webserver that invariably serves a page telling that the requested hostname *has been deliberately blocked*.
- *Failing IP*: a non-Internet-routable addresses such as *private address space* [Rekhter et al., 1996] or *shared address space* [Weil et al., 2012] as well as ordinary assigned IP addresses unrelated with the requested resource;
- *Error Page*: the returned IP address hosts a webserver that invariably serves a page telling that *the requested hostname is not existent or misspelled*;

The *symptom* that the client experiences is therefore different according to the replies it gets: only in the "Block Page" case the censorship is clearly notified, possibly providing a motivation or a law demanding it, and in some cases a reference to appeal the censorship.

If a DNS error is returned, a tech-savvy user can infer that something went wrong with the name resolution phase (which is indeed true).

The same happens for the "Error Page" case: against the definitions of the related protocols, the error is surfaced to the application level but still gives hints about the phase that failed.

When a “Failing IP” is provided, an error of type `network unreachable` or `host unreachable`, or a TCP connection error will be returned to the user, giving no information about the real event. In case the returned IP address is assigned to a host, the *symptom* will be different according to whether a service is listening at the transport port (usually 80, for HTTP requests) and how it will react to the HTTP request (a `HTTP 404` resource not found error).

This is but one example of how little transparency an user would experience when dealing with censored resources, and how simple detection tests (such as the one performed through the [Herdict] platform 3.3.1) can not give reliable information about the censorship technique.

DNS tampering is also used in two-stages censoring techniques, in order to divert selected traffic to a *surveillance device* that will inspect and possibly cause censorship (see Section 2.3.10).

This censorship technique has been documented since the early analysis of internet censorship [Dornseif, 2003] and confirmed in most of the field tests worldwide [ONI]; we too have found such results in the experimental validation (e.g. Section 5.2.1).

The *trigger* for this censorship technique is a `UDP` port 53 packet directed to the IP address of the misbehaving DNS recursive resolver and containing a DNS query of `type A` including the hostname belonging to the blacklist. In order to enforce censorship with this technique it is not necessary to have a *surveillance device* on the network path between the client and the *target*, as the client will issue a direct request to the recursive resolver (that acts as the *censoring device*). This on the other hand makes the circumvention of this censorship technique straightforward: changing the default recursive resolver will avoid the *censoring device* and thus leave open access to the Internet. Actually the change of the default (ISP-provided) DNS recursive resolver with other “open” resolvers is not rare, but can adversely impact the user experience [Ager et al., 2010].

DNS injection

Differently from DNS hijacking - performed directly at the recursive resolver - a more sophisticated technique is the *injection* of forged packets that imitate the legitimate response of the queried DNS server but providing fake data. Injection can happen at different locations of the network, not necessarily on the path between the client and the *target* and requires a *surveillance device* on the network path between the stub resolver and the recursive resolver or between the latter and the authoritative server that should provide the requested Resource Record.

Looking at Figure 2.3 it can be noted how a *censoring device* has the opportunity of replying to the stub server faster than the queried resolver: the first well formed message arriving to the client will be accepted, and a possible subsequent legitimate one will be ignored.

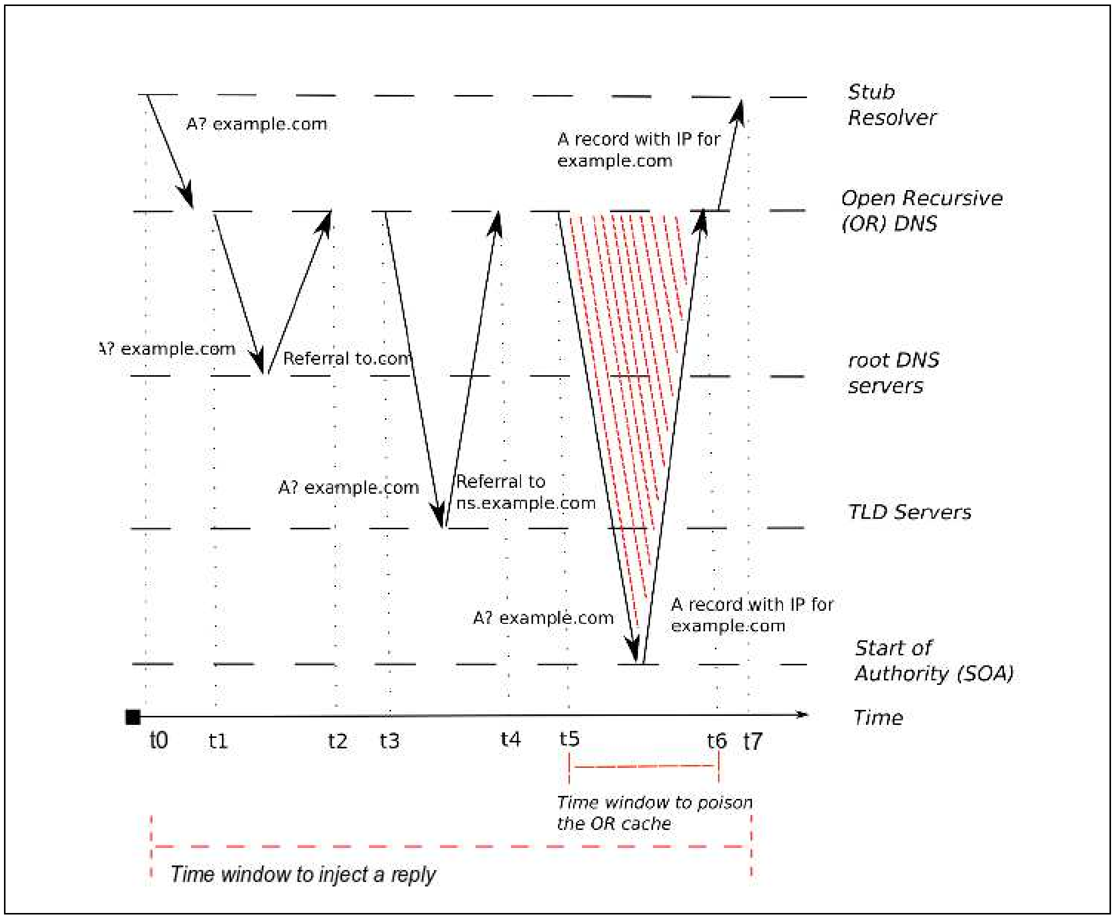


Figure 2.3: DNS tampering - *injection*. (Modified from Dagon et al. [2008, fig. 2] with the addition of the highlighted time window on the bottom: the interval allowing injection between the stub resolver and the recursive resolver.)

The *trigger* for this technique is similar to the hijacking performed at the ISP recursive resolver (`UDP` packet with destination port 53, carrying a DNS query of `type A` with the blacklisted hostname) but in this case there is no need to have the default DNS recursive resolver for the IP destination address: as long as the packet reaches the *surveillance* device the censorship will be applied. This makes ineffective the simple circumvention technique adopted against the DNS hijacking described before, as also the queries addressed to third party resolvers will be intercepted and replied with the tampered data.

The types of responses that are injected are the same as the aforementioned ones, and thus the same *symptoms* are experienced by the client. Due to the different mechanics, however, this technique can have much broader impact than intended, as found in [anonymous, 2012], where it is shown how censorship applied by transit ASes affects also DNS queries originating from foreign countries, finally censoring the *target* for peoples that are not subject to the censor's jurisdiction.

DNS-Sec DoS

The original design of DNS did not assume an hostile network environment, hence the weakness of this protocol in the face of tampering; to extend it while retaining compatibility with the existing infrastructure the Secure DNS (DNS-Sec) specification has been proposed [Atkins and Austein, 2004].

The adoption of DNS-Sec provides secure authentication of server response, thus preventing the possibility of injecting a forged response that could be accepted as valid. From an accessibility point of view, this ultimately prevents the access to the *target*, constituting a form of Denial of Service and succeeding in the censoring intent. Moreover, redirection to fake or warning content would be impossible and so would be "informed blocking": the only *symptom* a client would experience is a DNS-Sec error reporting that the name resolution system has been tampered with. In this case the expert user could be aware of the man-in-the-middle attack she is undergoing, but an user ignoring the network technicalities could as well blame the *target* for the failure.

Other possibilities for the censor to work-around DNS-Sec are also considered in [Vixie, 2012], but they are all discarded concluding that current DNS tampering techniques are not compatible with DNS-Sec adoption. Though current deployment of DNS-Sec may be limited [Lian et al., 2013], it is expected to grow as security concerns mandate it, thus enlarging the negative impact on transparency of DNS level censorship.

2.3.3 Packet filtering

We group under the term "packet filtering" all the censorship techniques whose *action* is to simply discard packets. With this technique the triggering packet is silently dropped causing a *symptom* of type `connection timed out` error. The *trigger* is data from the headers of *up to* the fourth level of the TCP/IP network stack (thus including also network layer). The motivation for associating *triggers* of different layers to a single censoring technique (while in principle they should be used to tell apart different setups) is that the enforcement of censorship based on the headers of these two protocols have little practical differences: packet filtering of varying complexity is a standard functionality provided by network devices ranging from switches to dedicated security appliances.

This technique requires a *surveillance device* on the path between the client and the *target* (as opposed to BGP tampering and DNS hijacking), and the *censoring device* must be in-line too (as noted, they are possibly the same device). In a stateless censoring system this technique can be used to block IP addresses of targets that are also subject to DNS tampering, so that if the client circumvents censorship in the DNS resolution phase (Figure 2.1, message exchanges up to "(3)") it is caught on the first packet of the TCP handshake (Figure 2.1, message exchanges "(4)") . It has been found *in the wild* [Dornseif, 2003, Murdoch and Anderson, 2008], but on the one side it has the shortcoming of blocking all services reachable at the same IP address (thus "overblocking" virtual hosts that, though having a different hostname, are hosted on the same IP), on the other side it requires the censor to collect all the IP addresses associated with the targeted resource and configure the *surveillance device* with rules for all of them. The same setup and *trigger* can be used to selectively route packets towards an off-path *surveillance device* in multi-stage censoring systems (see Section 2.3.10).

2.3.4 TCP connection disruption

The disruption of the communication during either the setup of the TCP connection (Figure 2.1, phase (4)) or during the subsequent packet exchange belonging to the same connection, i.e. sharing the same 5-tuple (source IP, destination IP, protocol=TCP, source port, destination port) .

Techniques enforcing this *action* leverage the connection-oriented nature of the TCP protocol: a notion of "state" of the connection is held at each endpoint and signals are used to set-up and tear-down the connection. The *censoring device* sends to the client packets that have the source IP of the *target* and the `RST` flag set, indicating that the connection is in a wrong state and thus has to be immediately terminated. The client will experience a *symptom* of type `connection reset` error.

The *trigger* for this technique contains the destination IP address or of the *target*

[Clayton et al., 2006] and possibly the transport level port numbers, to limit censoring to a specific application such as HTTP (port 80), HTTPS (port 443), SSH (port 22); this requires a *surveillance* device on the path between the client and the *target* (as opposed to BGP tampering and DNS hijacking).

This *action* can be used as in two-stage techniques (see Section 2.3.10).

2.3.5 Keyword blocking

A *surveillance device* that analyzes packets beyond the headers of IP and TCP is said to be performing Deep Packet Inspection. Different depths of inspection are possible depending on the payload that is analyzed (first packet, a given number of initial packets, all the packets); moreover different degrees of complexity in the analysis of such payload can be adopted (per-packet string matching, per-stream string matching, syntactical verification, semantic verification), progressing from a stateless to stateful inspection with increasing status information kept per connection. The more payload is analyzed and the more complex the analysis, the higher the resources required for the *surveillance device*[6].

The *action* performed by the censoring system can be of the same kind of the ones adopted in TCP-level filtering, but having an established TCP connection between the client and the *target* there is also the chance to provide application data, e.g. to redirect to a blocking page, or performing varying degrees of content mangling.

An example of enforcement of such kind of *action* for applications using HTTP is *HTTP tampering*, described in the following.

2.3.6 HTTP tampering

If a TCP connection is allowed to complete successfully (reaching the phase (6) in Figure 2.1, the censor has the opportunity of providing the client with an HTTP message that will be parsed by the client as it were coming from the queried *target*.

The HTTP protocol[7] messages are divided in two parts: the *header* and (not for all types of message) a body; a fundamental part of the header for response messages[8]is the *status code*, a numeric value of three digits characterizing the message type. According

[6]see [Risso et al., 2008] for a comparison of accuracy and resource consumption of different traffic classification methods

[7][Fielding et al., 1999]

[8]"response messages" are sent by the server when replying to "request messages", from the client.

to the different *triggers* that the *surveillance* device looks for in the request message, the different response messages that the *censoring* device sends back to the client, and the location of these components of the censoring system, the variants described in the following are possible.

The *action* of the *censoring* device in the case of HTTP tampering consists in sending an HTTP response message belonging to the following types, with some status codes grouped by the first two digits:

- code `30X` redirect, signaling that the requested resource is at another URL, reported in the "location" header field; this will cause the web browser to begin a new communication sequence (starting from phase (2) in Figure 2.1 to the given address; this constitutes an "HTTP redirect" and the experienced *symptom* for the user will depend on the result of the new communication;
- code `404` resource not found: the path of the requested URL does not correspond to a resource known to the server; the *symptom* experienced by the user will be a browser-provided error message describing the error;
- code `403` forbidden: the request is valid but the resource will not be provided; the body of the message can contain a description of the reason; the *symptom* experienced by the user will be the content of the page if present, or a browser-provided error message describing the error
- code `200` no error, signaling that the requested resource is provided in the body of the message; the browser will parse the content.

2.3.7 HTTP proxy filtering

A special case of semantic stateful inspection is constituted by a transparent proxy located in-line with respect to the path between the client and the *target*, by all means performing a MITM, forwarding only content that does not match the blacklisting rules. The proxy fully understands the application level protocol, hence the keyword matching can be done on specific statuses of the application protocol automaton, and on specific data fields, thus being highly selective and reducing possibilities of overblocking. The downside is that the transparent proxy must be in-line and that it is required significant processing power, otherwise performance impairing or altogether blocking could occur as a byproduct also for traffic that should not be censored.

An alternative is to have multi-stage censorship system, with the preceding stages in charge of pre-filtering (and blocking) connections or “hijacking” towards the *censoring device* the suspicious connections only (see Section 2.3.10).

2.3.8 TLS tampering

One widely adopted defense against various kinds of *Man-In-The-Middle*(MITM) attacks in accessing content on the Internet is provided by Transport Layer Security / Secure Sockets Layer (TLS/SSL)[Dierks and Rescorla, 2008]. This protocol operates over the transport layer of the TCP/IP stack and offers an authenticated and private channel to the application protocol (thus behaving as a lower sub-layer of the application layer). Besides HTTP, that we are considering in the simplified communication model represented in Figure 2.1, it can be adopted to secure other application layer protocols such as FTP, SMTP, XMPP, based on TCP[9].

Basically TLS performs a session setup phase using asymmetric cyphers whose keys (in form of SSL “certificates”) are verified by means of a trusted third party, the “Certification Authority” (CA). Once the negotiation of a symmetric cypher and session key is completed, the remaining communication will convey the encrypted data of the application level protocol.

TLS compelled certification creation attack

Although a number of weaknesses and countermeasures have been proposed in the past, a recent “politically sensitive” attack scenario has been presented in [Soghoian and Stamm, 2012] that could be easily employed by a censor in order to tamper with the communication (e.g. editing content) in ways unnoticeable to the user. The effectiveness of the attack is based on providing an SSL certificate for the site attesting a false identification, but still results as valid because it is signed by a trusted (but actually *misbehaving*) CA. In the attack scenario introduced in [Soghoian and Stamm, 2012] this CA can be compelled by government institutions to provide a site-specific certificate or even an intermediate certificate that can be used to generate valid ones for any website (hence the name of *compelled certification creation attack*).

[9]for applications relying on the UDP transport protocol there is a dedicated version, Datagram Transport Layer Security [Rescorla and Modadugu, 2006]

While injecting an *in*valid certificate would warn the user that some kind of issue is happening (still not declaring that it is intentional censorship), the use of a rogue certificate would completely hide the tampering and still provide the user with both tampered content and a false sense of confidence in its authenticity.

The *trigger* of this technique is an HTTPS request with an hostname for which the compelled CA has generated a rogue (fake but valid) certificate. The *symptom* is the reception of mangled content without the TLS protocol being able to alert for it.

To the best of the candidate's knowledge, there is no documentation about the use of MITM attacks by means of *compelled certification creation* aimed at censoring content, but only at accessing encrypted communications (e.g. in the Iranian GMail incident [Leyden, 2011]); nonetheless there is no theoretical obstacle to adoption for censorship in addition to the known surveillance application.

TLS failure

Analogous to the DNS-Sec case, if the censor tries to intercept the beginning of the communication that is protected with TLS and fails to provide a valid certificate then the TLS layer will warn the application that the validation of the certificate has failed and thus a MITM attack could have been attempted.

The *trigger* of this technique is an HTTPS request with an hostname for which the censor does not provide a valid certificate, and the *symptom* is an error (usually presented with an evident pop-up offering the user to either abort the connection or to override the failed certificate check and continue). If the user refuses to continue will not have access to the *target*. Again, an informed user is able to understand what happened, while an ordinary user could blame the *target* itself or the browser security features. If the user decides to ignore the validation error then the censor has access to the cleartext communication and thus can apply the *surveillance* and *censoring* techniques described before (again preventing the user from accessing the unmodified *target*).

2.3.9 Soft Censorship

Blocking the access to an online resource or service is an evident action, that if prolonged in time stands out as intentional, and possibly drawing attention and strong complains. Instead, gradually reducing the Quality of Service (QoS)[10] and thus the perceived per-

[10] [Kurose, 1993].

formances is much less evident, and it's harder to prove intentionality. Inconstancy of performance also adds up to the difficulties in measuring this action, and also on the frustration of the user, that ultimately will look for -allowed- alternatives: such is allegedly the case for the Chinese offer of online services replicating foreign analogous [J., 2011]. The intended effect thus, i.e. preventing the user from accessing some resource or service, is reached anyway.

In order to enforce this kind of *soft censorship*, also called *throttling*[Anderson, 2013], tools initially devised to guarantee QoS - and later used to violate *network neutrality* for economical advantage - are being employed [Aryan et al., 2013].

The *action* corresponds to the worsening of QoS parameters:

- increased packet loss
- increased delays
- increased jitter
- reduced bandwidth

One simple method to achieve these results would be to filter random packets along the path. In the case of TCP connections this would significantly impact the delays and throughput due to the connection-oriented nature of the protocol, bound to retransmitting lost packets and waiting for in-order reassembly. UDP instead would not suffer additional damage besides the packet loss, but the communication is still heavily affected in case the application protocol that is carried in UDP has its own loss recovery mechanisms.

Such *actions* can be implemented in routers and thus classified as a special case of "TCP/IP" filtering, and can adopt both a stateless or a stateful paradigm.

As for the other cases analyzed so far, a preferential location for the *censoring* device would be on national border connection gateways.

The *trigger* can be, like ordinary (blocking) censorship, related to specific *targets*, but a notable difference is that in this case the *trigger* can also be void, i.e. all communications, no matter the *target* or protocol, or content - will be subject to the *action*. This scenario is possibly based on external events (not elicited from the network) that cause a curfew-like control of Internet access [Aryan et al., 2013] reminding of the nation-wide disconnects experienced in the events of the "Arab Spring"Dainotti et al. [2011].

2.3.10 Multi-stage blocking

Having a *surveillance / censoring* device working in-line requires that it has to be deployed at country borders in order to intercept traffic directed to foreign countries (hosting *targets* that are legally out of direct reach for the censor). This setup poses two technical problems: (i) it has to process all the cross-border exchange traffic, potentially suffering performance issues; (ii) it constitutes a Single Point of Failure: any issue in this equipment would disconnect the served network from the Internet, with potentially high economic loss.

This problem has been solved in known censorship systems by employing multi-stage systems, e.g. preselecting at the first stage a much smaller fraction of "suspicious" traffic, discharging the second-stage device from processing large part of permitted traffic. This way the finer-grain blocking of the second-stage censoring device does not come at a high cost.

Another solution uses a deployment where the first stage is akin to a mirroring port just copying the traffic to an out-of-band *inspection* device, that thus does not impair the transmission performance of the original flows. Such a setup has a drawback, though: the enforcement of injection techniques is more challenging for the censor if the *censoring device* is not in-line with the path between the client and the target *target* and thus can not discard the request. In this setup the *censoring device* is engaged in a race condition against the legitimate response of the *target* server and its forged packets could arrive after the *target* reply; this would make the client ignore the censor packets due to the mismatch of the TCP sequence number[11].

In the following some examples of multi-stage deployments are described.

TCP filtering + BGP hijacking + HTTP proxy

One of the first descriptions of a multi-stage deployment is provided in [Clayton, 2006]: here the aforementioned reasons in favor of this kind of deployment are stated as motivation for the design of the system. The first-stage is triggered by destination IP address and TCP port number. In case of match, the packet is routed (by means of BGP) to an HTTP proxy that can access the hostname and the specific resource (second-stage *trigger*). Allowed URLs are fetched from the original *target* and served back to the client,

[11]the initial sequence number is randomly generated to to make harder for an off-path attacker to guess it [Gont and Bellovin, 2012].

while blacklisted ones will trigger an *action* of the kind reported in Section 2.3.7 - in [Clayton, 2006] requests are ignored, thus the client waits in vain until the *timeout* is struck.

DNS hijacking + HTTP proxy

This technique is triggered by DNS queries (UDP port 53) of *type A*, i.e. requiring the translation of a hostname belonging to a blacklist; the censoring DNS server replies providing an IP address that belongs to the second-stage surveillance device. Then the browser establishes a TCP connection directly with the second-stage surveillance device.[12] To an HTTP *GET* request including an URL belonging to a blacklist (secondary *trigger*) the second-stage censoring device replies with one of the *actions* seen in Section 2.3.7. If the requested URL was not blacklisted then the request is let pass.

Keyword blocking + TCP disruption

The *trigger* is constituted by a TCP/port 80 (HTTP) packet containing an `HTTP GET` request for the *target* URL[13] and as IP destination address the *target*'s address.

If the URL contains a blacklisted keyword then the TCP connection disruption

This deployment has been found operating in China and has been analyzed in detail, showing evolution in complexity over time [Clayton et al., 2006, Feng and Guo, 2012, Polverini and Pottenger, 2011, Verkamp and Gupta, 2012, Weaver et al., 2009, Xu et al., 2011], with different levels of sophistication in the craft of `RST` packets[14]. The analysis revealed stateful implementations for both stages: only the HTTP request belonging to a correctly established TCP connection triggers the censorship, while after it has been activated all packets sharing the 5-tuple (IP source and destination addresses, TCP transport protocol, source and destination ports) generates the TCP connection disruption *action* [Xu et al., 2011].

[12]If HTTPS is employed, at this point the browser attempts a TLS session establishment, that will fail unless a TLS compelled certificate creation attack has been performed (see Section 2.3.8). If the user, uncaring or unaware of the risk, allows the browser to accept the invalid certificate anyway, then the process continues as if HTTP were used.

[13]the payload of the TCP packet contains the strings of the query part of the URL (after the `GET` string) and the hostname part of the URL (following the `Host:` string).

[14]see [Weaver et al., 2009] for an experimental survey of the types of packets forged by *censoring devices*

2.4 Circumvention

A logical consequence from the awareness of censorship and progress on understanding its working details is the proposal of methods to dodge it, collectively named *censorship circumvention* (just “circumvention” in the following). Besides papers focused on circumvention itself, often papers discussing censorship and censorship detection add also the related analysis of possible circumvention methods: references to both cases are provided hereafter in the form of a brief survey of literature on circumvention.

Methods, tools and platforms have been specifically designed to counter censorship: in [Elahi and Goldberg, 2012] a taxonomy is presented that characterizes thirty among circumvention tools, platforms and techniques according to a number of properties, namely: the ability to make censorship damages outweigh its benefits, avoiding censor control over entities or traffic, avoiding censors surveillance, and other means along these main ones.

Another valuable source for scientific literature on censorship and circumvention is the webpage “Selected Papers in Censorship”[15].

The early analysis of network-based censorship techniques [Dornseif, 2003] cites a number of possible circumvention techniques: for each a brief analysis is presented, considering which censorship technique is circumvented and what is needed by the user to apply the technique, concluding that albeit several workarounds are possible they can be technically complex, burdensome or needing collaboration by a third party, thus resulting not easy to be applied for a common user.

A few techniques for circumvention of application-level keyword-based censorship are suggested in Crandall et al. [2007]. These techniques are based on the knowledge of the blacklist of keywords that trigger censorship, and are described as asymmetric techniques in the sense that are server-based and do not require changes in the client. We note that while some of them - namely IP packet fragmentation, insertion of HTML comments and varying characters encoding- exploit standard functionalities of the stack on the client that could virtually be unaffected and provide results equivalent to the non-mangled communication, other (captcha, spam-like rewording) likely will disrupt automatic functions such as indexing and text search and are meant solely for human interpretation.

In [Clayton et al., 2006] a technique is presented to circumvent a specific censorship (TCP-RST communication disruption) by ignoring the forged RST packets, also consider-

[15]www.cs.kau.se/philwint/censorbib/

ing TTL-based validation to identify forged RSTs from legitimate ones. The implications are discussed (adopting a non-RFC-compliant stack on both web server and client) and compared with the use of encryption. New censorship techniques and variants namely Revised Sequence Number Prediction Attack, Forged SYN-ACK Response (detected and described in [Polverini and Pottenger, 2011]) suggest that the proposed circumvention technique is no more effective.

A recent field survey on circumvention techniques in China has been published as technical report in [Robinson et al., 2013]: listed circumvention tools are Freegate[16], UltraSurf[17], web proxies, SSH, while only 16% used personal VPNs; according to the authors, a 15% that didn't tell the name of the tool possibly use GAppProxy[18], predecessor of GoAgent[19] tool.

Even if not specifically designed for censorship circumvention, anonymity technologies can and have been used to circumvent censorship: a recent survey on usage and geographical distribution of several technologies including proxy servers, remailers, JAP, I2P, and Tor is provided in [Li et al., 2013].

The availability of Internet Censorship detection and monitoring tools is a strict requirement for the development and evolution of circumvention technologies: this adds another motivation to the present Thesis work.

[16]http://www.dit-inc.us/freegate
[17]https://ultrasurf.us/
[18]http://gappproxy.sourceforge.net/
[19]https://code.google.com/p/goagent/ (documentation in Chinese)

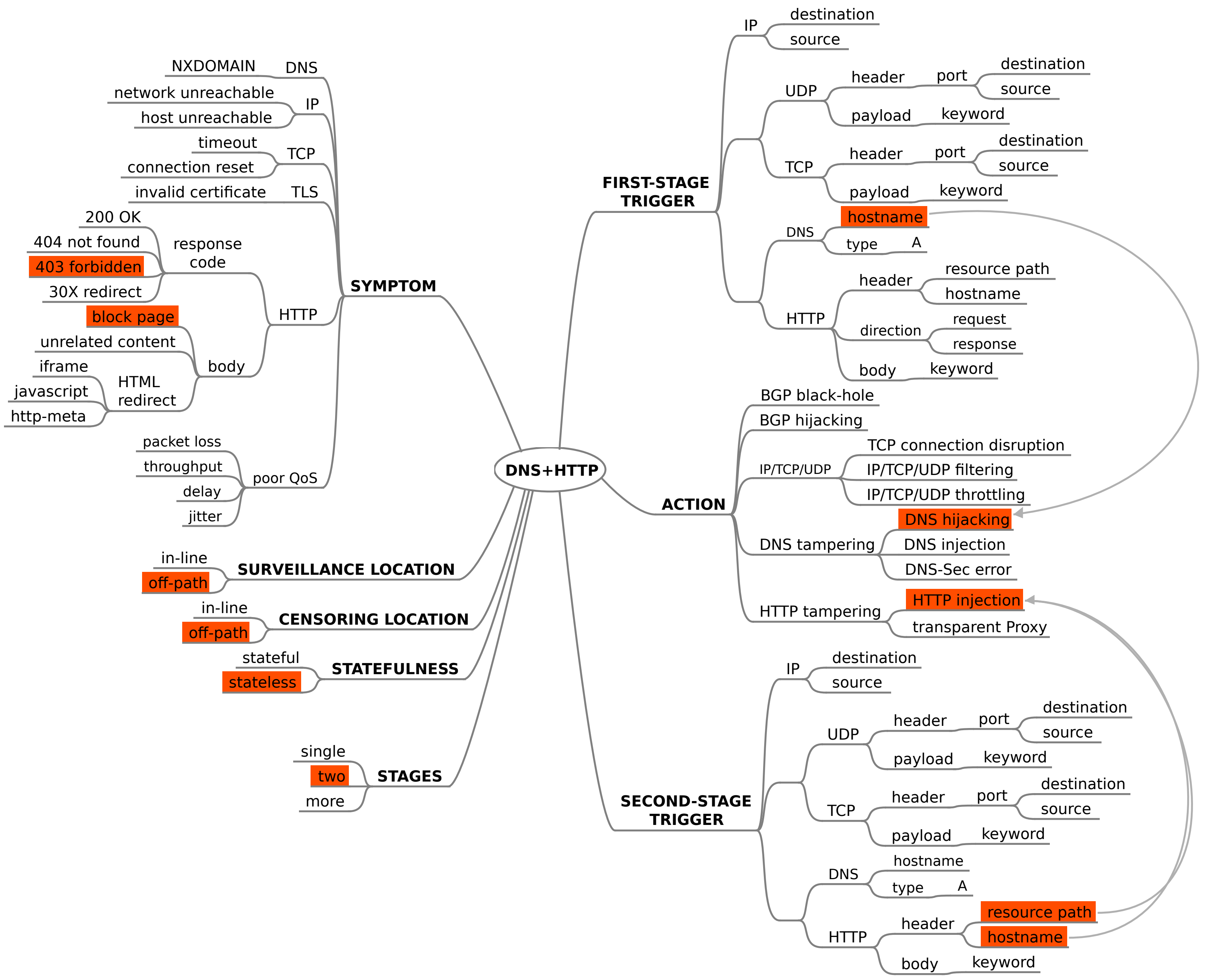


Figure 2.4: Characterization of Network-based Censorship Techniques - the two-stage technique is described in terms of the different axes: matched properties are highlighted.

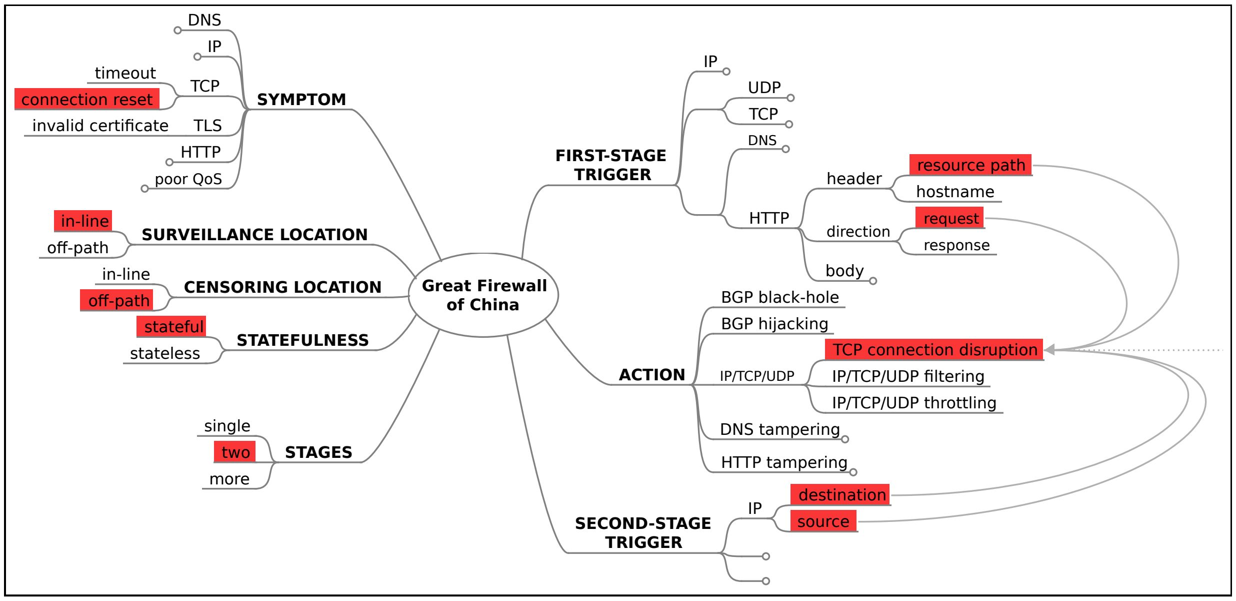


Figure 2.5: Characterization of Network-based Censorship Techniques - the two-stage technique (as detected in the analysis of the *Great Firewall of China* [Xu et al., 2011]) is described in terms of the different axes: unmatched section are collapsed, matched properties are highlighted.

Chapter 3

Detection of Internet Censorship

To be able to carry on an informed discussion and analysis of censorship, is of paramount importance the ability to assess and understand its actual usage. In fact significant aspects of censorship, such as its enforceability, its transparency and the accountability of the censors to the affected population, strongly depend on the technical details of the adopted censorship technique and thus evolve with the technology and real usage of it.

The evidence of censorship is fundamental to raise the awareness in international scenarios of the social cost of censorship methods. Finally, side effects of the application of censorship (dubbed "overblocking"), that play a major role in the ethical, political and economical feasibility of its enforcement, are also bound to the technical details of the adopted method.

In coherence with the definition of Internet Censorship we have adopted in Section 2.2, we consider the Internet Censorship Detection[1] as "*the process that, analyzing network data, proves the existence of impairments in the access to content and services caused by a third party (neither the client system nor the server hosting the resource or service) and not justifiable as an outage*". We implicitly include the collection of the suitable network traffic data in the Internet Censorship Detection as it is a fundamental phase of the process. We also note that, as stated in Chapter 2, in the present work the focus is on Network-based Censorship, and thus Detection refers, unless differently stated, to the related techniques (Section 2.1.3).

Detection is essentially based on the ability to tell the effect of the censorship from the "normal" uncensored result and from involuntary outages; for a class of detection methods (active detection), it requires also the possibility of intentionally triggering the

[1] hereafter also "detection", when no ambiguity would derive

supposed censorship system. The inference of the adopted censorship technique is inherent to identifying the type of third party causing the impairment and its differentiation from an outage.

With reference and in addition to definitions stated in Section 2.2, censorship detection techniques can be characterized considering two main aspects:

viewpoint : the role of the *probe* host in the client-server communication model:

- **client-based** : collected network traffic is initiated by the same IP address of the probe host;
- **gateway-based** : collected traffic has neither source nor destination IP addresses belonging to the same network of the probe; in the scenario of interest the *probe* host is a gateway towards Internet: all traffic between the served network and the Internet passes through it;
- **server-based** : collected network traffic is initiated by other addresses towards the same IP address of the probe host;

collection : the collection of network traffic data can be performed with *active* or *passive* techniques

- **active collection** : techniques that use client systems ("probes") to generate network traffic purposely crafted for possibly eliciting a censorship response (to be recorded and analyzed).
- **passive detection** : techniques that collect traffic data from network application logs or traffic traces captured on the device ("probe") in order to look for evidence of censorship events;

A special case crossing these definitions is the usage of *active* methods towards a controlled destination: as both sides of the communication are controlled (both qualify as "probe") then traffic can be collected also at the receiver side (*passive* collection method). As for the *viewpoint*, if both edges are controlled then once the server receives client traffic can also respond with a purposely crafted reply (e.g. containing a *second-stage trigger* for a stateful censorship system); finally, probes can switch role, allowing directed testing of the network paths in-between. This setup is akin to the one used in the methods of *network tomography* [Castro et al., 2004], thus in analogy we name it "censorship tomography", as a special case of *active* methods. In fact even though *censorship tomography* implies logging of received traffic, the possibility to generate traffic purposely forged to trigger some specific mechanisms is the essential property that characterizes *active* methods.

3.1 Active detection methods

For *active* censorship detection tools, the algorithm consists in the following steps:

- generate traffic towards a *target*, supposed reachable;
- if the request gets back no results or an error then censorship is inferred;
- if the received content is equal to a Ground Truth or satisfies a similarity criterion, then the *target* is considered reachable and not subject to censorship

In order to analyze the censorship technique, variations on the request can be made to pinpoint the *trigger*: by comparing the *symptoms* collected for the different tries, information about both the *trigger* and the *action* is obtained and then existence and properties of the *censoring system* can be inferred. We stress that, without the ability to selectively investigate the censoring technique mechanics, a detection tool can hardly tell censorship from outage, as it is the coupling (*trigger*,*symptom*) confronted with the case (*non-trigger*,*expected uncensored behavior*) that can surface the existence and nature of a censoring infrastructure and thus the intentionality of the communication impairment.

3.1.1 DNS resolution

A widespread censoring techniques involves the DNS resolution process, that is involved every time a symbolic name for the host is provided (see Section 2.3.2). Possible variants involve ISPs altering their copy of the distributed DNS database so that a wrong reply is given to clients ("DNS hijacking") ; or a in-line *censoring device* intercepting the DNS queries and replying in place of the intended DNS server ("DNS injection").

From the detection point of view the cases of the complacent ISP and the one of the intercepting device can be told apart by querying alternative Name Servers hosted outside of the censored network [Nabi, 2013]: if the forged Resource Record is only in the database of the ISP's default Name Servers, the alternative Name Server will provide different (correct) answers; in case an intercepting device is actively tampering the DNS traffic then all responses will be equivalent (and wrong), and detection will be possible only by comparing results with the ones collected from probes outside the censored network.

Limiting the analysis to just the Name Resolution phase, in case of DNS tampering a comparison with results of measurements from other countries will show that a given (censored) host name is resolved to two different sets of IP addresses. Unfortunately, this is also the case for DNS-based load balancing and more in general performance enhancement usually provided by Content Delivery Networks (CDNs) [Pathan and Buyya, 2008]. Therefore a detection test that compares the set of resolved addresses of a hostname among different countries would systematically suffer of false positive errors specially for high-traffic addresses (mostly likely to leverage CDNs). This test can instead be used as an exclusion criterion: if the two sets are identical, then **no DNS-based censorship** has been applied for the considered resource.

In [Gill et al., 2013] a method is proposed to infer DNS tampering (there named "DNS redirection") by considering all the hostnames resolved to one same IP address in the test results, and count the number of different ASes the same hostnames are resolved to by a trusted DNS server: if the AS count is greater than an empirically set threshold [2] then the response of DNS tampering is given.

3.1.2 IP reachability

Reachability of a *target* at the network level has been performed since the early days of the deployment of Internet by means of the standard utilities `ping` and `traceroute` [Jacobson, 1999].

The first one is an user-level interface to send ICMP `echo requests` messages to an host, receiving back an `echo reply` demonstrating the mutual reachability of the two, i.e. that directed paths forth and back exist between the sender and the receiver.

The original traceroute also used ICMP echo request packets, but with an increasing TTL counter, so that at each expiration an ICMP `time-exceeded` error from the last reached router would be returned, altogether with the router IP address; basically collecting the sequence of addresses the path between the sender and the destination would be discovered hop-by-hop.

These techniques can carry only *triggers* of kind *source/destination IP address*, thus they are useful to pinpoint censorship techniques of type *IP-based filtering* (Section 2.3.3).

There are several possible causes of inaccuracy or unresponsiveness [Luckie et al., 2008,

[2] the threshold is set to 32 ASes considering the rate of growth of the percentage of blocking detected with other methods [Gill et al., 2013, appendix A.1].

Marchetta and Pescapé, 2013] that would lead to false positives when using this techniques as censorship tests), thus using them alone has limited usefulness.

Both `ping` and `traceroute` have been used in [Feng and Guo, 2012] as a preliminary active detection technique.

Variations on the `traceroute` technique are described in Section 3.1.5.

3.1.3 TCP reachability

In order to test if censoring techniques of type *packet filtering* or *TCP connection disruption* (Sections 2.3.3 and 2.3.4 respectively) are employed by the censoring system all is needed is a tool able to log transport and network level errors when sending TCP packets with the suitable *trigger*. A tool to start probing with the minimum *trigger* towards a *target* has just to try a three-way handshake: `netcat` [3], a mature ad widely used tool for network diagnostic and security testing, has been adopted for this purpose [Clayton et al., 2006].

Often the detection techniques adopted in research papers or in the provided platforms and tools just leverage the application level log files and error reporting functionality to detect the disruption of TCP connection [Aryan et al., 2013, Gill et al., 2013, Nabi, 2013, Polverini and Pottenger, 2011] e.g. triggering the system directly by sending a - possibly innocuous - HTTP GET request (see following section). At the other end of the spectrum of possibilities there is the technique adopted in [Khattak et al., 2013], generating different of combination of initial packets (with different flags set) in order to test a *censoring* device statefulness.

3.1.4 HTTP tampering

The detection of censorship techniques of type *HTTP tampering* (Section 2.3.6) is the most frequent in literature, being considered virtually by all the censorship-related papers cited so far.

The common procedure is to use an application to request a resource by means of the HTTP protocol, the *trigger* being set in the `header` section of the HTTP request, either in the `query` part of the GET request - that is interpreted as a path to a resource on the server -, or in the `Host:` header field - that is interpreted as the hostname and

[3]http://netcat.sourceforge.net/

used to identify one specific website if many are hosted at the same IP address ("virtual hosting").

The main source of variation is the specific tool adopted to generate an `HTTP GET` request, comprising Python scripting [Filastò and Appelbaum, 2012, Gill et al., 2013, Nabi, 2013], the command line common unix utility `wget` [Polverini and Pottenger, 2011], or more exotic tools such as `fragroute` [4] [Park and Crandall, 2010] and `scapy` [5] [Crandall et al., 2007, Khattak et al., 2013].

In case a *censorship tomography* setup is adopted, an helper server is used to receive the requests (usually with no *trigger*) and reply back with a blacklisted keyword in the content of the response HTTP message. Such a setup has been proposed in [Filastò and Appelbaum, 2012] using programmable back-ends, and previously adopted by Park and Crandall [2010] to detect HTTP response filtering in the Great Firewall of China (found to be dismissed). Recently Khattak et al. [2013] have adopted this setup to investigate the details of statefulness (and more in general the vulnerability to circumvention) of the GFC.

In case the censoring system successfully submits a -potentially mangled- content, the ultimate detection technique to check whether a resource has been tampered with would be to compare the received content against a Ground Truth. If the content is an HTML page of a dynamic web site a verbatim comparison would almost surely fail, as there are variable components such as timestamps, localization effects, and in general every dynamic content managed through server-side scripting would vary the HTML code downloaded by different clients. A tentative to overcome this variability has been proposed in [Sfakianakis et al., 2011] that uses three tests on content: (i) an MD5 hash is taken for both the retrieved and the Ground Truth version: if they coincide there is no censorship; otherwise the relevant content is extracted for both webpages by means of the *Readability* [6] algorithm and (iii) the result is compared with a fuzzy hashing technique [Kornblum, 2006] to obtain a similarity score to base the decision on.

[4]http://www.monkey.org/~dugsong/fragroute/
[5]http://www.secdev.org/projects/scapy
[6]https://www.readability.com/developers/api

3.1.5 Topology-aware Detection

Some detection techniques focus on the topology of the network when probing for censorship. In fact variations of the ICMP-based `traceroute` technique leverage other kind of probing packets (TCP or UDP) using either varying port numbers (UDP) or different sequence numbers (TCP) to identify the hop that elicited the ICMP `time-exceeded` error. Thus a TCP packet initiating a connection (or carrying payload in an established connection) can be sent with increasing TTL to discover if and where (in terms of path hops) the blocking is enforced.

An example of these techniques can be found in [Xu et al., 2011], where the location of censorship-enforcing boxes is found by exploiting the behavior of the specific censoring system - namely, the Great Firewall of China (GFC). The peculiar behavior in discussion is related to the *statefulness* of the censoring system, as detected by [Clayton et al., 2006] and recalled briefly in the following.

In the GFC not all the packets are inspected by the system, but only the ones occurring in a correctly established TCP connection to a webserver (TCP port 80); in such connections Deep Packet Inspection techniques are used to match character strings (keywords belonging to a blacklist): once a blacklisted string is found, forged RSTs directed to both endpoints shut down the connection, and every further connection between the same endpoints is replied with forged RSTs regardless of the packets content, for a fixed timespan. Thus the censoring system "remembers" the connection and behaves differently according to the kind of communication that happened before, in other words the *state* of the connection is kept, hence the *statefulness* of the system.

The statefulness of the Chinese censoring system has changed over time: Crandall et al. [2007], Polverini and Pottenger [2011], Xu et al. [2011] found it being stateful with no exceptions, as a TCP packet containing an `HTTP GET` request with a blacklisted string would constitute a *trigger* only if sent after a valid TCP connection establishment.

In the characterization of censorship techniques that we have proposed in Section 2 this censoring system adopts network-based censorship summarized in Figure 2.5.

The probing technique in discussion is aimed at detecting and topologically locating devices that enforce keyword-based censorship. It uses a *first-stage trigger* of type `TCP-port-80`, `HTTP-header-keyword` to activate filtering towards a website inside the censored network; after the activation a `traceroute`-like sequence of packets with *trigger* simply *IP-destination,TCP-port-80* is sent until the reception of an `RST` signals the finding of the censoring device. More detail is provided in the algorithm pseudocode in Fig. 3.1.

```
for target in targetlist do
    check HTTP reachability with neutral GET
    if target is reachable then
        try HTTP GET with ‘‘FALUN’’ keyword
        wait 5 seconds
        if no RST is received then
            target is whitelisted
            next for
        else repeat:
            hop=hop+1
            send ACK to target with TTL=hop
            if response is RST then
                censorIP=(last saved IP)
                censordistance=hop
                exit repeat
            elseif response is ICMP Time Exceeded then continue repeat
           end repeat
        end if
    end if
end for
```

Figure 3.1: Topology-aware detection: pseudocode describing the location algorithm for Chinese Great Firewall presented in [Xu et al., 2011] (as interpreted by the writer, from the textual description).

3.1.6 Soft-censorship detection

Detection of soft censorship activities requires the possibility to measure impairments in access to online services and resources. This kind of detection is rooted in techniques from the field of performance measurements and Quality of Service estimation and monitoring.

An example of detection of *throttling* is found in [Anderson, 2013], in which the author applies statistical analysis on measurement data collected by μTorrent clients whose IP addresses are geolocated in Iran. The performed measurements are executions of Network Diagnostic Tool [Carlson, 2003] tests against M-lab servers outside the country under analysis (mostly Greece , U.S.A. and U.K.). Considered performance parameters are Round-Trip Time (minimum and average), Packet Loss, Network-limited Time Ratioand Network Throughput. Measurements are aggregated along three axes: country-level, ASN and network prefixes, control group (defined as “logical, coherent groups of networks and clients based on common characteristics, such as the nature of the end user or performance”, assuming that sets of privileged users - government agencies, banks, commercial customers - are subjected to different policies than the rest). The detection of significant events (suggestive of censorship activities) is based on thresholds (trend-based minimum and maximum bounds) and variance among different aggregates (assuming that natural variance is high, while when external limitations are imposed variance will be low).

3.2 Passive detection methods

3.2.1 Server-based Detection

Server-based Internet Censorship Detection methods belong to the *passive detection* category, taking the evidence on which performing the analysis from traffic traces and application logs. Such data collected at a server is indipendently generated by the customers when accessing the provided online services, thus it is limited in two aspects: (i) the only *target* is the server itself; (ii) the *triggers* are limited to the service protocol. Depending on the service architecture it can constitute a single viewpoint (service hosted on one single host) or multiple viewpoints (for a service hosted on a distributed platform, by means of a Cloud platform or a CDN).

The main server-side method for detection of censorship is applying statistical analysis to the number and origin of clients connecting to the server. This kind of detection relies on the hypothesis that censorship events are country-wise in scope, and requires the possibility of tracking at least the country that is source of the connections, e.g. relying on *IP address - to ISP - to country* mappings, also named *IP Geolocation*[7]. Examples of statistical server-based censorship detection activities are the *Google Transparency Report - Traffic* page[8] and the *TOR metrics portal*[9].

Another example of server-side censorship detection is the case of google analysis on reported malfunctioning of the search engine from mainland China users[10]: the presence of some specific characters in a search query caused the connection to google website to be interrupted for a minute or more, showing "The connection was reset" error. Google response was to warn the user when the "sensitive" keywords (characters possibly contained in simple everyday use words). This practice has been quietly discontinued, as reported by Greatfire [GFC, 2013][11].

[7]An example of service offering this mapping both as for-pay service or in lower-quality unsupported public available version, is given by MaxMind http://dev.maxmind.com/geoip/geolite

[8]http://www.google.com/transparencyreport/traffic

[9]https://metrics.torproject.org

[10]described in a blog post in the official Google blog: http://insidesearch.blogspot.tw/2012/05/better-search-in-mainland-china.html

[11]https://en.greatfire.org/blog/2013/jan/google-bows-down-chinese-government-censorship

Self-exposed server-based ***censorship***

A peculiar case to be considered is a form of server-based *detection* of server-based *censorship*. Explicit requests of blocking target content are issued to Online Service Providers: a notable example is given by Google Transparency Report - Removal Requests [12] that discloses a summary of requests from governments or from copyright owners to block access to specific content. While the actual targets are not disclosed, a categorization of removal requests is done according to the reason and the type of requester and the related statistics are provided. At the time of writing (March 2014) considered reasons are most prominently Defamation, Privacy and Security, and then (each accounting for less than 5% and decreasing): Government Criticism, Impersonation, Adult Content, Hate Speech, Violence, Copyright, National Security, Religious Offense, Trademark, Electoral Law; Other (about 18%). A breakdown is provided according to the origin of the request: "court orders" or "executive, police, etc.", the first type being prominent for category "Defamation". For each country also the extent of non-compliance with the requests is given, along with the reason for not abiding by the requests. An interesting "out-of-band" channel is offered to users willing to notify the unavailability of Google services (and possibly of Internet connection): three phone numbers are provided to leave a voice message that will be automatically tweeted with hashtag indicating which region the calling number is located (when detected).

3.3 Censorship Detection Platforms and Tools

The tools and platforms employed for Internet Censorship Detection or Monitoring are not numerous but quite heterogeneous. A common baseline can be abstracted from their nature of tests: as such the detection can be broken down in the same components and phases described in the previous section. Besides this, they differ in the approaches that have been adopted such as the degree of automation adopted, ranging from testers having to run manual checks [Feng and Guo, 2012] up to virtually unmanned censorship monitoring platforms [Sfakianakis et al., 2011], or between centralized control `Herdict` to completely distributed [Filastò and Appelbaum, 2012].

[12]http://www.google.com/transparencyreport/removals/government

3.3.1 Herdict

The "Herdict" project [Herdict] is a crowd-sourced censorship monitoring project. Its main interface consists of a website allowing users to report about "accessibility"of URLs from within their browser; this way the platform leverages crowdsourcing both for the collection of *targets* of interest for the users, and by having the users to perform an application-level censorship test.

The URLs to be checked are both suggested by a limited number of affiliated organizations and provided by the users themselves. In Figure 3.2 a screenshot [13] is shown of the form for the submission of a test verdict: by clicking either on "accessible" or "inaccessible" the user provides a verdict about the reachability at *application* level. The simple mechanics of this method consists in providing the user some URLs of interest and receive, together with her verdict on accessibility, also the IP address of the user (as her browser is connected to the Herdict website), and a few accessory browser-related data, so that the user location can be inferred through geolocation and the verdict can be aggregated by the *source IP* of the user/probe.

An additional interface is provided by means of a browser module ("plug-in") that: (i) collects and sends to the Herdict platform the URL of each website visited by the user, returning a response about the accessibility of that website according to the Herdict database; (ii) allows the user to quickly report whether the website is accessible or not. The first functionality can be disabled by the user. Moreover a registration form is provided to the user to describe her location (that will be associated with all the user's report). This constitute another form of metadata collection about the condition of the accessibility measurement.

Given the mechanics of this detection method, it can be considered as testing the censorship of the targets at *application level*, as the whole protocol stack must have worked unimpeded in order to provide the final result of the webpage rendering (thus including HTML and script processing). Being more precise, as user judgment is involved in assessing whether the results she sees are to be considered an "access" or not, this should be considered an "user-level" [14] censorship test.

[13] as of March 2014

[14] a humorous expression with some popularity is "at layer 8", with reference to the 7 layers of the ISO/OSI network model [Ross and Kurose, 1999], the upper one being the application protocol (roughly corresponding to the top sublayer of TCP/IP application layer).

This project has both interesting strong points and inherent shortcomings. The strong aspects can be summarized as

low adoption bar: the user doesn't have to install any software besides a JavaScript-enabled [15] browser; moreover no technical skills or specialistic knowledge is required to operate the tests due to their basic nature;

quasi-real-time reports: results of checks from all the users are published in easy to understand maps and graphs, besides downloadable CSV files;

crowd-sourced target lists: the users themselves can submit URLs that are of interest to them and contextually report on their reachability, thus increasing the list of considered URLs (though the lack of selection on this set is an issue).

Weak aspects of this project are intrinsic of its design:

single, basic, subjective test: the project is designed to perform a simple web page *visibility* test, i.e. the user is requested to state if the target URL is visible or not: this on the one side relies on the expectations of the user on the appearance of the webpage and its correct rendering in the browser frame, on the other doesn't give any detail of the reason/technique of blocking;

unselected input: as target URLs are provided by users themselves with no filtering or selection, irrelevant URLs (of no interest for the censors to block) or simply misspelled ones (uselessly crowding the "unreachable" list) can and are introduced; filtering out these cases is not a trivial task;

error or falsification prone output: for an user to report wrong results is as easy as clicking "accessible" instead of "inaccessible", so both mistakes and falsification of results are trivially introduced: this worsens the issue of verifiability of results, common to most if not all the censorship monitoring platforms that rely on user input;

user activity required: the user must actively go to the project website and actively endure the reporting procedure: this implies that there is no way to control the time

[15] [Hoehrmann, 2006]

or the targets that are checked; moreover the users have minimal incentives to perform the checking and to come back and perform checks again besides the obvious ethical motivation; this is partially mitigated by the plug-in browser interface, limiting the user action to the minimum (a click on the reporting button), but still the user intentional activity is required.

3.3.2 CensMon

A complete architecture named *CensMon*, specifically designed for censorship detection, is presented in [Sfakianakis et al., 2011]. As a monitoring architecture, it is designed for continuous and automatic functioning, thus the "needle in a haystack" problem of selecting *targets* worth checking has been addressed by feeding the system with URLs automatically harvested from online sources. An abstracted representation of the *CensMon* architecture is shown in Fig. 3.3.

One significant contribution from [Sfakianakis et al., 2011] is the presentation of the first architecture specifically designed for Censorship Monitoring (opposed to spot detection). The study of this architecture has inspired UBICA for the closed-loop control design, albeit this aspect in *CensMon* was minor and considered at a very basic level.

Another contribution is the practical solution they provide for the "needle in a haystack" problem [Invernizzi et al., 2012] (though they do not elaborate on this aspect, and simply describe the source of information used to gather *inputs* for the detection algorithm). A brief description of such sources is reported in the following.

User Input: collection of URLs provided by the users of the system through the user interface

OpenNet Initiative's Herdict Web [Herdict]: periodical scraping of the online report of tested URLs.

Google Alerts[16]on selected topics: subscribing to the topics internet censorship, net neutrality, freedom of speech and human rights, a daily report by email on the relevant search results on the topics is fetched by IMAP client, processed and fed to the system.

[16]http://www.google.com/alerts

Internet Trends: Google Hot Trends [17] and Twitter [18] are scraped for keywords; these are used to query google and take the top-10 results for each trend. No details are given in [Sfakianakis et al., 2011] on how the keywords are inferred from the trends, we speculate that techniques for keyword extraction like Latent Semantic Analysis [Crandall et al., 2007] can be applied.

Search of ONI categories: the list of categories that the Open Net Initiative [ONI] has defined for censorship targets is considered to extract 10 keywords (news outlets, freedom of speech, entertainment, government, terrorism, porn, gambling, religion, net neutrality and human rights); for each the top-100 results from the Google search engine are considered.

3.3.3 OONI architecture

. The OONI project [OONI] is a censorship detection architecture that has been presented in [Filastò and Appelbaum, 2012] and at the time of writing (March 2014) is in active development. It is a Free Software project, part of the wider [Tor Project] with which it is tightly integrated.

The main component of the architecture (Figure 3.5) is the *ooni-probe*, that can perform several different tests in trying to access either the *target* or one of the back-ends (*oonib* in the figure). The *oonib* component constitute a deployment of servers with the role of helpers for tests that require the control of both sides of the communication. Moreorver they serve as repositories for the report of test results.

The tool is written in Python, benefiting from the high-level libraries available for this language to deal with networking.

Censorship detection tests that are supported by ooniprobe are:

- Content Blocking Tests
  - DNS Consistency
  - HTTP Requests
  - TCP Connect
- Traffic Manipulation Tests
  - HTTP Invalid Request Line
  - DNS Spoof
  - HTTP Header Field Manipulation

[17]https://www.google.com/trends/hottrends
[18]www.twitter.com

  - Traceroute
  - HTTP Host

The first group (Content Blocking) can be performed by the probe used as a standalone, while the second group (Traffic Manipulation) requires the interaction with an helper server.

The Content Blocking tests present two phases: (i) generation of probe traffic; (ii) comparison of outcome against a ground truth.

The ground truth is obtained by means of the application Tor [Dingledine et al., 2004], a transport-layer proxy providing access to an overlay network designed for privacy and used also for censorship circumvention. In fact the Tor application running on the same host of the probe offers to the local network applications a SOCKS 5 [Leech et al., 1996] proxy server that tunnels TCP and UDP traffic in an encrypted circuit used to traverse the *surveillance* device without exposing any *trigger*. Resources fetched through the tunnel are considered as not tampered.

3.3.4 Tools for Censorship Detection

The complexity and specialization of Internet Censorship detection tools can vary. At one end of the spectrum there is direct usage of applications (such as web browser, VPN clients, overlay networks, etc.) to manually check their functioning. At a basic level, each application can be considered an *active* testing tool of the application itself: tried from a probe if it succeeds then is not censored, otherwise *may be* censored or not.

This approach has some evident shortcomings, namely:

- it is not possible or hard to tell intentional disruption of expected behavior from technical issues;
- requires a full working installation for the tested application (limiting the deployment possibilities);
- it is hard to automate;
- the service information report of the application can be insufficient or hard to interpret, not being designed for censorship detection.

In fact one of the considered platforms for censorship detection and reporting, [Herdict], is based on web-browser usage by users actively checking websites reachability. The platform itself is a mean to address the shortcomings related with direct use of browsers as censorship detection tools, adding the missing functionalities that make *Herdict* a platform and not a tool: (i) collection of targets; (ii) multiple viewpoints; (iv) reporting of results.

Compared with detection platforms, tools are more limited in functionality, being focused mainly on the measurement aspect, and lack most if not all of the automation facilities offered by platforms, such as results collection, integration and analysis.

Samizdat

The tool used to gather the data analyzed in [Nabi, 2013] has been made available by the author, and is hosted on GitHub [19].

The tool consists in a Python script that downloads from a blog URL[20] the list of targets to check for censorship then applies a list of tests of type DNS resolution, alternate DNS resolver check, web content access, and a variation on keyword-based web access.

rTurtle

The data backing the analyses performed by *The OpenNet Initiative* [ONI] has been collected by means of an undisclosed client, but a detailed analysis of the dataset has been published in [Gill et al., 2013]. In this paper an algorithm for the detection of censorship based on collected data is reported (see Fig. 3.6): if the test finds evidence of blocking possible motivations are: DNS blocking, IP blocking, No HTTP Reply, RST, Infinite HTTP Redirect, Block page. From the description of the tests it can be inferred that this tool performs HTTP GET requests towards given *targets* and collection of the HTTP replies or application-level logging of errors.

Alkasir

Alkasir is mainly meant as a circumvention tool dedicated to a restricted list of websites. The tool is part of a system comprising a website [alkasir], a client application, and an undisclosed set of proxies that are used to tunnel the user traffic towards blocked websites. The tool is not open source, while it is downloadable for free. The list of URLs to which access is granted is managed on a per-country basis: users are allowed to tunnel their traffic through the alkasir proxies provided that the following conditions are all met:

[19]https://github.com/ZubairNabi/Samizdat

[20]http://propakistani.pk/wp-content/uploads/2010/05/blocked.html

- the user herself or another user from the same country has reported the URL as blocked
- the operators of the alkasir system have policied the URL as complying to the alkasir policies for URL submission[21]
- the alkasir systems validates that the submitted URL is blocked from the user country

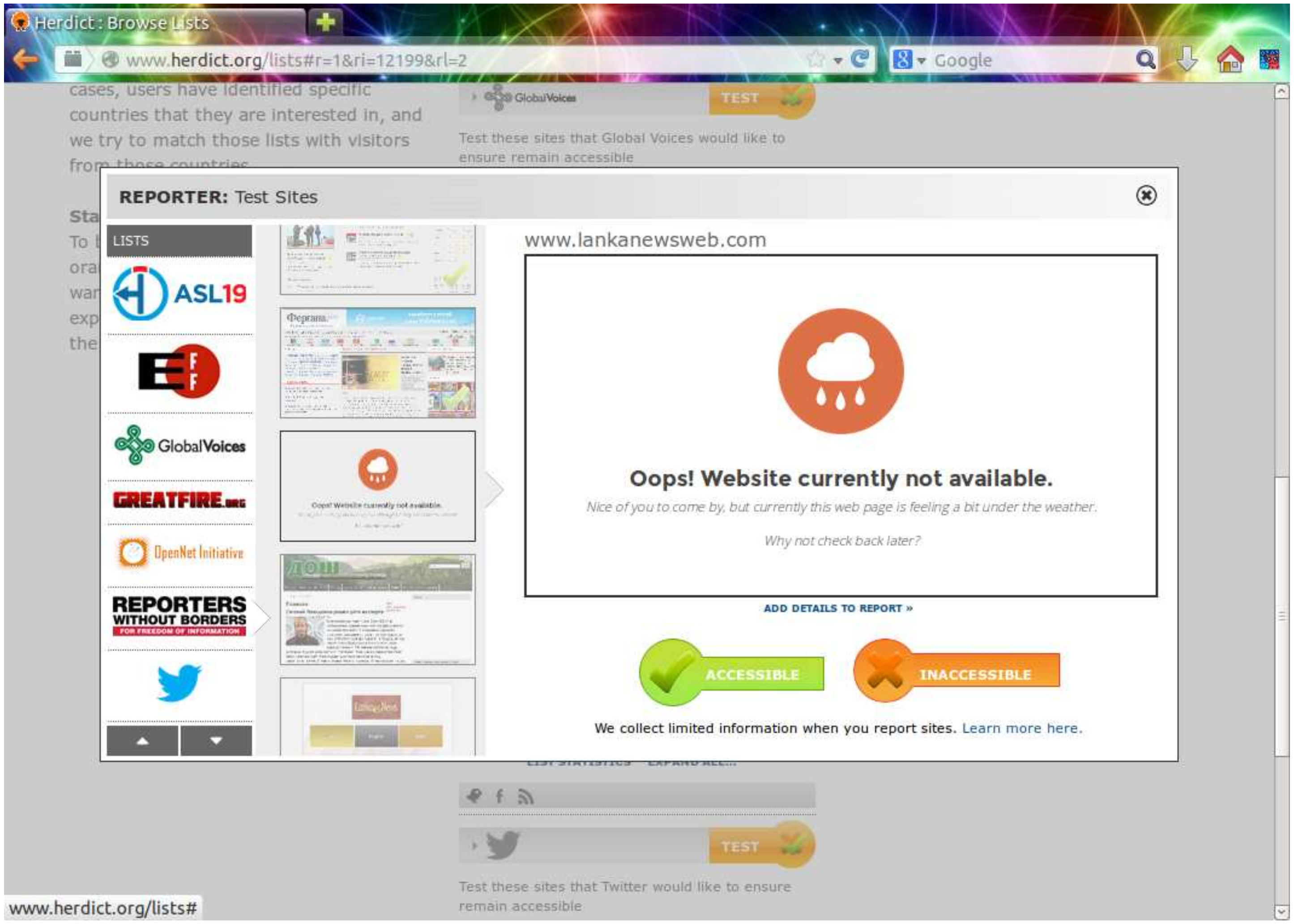
The submission of the URLs is allowed only through the Alkasir client, in the form of a URL list. The user will be notified if and which URLs have been validated as being blocked from her country and compliant with the URL submission policy.

The detection technique of the Alkasir tool is not disclosed. Nevertheless this system has been considered among censorship detection tools as it actually serves as a detection service, by publishing their blocking results.

A report of URLs considered as blocked by the system is given in the form of *google map* embedded in the product website, reachable at the URL https://alkasir.com/map. Web pages reporting per-country lists of blocked URLs are also provided.

The shortcomings of the Alkasir system as a censorship monitoring platform derive mainly from its intended usage as a censorship *circumvention* tool: the analysis and reporting of censorship is intended as a mean to limit the traffic that users can impose to the alkasir servers, and does not have a monitoring objective besides this. The reason, as stated in the website FAQs[22], is that having limited resources (bandwidth, proxy servers), the restriction of circumvention only to selected categories of resources that actually need it is necessary to provide the service to a larger user base.

[21]https://alkasir.com/policy
[22]https://alkasir.com/faq


Herdict : Browse Lists

www.herdict.org/lists#r=1&ri=12199&rl=2

cases, users have identified specific countries that they are interested in, and we try to match those lists with visitors from those countries

Test these sites that Global Voices would like to ensure remain accessible

REPORTER: Test Sites

www.lankanewsweb.com

Oops! Website currently not available.

Nice of you to come by, but currently this web page is feeling a bit under the weather.

Why not check back later?

We collect limited information when you report sites. Learn more here.

Test these sites that Twitter would like to ensure remain accessible

www.herdict.org/lists#

Figure 3.2: The *Herdict* platform: a screenshot of the submission form. On the right a frame showing the *target* as retrieved by the browser (in an i-frame), and below the two buttons to report: "accessible" or "inaccessible". On the left a menu of URL lists proposed by affiliated associations is shown.

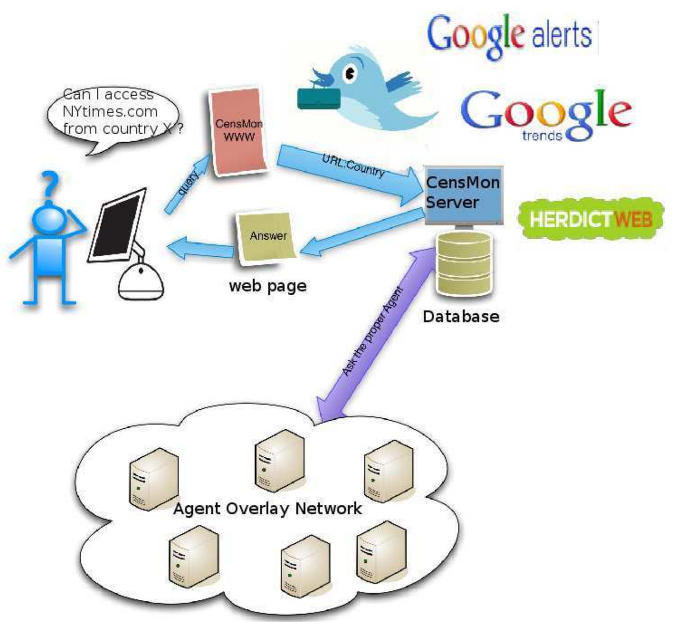


Figure 3.3: The CensMon architecture (from [Sfakianakis et al., 2011] Fig. 1).

INPUT: URL

1. distribution of URL to probes
2. DNS resolve → output: noerror, noxdomain, timeout, connrefused
3. TCP connect to resolved IP, port 80
4. URL filtering: keyword=URL postponed to WebHelper URL (expecting 404not found)
5. HTTP access to URL, get status code, HTML (if redirection: last URL and IP visited)

Server-side analysis:

6. DNS tampering:
   - 6.1. (whois from server)
   - 6.2. coherence of DNS resolution from different probes
7. HTML content check:
   - 7.1. md5sum for coherence among probes
   - 7.2. arc90 readability extraction → fuzzy hashing → comparison among probes
8. inaccessibility event → probe that got it keeps monitoring the URL

Figure 3.4: *CensMon* [Sfakianakis et al., 2011] detection procedure.

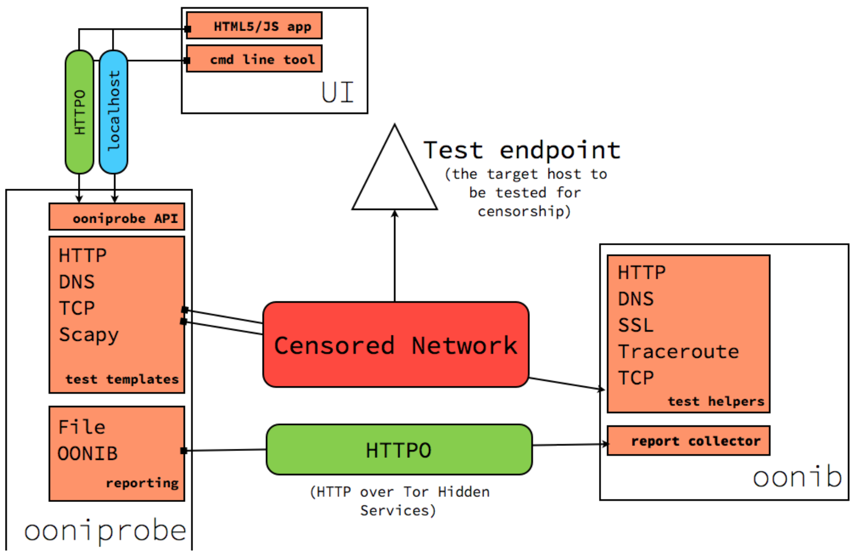


Figure 3.5: OONI architecture, from [Filastò and Appelbaum, 2012]

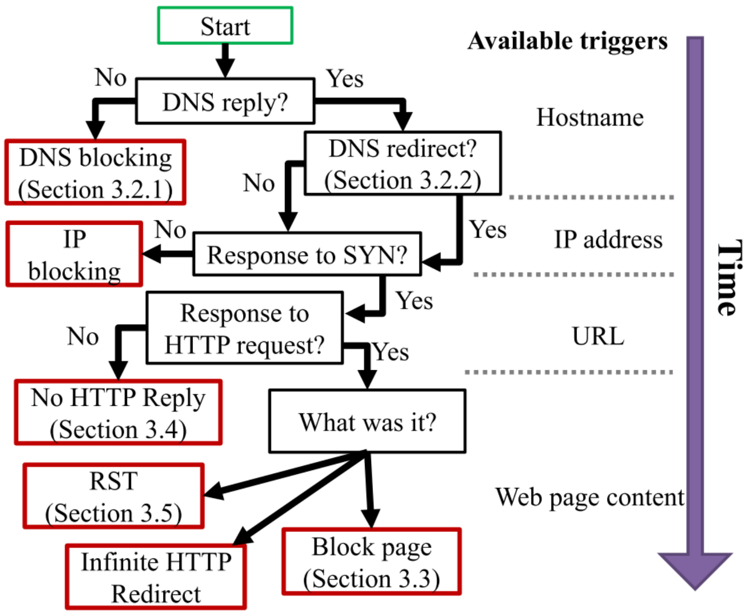


Figure 3.6: The Detection Algorithm on ONI data (from [Gill et al., 2013] Fig. 2).

Chapter 4

The UBICA platform

4.1 UBICA design properties

From the analysis of the available platforms it has emerged that each of them presents a number of shortcomings, and none fully embodies a platform performing the complete cycle of censorship monitoring activities from the collection of targets to the publication of responses. UBICA has thus been designed to be a *sustainable* platform for censorship monitoring. By sustainable we mean that it has to: (i) foster user adoption (being directly useful to the user); (ii) generate the less probe traffic for both probes and targeted services. Indeed disregarding user participation and generated network load is a notable deficiency of known censorship detection and monitoring platforms, thus these goals constitute an advance on the state of the art with the intent of providing a valuable contribution to the human rights watch activities. The addition of sustainability to the base requirements for a censorship monitoring platform leads to the list of desired properties and features reported in Fig. 4.1.

- incentives to user adoption;
- user privacy consciousness;
- support for a diverse probe set (multiple vantage points and sources of information);
- platform-wide test management:
  - probes grouping;
  - per-group test upgrade;
  - per-group test activation;
  - per-group test parameters tailoring;
- an expandable censorship detection test set
- a closed-loop control system
- consider resource as *un*censored unless censorship evidence
- a reporting system with different interfaces:
  - for the users running the probes, with user-centered detailed data
  - for the researchers running the platform
  - for the public, with aggregate data graphs and maps
  - for the public, in open data formats, with raw anonymized test results

Figure 4.1: Design properties and features of the UBICA platform.

Other aspects have been set aside from the primary goals, but have been considered in order to be supported:

- resistance to censorship of the platform itself
- distribution of probes in censoring countries

According to Burnett and Feamster [2013], the following principles should be adopted to help guarantee robust and accurate monitoring of censorship:

1 Correlate independent data sources.

2 Separate measurement from analysis.

3 Separate information producers from information consumers.

4 Employ “dual-use” scenarios wherever possible, to improve robustness.

5 Adopt existing techniques to provide robust channels for measuring, reporting, and accessing reports.

6 Heed and adapt to continually changing political and technical threats.

These principles while have not been a blueprint for the design of the UBICA platform still apply to its prototypical implementation: specifically principle 1 is fulfilled in using sources for the collection of targets list that include also a reachability information associated with targets; moreover the adoption of client types whose behavior can be trusted (PlanetLab nodes and BISMark routers) enhances the diversity of information sources; principle 2 is strictly respected, as there is an explicit separation between the evidence collection phase and the censorship analysis and report; the same is valid for principle 3, embodied in the different interfaces available for reporting results of evidence collection and analysis; principle 4 is verified in the UBICA platform by leveraging the BISMark and PlanetLab distributed platforms (designed and used for network experiments and network performance measurements) as hosts for UBICA probes; finally the automatic update functionality for both tests and clients helps in fulfilling principle 6 and in adopting the means considered in principle 5;

The main objective of the UBICA platform is to provide the users with a *censorship monitoring* system that presents both a report on world-wide Internet censorship status and a quick view of censorship from her point of view (user-based). Three categories of users are considered: *researchers*, that run and operate the platform backend infrastructure; the *public users*, provided with coarse-grained (country-level) reports; and *volunteer users*, hosting UBICA clients and allowed the access to finer-grained reporting. In order to gather data, the platform leverages a distributed deployment of probes belonging to different kinds (router-based, headless client, gui-client) that are orchestrated by a central management server. The platform is designed to provide: (i) dynamically updated censorship tests; (ii) dynamically updated targets to be verified; (iii) support for different types of probing clients; (iv) client-tailored tests; The execution of the tests is managed by a scheduler that has the objectives of maximizing the coverage of targets while keeping a low volume of probing traffic from each probe and towards each target.

With respect to the state of art, the goals and the properties of the platform are novel, as no published censorship detection platform present all of them. Main novelties reside in: (i) the continuous analysis of evidences; (ii) the interface with the user, providing an incentive for the expansion and durability of the probe set and user base; (iii) the dynamic scheduling of the evidence collection tests.

4.1.1 Main differences with state of art

The platform most similar in goals and properties to the UBICA platform is *CensMon* [Sfakianakis et al., 2011]; with regards to it, UBICA presents significant differences, related with the following properties of UBICA missing from *CensMon*:

- a diverse probe set (expanding on the one available to CensMon, Planetlab nodes);
- complex test management (upgrade, selective test activation, per-probe test parameters tailoring, aggregates of probe - campaigns);
- incentives to user adoption (an aspect structurally missing from CensMon as its probes are not intended to be run on users premises);
- a reporting system, both for users and for the public (it can be imagined that *CensMon* has an interface for queries and reports, even if it's not described in the paper).

Due to the its interface to humans, UBICA includes components that are missing from CensMon, namely a Graphical User Interface for the probe, and a Reporting Portal with three interfaces according to the role of the user (public, volunteer, and researcher).

Another comparable architecture is OONI, with which the main architectural differences of UBICA are

- centralized control
- test distribution and execution management
- server-based censorship analysis on the whole DB
- closed control loop
- not coupled with Tor (but possible integration)
- incentives for users adoption through probe GUI

An implementation difference between UBICA and the OONI architecture lies in the probe: while *ooniprobe* is written in Python leveraging a powerful but voluminous library (Twisted Matrix [1]) as a core of its event-driven networking engine, UBICA deploys different kinds of probes that can be as lightweight as an ASH [2] script calling half dozen ubiquitous UNIX utilities.

Some differences derive from explicit design choices: OONI aims at being highly distributed with no control from a central entity. Moreover OONI's goal is to provide an open

[1]twistedmatrix.com

[2]Almquist SHell, a basic POSIX shell implemented in `busybox` and `dash`.

platform for generic censorship tests: this led to the choice of Python and the Twisted Matrix framework to leverage an high-level event-driven programming platform; on the other side this has raised the requirements for running the software probe, making it not suitable for embedded systems such as home gateway routers. Another goal difference is that UBICA is a **monitoring** platform (much more like *CensMon*) whereas OONI, albeit being generic and extensible, is a **detection** platform, thus neglecting or demanding to other external systems aspects such as target collection, scheduling, and data analysis and reporting.

4.2 Architecture design and implementation

The UBICA platform is designed having in mind different kinds of stakeholders: the common *citizen* that wants to be informed about blocked resources and service from her point of access to the Internet; the *journalist* or activist that looks for data about the freedom of information in her country or other countries; the *researcher* in the field of networking and social sciences, interested in worldwide evolution of Internet censorship. The platform is based on the participation of *volunteers* that provide a limited part of their bandwidth and CPU resources (as well as a few MB of disk space for temporary files) running an UBICA probe, to perform the gathering of information about censorship as experienced by their connection points. The *volunteer* may be interested in remaining anonymous, and in order to protect her privacy the platform avoids using the probe IP address in data processing and reporting, nor employs any identification data. Another kind of stakeholder, though not explicitly considered in the design as an independent actor, can take advantage of UBICA measurement: the provider of services that checks her information from service logs against measurement towards her service from different vantage points. This kind of user has no specific treatment and its usage of the platform can be absorbed in the “journalist” kind of user. Another category of stakeholders is represented by the censors themselves (governs, institutions, employers); these in principle could be uncaring of publication of information about censorship as long as it remains effective, but could as well be willing to limit the knowledge about the specific targets and techniques, or willing to hide the presence of censorship altogether. The UBICA platform is not designed to circumvent censorship of its services, but can adopt techniques of passive censorship detection in case this would happen (Section 3.2).

Actor	Detail	Suggestion
Volunteer - Router	connection	moderated
Volunteer - Client	connection	automatic
Volunteer - Standalone	Country	none
Non-Volunteer	Country (if available)	none

Table 4.1: Actors and Use Cases: the *detail* of the reports and the possibility of *suggesting* new resources vary according to the Actor.

These stakeholders interact in different ways with the platform, mainly differentiated by the fact that there is a *volunteer* running a probe from within the same access network (defined as a range of IP addresses assigned to an access provider as returned by WHOIS database query). This is summarized in Table 4.1.

The use cases describing how these actors interact with the platform are listed in the following section.

4.2.1 Use cases

The common citizen

The generic *user* can access UBICA by means of the main web portal, that shows information about the recent state of censorship as detected by the UBICA platform. The report is specific for a country (preferentially detected by geolocating the user IP or provided explicitly by the user).

The different actors that interact with the UBICA platform are represented in the Use Case Diagram in Figure 4.2.

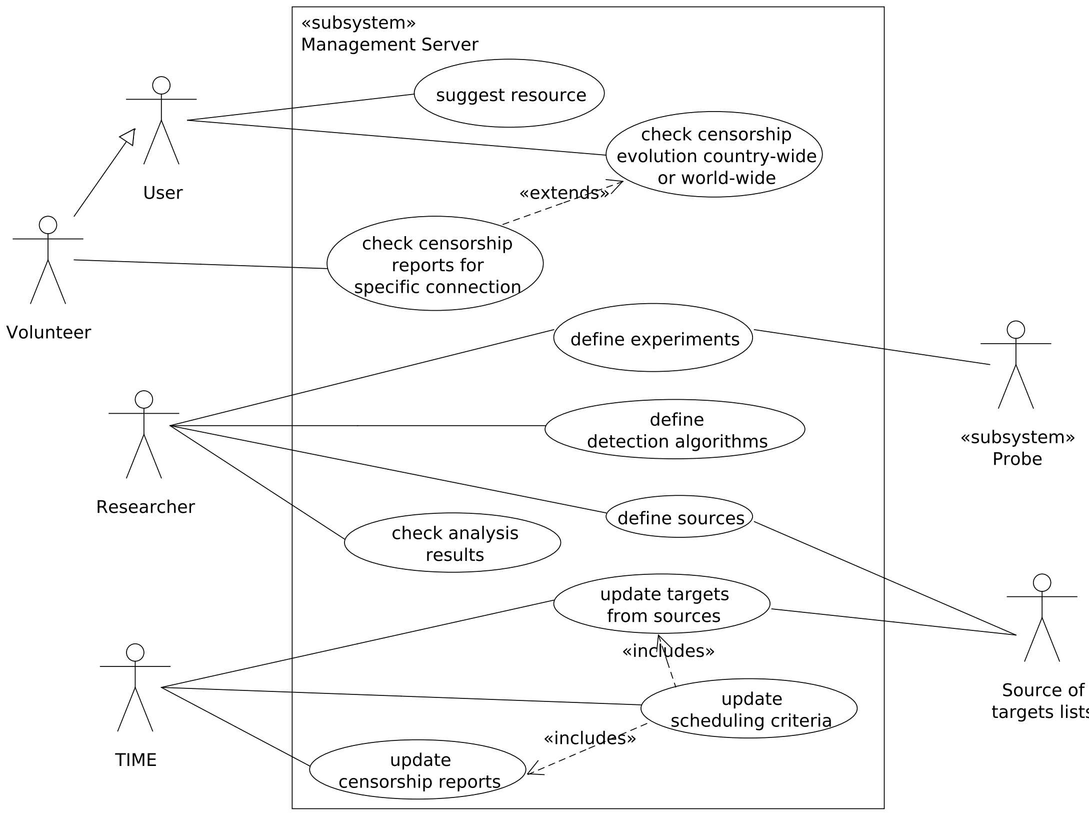

«subsystem»
Management Server
suggest resource
check censorship evolution country-wide or world-wide
«extends»
check censorship reports for specific connection
User
Volunteer
define experiments
define detection algorithms
define sources
check analysis results
update targets from sources
«includes»
update scheduling criteria
«includes»
update censorship reports
Researcher
TIME
«subsystem»
Probe
Source of targets lists

Figure 4.2: The UBICA Use Case Diagram.

Use Cases involving the Probe subsystem of the UBICA platform are represented in the Use Case Diagram in Figure 4.3.

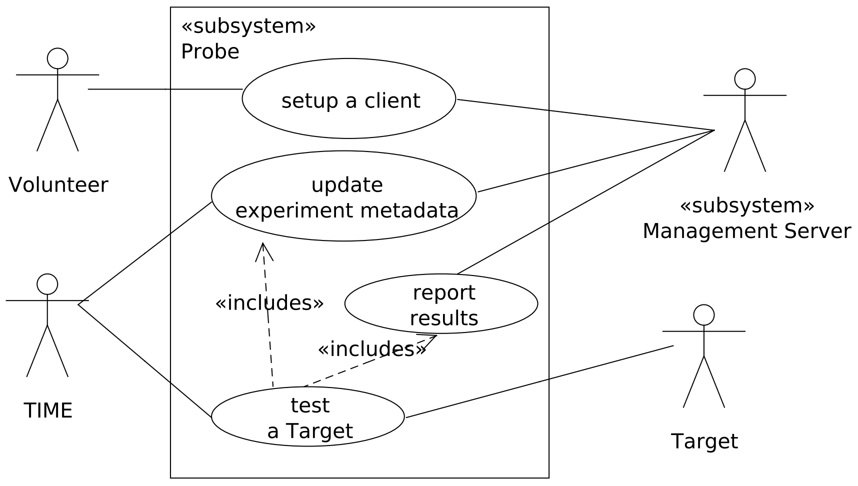


Figure 4.3: The Probe subsystem: Use Case Diagram

A schematic diagram of the UBICA architecture in shown in Figure 4.4.

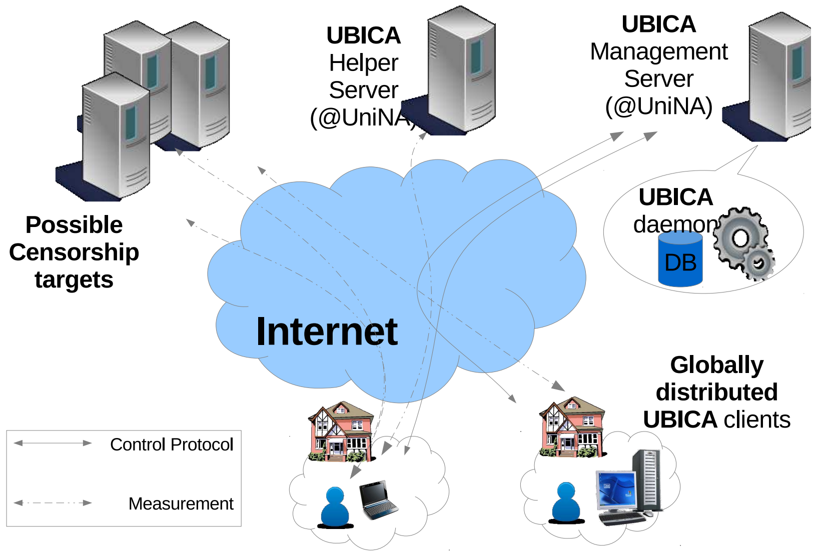

UBICA Helper Server (@UniNA)
UBICA Management Server (@UniNA)
UBICA daemon
DB
Possible Censorship targets
Internet
Globally distributed UBICA clients
Control Protocol
Measurement

Figure 4.4: The UBICA architecture diagram.

4.2.2 Control loop

The UBICA management server operates following a control cycle that comprises several phases, namely:

1. Collection of Targets
2. Scheduling of evidence collection
3. Evidence collection
4. Censorship Tests
5. Evidence reporting and data export
6. Update Targets and Scheduling criteria

During these phases, in order to perform its operations, the management server interacts with external components such as sources of information, the platform probes, and clients of the portal services. A visual representation of the control cycle that is executed by the management server that governs the UBICA platform, and its relations with external elements, is shown in Figure 4.5.

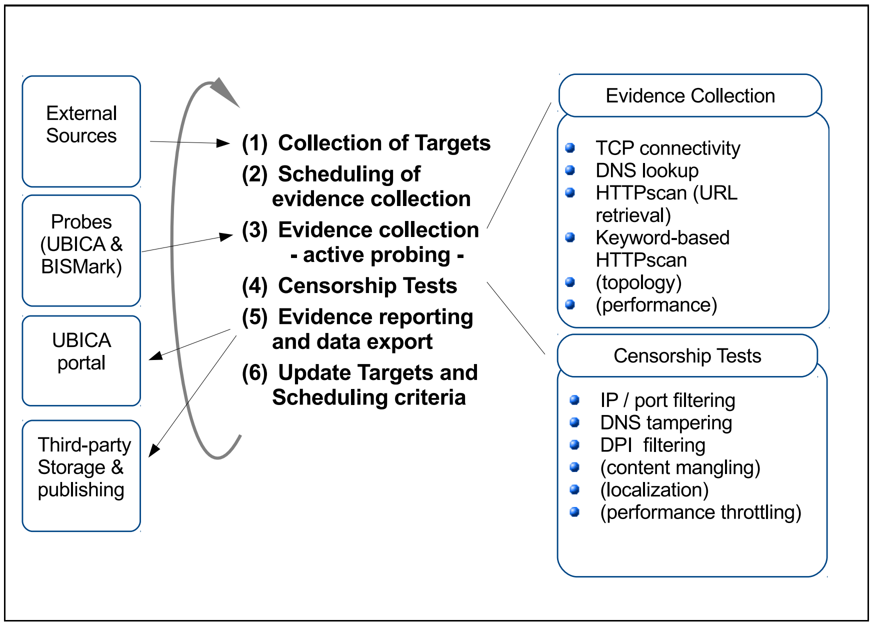


Figure 4.5: The UBICA architecture: the control cycle.

4.2.3 Probe types

Router-based probes

The candidate has designed, implemented and deployed a version of the probe software that can be run in the customized OpenWRT environment provided by the *BISMark* platform. Specific requirements for this kind of probes are related with the limited resources available for the device, above all the storage space and the RAM memory space: the model adopted for the probes is equipped with 8MB Flash memory card and 64MB RAM partially used as RAM disk to accommodate temporary data.

These conditions led to the following design choices:

- use of standard UNIX utilities whenever available (most already included in the adopted OS distribution;
- composition of several tools by means of Ash scripting;
- limited processing on the probe.

The embedded probes share most of the evidence collection code with the other UBICA probes, the main differences being related to the scheduling and the deployment procedure.

Being developed as a BISMark component, the probe leverages the distribution platform for the BISMark experiments and thus is provided as an OpenWRT package[3]. The package comprises a `Makefile` taking care of moving the files in their place in the filesystem and setting up the scheduling.

As regards scheduling, the heartbeat connection is tried in cycles spaced by 6 hours, each cycle consisting in 6 executions spaced by 10 minutes; in `crontab` notation:

```
*/10 3,9,15,21 * * * /tmp/usr/bin/censorship-testing
```

The actual execution of evidence gathering procedures is always controlled by means of the *management server*, and the aforementioned scheduling puts an upper bound to the frequency of target distribution and testing rates.

The inclusion in the BISMark platform implies that other active probing experiments can be running on the same device: in order to avoid mutual interference, a dedicated mutual exclusion mechanism is introduced for this version of the probe, as can be seen in the excerpt of listing in Fig. 4.6.

[3] Publicly available at https://github.com/sburnett/bismark-packages/tree/f383d68fdee2c5fbd027114c0cf355dc79e83f5a/utils/pakistan-censorship-testing

```
tries=1
while [ $tries -lt $maxlocktries ]; do
  date >&2
  echo "expiring previous active locks ..." >&2
  expire_active_measurements_lock ; # just in case
  echo "try to acquire active lock ..." >&2
  if acquire_active_measurements_lock ; then
    echo "active lock acquired ..." >&2
    break
  else
    echo "FAILING to acquire active lock ..." >&2
    echo "waiting $waitlock seconds" >&2
    tries=$(( tries + 1 ))
    sleep $waitlock
  fi
done
if [ $tries -eq $maxlocktries ]; then
  echo "WARNING could not acquire Active lock in $maxlocktries tries ( $(( 
      waitlock * maxlocktries )) s)." >&2
  echo "ABORTING test." >&2
  exit 42
fi
```

...

```
# end of critical section. after this just an ftp upload is performed.
echo "releasing active lock ..." >&2
if release_active_measurements_lock; then
  echo "lock released." >&2
else
  echo "ERROR $? releasing lock" >&2
  # weird. let's pretend it never happened.
fi
```

Figure 4.6: Listing of router-based client: code implementing the Mutual Exclusion for active probings on BISMark platform, enclosing the critical section where active measurements are performed.

This implies that there is a legitimate cause for the lack of reports from BISMark probes that have successfully checked with the Management Server and have been assigned a list of evidence gathering tests: it can happen that other active experiments are being run on the probe, thus the execution of the test has been delayed until aborting. As it can be seen in the listing in Fig. 4.6, the acquisition of the mutex lock is tried up to $*maxlocktries* times (default 5), each time waiting for $*waitlock* seconds (default 10) after a failure.

Other minor differences lie in the absence of failure report regarding test running as errors are captured by the script itself and appended to the testing report: in case the

BISMark platform fails to run the script, no reports are uploaded at all, and if any partial data collection has been performed it is cleaned up to prevent storage exhaustion.

A clue of this event can be found in missed reports for probes that did regularly contact the management server with the heartbeat connection and were scheduled for testing at least one target.

UBICA client

The probe version that leverages best the features of the UBICA platform is dubbed ”UBICA client” and is distributed as a binary developed for Linux (tested on Ubuntu versions), Windows and Mac OS X operating systems. It is based on the client of the HoBBIT platform [de Donato et al., 2014], designed for network performance measurements. The specific features provided by this kind of probe are related with a tighter integration with the Management Server and a richer interaction with the user, plus management-friendly properties and can be summarized as:

- a Graphical User Interface (GUI)
- automatic update of experiment definition
- automatic update of the client code
- detection of mobility (and user interaction)
- detailed debug report for measurement failures

With respect to the HoBBIT client, the main differences lay in the initial registration phase, and in the reporting of the experiment results. In the registration phase (activated at the first usage of the client and checked periodically) a number of data about the user location and connection contract are asked to the user by means of a graphical form; when these are communicated to the Management Server, an identification code is returned to associate the client with a class of vantage points (depending on the access network and an user-supplied classification of her connection). In order to better protect the privacy of the UBICA user, with the side benefit to improve the usability of the application, this registration phase is limited to ask the “location type” of the connection by selecting it from a list, while the access network is derived server-side by storing the range of IP addresses the client address is assigned from. The reporting of the experiment results is *not* performed, as in the HoBBIT client, by means of a direct XML interaction with the Management Server, but instead through the upload of a cumulative, self-coherent archive including the input parameters and the resulting data from the experiment. This is done both to decouple the request of experiment metadata from the execution of the experiment and the upload of the results, and to more easily support the heterogeneous set of probes that can participate in the collection of evidence data.

The overall control cycle of the UBICA client is represented as Activity Diagram in Figure 4.7. This control flow is repeated endlessly during the execution of the application unless the user explicitly closes it.

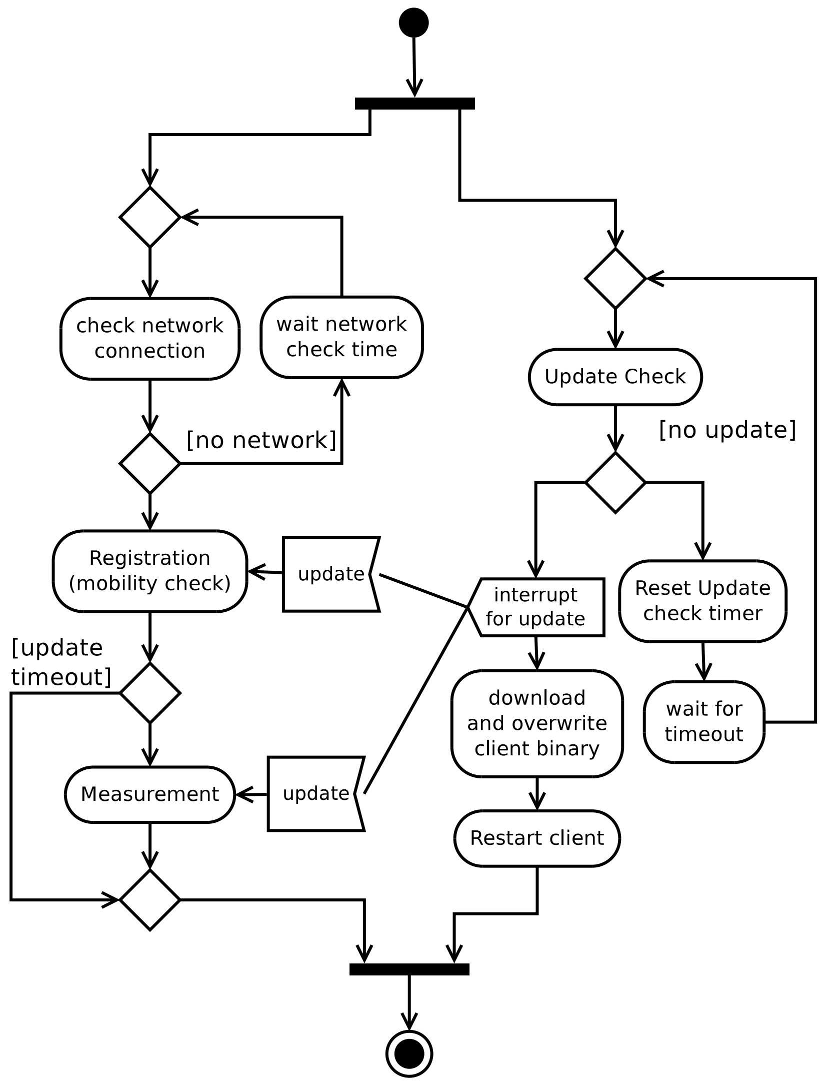


Figure 4.7: UBICA client: Activity Diagram - overall control cycle.

The main activities performed during the control cycle are: (i) check for presence of a newer version of the client and if available update it (*Update* activity); (ii) check of user network mobility and registration of the new measurement environment (*Registration* activity) and (iii) the management of measurements (*Measurement* activity). The *Update* activity is subject to a timeout then takes precedence over the other activities. This is a cautionary approach: in case a bug affecting the behavior of the old client or the quality of the measurements has been fixed, the old client is dismissed as soon as possible in favor of the updated one. The drawback of this approach is that all the updates are considered of maximum priority. Future enhancement could implement two categories of updates: preemptive and non-preemptive ones, the former to be reserved for important fixes while the latter can be deferred after the completion of measurements or even made optional according to user choice.

In the following each of the three main activities are described in further detail.

Automatic client Update The first phase of the UBICA client control flow is the check for newer versions of the binary for its same platform, and if present download it, overwrite the binary and restart itself. This function is activated either when the client is run, or on a periodic timing ($4h$ maximum). In case an error occurs (either for lack of connectivity to the server, or a server error), the client recovers nicely by skipping the update, that will be attempted again after the update period times out. After the update check, the presence of a working network connection is verified again, and if available the control flow continues with mobility detection / registration.

Registration - Mobility detection The censorship monitoring performed by the UBICA platform leverages the knowledge of the network the client measures from, with the granularity of an address range (e.g. `192.168.1.0-192.168.1.255`). In case of a nomadic user (changing his connection network) the platform detects the potential change of network and optionally prompts the user with a question about the “location type” he is at, then obtains from the Management Server a connection ID that will be used from this time on to identify the type of client.

The control flow of the registration phase is shown as an Activity Diagram in Figure 4.8. The activities are positioned on different lanes according to the object that is responsible for it, with the exception of the rightmost lane that represents the Management Server. Interactions between the client and the Management Server are performed by means of synchronous messages exchange through the network.

The detection of mobility (i.e. the user changing access network) is based on the detection of the MAC address of the gateway router: a change to this layer-2 address is considered as a possible change of network. In order to detect the change of the MAC address without revealing information to the platform, a hash value derived from the MAC address is stored along with the "location type" provided by the user, in a file in the installation directory of the client. This (key,value) local database is queried each time a connection check is performed: in case a match is found the corresponding location is considered and the user is not bothered with questions, when no match is found the nature of the new location is asked to the user, the database is updated and the updated connection ID is requested to the server.

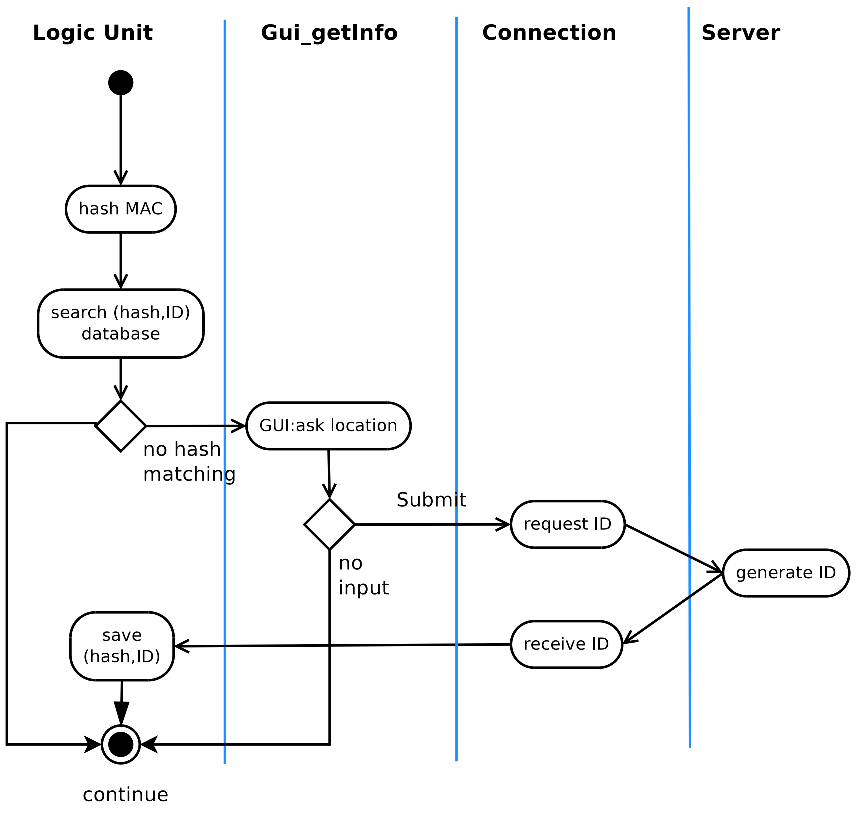


Figure 4.8: UBICA client: control flow of the *Registration* activity.

Measurement The *Measurement* activity constitutes the part of control flow that takes care of checking with the Management Server the list of measurements to be performed, update the measurement definitions if necessary, update the measurement parameters if necessary, perform the measurements and report back to the Management Server the results. The measurement activity of the UBICA client is represented in Figure 4.9 and described hereafter.

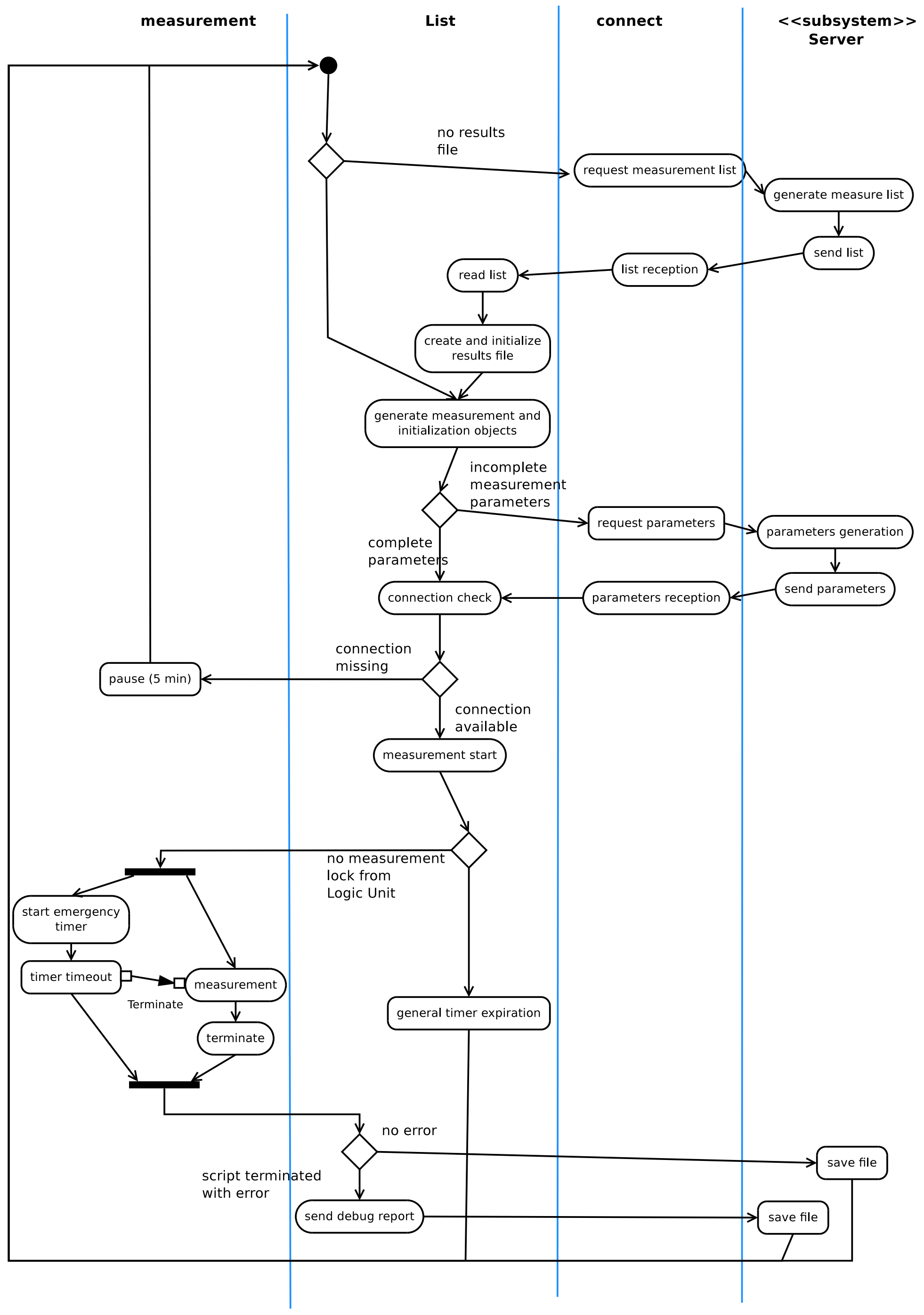


Figure 4.9: UBICA client: control flow of *Measurement* activity

The Activity Diagram is divided in lanes according to the object in charge of the execution of the tasks, except for the rightmost lane that represents the Management Server. Interactions between the client and the Management Server are performed by means of messages exchange through the network.

In the beginning the presence of an up-to-date list of measurements to be performed is checked, and if missing it is asked to the Management Server. Each measurement in the list instantiates one object of class *measurement*, that will be in charge of the actual execution of the experiment, that is defined by a `bash` script. In this phase experiment-specific dynamic parameters can be requested to the Management Server, completing the setup. Before the start of measurements, the presence of connectivity to the Internet is checked: if the test fails then the execution of experiments is postponed for a given time (5 minutes), then the cycle is restarted, skipping the request of the list of measurement to the Management Server. If the connection check passes, the measurements are started and optionally repeated according to the order and the repetition frequency specified in the measurement list. For each measurement an expected duration is specified in the experiment parameters: this sets a countdown from the beginning of the experiment that on expiration causes an abrupt termination of the measurement. This is motivated by the exceptional nature of a duration exceeding the expected time that is a symptom of a malfunction and can impair the execution of the other measurements and ultimately the functionality of the application. The causes for an abrupt interruption of a measurement are:

- an error occurred during the experiment script execution
- the timeout for the specific measurement has expired
- the user has requested the interruption from the GUI interface

If the experiment terminates successfully, a report file is uploaded to the Management Server. If an error occurred during the execution of the experiment, a debug report is uploaded instead.

4.2.4 Target collection

Internet Censorship detection requires the fundamental process of collecting sets of potential targets to be checked. These sets are needed both when applying active censorship detection techniques, and when applying passive ones: in the first case the target set has to be converted in suitable triggers (URLs, domains, IPs or keywords) that are likely to elicit censorship reaction, in the second case the set will limit the scope of data collection to make the scraping, processing and storage tractable.

Targets of censorship can be grouped, from a user's viewpoint, in two broad categories: *resources* and *services*. A *resource* is a content available on the web (a text, an image, or other kinds of documents) and is well represented by an URL. A *service* is a network application, including -but not limited to- web applications.

Services

A service (or network application) can be characterized by the application protocol it adopts and by the end points of the communication (the servers); this means that the censor has multiple possibilities to impair an application, i.e. there are many different *triggers* for the censor to exploit.

Testing the censorship of a service is equivalent, in the UBICA model, to test the reachability of the list of host servers associated with the application (as *test targets* using the related potential *triggers*. This equivalence is built in three phases of the UBICA processing cycle: *scheduling of experiments*, *censorship analysis* and *evidence reporting*. The tests will be scheduled to be run in a single experiment, in order to have a snapshot measurement for the application.

The detection of censorship for targets of type *service* is performed in two stages:

1. the same algorithms used for *resource* censorship detection are applied on a subset of the available tests, selected by choosing the specific *service* in the many-to-many relationship *service-test*; the outcome of this phase is a list of tuples (resource, test, confidence %);
2. a weighted average is taken of the list of tuples aggregated by *test*, producing a single output value (service, verdict);

Automatic collection

The collection of targets is performed periodically by the management server. A list of *sources* is part of the Knowledge Base, each entry presenting the fields:

- last check timestamp
- check interval
- retrieval method

where the timestamp and the interval are controlled and if enough time has passed since last check a new retrieval is done.

The *retrieval method* is a pointer to a script that is in charge of downloading, parsing and inserting in the database the updated *targets*. It can be as simple as an URL pointing to an API, when the source of the target lists provides one, or a web scraping script that parses the HTML in order to extract URLs.

Examples of sources periodically checked by the UBICA platform for targets collection are [Herdict], *The Great Fireall of China*, as well directories of open Web Proxies.

Future development can include automatic keyword collection mechanism such as the one presented in [Crandall et al., 2007], or list of URLs from search engines and Social Networks such as [Sfakianakis et al., 2011].

4.3 Target and Scheduling update

A target (e.g. of the type resource, identified by an URL) is associated with many entities (e.g. multiple IP addresses of the webserver) and multiple triggers: destination socket, protocol information, payload strings. Its essential characteristic, though, is its potential of being censored in some country, with some technique. This nature is best captured by the *cycle of life* of a target in the UBICA platform, that is represented as a Finite State Machine in Figure 4.10.

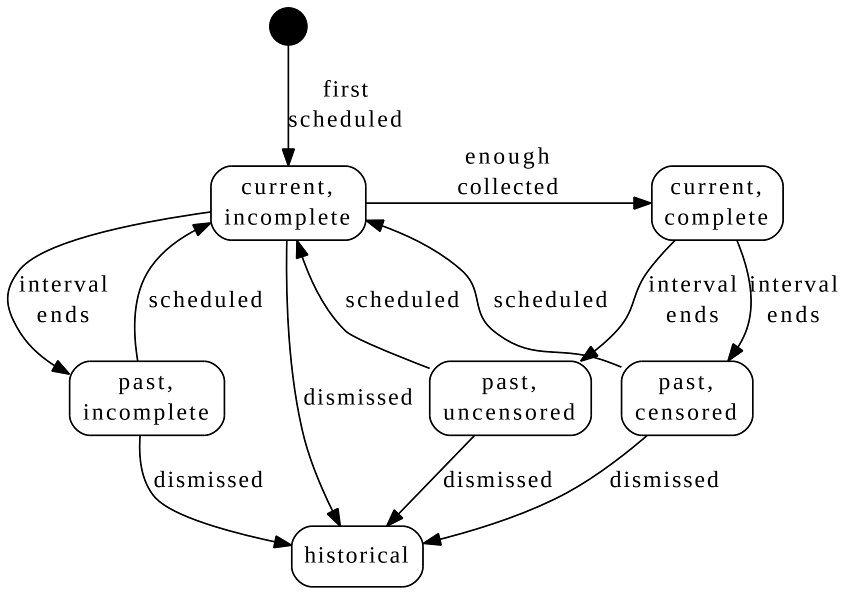


Figure 4.10: Cycle of life of a *target* - Finite State Machine

The states that describe a target are:

new the target is known to the platform, but has never been tested

past,incomplete the target has been tested in the past, but not enough information was collected to issue a verdict (assumed uncensored)

past,uncensored the target has been tested in the past, and found *not censored*

past,censored the target has been tested in the past, and found *censored*

current,incomplete the target is being tested in the current measurement interval, and not enough information has been collected yet to issue a verdict

current,completed the target is being tested in the current measurement interval, and enough information has been collected to evaluate it

historical the target is not available for scheduling anymore

4.3.1 Scheduling of evidence collection

The centralized control paradigm adopted in the UBICA platform allows to coordinate measurements across the available probes.

A probe notifies its availability to perform measurements by contacting periodically the Management Server, asking for a list of tasks to perform. The Management Server, according to the *viewpoint* of the contacting probe (the network the probe connects from) selects a list of *targets* that still haven't completed a number of measurements from the given *viewpoint* (thus being in state *incomplete*, Figure 4.10). If no *targets* are in the *incomplete* state, a random selection of targets that in the previous analysis interval were classified as *censored* from the given *viewpoint* are chosen.

4.3.2 Evidence collection

The collection of evidence of censorship is performed through *active measurements* from the probes, i.e. as a consequence of a scheduled experiment, a probe generates network traffic towards the target hosts purposefully for triggering possible censorship, and records the response it receives. The kind of traffic that is generated is flexible thanks to the architecture design: it can be both the result of execution of standard implementation of network utilities, such as `nslookup`, `netcat`, `curl`, `traceroute`, or -if necessary- an ad-hoc binary can be deployed.

For each experiment execution a given number of target hosts are considered, depending also on the client type.The total experiment duration is enforced by a timeout that terminates all running measurements. This has been devised as a safeguard against measurements that fail to return either with a result or with an error, and can possibly remain running wasting memory and CPU resources. The measurement from each probe towards the different target hosts are started at different times, uniformly distributed in an interval of 40 seconds, across a total experiment duration of 220 seconds. These values have been chosen heuristically to reduce the total probing time (thus affecting the less is possible the user experience) and still allowing for a fair number of target hosts to be checked.

The different types of measurements that have been performed are listed in the following.

DNS resolution

Happening first in the communication reference model as presented in 2.2.1, DNS resolution affects the remaining phases thus has specific importance and informative power. To collect clues about this phase a name resolution is elicited: given a Fully Qualified Domain Name, a DNS request of `type A` is issued from the probe towards its default resolver. The tool used to issue the request is `nslookup` (from the `bind-utils` package for the Linux-based platforms, and the default implementation on Windows systems [4]). A typical response is printed in Fig. 4.11: it can be noted that it reports the IP address of the server replying the query, as well as a list of name resolutions.

```
Server: 130.194.1.99
Address: 130.194.1.99#53

Non-authoritative answer:
Name: 2photo.ru
Address: 217.65.3.83
Name: 2photo.ru
Address: 217.65.3.84
Name: 2photo.ru
Address: 217.65.3.85
Name: 2photo.ru
Address: 217.65.3.82
```

Figure 4.11: Example of `nslookup` output.

[4]we decide to use default implementations where available in order to emulate more closely the experience an user of the system would have.

DNS service provider	Resolver IP address
Google	8.8.8.8
Comodo	8.26.56.26
OpenDNS	208.67.222.222
Level3	209.244.0.3
Norton	198.153.192.40

Table 4.2: Open DNS resolvers used as *control* in UBICA .

An example of error output that can be reported while performing this test can be seen in Figure 4.12.

```
Server: 127.0.0.1
Address: 127.0.0.1#53

** server can't find 18starjp.blogtan.net: SERVFAIL
```

Figure 4.12: Example of `nslookup` output in case of errors.

Possible outcomes from this test are:

NOERROR : no error conditions have been detected [Mockapetris, 1987a]

SERVFAIL : "server failure" - the name server was unable to process the query [Mockapetris, 1987a]

NXDOMAIN : "no such domain" - the domain name referenced in the query does not exist [Mockapetris, 1987a]

NOANSWER : the server replied, but provided no answer for the requested query (that in the tests is of `type A`)

TIMEOUT : the server did not reply before the timeout (1 minute)

This kind of evidence is the basis for the detection of DNS tampering techniques described in Section 2.3.2.

In order to distinguish among different DNS tampering techniques, the same request is issued from the probe also towards a list of open resolvers, used as *control* resolvers from inside the censored network. The list of open resolvers is the same as used in [Nabi, 2013] and is reported in Table 4.2.

TCP reachability

In order to check for filtering triggered by *IP:port* this phase of the test tries to establish a TCP connection starting a three-way handshake, with a given timeout. The tests takes as parameters a socket *targetIP:port* and a timeout value in seconds, that has been set by default to 15. The choice of this default value is motivated by the consideration that times of the order of tenths of seconds only for the establishment of the connection make it practically unusable, even though network equipment can tolerate higher delays [5].

In the case the target server is reachable and the considered port has a process listening on it, a full establishment and teardown of a connection is performed: a total of 6 TCP packets carrying no payload are exchanged between the probe and the destination as per Fig.4.13.

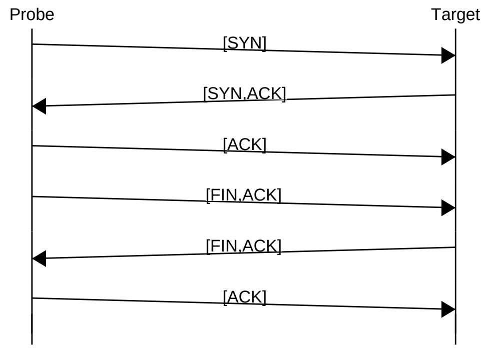


Figure 4.13: TCP connection test - Sequence Diagram of the full three-way handshake and teardown.

Possible outcomes from this test are:

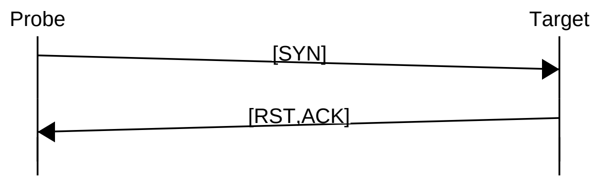
Open : the probe completed the connection establishment;
RST : a packet with RESET flag has been received;
Timeout : no answer has been received before the timeout expired;
Network error : network unreachable or destination unreachable;

The scenarios that explain the failure cases are shown in Fig. 4.14. We notice than the fact that the case of out-of-band censorship devices forging `RST` packets to disrupt the connection (Fig.4.14b) could be detected with specific and non-trivial techniques (as described in [Clayton et al., 2006, Khattak et al., 2013, Polverini and Pottenger, 2011]) and told from a legitimate unavailability of the service (Fig.4.14a), but this requires the inspection of packet-level traces of the communication. The collection of such traces requires administrator rights on the user's system and poses high privacy and security

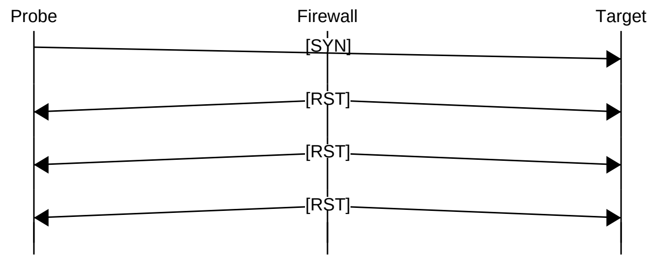
[5]e.g. cisco security appliance ASA 5500 waits a default 30*s* for a new TCP connection to be established (see http://www.cisco.com/en/US/docs/security/asa/asa82/configuration/guide/conns_connlimits.html).

risks, as the application would have access to the whole traffic originated or received on the system. To avoid these concerns and to ease the installation of the UBICA client to the end users we made the design choice of not requiring administrator rights, limiting the level at which network data can be read. As a consequence of this, in the presented platform the analysis on RST packets that could tell a censor action from a legitimate technical unavailability can not be performed, because the results of the TCP-connectivity test is collected at application level, as it is provided through the Operating System API. Therefore to tell legitimate unavailability and technical issues from active censorship we rely on multi-point and long-term probing (space and time diversity).

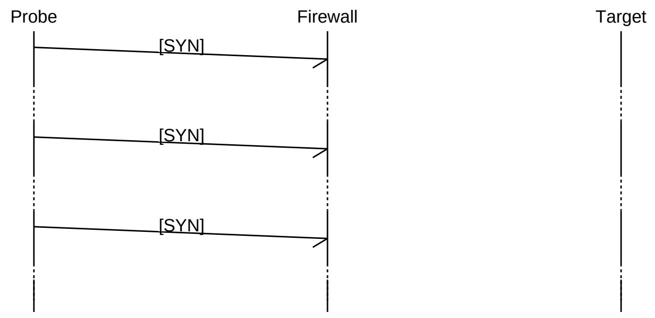
As this test can be affected by network conditions, in the case the result is not *Open* the test is repeated in a short time, until the maximum number of repetitions is reached (set to 3 in the experiments). This way a single *Open* result will signal that a TCP connection can be established, while to characterize the target service as *unreachable at TCP level* it will be necessary to sum up three subsequent failures.



(a) Legitimate RST



(b) Censorship RST



(c) Timeout

Figure 4.14: TCP connection test - Sequence Diagrams of failure cases: (a) the expected scenario causing a RST; (b) the case of out-of-band censoring device; (c) the case of in-line filtering device (also common firewall devices and personal firewalls).

HTTP reachability

This test has usually been performed after a TCP-connectivity test towards the IP address and port of the webserver. The trigger used in this test is an `HTTP GET` request: the response (of lack of it) from the server is collected, along with application level values.

Before the issue of the actual request, two parameters are set dynamically: the `User-Agent` string and the IP address of the target server. The HTTP header field `User-Agent` is chosen randomly in a list of the most common user agent strings, according to Eckersley [2010] and similarly to what specified in [ooni-specs]; the list of used strings is shown in Tab. 4.3. The main purpose of this is to blend the request among the

`Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.7) Gecko/20091221 Firefox/3.5.7`
`Mozilla/5.0 (iPhone; U; CPU iPhone OS 3 1 2 like Mac OS X; en-us) AppleWebKit/528.18`[...]
`Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6`
`Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6`
`Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6`
`Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.9.2) Gecko/20100115 Firefox/3.6`
`Mozilla/5.0 (Windows; U; Windows NT 6.1; de; rv:1.9.2) Gecko/20100115 Firefox/3.6`
`Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.9.2) Gecko/20100115 Firefox/3.6`
`Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.7) Gecko/20091221 Firefox/3.5.7`
`Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.9.1.7) Gecko/20091221 Firefox/3.5.7`[...]

Table 4.3: List of `User-Agent` strings used in the HTTP reachability test. Sources: [ooni-specs], [Eckersley, 2010].

ones issued by common users: a server that discriminates on the basis of the `User-Agent` field would affect a significant part of the *western* user base. For the specification of the IP address of the target server two different criteria are used: (i) when testing the reachability of a resource at a given - trusted - server, the IP address is directly specified in the input data, bypassing DNS resolution to avoid DNS tampering; (ii) when deepening the collection of evidence of DNS tampering, the IP address is taken from a DNS resolution performed locally from the probe towards its default resolver. This allows to use the same code for both purposes, varying just the data that is provided in input to the test.

The behavior in case of `HTTP Redirection` is set to following the `Location` header field, i.e. a new `GET` request will be issued towards this other URL. This event is revealed by both a counter of redirects and the list of HTTP headers related to the series of requests and saved in the results report.

Other forms of automatic redirection are provided in HTML by means of a number of mechanisms, including [Chellapilla and Maykov, 2007, W3C, 1999, WASC, 2011]:

- "refresh" `meta` tag specifying an URL
- framing (in a frameset)
- framing (in an i-frame)
- client-side scripting (e.g.) JavaScript [Hoehrmann, 2006]

We group all of these under the umbrella-term "HTML-based redirection" (as opposed to HTTP-based one). In order to follow these kinds of redirects the processing of HTML code (and possibly embedded scripting code) is necessary; such processing is *not* performed in the HTTP reachability test, that processes only `30X` HTTP redirects. This way the HTML (and possible scripting) code is downloaded as resource content and reported for analysis.

According to our preliminary measurement campaigns, stopping content download at this stage does allow to detect censorship actions that are triggered by hostname, server

IP and path of the requested resource, and provide as reply to the browser request a redirection in place of the requested content (see Section 2.3.7). In fact the returned content is usually much smaller in size than the requested resource (see Section 3.1.4) and consists in the code for the HTML redirection.

A possible limitation of this design choice is related to the possibility that the target resource legitimately contains HTML-based redirection; moreover this mechanism has consequences in the interpretation of the results by the user. A discussion of this case is reported in Section 5.5.1.

In case of SSL connection (when the target URL presents the `https` protocol indication), the validation of the certificate is *disabled*, i.e. the connection will be established even when it is not possible to validate the server certificate, or if the validation would fail. This is done in order to get results even in case of certificate expirations, misconfigurations and from a possibly malicious interlocutor (*TLS tampering*, see Section 2.3.8).

Other settings regard timing boundaries, that are enforced in order to limit the overall evidence gathering time, as the reporting of the results to the management server is bound to the termination (either natural or forced) of all the network tests. The establishment of a TCP connection has a timeout of $15s$ (coherently with the TCP connection test). The whole transfer is bound to finish $10s$ before the evidence gathering session timeout, in order to give the application the chance to exit cleanly (with a timeout error) without being forcibly terminated. The size of the downloaded resource is also bound to a maximum, that in the experimental evaluation has been set to $5MB$ as a safeguard. The tool used to issue the request and collect application level information is `curl`, a mature and widespread command line tool for "getting or sending files using URL syntax" [6].

The report from this test will include the values listed in Table 4.4, mostly from the `curl` interface.

Besides the aforementioned output values, a number of coarse grained outcomes are possible, notably in case of absence of returned content. In such case, trying to fetch the resource pointed by the URL has failed, for the following possible reasons:

- an HTTP error occurred (recorded in `http_code`)
- a network error occurred (recorded in `curl_exit`)
- maximum number of redirects reached (recorded in `curl_exit` and `num_redirects`)
- a timeout is reached (recorded in `curl_exit`)

[6]http://curl.haxx.se

Value	Description
`content_type`	The Content-Type of the requested document, if there was any.
`http_code`	The HTTP code returned for the last request (a 30X will be followed, thus usually will not appear).
`num_redirects`	Number of redirects that were followed in the request.
`size_download`	The total amount of bytes that were downloaded.
`size_header`	The total amount of bytes of the downloaded headers.
`size_request`	The total amount of bytes that were sent in the HTTP request.
`size_upload`	The total amount of bytes that were uploaded.
`speed_download`	The average download speed that curl measured for the complete download. Bytes per second.
`speed_upload`	The average upload speed that curl measured for the complete upload. Bytes per second.
`time_appconnect`	Seconds from the start until the SSL handshake to the remote host was completed.
`time_connect`	Seconds from the start until the TCP connect to the remote host (or proxy) was completed.
`time_pretransfer`	Seconds from the start until the file transfer was just about to begin. This includes all pre-transfer commands and negotiations that are specific to the particular protocol(s) involved.
`time_redirect`	The time, in seconds, it took for all redirection steps include name lookup, connect, pretransfer and transfer before the final transaction was started. time_redirect shows the complete execution time for multiple redirections.
`time_starttransfer`	Seconds from the start until the first byte was just about to be transferred. This includes time_pretransfer and also the time the server needed to calculate the result.
`time_total`	Seconds for the whole operation to complete.
`url_effective`	The URL actually requested (the last of redirects)
`useragent`	The string chosen as `User-Agent` HTTP field in the request.
`hostIP`	The IP address of the server destination of the HTTP request.
`curl_exit`	The exit error code of the application.
`content`	The data returned as content of the HTTP response.
`headers`	The data returned as the *header* fields of the HTTP responses (appended, for multiple exchanges in case of redirects).

Table 4.4: Output values from the HTTP test. Except last five, they are obtained by means of the `curl` API and the descriptions are adapted from the application *man page*; last five are the result of the setup phase before the actual GET request and data received in the responses.

	DNS	Host	GET string	Description and rationale	Censorship evidence
0	local	from URL	from URL	normal request	no content, blockpage, errorpage
1	**trusted**	from URL	from URL	normal request toward a trusted IP address	if censored: DNS-tampering
2	**trusted**	from URL	**random**	request of a plausibly non-existent resource on target hostname, toward a trusted IP	if not HTTP 404 response, censorship triggered by the *hostname*
3	**trusted**	**random**	from URL	request of resource from plausibly nonexistent domain, toward a trusted IP	if not HTTP 404 response, censorship triggered by the *resource path*
4	**trusted**	**random**	**random**	request of resource from nonexistent hostname, toward a trusted IP	if not HTTP 404 response, censorship triggered by the *IP address and GET method*

Table 4.5: HTTP test variations: by substituting specific parameters it is possible to infer the one(s) that trigger the censorship.

These provide the evidence for censorship detection tests based on content analysis.

The control over IP address towards which the request is issued and on the header fields of the request allows for a number of secondary tests that leverage the HTTP test to deepen the analysis in case of censorship suspicion.

In order to infer the trigger that causes the censorship mechanism, in case of censored content a sequence of purposely forged HTTP GET requests can be issued according to Table 4.5.

Case 3, when the hostname is substituted with a random string, can generate two legitimate (i.e. in absence of censorship) responses: an HTTP 404 error, or an HTTP 200 confirmation, and a resource in the body. The last is caused by the Virtual Host matching algorithm [7] and can happen when:

there are no virtual hosts : the resource returned will be the same of case 1

the default virtual host is the original hostname : the resource returned will be the same of case 1

the default virtual host has the same resource : the resource returned can be different from case 1

Ambiguity in this case can be dealt with by comparing the response with what happens in case 1. If in case 1 the blocking resulted in no resource (and error X), and same for case 3, the (same) censorship is still triggered while if a resource is returned only in 3 then the trigger was in the hostname and could be circumvented if no virtual hosting is

[7]Virtual Hosts are chosen by matching the server IP address (in case of multi-homed hosts) and then on HTTP host field value, falling back to the default website on the IP if no hostname matches on it (e.g. see http://httpd.apache.org/docs/2.2/vhosts/details.html).

running on the target webserver (by specifying the trusted IP and a wrong hostname). If in case 1 the blocking resulted in a resource (blockpage, errorpage or mangled page), then in case a resource is returned in 3, it must be compared with it: if matches, the (same) censorship is still triggered, if it is different, then the trigger was in the hostname and there is possibility of circumvention. Last possibility is when case 1 resulted in a resource and case 3 in an HTTP 404 error: then clearly the blocking was triggered by the hostname, but no circumvention is possible (as virtual hosting is active).

Drawbacks of this testing approach are that it causes errors of wrong pages requests in the target servers (if the request reaches them) and it is fingerprintable. The first aspect can be mitigated by choosing the suitable probing rates for the considered target. The fingerprintability can be mitigated by using, in place of the random substitute strings for resources, URLs that are at the same time: similar to the original ones, not containing the same strings (that could trigger a pattern-based DPI), and known to be missing from the target host, and thus potentially cause a revealing 404 error. As the concealing of probing activity has been excluded from the stated goals for the platform, the implementation of this enhancement has been not conducted for the prototype realization used in the experimental validation.

SSL certificate check

Possibilities A way to detect this type of censorship can be based on the content analysis, as it should lead to altered or no content provided that the invalid certificate is not checked. To detect the specific censorship technique (and thus identify the level at which the censor acted) a copy of the SSL certificate can be compared with the one provided by a *ground truth*, supposedly from outside the censornet. It is worth noting that knowing that this censorship technique is applied does not help much in circumventing it, because even if the rogue certificate is detected (e.g. by adopting the *certificate pinning* practice), the connection can not continue or is known to be tampered with: from this point of view it could be an effective censorship technique. Nonetheless, even if the access to the original content is prevented, a strong evidence of censorship is surfaced, revealing that the proper communication standards and rules have been not respected, and the trust of the user has been betrayed.

Previous research of MITM attacks by means of forged certificates can be found in [Holz et al., 2011], where a survey of SSL adoption in 2010 and 2011 has been performed on the Alexa top 1M sites of the time, by actively probing from PlanetLab nodes the HTTPS services on the hosts and analyzing the properties of the returned certificate. The check for MITM attacks was indeed a secondary goal for the authors, that were performing a broader analysis on the status of TLS/SSL adoption over time. They defined as *suspicious* cases the ones that provided a different certificate while the certificates were identical if retrieved from all but at most 2 other locations. The authors did actually find a higher number of *suspicious* cases when probing from Beijing, Moscow and Shanghai, with respect the other European or American probes locations, but did not conclude that they were due to MITM attacks, blaming administrative difficulties, motivating that with the consideration that Shanghai suspicious cases corresponded to domains not identifiable as sensitive (neither political, nor religious, or circumvention or communication topics were involved). From the point of view of censorship detection, main shortcomings of the method adopted by the authors of [Holz et al., 2011] derive from the fact that they (knowingly) rely on local DNS resolution and thus could be subject to both CDN-based localization and to DNS tampering.

In the UBICA platform the evidence collection for this test is implemented by means of the `openssl` tool [8], used to retrieve the certificate of the given server from the probe.

In order to differentiate the cases when DNS tampering is in action or not, and to avoid false positives due to CDN and similar localization mechanisms, the IP address of the server is specified explicitly, while the hostname is specified by means of the `SNI` (Server Name Indication) extension [3rd, 2011] in case the host is serving multiple Virtual Hosts. Thus the SSL check takes 2 input parameters (server IP address and hostname) and outputs 4 values: the `X.509` certificate in PEM format, a flag signaling whether a *ground truth* IP address has been specified or a local DNS resolution has occurred, the standard error of the SSL handshake (reporting possible errors and other metadata).

In [Winter, 2013] a test similar to the SSL check proposed in the UBICA architecture is described, with the difference that DNS resolution is not controlled in performing the test (it is not performed at all if a DNS tampering is preventively detected).

Traceroute

Traceroute [Jacobson, 1999] and its variants are active probing tools using network packets forged with an increasing value of the Time-to-Live (TTL) field to solicit an ICMP Time

[8]https://www.openssl.org/

Exceeded error message one hop at a time. Their use for censorship detection is described in Sections 3.1.2 and 3.1.5.

The UBICA platform generally uses the traceroute tool provided with the Linux operating system, that employs UDP packets and does not require administrator privileges. If such privileges are available, the `hping3` [9] implementation of traceroute is employed, as besides using TCP and ICMP protocols, it is capable to set arbitrary data in the payload of the probe packets.

The *traceroute* measurement requires as mandatory input an IP address, and optionally (provided that administrator privileges are granted) a protocol, a port, and a data payload; the typical output is reported in Figure 4.15.

```
HPING 143.225.229.169 (eth1 143.225.229.169): S set, 40 headers + 0 data bytes
hop=1 TTL 0 during transit from ip=192.168.0.1
hop=1 hoprtt=1.0 ms
hop=2 TTL 0 during transit from ip=213.205.16.53
hop=2 hoprtt=21.2 ms
hop=3 TTL 0 during transit from ip=94.32.137.129
hop=3 hoprtt=20.3 ms
hop=4 TTL 0 during transit from ip=213.205.17.181
hop=4 hoprtt=21.2 ms
hop=5 TTL 0 during transit from ip=94.32.135.137
hop=5 hoprtt=25.7 ms
hop=6 TTL 0 during transit from ip=193.201.29.15
hop=6 hoprtt=25.4 ms
hop=7 TTL 0 during transit from ip=90.147.80.58
hop=7 hoprtt=49.8 ms
hop=8 TTL 0 during transit from ip=90.147.80.166
hop=8 hoprtt=29.7 ms
hop=9 TTL 0 during transit from ip=90.147.80.150
hop=9 hoprtt=35.8 ms
hop=10 TTL 0 during transit from ip=193.206.130.10
hop=10 hoprtt=62.8 ms
hop=11 TTL 0 during transit from ip=143.225.190.145
hop=11 hoprtt=28.7 ms
hop=12 TTL 0 during transit from ip=143.225.190.97
hop=12 hoprtt=51.0 ms
len=46 ip=143.225.229.169 ttl=52 DF id=33934 sport=0 flags=RA seq=12 win=0 rtt=43.1 ms

--- 143.225.229.169 hping statistic ---
13 packets tramitted, 13 packets received, 0% packet loss
round-trip min/avg/max = 1.0/32.0/62.8 ms
```

Figure 4.15: Evidence collection - hping3 traceroute

4.4 Analysis Engine

The primary goal of the UBICA platform is to detect censorship, i.e. given a target resource and a viewpoint (a country / ISP / network address range) tell if the resource is

[9] a command-line network diagnostic tool http://hping.org

censored, with what kind of technique, and with what degree of confidence. Given a target resource and a probing vantage point, for each analysis interval the ultimate outcome is one of the following:

insufficient data : some data, but no enough to run the algorithm
not censored : verdict with a confidence index
censored : verdict with a confidence index

4.4.1 Evidence Data

In order to reach a verdict about the resource the results from clue collection are evaluated. Such data is considered in analysis tasks dedicated to specific phases of the communication model (Section 2.2.1 that create the intermediate data used for the censorship detection algorithm. The output of each analysis task is available to the *researcher* through the Management Interface, that constitutes an abstracted interface for both the access to the raw data and the setup of the parameters of the analysis algorithm.

Data cleansing

One guiding principle in the platform design has been the base hypothesis that a resource is *not* considered censored unless contrary evidence is available. Considering the detection algorithm as a binary classification process, this is equivalent to say that the null hypothesis is that the resource is *not* censored, and the detection algorithm will preferentially generate Type-II error (rejection of a real censorship occurrence) over Type-I ones (false alarm).

Most of the considered measurements are end-to-end tests, that can fail for a number of unintentional reasons (faults, misconfiguration, catastrophic events, massive malicious activity). Research in Internet-scale outages such these [Jun and Brooks, 2011] has considered their effects as negligible after 48 hours. In order to account for such temporary conditions, a tentative time span for analysis has a been set to 14 days: approximately one order of magnitude longer as a cautionary measure, and chosen as a multiple of the week to take into account possible anthropic patterns. Despite the long time averaging interval, the temporal analysis algorithm is designed in order to have half-size scope: specifically even if a 14 days long interval has been set, time variations with 1 week scope are detected (see Section 4.4.3).

	DNS				TCP			HTTP	
Outcome	Error	Failing IP	Block IP	Error IP	Error	RST	Timeout	No reply	Undersize
Threshold	70%	70%	0%	70%	50%	70%	70%	70%	70%

Table 4.6: Data Cleansing: coherence thresholds

A minimum number of tests to be considered for said time span can be set (in the experimental evaluation defaults to 10 tests over a 2 week timespan [10]) so that multiple samples for each experiment are available and the effect of occasional outages is averaged.

For each test outcome a minimum percentage threshold can be defined: if the relative frequency of occurrence of the outcome does not surpass its threshold the outcome is *not* considered for the analysis. These thresholds can be considered indicators of the coherence of the outcome and are to be tuned according to the amount of false positives generated by over-threshold results. For the experimental evaluation they have been set to the values reported in Table 4.6. It is worth noting that the thresholds are set depending only on the type of outcome and are valid for all the resources and for all the viewpoints. A possible enhancement is to specify per-(viewpoint,test) threshold and evaluate its impact on the false positive rate, that has been left for future research.

Another pre-processing task that is performed regards hostnames that result unreachable from all the ISPs worldwide. The unreachability can be either at the DNS level (returning DNS errors or Error Pages) or at the TCP level (connection error) or at the *HTTP content* level (no content). This is explained by the fact that during the monitoring time some websites are decommissioned, and thus either (i) the hostname itself is no more present in the DNS global database (expired domains) or (ii) no server is available to reply or (iii) no content is provided. If the unreachability is at DNS level worldwide, these hostnames and the resources that are hosted at them are signaled for manual inspection: the operator can then mark them as *dismissed* (see Figure 4.10) and have them excluded from the censorship analysis and future scheduling of tests.

It is worth noting that server-side censorship, i.e. censorship applied either by the content manager, or the service host, could generate such a result: the UBICA platform in the presented design does not try to detect this kind of censorship and its verdict, coherently with the cautionary principle, is of *not* reporting the case as censorship.

[10]this is a tentative value that heuristically has yield no false positives in the experimental evaluation; it is consistent with -and more cautionary than- a similar threshold of 3 samples per week used in [Gill et al., 2013].

4.4.2 Detection Algorithms

The detection algorithms that are employed in the UBICA platform integrate different kinds of information, coming from:

- the specific experiment (viewpoint,target) under evaluation
- other experiments with the same target but different viewpoints (in the same ISP, in the same country, globally)
- the knowledge base (knowledge from external sources or provided by manual inspection)

The detection algorithm takes as input a triplet (viewpoint,target,timespan) and outputs a *verdict* comprising:

- statistics of the censorship *index* (minimum, average, maximum)
- the list of censorship techniques detected

The censorship *index* is a summarizing value that accounts for the number of different censorship techniques that have been detected and the percentage of times they have been detected. It is meant to serve as an overall index of the technical effort that a censor has put into preventing the access to that *target* and the effectiveness of these efforts.

The list of censorship techniques is built by analyzing the outcomes of the measures as described in the following.

Pre-processing of DNS measures

The results of DNS measures (see Section 4.3.2 for details) consist in the response to a query of *type A* that is issued towards either the user's default DNS resolver, or to a group of open resolvers (used as control). For both kinds the analysis steps are the same.

The answer of the query is classified as:

DNSerror : the query resulted in an error, no IP address was returned

Failing IP : the returned IP address belongs to a list of known addresses that will cause a TCP connection error; this list comprises non-publicly-routable addresses such as private addresses;

Error Page : the returned IP address hosts a webserver that invariably serves a page telling that the requested hostname is not existent;

Block Page : the returned IP address hosts a webserver that invariably serves a page telling that the requested hostname *has been deliberately blocked*, possibly reporting the reason and the law under which this has been done.

Plausible DNS result : if none of the previous ones verified, i.e. an IP address has been returned and it is not present in any of the known lists.

For each class the percentage is taken on the specified time span, this percentage will be compared with the *coherence threshold* described in Section 4.4.1 to decide whether the results will be considered for the final censorship verdict. This is done because almost all the outcomes could be returned in case of technical issues or outages, and thus some tolerance for faults has been factored in the design. A notable exception is for the case of *Block Page* class, for which the censorship is explicitly stated and is not associated with outage scenarios; therefore a threshold of 0% has been chosen for it.

The described summarizing pre-processing constitutes the DNS Analysis View (and is implemented as a *materialized view* in the database).

Pre-processing of TCP measures

The analysis of the TCP measures is straightforward (see Section 4.3.2 for the collection of base data), and consists in grouping the outcomes in broader classes:

RST : three subsequent connection tries have been shut down with RESET packets;

Timeout : three subsequent connection tries have timed out;

Error : other network errors have occurred;

Reachable : a TCP connection has been correctly established (complementary to the other cases);

The described summarizing pre-processing constitutes the TCP Analysis View (and is implemented as a *materialized view* in the database).

Pre-processing of HTTP measures

The results of *HTTP reachability* measurements (Section 4.3.2) are analyzed to infer information about content. Given a time interval and a *target*, the summaries and the percentage of occurrences of the following events are computed:

- a content has been downloaded (percentage);

- median of the downloaded content (in bytes) - samples with no downloaded content are ignored;
- couple (HTTP redirect count, percentage);
- couple (HTTP response code value, percentage) - in case of HTTP redirect this refers to last retrieved resource;
- couple (application-level error code, occurrence).

The described summarizing pre-processing constitutes the Content Analysis View (and is implemented as a *materialized view* in the database).

Censorship verdict

Given a time interval and a *viewpoint*, for each *target* a *verdict* is obtained on whether it is censored or not, and if it is, with which techniques. The *viewpoint* represents the network hosting the probe, and can be set to different aggregation levels, such as *address range*, *ISP*, *Country*; results collected from one *viewpoint* are compared with “the rest of the world”, i.e. the complement to the *viewpoint* of the set of all measures; in the following for easy of exposition and coherence with the displayed results the aggregation level *ISP* has been considered.

The algorithm is a cascade of three steps, with results from one step used in the subsequent one. For each technique that has been appended to the list of detected techniques the related percentage of occurrence is summed to a temporary counter that will be used to calculate the overall confidence index.

The first step is the evaluation of DNS tampering. Only results crossing the *coherence threshold* are considered (4.4.1), and every condition of the form *isDNSblockIP(D)* is a shorthand for “the percentage resulting from pre-processing of *DNSblockIP* outcome for *D*efault resolver has crossed its specific *coherence threshold*”; analogous meaning has the condition *isDNSblockIP(C)*, but referred to *C*ontrol resolvers; and the expressions ending in “_xavg” mean that the percentage is the result of an average on all ISPs but the one under analysis, thus *isDNSerr(C)_xavg* means that the average of Control resolvers calculated on the complement to current ISP has crossed the threshold for *DNS error* .

```
if isDNSblockPage(D) then
    if isDNSblockPage(C)
    then append_to_techniques ( DNSblockPage_injection )
    else append_to_techniques ( DNSblockPage_hijacking )

if isDNSfailIP(D) then
    if isDNSfailIP(C)
    then append_to_techniques ( DNSfailIP_injection )
    else append_to_techniques ( DNSfailIP_hijacking )

if isDNSerrorPage(D) then
    if isDNSerrorPage(C)
    then append_to_techniques ( DNSerrorPage_injection )
    else append_to_techniques ( DNSerrorPage_hijacking )

if isDNSerror(D) then
    if isDNSerror(C)
    then
        if ( not isDNSerror(D)_xavg ) and ( not isDNSerror(C)_xavg )
then append_to_techniques ( DNSerror_injection )
    else
        if ( not isDNSerror(D)_xavg )
then append_to_techniques ( DNSerror_hijacking )
```

Figure 4.16: Censorship verdict algorithm - step 1 (DNS analysis)

The second step is the detection of packet filtering and TCP connection disruption (Sections 2.3.3 and 2.3.4). As the TCP measurement is performed towards the IP address returned by the default DNS resolver, whenever a DNS response of type *failing IP* is returned then the TCP connection is meant to fail: this should not be accounted as due to TCP-level tampering, thus a correction is made on the percentages on which apply the technique-specific threshold.

Given the following definitions

- F the total number of failures at TCP level
- D the count of *DNS failing IP* occurrences
- C the count of TCP failures not due to *DNS failing IP*
- th the threshold for TCP failures
- T the total number of tests
- R the number of TCP tests not preceded by *DNS failing IP*

the threshold crossing that would have been verified as

$$\frac{F}{T} > th$$

is instead evaluated as

$$C = F - D$$

$$R = T - D$$

$$\frac{C}{R} > th \quad \Rightarrow \quad C > th \cdot T - th \cdot D$$

Besides the threshold adaptation and percentages computed on a reduced number of tests, the different outcomes are evaluated similarly to the previous step, possibly adding new detected *symptoms*: `TCP-Timeout`, `TCP-RST`, `TCP-errors`, and summing their relative occurrences (calculated with respect to the reduced number of tests) in the confidence counter.

The third step regards the detection of *HTTP tampering* techniques (Section 4.3.2). It is composed of two parts: the evaluation of cases when no content was retrieved, leading to `HTTP-noContent`, and when a content has been retrieved but its size is smaller than a give threshold compared with the average of the size computed on all ISPs but the one under analysis, leading to a condition of `HTTP-undersize`.

As for the case of TCP, the threshold for `HTTP-noContent` has to be adapted accounting for the failures due to both *DNS-failingIP* and TCP errors.

The final *verdict* returns the list of detected techniques, each with the related percentage of occurrence, and a confidence index, calculated as the sum of all the (scaled) percentages and normalized to 0.99.

4.4.3 Time Analysis

Censorship is a dynamic phenomenon, as targets and censorship techniques vary in time. A censorship monitoring platform must explicitly address this characteristic in order to be able to both reliably detect censorship and signal when a change has occurred. The UBICA platform satisfies these requirements by performing censorship detection tests on given time intervals. A “jumping window” scheme is used to partition the available reports in a series of consecutive evenly-sized intervals.

To take into account possible border effects, a second series of contiguous time intervals is considered, having the same duration of the first one but displaced by half length. This way each interval of one series is centered on the border between two consecutive intervals of the other series. As a consequence the granularity of time analysis is of one half-interval, but employs results gathered on a full interval span. The interval duration for the experiments has been set to a default of two weeks, as a compromise between a small granularity and the size of analysis data that has to be stored. Considering a se-

Overlapping outcomes A C B	Disjoint verdict $D_1 D_2 D_3 D_4$	Description
000	0000	same as if considering only A and B
001	0001	delays detection by one half interval
011	0011	same as if considering only A and B
111	1111	same as if considering only A and B
110	1100	same as if considering only A and B
100	1000	anticipates end of censorship by one half interval
010	0110	signals censorship in the middle (undetected in disjoint intervals)
101	1001	signals absence of censorship in the middle (undetected in disjoint intervals)

Table 4.7: Time analysis: response using overlapping time intervals - Three full-length intervals (two disjoint A,B plus the overlapping one C in the center) exhibit presence (1) or absence (0) of censorship evidence; this is translated in four verdicts on the disjoint half-length intervals D_1, D_2, D_3, D_4.

quence of two disjoint intervals A and B each divided in two half intervals (A1,A2,B1,B2) and one interval C that overlaps A2 for its first half and B1 for its second half, the algorithm maps the censorship analysis results of the sequence f(A,C,B) onto the sequence D1,D2,D3,D4. The association between the possible combinations of censorship responses over two subsequent intervals and the central overlapping one is shown in Table 4.7 with the corresponding interpretation given by the algorithm. The primary criterion is conservative: for the half intervals with overlapping the verdict is for censorship if both A and C gave censorship (respectively C and B). This delays the detection of censorship or anticipates the end of detected censorship with respect to the dates obtained by considering only A and B. Another effect of the proposed mapping is to highlight variations in censorship (or its absence) that would go undetected considering only A and B. Time analysis applies iteratively this algorithm to subsequent triplets of overlapping intervals, so that the verdict on each half interval is always evaluated on the results from two overlapping full intervals.

4.5 Report Interface

The ultimate aim of the platform activity, the censorship detection results, are presented to the users in the form of graphical reports, provided by means of a dedicated interface. The Report interface is implemented as a web application, employing as a back-end the Apache webserver connected with a relational database (Postgres), and as a front end a JavaScript-enabled web client. This choice allowed to leverage the availability of mature open-source software and high-level client-side graphic libraries [11] , as well as limiting to the minimum the necessity of installing external software in order to access the platform report functionalities.

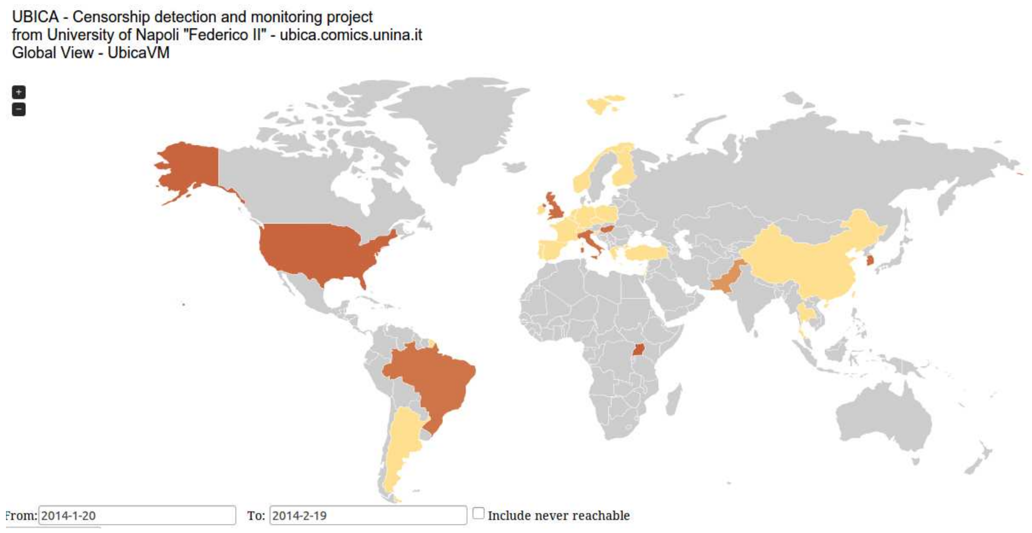
Two main views of the data are presented, a “Global” view and an “User-centric” view, depending on the network address of the user: if the request to the report interfaces comes from a network that hosted an UBICA probe, then the “User-centric” view is chosen, otherwise the generic “Global view” is shown. This is performed by matching the client IP address with the address ranges associated with the UBICA probes.

4.5.1 Global View

The *Global* view presents the analysis results regarding all the countries from which enough data has been provided to perform a censorship analysis. Aggregated results are shown in form of graphs and refer to the last month of analysis; the couple *from,to* of input fields allows to specify another time interval. The view is divided in three areas: a world map (on top), a bar-chart area (in the middle) and a time-series graph at the bottom.

The world map At the top of the report view a map of the globe is shown, with the world countries boundaries (Figure 4.17). Each country is either colored gray (if no enough data is available for a censorship analysis from that country) or is colored with a shade of red with increased intensity the higher the evidence of censorship is detected. With a mouse click on the area of a country a view is shown of results that are specific to the chosen country. This view is similar in aspect to the User-centric view, to which we refer for further details.

[11]For the graphical rendering of report data the JavaScript library *highcharts* (http://highcharts.com) has been employed.


UBICA - Censorship detection and monitoring project
from University of Napoli "Federico II" - ubica.comics.unina.it
Global View - UbicaVM

From: 2014-1-20 To: 2014-2-19 ☐ Include never reachable

Figure 4.17: UBICA Interface, Global View Reports - top of page: world map with evidence of censorship (gray countries lack enough data for an analysis; red countries have data, the darker the color the more evidence of censorship is available. Different countries are *not* to be compared, as the number of tests and targets usually differ.

The bar charts The center of the view (Figure 4.18 is occupied by three bar-charts showing (clockwise from top left): top tested countries, top 100 tested URLs, top 100 failing URLs. Each item of the chart has two bars associated: “reachable” (in green) and “unreachable” (in light red) that represent the number of access tests that resulted in a success or in a failure (hint for possible blocking). In order to compare items that have been tested a different number of times, a version of the same data expressed in percentage of the number of tests per item is provided (Figure 4.19). By hovering with the mouse pointer over one item’s bar the data values (full item name, reachability count/percentage and unreachability count/percentage) are shown; by clicking on it a pop-up window is presented, showing the detailed summaries of test results for the specific item (Figure 4.20). All the charts have a sliding bar that allows to scan over a range of items beyond the first displayed.

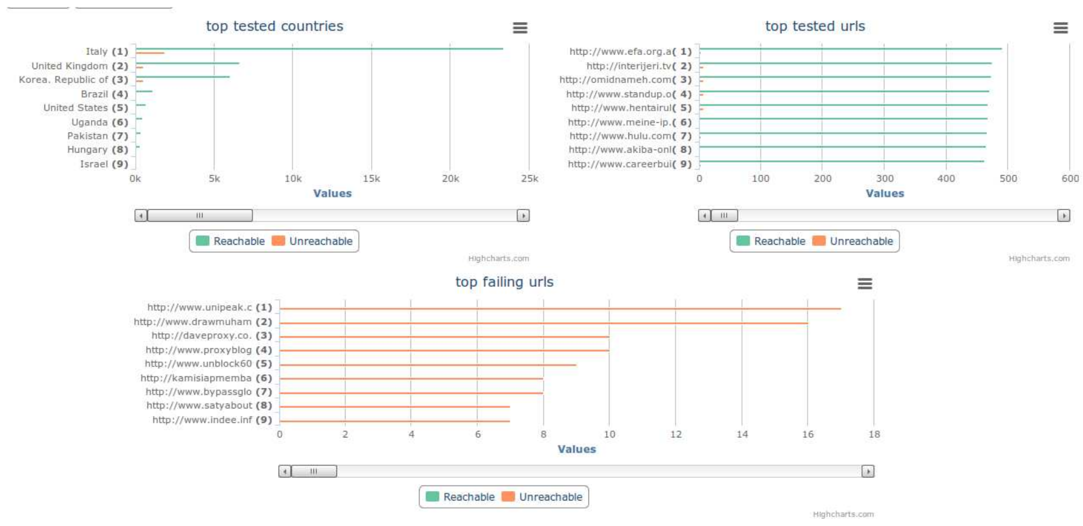


Figure 4.18: UBICA Interface, Global View Reports - middle graphs: bar charts of top tested countries, top tested URLs and top failing URLs (clockwise, from top left). Results are the number of tests for the specific item.

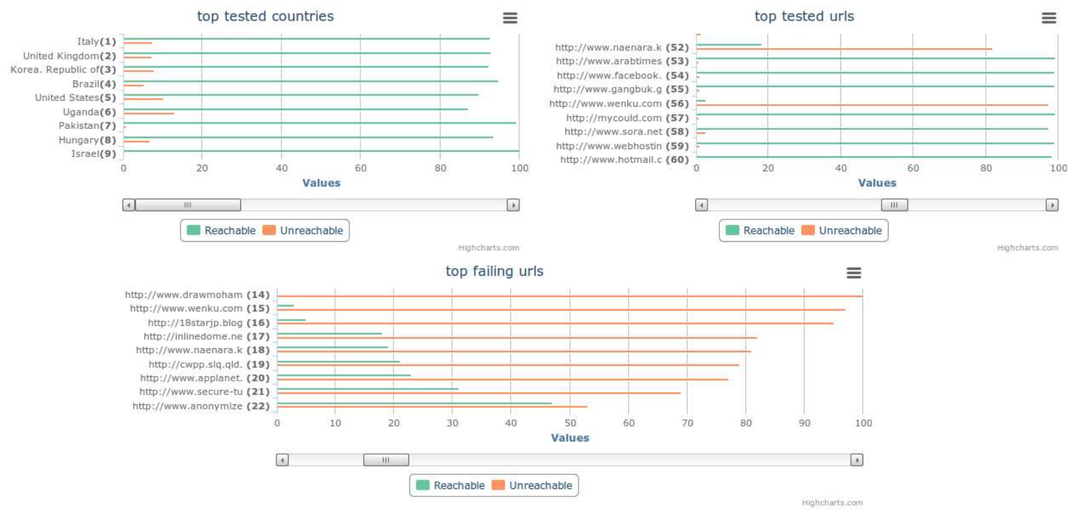
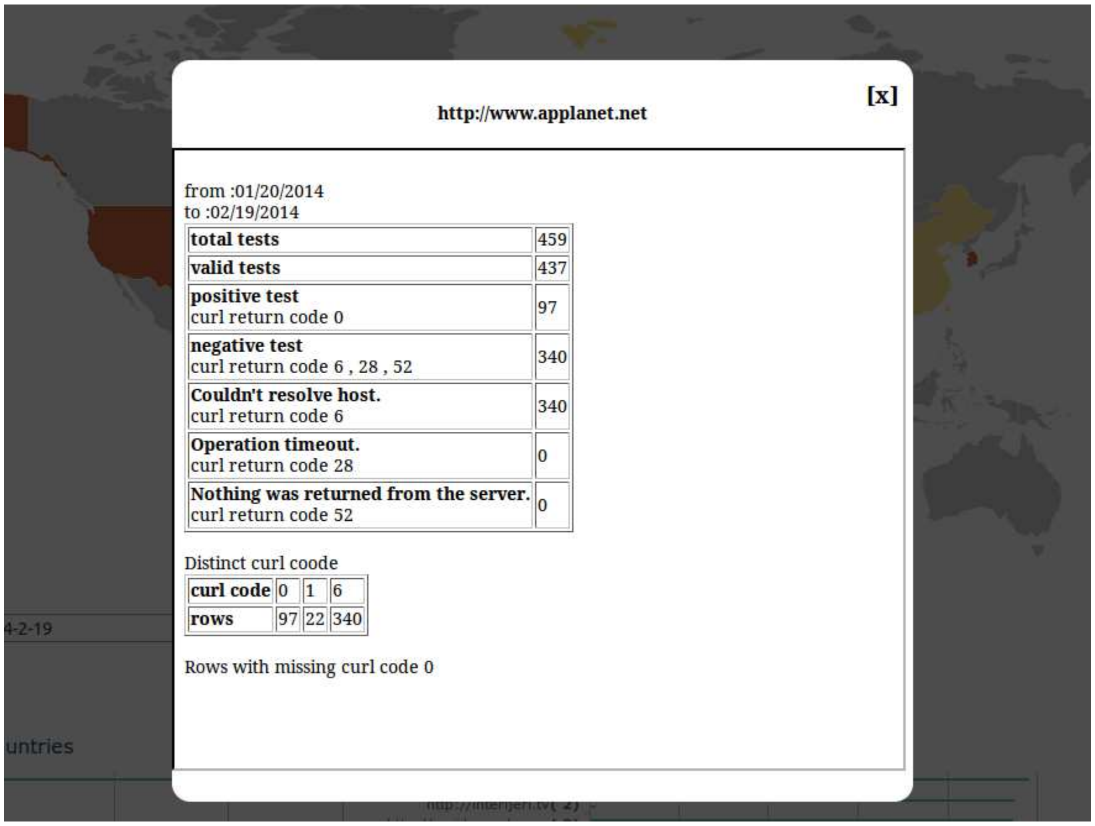


Figure 4.19: UBICA Interface, Global View Reports - middle graphs: bar charts of top tested countries, top tested URLs and top failing URLs (clockwise, from top left). Results are in percentage over the total number of tests for the specific item.



[x]

http://www.applanet.net

from :01/20/2014
to :02/19/2014

total tests	459
valid tests	437
positive test curl return code 0	97
negative test curl return code 6 , 28 , 52	340
Couldn't resolve host. curl return code 6	340
Operation timeout. curl return code 28	0
Nothing was returned from the server. curl return code 52	0

Distinct curl coode

curl code	0	1	6
rows	97	22	340

Rows with missing curl code 0

Figure 4.20: UBICA Interface, Global View Reports - clicking on a bar-chart item a pop-up window with the details of the specific item appears

The time-series chart The time-series chart at the bottom of the view shows the aggregate results of tests over time (Figure 4.21). Hovering the mouse pointer on the graph shows the timestamp and the results available for that time. Below the time chart two controls allow to set the time interval to be shown in detail.

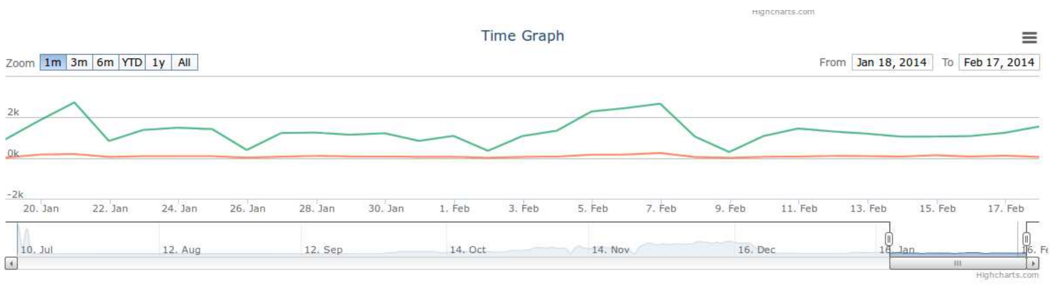


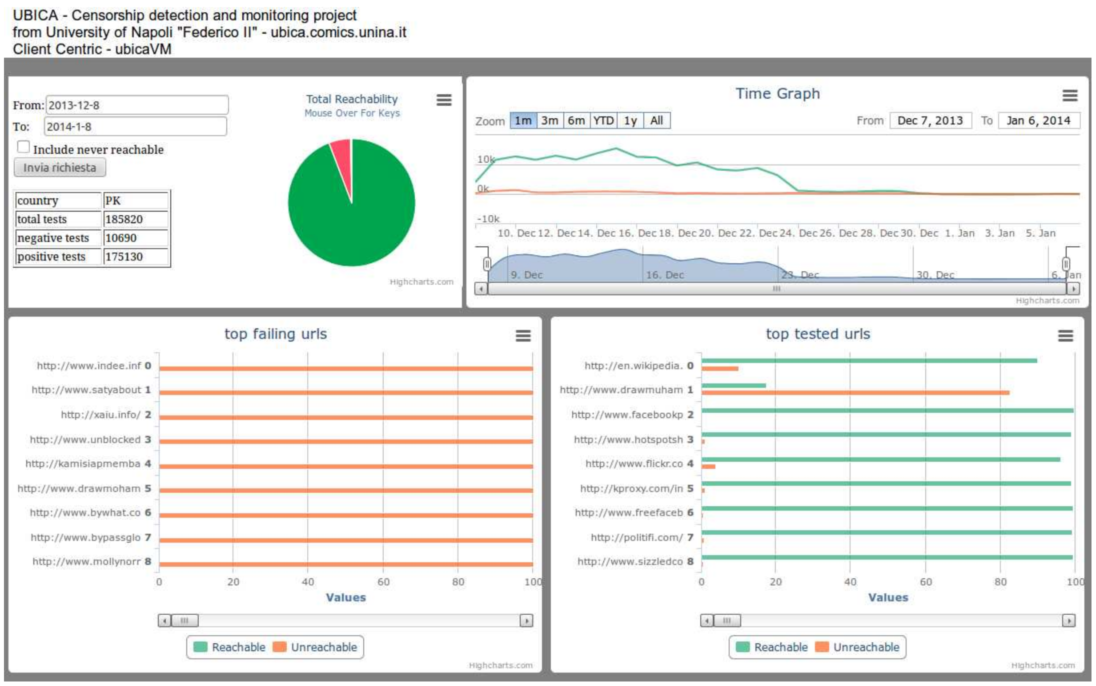
Figure 4.21: UBICA Interface, Global View Reports - bottom: time series graph of reachability tests

4.5.2 User-centric View

The *User-centric* view shows the analysis results regarding the specific network from which the client is querying the Reports Interface (as detected by the server). This is possible if the network (intended as a range of IP addresses advertised on public databases as WHOIS as belonging to a given ISP) has hosted an UBICA probe that uploaded a minimum number of reports [12] to the management server. This view is similar to the *global* view, as it shows a time series graph, and two bar charts showing the top 100 failing URLs (bottom left) and top 100 tested URLs (bottom right); moreover in the top left part a pie chart shows the overall percentage of reachability. All data refers to the specific network and the considered time span defaults to the last month for which data is available. A *from,to* couple of input fields allows the specification of the desired time range. All the graphs are a specialization of the ones available in the *Global* view, thus provide the same functionalities (sliding bars to scan the top 100 items, numeric values for an item when the mouse hovers on its bar, and item-specific details when the bar is clicked).

The same graphs are shown for data aggregated at *country* level when a tested country is clicked in the *Global* view.

[12]set by default, and in the analysis here described, to the value 10.



UBICA - Censorship detection and monitoring project
from University of Napoli "Federico II" - ubica.comics.unina.it
Client Centric - ubicaVM

From: 2013-12-8
To: 2014-1-8
☐ Include never reachable

country	PK
total tests	185820
negative tests	10690
positive tests	175130

Figure 4.22: UBICA Interface, User-based Reports

4.5.3 Analysis dashboard

The Analysis Interface has been designed to allow manual inspection of the results, highlighting conditions that can potentially lead to clues of censorship and of the techniques possibly adopted to enforce it.

In a preliminary phase, the Analysis Interface has been adopted to check the data and help define the censorship detection algorithms. In the following sections we will show selected examples from this use case.

The TCP analysis report

In order to collect targets for the detection of Transport-level censorship techniques the following algorithm is proposed:

1. select URLs whose DNS results suggest blocking (everything besides plausible IP addresses), and *no* TCP reachability issues
2. compare the returned IP address against the ones returned from probes in other countries / ISPs (different ISPs may have different censorship-enforcing boxes IPs, but also redirect to the same country-wide IP as for BlockPage cases).

3. non-matching addresses will be added for TCP-connectivity test; these tests will be marked as "driven by DNS-tampering-suspicion" as opposed to the ones deriving from DNS resolution from the default resolver;

The expected result is that by comparing, for a given resource (URL) the results of TCP reachability for default-DNS set, control-DNS set and foreign-DNS set it should become evident if there is TCP-level blocking applying the following criteria:

- if all the three sets have comparable percentages of TCP reachability (differenced less than a given tunable threshold) then we assume *no TCP-level censorship*
- if the *foreign* set shows a smaller percentage of TCP reachability (differenced more than said threshold) compared with the *default DNS set* then we consider it a potential evidence of *TCP-level censorship*, triggered by the IP:port sockets belonging to the *foreign* set; moreover this validates the evidence of DNS-based censorship and can suggest new IP addresses of type *BlockPage* or *ErrorPage*.
- the same criterion is applied to the *control* set if it does not match with the *foreign* set.

Inference of the used censorship techniques

The report on both Content-related tests and DNS-related ones is shown in Figure 4.23: data is aggregated per target URL, and geolocation data (Country and ISP, as derived from IP address) of the probe that performed the measurements; shown data are:

Tests : the total number of tests performed (i.e. the total number of reports uploaded to the platform);

NULL content % : the percentage of *webcontent* tests that resulted in no content;

Content length : the 50% percentile of content length as reported in *webcontent* tests;

DNS Plausible % - D : the percentage of *DNSresolution* tests towards *the probes default resolver* that resulted in responses *not* suggesting any tampering;

DNS Plausible % - C : the percentage of *DNSresolution* tests towards *Control resolvers* that resulted in responses *not* suggesting any tampering.

From the comparison of *content length* field across different rows it becomes evident that in Pakistan one provider (namely, *National WiMAX/IMS environment*) blocks the access to shown URL, as no content has ever been downloaded in 74% of 27 tests, while for other countries / ISPs this happened in percentages ranging from 0 (Italy) to 6 (Pakistan, other ISP). Moreover it can be seen that the DNS test was not able to detect any tampering, so either the censorship techniques for this URL does not involve DNS tampering, or the DNS tampering detection algorithm was not able to detect it. In the first case, taking the DNS results at face value, we can motivate it by noting that the ISP name suggests that the connection to the Internet is provided by means of WiMAX technologies, that are known to employ Performance Enhancing Proxies (PEP) middleboxes: these could constitute an almost cost-free censoring device allowing the provider to enforce URL-triggered DPI-based censorship. If we instead suppose that DNS tampering detection has failed, a DNS poisoning / injection could be enacted providing a resulting IP that is of type "Failing IP" (as the "ErrorPage" and "BlockPage" IP provide HTTP content).

Manual inspection of these cases can also lead to validation or improvement of the DNS tampering detection algorithm, e.g. by allowing the discovery of a new IP of type "Failing IP" that could be used to block the access to the target URL: this procedure will enable *tracking* of the IP addresses involved in DNS tampering, updating the censorship tests as the actual implementations of censorship change over time.

4.6 Known Issues and Limits

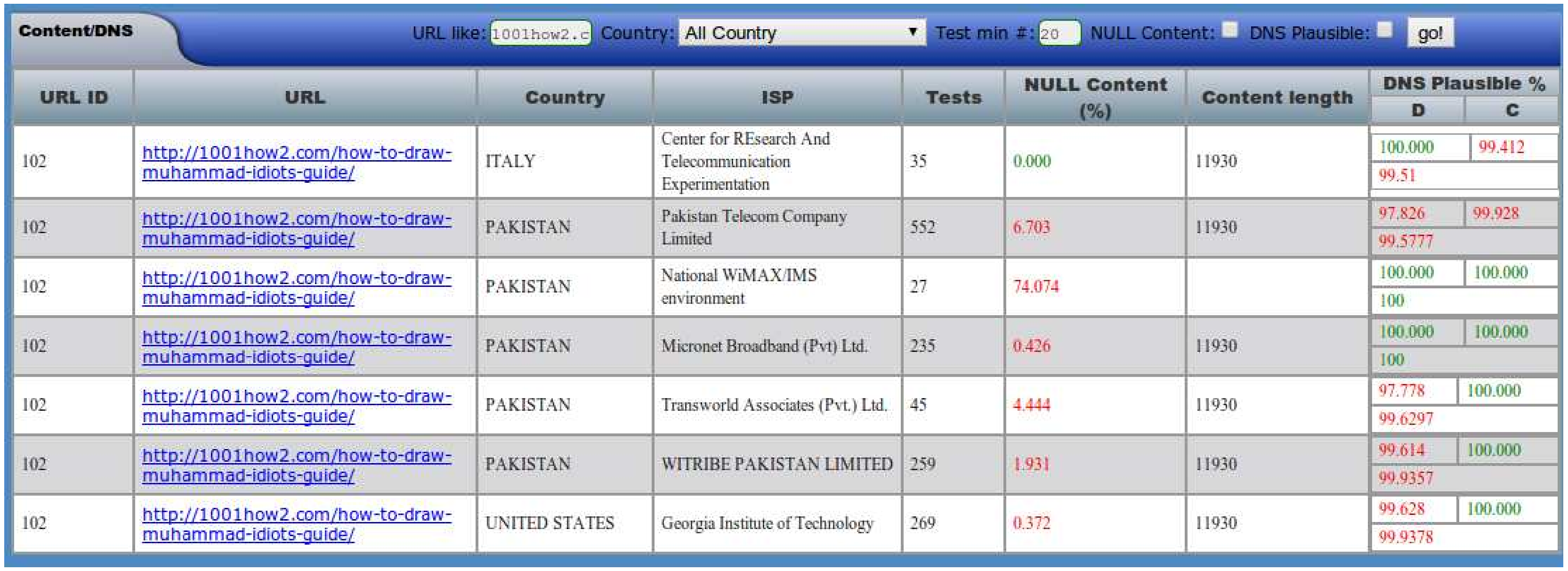
The design principles that have been chosen, while presenting the desired properties of control that allow for a flexible management of the probes and of the experiments, have some intrinsic drawbacks. These are rooted mainly in the centralized nature of the deployment, embodied in the Management Server, and in the crowdsourced nature of the data processed by the system. The centralized nature of the adopted deployment makes it more prone to detection and blocking by the censors. The phases of experiment update and input parameters download, as well the upload of the result reports to the server, can be tampered with or altogether impeded. Possible strategies for mitigation of detection and blocking are:

- encrypt communications between clients and management server
- distribute the server interface on a general purpose cloud (to avoid trivial IP-based blocking)

- adopt address distribution services for the management server address analogous to the ones adopted by the Tor network (email, mirrored index servers)
- adopt indirect channels to upload or by other means publish the result reports (e.g. using delay-tolerant anonymous communications such as i2p [Timpanaro et al., 2012].

The other intrinsic limitation is the crowdsourced origin of the analyzed information: the data are provided by anonymous users that are considered as *trusted*, in the sense that there is no direct validation of provided data, nor authentication and identification are adopted, for privacy reasons (thus a reputations system is not a viable solution).

As for other projects that rely on information that can not be validated, or provided by anonymous users that can not be held responsible for supplied data, a form of mitigation is based on statistical analysis: the more users support the same data, the more these are considered reliable. This is implemented in the UBICA platform by weighting the "confidence" value by the percentage of reports supporting the final decision.



URL ID	URL	Country	ISP	Tests	NULL Content (%)	Content length	DNS Plausible % D	DNS Plausible % C
102	http://1001how2.com/how-to-draw-muhammad-idiots-guide/	ITALY	Center for REsearch And Telecommunication Experimentation	35	0.000	11930	100.000	99.412
							99.51	
102	http://1001how2.com/how-to-draw-muhammad-idiots-guide/	PAKISTAN	Pakistan Telecom Company Limited	552	6.703	11930	97.826	99.928
							99.5777	
102	http://1001how2.com/how-to-draw-muhammad-idiots-guide/	PAKISTAN	National WiMAX/IMS environment	27	74.074		100.000	100.000
							100	
102	http://1001how2.com/how-to-draw-muhammad-idiots-guide/	PAKISTAN	Micronet Broadband (Pvt) Ltd.	235	0.426	11930	100.000	100.000
							100	
102	http://1001how2.com/how-to-draw-muhammad-idiots-guide/	PAKISTAN	Transworld Associates (Pvt.) Ltd.	45	4.444	11930	97.778	100.000
							99.6297	
102	http://1001how2.com/how-to-draw-muhammad-idiots-guide/	PAKISTAN	WITRIBE PAKISTAN LIMITED	259	1.931	11930	99.614	100.000
							99.9357	
102	http://1001how2.com/how-to-draw-muhammad-idiots-guide/	UNITED STATES	Georgia Institute of Technology	269	0.372	11930	99.628	100.000
							99.9378	

Figure 4.23: Analysis Interface: summarized results of Content and DNS tests are aggregated by target URL and probe Country and ISP; shown data are: the number of tests successfully reported, 50% percentile of web content length, and percentage of supposedly *not* tampered DNS resolutions for both *probe default (D)* and *control (C)* resolvers. It can be noted how in Pakistan one provider (National WiMAX/IMS environment) blocks the access to shown URL (by comparing with other countries/providers content length), and also infer that blocking is not done at DNS level.

Chapter 5

Experimental Validation

The platform described in the previous chapter has been implemented in a prototype presenting all the functional features as designed. A subset of the designed evidence collection tests have been implemented, namely: *DNS resolution*, *TCP reachability* and *HTTP reachability*.

The management server, hosting also the UBICA portal with public interfaces has been configured in the laboratory of Networked System at Department of Electric and Information Technologies Engineering at University of Napoli “Federico II”, with public IP address in the GARR (Italian Academic and Research telecommunication network). A repository server and a backup server within the same premises have been set up.

With the help of professional and personal contacts a limited number of software probes have been deployed in different countries worldwide, plus more than a dozen BISMark routers [Sundaresan et al., 2012] for an experimental deployment in Pakistan and two control routers, one in Italy and another in U.S.A.; the distributed experiments platform PlanetLab [Chun et al., 2003] has also been employed, deploying UBICA probes in the most diverse set of countries available at the time of the experiments.

The measurement campaigns have leveraged a different sets of UBICA probes (due to some churning in availability of the platforms and in user participation), summing up to more than 200 probes at different times, constituted by: 47 clients with GUI (run by volunteers both in Italy and abroad); 188 headless clients (of which 19 run by volunteers worldwide and 169 in PlanetLab nodes) and 16 BISMark home routers run by volunteers (mostly in Pakistan). Measurements have been made from 31 different countries, testing more than $16K$ different *targets* (about $15K$ different hostnames) on a timespan of 4 months (during multiple shorter measurement campaigns).

5.1 Global experimental campaigns

5.1.1 Country-wise *HTTP reachability*

A preliminary test of the platform functionality has been performed using the *HTTP reachability* collection test, while analysis has been performed directly on the raw data, verifying the HTTP return code (`200 OK`) and the actual download of content. The SQL code used to extract the data is reported in Figure 5.1.

Postprocessing of *HTTP reachability* data with this simple algorithm constitutes a detection technique comparable with the [Herdict] method.

The *targets* to be tested have been taken from the [Herdict] website (the most recent 600 URLs reported as unreachable) by leveraging the exposed API to query the database of reports.

In a measurement campaign started on July 16th, 2013, 22:25 UTC and spanning almost 3.5 hours over 118 globally distributed PlanetLab nodes we have collected the results of the above mentioned *content reachability* test: results are shown in Table 5.1.

By observing the reachability percentages, plotted in Figure 5.2, we can notice a spike of lack of content in requests issued from within China (CN, 70.5%). Only another country shows *unreachability* level greater than 50%: Bangladesh (BD, 69.1%). In this case the combination of having a single probe (see Table 5.1) with a limited time span for the detection does not allow to infer if a temporary connectivity issue for the probe is the cause.

The experiments have verified the main functionalities of the platform, specifically:

- the collection of *targets*;
- the controlled distribution of sublists of *targets* to the probes as they query the management server;
- the execution of the measurement tests from the probes;
- the reporting of measurement test results to the Repository Server
- the processing of the aggregated result data.

probeCC	probes	tests	reach	unreach	reach%	unreach%
US	69	69875	52579	17296	75.2	24.8
CN	10	11697	3446	8251	29.5	70.5
JP	6	6044	4396	1648	72.7	27.3
NZ	6	6569	3533	3036	53.8	46.2
CA	3	3032	2478	554	81.7	18.3
RU	3	3035	2043	992	67.3	32.7
UY	3	3091	2129	962	68.9	31.1
XX	3	3056	2340	716	76.6	23.4
BR	2	3025	2375	650	78.5	21.5
GB	2	2076	1455	621	70.1	29.9
HK	2	1351	1126	225	83.3	16.7
AR	1	1006	738	268	73.4	26.6
AU	1	1014	780	234	76.9	23.1
BD	1	1156	357	799	30.9	69.1
CZ	1	985	829	156	84.2	15.8
EC	1	678	565	113	83.3	16.7
IN	1	1011	837	174	82.8	17.2
KR	1	1031	780	251	75.7	24.3
SE	1	518	402	116	77.6	22.4
TR	1	686	572	114	83.4	16.6

Table 5.1: HTTP reachability and content presence analysis: test statistics collected from PlanetLab

cc	probes	urls	avgTriesPerURL	TriesURLprobe	avgreachperc	avgunreachperc
US	69	298	194.38	2.8	76.81	23.19
CN	10	298	29.06	2.9	35.23	64.77
JP	6	296	17.64	2.9	75.11	24.89
NZ	6	298	17.3	2.9	60.49	39.51
UY	3	295	8.84	2.9	71.16	28.84
BR	2	295	8.83	4.4	80.97	19.03
CA	3	295	8.83	2.9	83.98	16.02
RU	3	296	8.82	2.9	70.3	29.7
XX	3	296	8.69	2.9	77.96	22.04
JO	2	298	6.0	3.0	0.0	100.0
GB	2	296	5.86	2.9	71.85	28.15
HK	2	295	3.93	2.0	85.25	14.75
00	1	298	3.4	3.4	19.3	80.7
BD	1	297	2.98	3.0	35.69	64.31
IN	1	295	2.95	3.0	85.31	14.69
KR	1	295	2.94	2.9	79.21	20.79
AU	1	295	2.94	2.9	79.83	20.17
AR	1	295	2.93	2.9	76.04	23.96
CZ	1	295	2.92	2.9	85.09	14.91
TR	1	294	1.96	2.0	85.54	14.46
EC	1	294	1.94	1.9	85.2	14.8
SE	1	298	1.74	1.7	81.94	18.06

Table 5.2: Presence of content: test statistics collected from PlanetLab

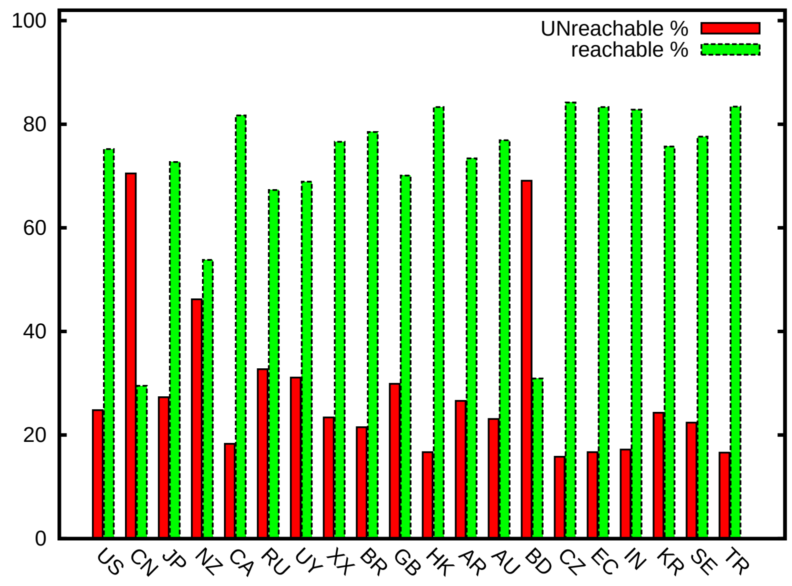


Figure 5.2: HTTP reachability test (PlanetLab) - percentage of web content downloading test that return HTTP code `200 OK` and content size greater than zero (green) and percentage of tests not returning any content.

5.1.2 Tor application censorship

One novel test we have implemented is the blocking of Tor application. It is an integrated test in the sense that its outcome depends on the execution of several basic tests, towards multiple targets. The definition of this test is mainly based on Winter [2013].

Collection: DNS resolution of

- Tor home page and mirrors
- Tor-based circumvention techniques and tools pages
- Tor overlay node list webpage

Collection: HTTP GET of

- Tor home page and mirrors
- Tor-based circumvention techniques and tools pages
- Tor overlay nodes list webpage
- Validated Tor overlay nodes list from Directory Authorities

Collection: TCP reachability of

- Tor nodes (relays)

Node	Consensus OK	ping ok	ping FAIL
pl2.6test.edu.cn	0	0	10
pl2.pku.edu.cn	0	0	10
pl2.zju.edu.cn	0	0	10
planet1.dsp.ac.cn	0	0	10
planet2.dsp.ac.cn	0	0	10
planetlab1.buaa.edu.cn	0	0	10
planetlab-1.sysu.edu.cn	0	0	10
planetlab2.cqupt.edu.cn	0	0	10
planetlab2.ustc.edu.cn	0	1	9
pln.zju.edu.cn	0	0	10
planetlab1.cqupt.edu.cn	0	0	10
pl1snu.koren.kr	1	0	9
csplanetlab4.kaist.ac.kr	9	1	0
planetlab1.eee.hku.hk	10	0	0
plab1.cs.ust.hk	9	1	0
pl02.comp.polyu.edu.hk	9	1	0
planetlab2.iitkgp.ac.in	10	0	0
planetlab1.iitkgp.ac.in	10	0	0
planetlab2.bgu.ac.il	10	0	0
planetlab-2.cmcl.cs.cmu.edu	9	1	0
planetlab6.csail.mit.edu	10	0	0
planet2.ku.edu.tr	9	1	0

Table 5.3: Tor censorship test: reachability of Directory Authorities.

- Tor Directory Authorities

The pseudocode describing the test is reported in Figure 5.3. The probe set we used for this test is the *PlanetLab* one.

In Table 5.4 the results of Relay reachability are reported: for an updated list of 851 relays a TCP connection is attempted to the IP address and port that are advertised on Directory Authorities. From the results it can be seen that probing hosts in China exhibit a much higher percentage of blocked relays with respect to other countries. A notable result can be seen for Korea, whose two available servers show opposite behavior: one (pl1snu.koren.kr) mostly sees blocked relays, while the other (csplanetlab4.kaist.ac.kr) has a ratio of blocked relays of 0.12 and thus analogous to non-censored case.

5.2 Censorship in Pakistan

In [ONI] country profiles Pakistan is classified as applying “selective filtering”, showing consistent level of censorship and tight control on Internet communications across the national border. The government body Pakistan Telecommunication Authority (PTA) is in charge of the management of the Pakistan Internet Exchange, the exchange point connecting the country to the rest of the Internet, and maintains a blacklist of URLs to be censored [Nabi, 2013]. According to the last report from [The OpenNet Initiative]

Node	TCP OK	TCP FAIL	fail ratio
pl2.6test.edu.cn	54	797	0.94
pl2.pku.edu.cn	47	804	0.94
pl2.zju.edu.cn	48	803	0.94
planet1.dsp.ac.cn	51	800	0.94
planet2.dsp.ac.cn	53	798	0.94
planetlab-1.sysu.edu.cn	50	801	0.94
planetlab2.cqupt.edu.cn	51	800	0.94
planetlab2.ustc.edu.cn	5	846	0.99
pln.zju.edu.cn	51	800	0.94
planetlab1.cqupt.edu.cn	48	803	0.94
pl1snu.koren.kr	47	804	0.94
csplanetlab4.kaist.ac.kr	752	99	0.12
planetlab1.eee.hku.hk	748	103	0.12
plab1.cs.ust.hk	746	105	0.12
pl02.comp.polyu.edu.hk	768	83	0.10
planetlab2.iitkgp.ac.in	748	103	0.12
planetlab1.iitkgp.ac.in	754	97	0.11
planetlab2.bgu.ac.il	749	102	0.12
planetlab-2.cmcl.cs.cmu.edu	769	82	0.10
planetlab6.csail.mit.edu	765	86	0.10
planet2.ku.edu.tr	747	104	0.12

Table 5.4: Tor censorship test: TCP reachability of Tor overlay nodes (relays).

Node	HTTP reachable	HTTP FAIL
pl2.6test.edu.cn	2	2
pl2.pku.edu.cn	2	2
pl2.zju.edu.cn	2	2
planet1.dsp.ac.cn	0	4
planet2.dsp.ac.cn	0	4
planetlab-1.sysu.edu.cn	2	2
planetlab2.cqupt.edu.cn	2	2
planetlab2.ustc.edu.cn	2	2
pln.zju.edu.cn	2	2
planetlab1.cqupt.edu.cn	2	2
pl1snu.koren.kr	1	3

Table 5.5: Tor censorship test: HTTP reachability of websites describing circumvention techniques and tools. Only probes that had unreachable relays (see Table 5.4) perform this test.

blocked resources belong to the classes: Religion, Sexuality and Politics.

5.2.1 Preliminary analysis

In the period between Sept 9th and 23th 2013, with the assistance of a local researcher in Pakistan, we have conducted a measurement campaign from a BISMark router located at the researcher's house. Measurements were performed at intervals of 6 hours, more precisely at 3AM, 9AM, 3PM, 9PM UTC, corresponding to 8AM, 2PM, 8PM, 2AM of the local timezone (UTC+5). The choice of the sampling period and the time of the day has been guided by possible diurnal patterns in network congestion. At roughly the same time, another BISMark router located in the USA was performing the same measurement on the same set of URLs.

DNS resolution analysis

On a total of 396 hostnames, 270 (68%) are identically resolved from inside PK and USA, thus excluding the occurrence of DNS-related censorship for the related resources. For the remaining resources the analysis has exploited results from the *content size* tests.

Content size analysis

Considering the size of the resource (webpage) that has been retrieved, and averaging on all measurements from within a country, we expect to find a significant difference between different countries if one of the two is censoring the content by means of a "blocking page" (see 2.3.7). For each URL u the average resource size per country $s_{u,PK} = \frac{\sum_{uinPK} size(u)}{|PK|}$ is calculated and divided by the corresponding size averaged on all the other countries (just USA in this case); see Figure 5.4 for the SQLite implementation of the database query. Indeed by considering the empirical CDF of such ratio (Figure 5.5) we can see that while most of URLs show a comparable average size, there is an interesting fraction of URLs whose size is much smaller in Pakistan with respect to US.

The empirical probability mass function distribution (Figure 5.6) shows clearly two modes: one centered in 1 and a smaller one close to zero. While the variability around 1 can be considered as expected differences in parts of the HTML code that are updated in the dynamic generation of the resource, the relatively big variations that lead to the mode close to zero hint to a different phenomenon, on which we will focus to find evidence of censorship.

URL	$< size_{PK} >$	$< size_{US} >$	Ratio
https://barenakedislam.wordpress.com	453.0	49095.63	0.01
http://www.internationalfreepresssociety.org	443.5	38085.32	0.01
http://ninjaproxy.com	342.45	14085.42	0.02
NinjaProxy.com	342.39	13154.06	0.03
http://www.similarsites.com	375.33	13701.44	0.03
http://www.youtube.com	4183.91	144177.2	0.03
http://www.freefacebookproxies.com/	9041.17	241485.33	0.04
http://friendlyatheist.com	7881.34	205294.23	0.04
http://www.loonwatch.com	2661.73	65075.19	0.04
http://www.sodahead.com	3575.67	73969.7	0.05
http://www.hotspotshield.com/	731.8	10789.91	0.07
http://face-of-muhammed.blogspot.com/	6208.7	85342.93	0.07
http://www.foxnews.com	4705.53	63425.26	0.07
http://www.buzzfeed.com	22097.93	287001.77	0.08
www.freefacebookproxies.com/	18245.93	233254.73	0.08
www.hotspotshield.com	870.1	10632.97	0.08
http://www.cagle.com/news/muhammad/	3594.5	40974.12	0.09
www.smugbox.com/facebook/login/stories?id=226	1883.93	21455.95	0.09
http://www.faithfreedom.org/Gallery/28.htm	1438.93	15423.32	0.09
http://www.turbohide.com/	896.91	8744.12	0.1
http://www.internationalfreepresssociety.org/	16541.03	156461.16	0.11
www.unblockbook.net	812.48	6348.47	0.13
http://www.thesecretninjaproxy.info/	469.79	3416.17	0.14
www.kproxy.com.	647.47	4694.55	0.14
www.kproxy.com	666.39	4618.71	0.14
http://www.unblock-facebook.net	840.26	5783.3	0.15
www.blockedsiteaccess.com	1271.46	7780.19	0.16
www.proxyserver.it	395.26	1055.39	0.37
http://www.proxyblog.cn/	5308.0	13474.44	0.39

Table 5.6: Selection of URLs whose content size ratio (size collected from PK divided size collected from US) is lower than the threshold (0.5); size is averaged for a given country, and measured in *Bytes*; URL path may be truncated for presentation constraints.

In order to differentiate between the two modes we choose a threshold of 0.5, falling amid the two. An excerpt of the URLs whose size ratio falls below this threshold (in total 56, of which 28 are *youtube* videos) are reported in table 5.6.

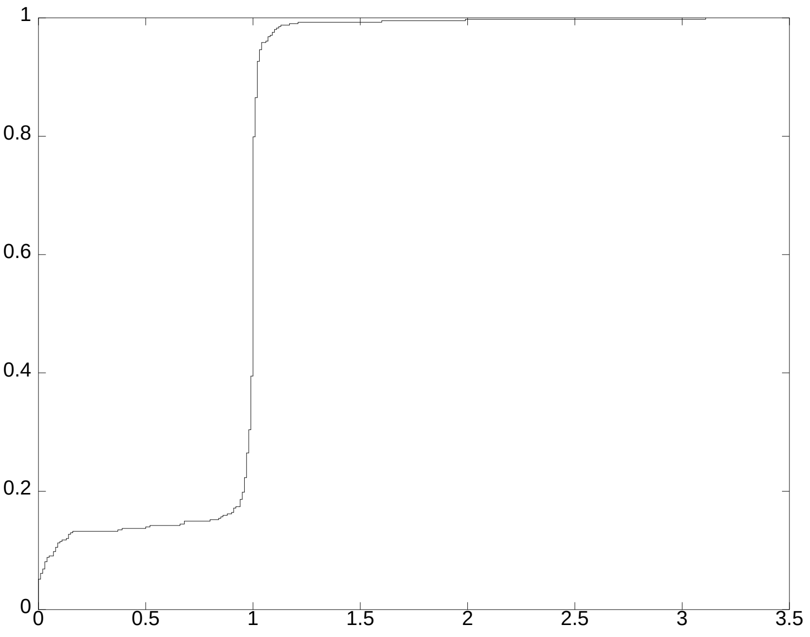


Figure 5.5: Empirical CDF of content size ratios (PK versus US) for each URL

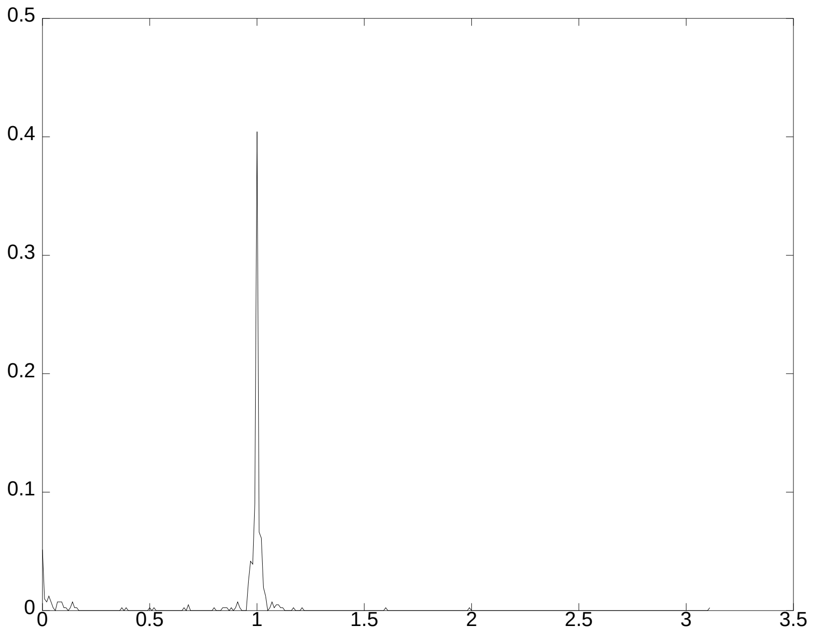


Figure 5.6: Empirical PDF content size ratios (PK versus US) for each URL, tested URLs are from Nabi [2013] (468 URLs)

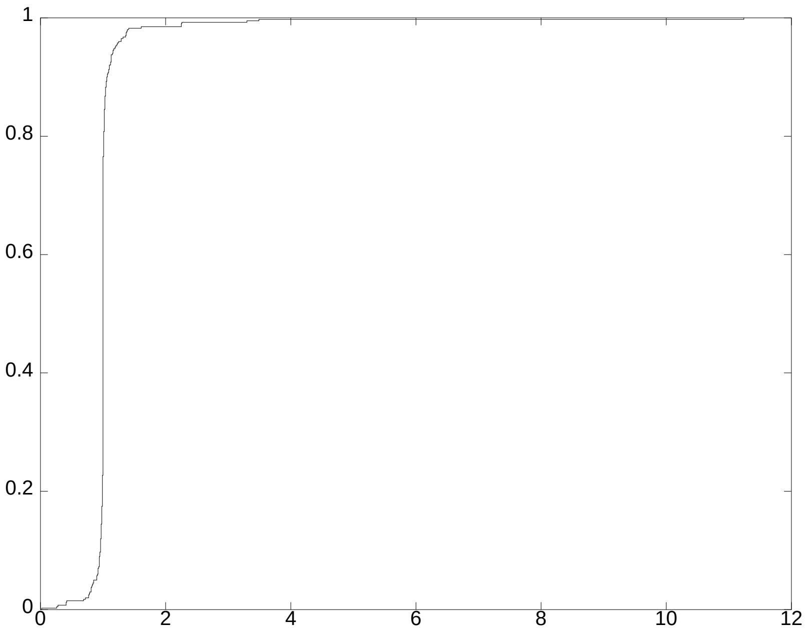


Figure 5.7: Empirical CDF of content size ratios (IT versus US) for each URL

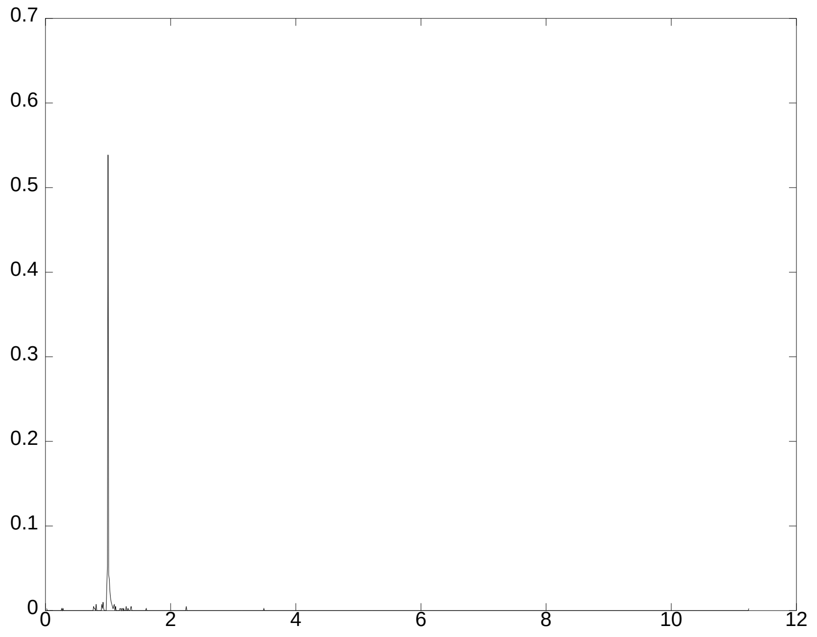


Figure 5.8: Empirical PDF content size ratios (PK versus US) for each URL, tested URLs the same checked in Pakistan

Taking one of the URLs selected through the *average content size ratio* test, namely *ninjaproxy.com* (accounting for 343*Bytes* in Pakistan and 14753*Bytes* from USA), by looking at the HTML code received by the client in Pakistan(Figure 5.9) it is possible to confirm that it is completely different from the one retrieved from outside Pakistan (Figure 5.10. Indeed censorship has been enacted providing a webpage with iframe redirection to a blocking page. These results are consistent with Nabi [2013] and the analysis in the report by [The Citizen Lab] on this country.

```
 1 <iframe
 2 src="http://202.125.134.154/webadmin/deny/?
 3 dpid=1
 4 &dpruleid=78
 5 &cat=104
 6 &ttl=0
 7 &groupname=PTCL2
 8 &policyname=PTCL2-policy
 9 &username=MMBB-22
10 &userip=XXX.XXX.XXX.XXX
11 &connectionip=127.0.0.1
12 &nsphostname=KHI494-NSP-04
13 &protocol=policyprocessor
14 &dplanguage=-
15 &url=http%3a%2f%2fwww%2eninjacloak%2ecom%2f"
16  width="100%" height="100%" frameborder=0>
17 </iframe>
```

Figure 5.9: HTML code of censored webpage (ninjaproxy.com) in PK; *newlines* have been added for clarity, the code consists of a single line; at line 10 the retrieved HTML shows the actual IP address of the probe, that has been hidden in this display with "XXX".

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.
dtd">

<html>
<head>

<!-- stop obfuscation -->
<SCRIPT TYPE="text/javascript" SRC="http://ad.
xtendmedia.
com/st?ad_type=pop&ad_size=0x0&section=1525666&banned_po
p_types=28&pop_times=3&pop_frequency=3600&pub_url=${
PUB_URL}"></SCRIPT>

<script type='text/javascript'>
var adParams = {a: '11841072', rt:'generic',
serverdomain: 's.m2pub.com', closeButton: 'Left',
backgroundColor: 'transparent', size: '728x90'  ,
context:'c12310043'  };
</script>
<script type='text/javascript' src='http://creative.
m2pub.com/matomy/scripts/catfish/catfish.js'></script>
  <script type='text/javascript'>
  var myArray=new Array();
    myArray[0] = '%0n%0n%0n%0n%09<fpevcg
    ynathntr=%22WninFpevcg%22>%0n%09gel %7o%0n%09
    ine k = gbc.ybpngvba.ubfg;%0n%09%7q pngpu %28r%29
    %7o%0n%09    gbc.ybpngvba.ercynpr%28%22uggc://%22+
    ybpngvba.ubfg%29;%0n%09%7q%0n%09</fpevcg>%0n%0n%09<
    yvax ery=%22fubegphg vpba%22 uers=%22snivpba.vpb%22>
    %0n    <gvgyr>Avawn Pybnx %7p Snfg, serr, nabalzbhf
    jro oebjfvat jvgu AvawnPybnx.pbz</gvgyr>%0n    <
    fglyr glcr=%22grkg/pff%22>%0n    <!--%0n     obql
    %7o%0n            onpxtebhaq-pbybe: #000000;%0n
    pbybe: #ssssss;%0n            sbag-snzvyl: ireqnan,
    gnubzn, nevny, fnaf-frevs;%0n            sbag-fvmr:
    13ck;%0n      %7q%0n      n, n:npgvir, n:ivfvgrq%0n
    %7o%0n            pbybe: #ssO400;%0n      %7q%0n      .
    heyva %7o%0n            sbag-fvmr: 13ck;%0n           sbag-
    snzvyl: ireqnan, gnubzn, nevny;%0n            cnqqvat:
    3ck 3ck 4ck 5ck;%0n            obeqre: fbyvq 1ck
    #000000;%0n      %7q%0n      .heytb %7o%0n            sbag-
    jrvtug: obyq;%0n            sbag-snzvyl: ireqnan,
    gnubzn, nevny;%0n      %7q%0n      .fz %7o sbag-fvmr:
    11ck; %7q%0n      .kfz %7o sbag-fvmr: ';
```

Figure 5.10: HTML code of the original webpage (ninjaproxy.com) from US (fist 15 lines)

5.2.2 Censorship in Pakistan

Hereafter we report results from the automated analysis of data collected for 540 URLs over 5 different Pakistani ISPs. A selection of notable results is described.

The case of Youtube

An example of blocked URL showing interesting differences among ISPs is the streaming video platform - with content and comment sharing from users - *Youtube* (www.youtube.

com, integrated with the social network *google plus* and the search engine google).

An overall results report for the URL of a resource on the *Youtube* portal as tested from different ISPs in Pakistan is shown in form of a bar chart in Figure 5.11.

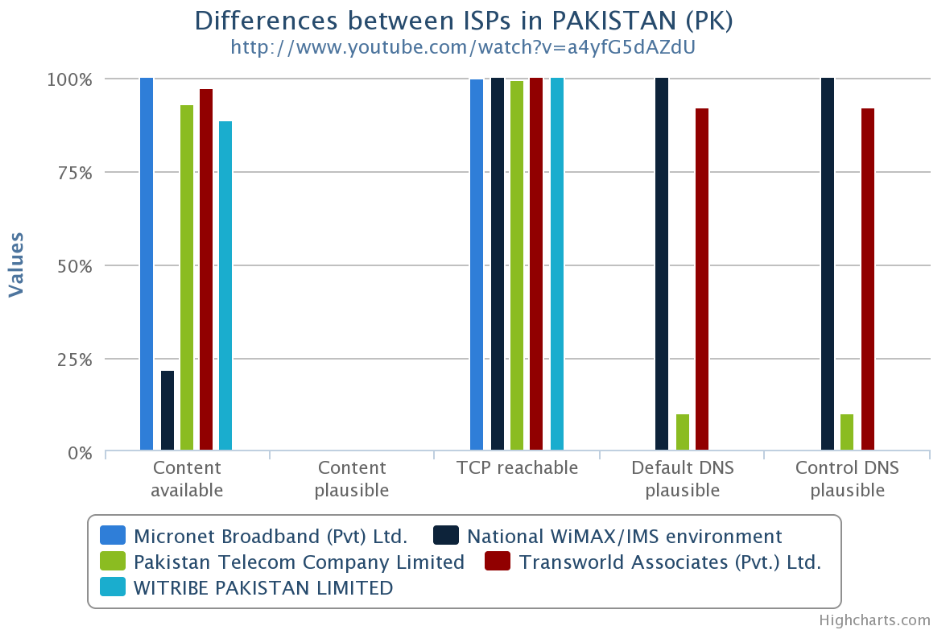


Figure 5.11: Censorship in Pakistan: Youtube - different censorship techniques

It is evident, from the lack of bars in the second aggregate (with label “Content plausible”) that this resource is never reachable, even though for all but one ISP a resource is returned when performing an HTTP request (first aggregate of bars, labeled “Content available”).

We recall that “Content plausible” is the percentage of URLs that passed the *size ratio* censorship test, and thus present a content size comparable to the average on all countries. The outcome of this test is represented in Figure 5.12 as a CDF of the ratio of the size of the downloaded content in one sample over the global average of such size. The CDF for Pakistan is shown (in dark blue) along with other countries for comparison: Italy (cyan) and U.S.A. (in green); the aggregation level is *country*, thus considering samples for the whole nations regardless of the ISP. The graph shows clearly that the size ratios in Pakistan are close to 0 (i.e. the content size is very small compared with the global average) with relative frequency 1 (always), while for both the other countries the occurrences fall close to 1 (thus same content size as the global average) with relative frequency greater than 0.9 (for U.S.A. 0.91 for a size 1.23 times the average, for Italy 0.95 for a size 1.11 times the average). Comparing with the size ratio threshold (set to 0.3 in the prototype test) we notice that the test has correctly separated results in Pakistan from the ones in the other countries. Moreover as the detected condition is above the coherence threshold the reported results are consistent over each country dataset.

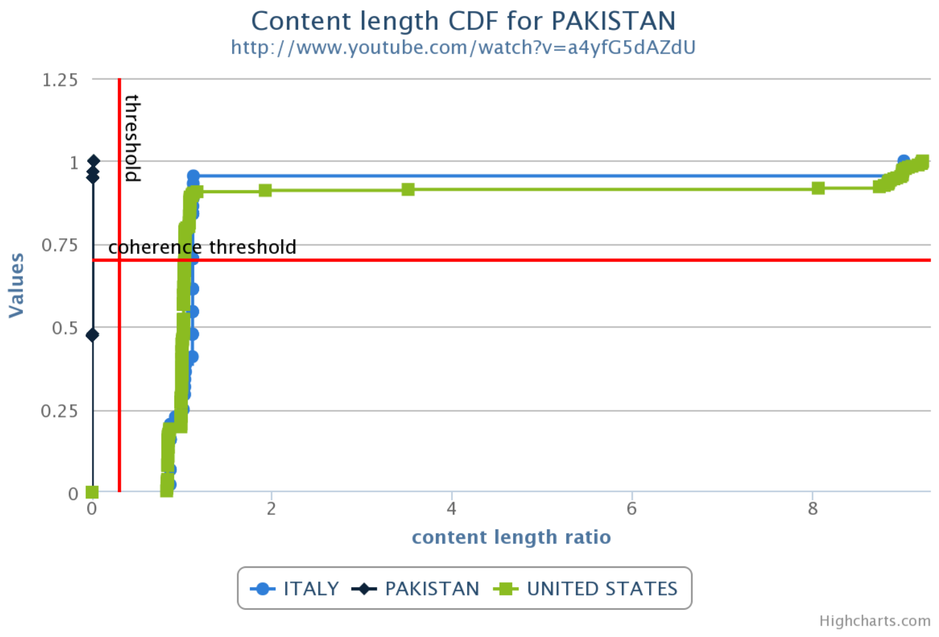


Figure 5.12: Censorship in Pakistan: Youtube - Empirical CDF of content size ratio; the abscissa is the ratio of a content size sample over the global average; the vertical threshold separates samples that fail the test (smaller than the threshold) from the ones that pass (greater); results are considered in the final verdict only if the coherence threshold (horizontal red line) is reached.

TCP-level tests (third aggregate, label "TCP reachable") show almost 100% reachability for all the ISP, thus either no censorship is enacted at this layer, or DNS tampering precedes it.

By considering the *default* DNS results for two ISPs "Micronet Broadband (Pvt) Ltd." and "Witribe Pakistan Ltd." no result yields a plausible IP address (i.e. neither a known block page or a failing IP, nor a DNS error), similarly for "Pakistan Telecom Company Ltd." only 11.7% is plausible. These ISPs clearly block the resource with *DNS tampering*. The DNS overall results show equal values for the *default* and the *control* resolvers, thus the inferred technique is DNS *injection*. The ISPs "Transworld Associates" (red in Figure 5.13) and "National Wi-Max/IMS" (dark blue) do not perform *DNS tampering* on the resource under analysis; yet for both the *content size ratio* analysis has detected censorship: an *HTTP tampering* technique has been applied.

To gather information regarding the *symptom* the user gets in the censored networks, we leverage the detailed DNS analysis, shown in Figure 5.13. It can be noted that, while two ISPs (namely, "Micronet Broadband (Pvt) Ltd." and "Witribe Pakistan Ltd.") both use DNS tampering to provide the user with an explicit *block page*, the ISP "Pakistan Telecom Company Ltd." returns an address that will likely cause an error (either at TCP-level or an HTTP-404), thus confounding the customer without providing explicit notification of censorship.

From the comparison between the summarized view (Figure 5.11) and the DNS analysis details (Figure 5.13) the behavior of one ISP ("Pakistan Telecom Company Ltd.", in red) seems inconsistent with the expected *symptom*, as the detected technique ("DNS injection - failing IP") should have elicited an error, and not the high percentages found both in *TCP reachable* and *Content available* bars (5.11). By inspecting the collected evidence data it resulted that the IP address returned by the ISP under analysis is `127.0.0.1`, corresponding to `localhost`, i.e. for each machine is the address of the machine itself (network level loopback); while other "specialized" network address ranges [Cotton and Vegoda, 2010] are unlikely to be assigned to active hosts in the same LAN of the probe, `localhost` for sure is, and the outcome of a TCP connection to the port 80 and possibly an HTTP request depend on the presence of a service listening on that port, and the response the service will return, if present. The inspection confirms the verdict of the platform, that detected censorship and the actual technique *DNS injection* regardless of the misleading *symptoms* (no errors at any level of the stack - DNS, TCP, HTTP).

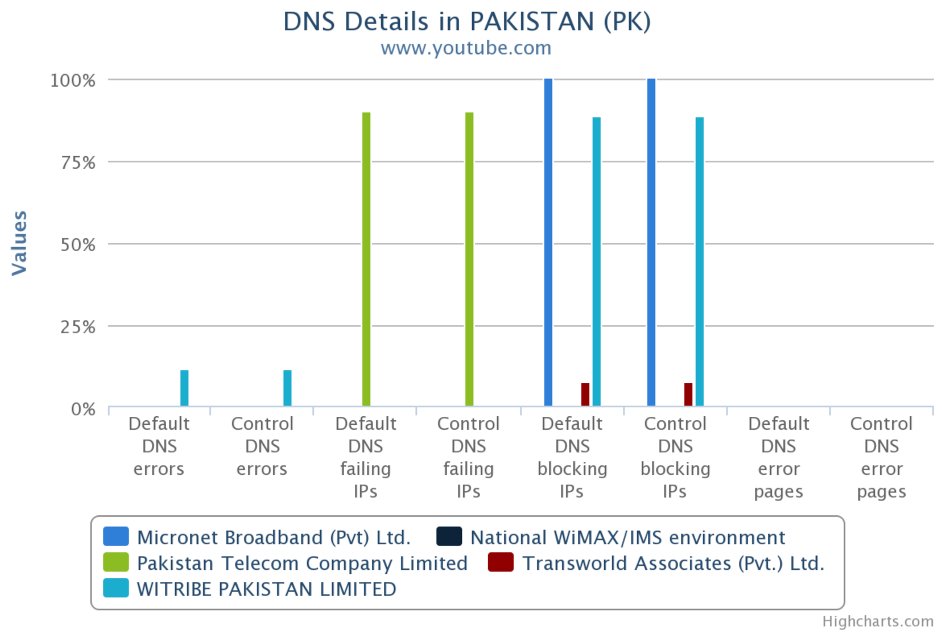


Figure 5.13: Censorship in Pakistan: Youtube - detail of DNS analysis.

5.3 Censorship in Italy

The measurement campaign performed for the validation of the platform have confirmed that Internet censorship in Italy is enforced mainly against websites proposing online gaming and betting and copyright infringement; another significant motivation for censorship is the block of child pornography, but due to ethical issues in potentially involving volunteers in police investigations the latter has been not tested.

From collected data it is evident that no centralized censoring infrastructure is present, as censoring is detected for different ISPs starting and ending at different times, and censoring techniques are sometimes different (in the vast majority DNS hijacking, and case-specific TCP blocking).

The Italian Agency for State Monopolies (AAMS) [1] provides an official list of domains[2] that have been blocked because of infringement of the Italian laws on online gaming and betting (that require a state license).

Another -but non-official - source is provided by an independent researcher in his

[1]Amministrazione Autonoma dei Monopoli di Stato, eee.aams.gov.it

[2]http://www.aams.gov.it/sites/aams2008/files/documenti_old/private/downloads/documentazione/scommesse/Elenco_siti_inibiti/elenco_siti_inibiti.rtf

"observatory on censorship" website[3] where a list of censored domains together with the authority that issued the censoring order and the date it was issued are reported.

The measurement campaign has been performed on 5 ISPs testing more than 840 target URLs, of which only 59 resulted not accessible or with a content differing from the one retrieved outside of Italy. In the case of blocks of websites proposing online gaming and betting the block is explicit (by means of a blockpage), while for websites related to *file sharing* the block is not motivated, resulting in a network error or a website describing a generic error.

The censoring technique used most across all the tested ISPs is DNS hijacking whose effect is graphically shown in Figure 5.14, in which DNS resolution requested to the probe default resolver is compared between probes from inside Italy (red lines) and USA (blue lines).

A few specific examples are described in the following.

5.3.1 Betting and gaming

The website bet365.com will be used as a representative of the *betting and gaming* website class.

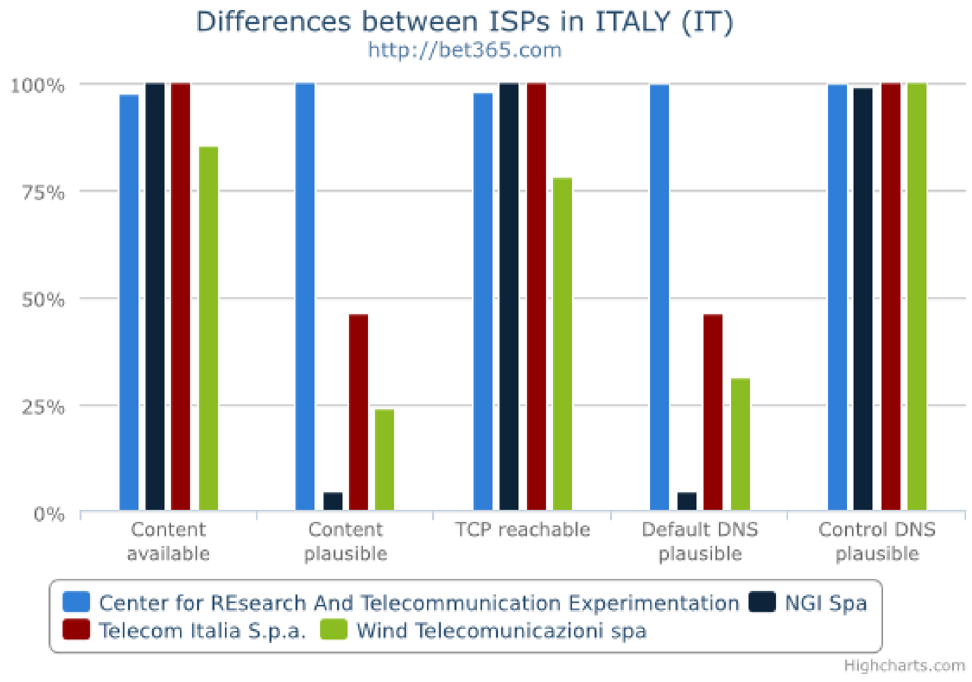


Figure 5.15: Censorship in Italy: gaming websites - censorship techniques

[3]http://censura.bofh.it/elenchi.html

The results of censorship analysis algorithms for the resource bet365.com is reported in Figure 5.15. We can see that for the ISP “NGI” the percentage of DNS resolutions performed the probe’s default resolver is as little as 4.5%. This is reflected by an analogous percentage of content of plausible size. From the same graph it can be seen that also for “Wind Telecomunicazioni” and “Telecom Italia” providers there are low percentage of plausible DNS resolution (31.2% and 46.1% respectively) and similar percentages of plausible content size (23.8% and 46.1% respectively). Only for the “Center for REsearch And Telecommunication Experimentation” ISP, that is user of the *GARR* [4], ISP both the DNS resolutions and the downloaded content size are always plausible, showing no censorship on its network for the considered resource. For the others, by comparing the result between the default DNS resolver and the control ones it can be inferred that the technique is of type DNS hijacking, as no control DNS is affected.

We can dig into deeper detail by inspecting the results of the DNS analysis, reported as a bar chart in Figure 5.16. Here we can see that for all the three ISPs censoring the resulting DNS response belongs to the list of known *blockpages*, thus the adopted censoring technique has the effect of presenting the user with a block webpage explicitly telling her of the censorship.

[4]The “GARR” is the Italian Academic and Research telecommunication network, a non-profit organization founded to manage an high speed infrastructure to connect to the Internet for the Italian academic, research and education communities. Website: http://www.garr.it

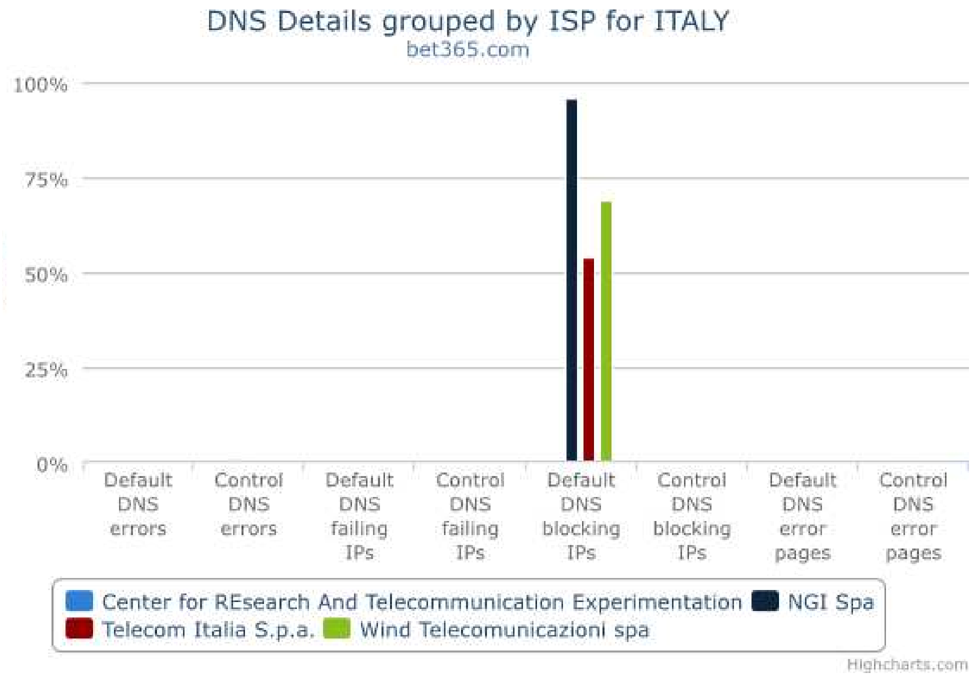


Figure 5.16: Censorship in Italy: gaming websites - detail of DNS analysis for bet365.com

From the Figure 5.16 it can also be noted that with the exception of "NGI Spa", with 95.4%, no ISP gives percentages close to the totality. The possible causes of this behavior can be: (i) a variability of the censor behavior in the analysis time interval (beginning or ending of censorship) (ii) heterogeneity of the probe environment at a granularity smaller than the *ISP* level.

The temporal evolution of the case under description is drawn in Figure 5.17: it can be seen that the oscillating results between reachability (upper line) and unreachability is limited to the *default* resolvers (the first two entries in the key, prepended with "DEF:"), while the *control* resolvers always report the domain as uncensored. It can be noted that the default DNS server address - as reported in the DNS reply - in the case under analysis corresponds to `localhost`: a local caching application such as `dnsmasq`[5] is in function on the probe system, preventing the collection of the local default resolver.

[5] `dnsmasq` is an open source DNS cache and forwarder, installed by default on several distribution of Linux, including OpenWRT and Ubuntu, main OSes for the UBICA probes. Website: http://www.thekelleys.org.uk/dnsmasq/doc.html

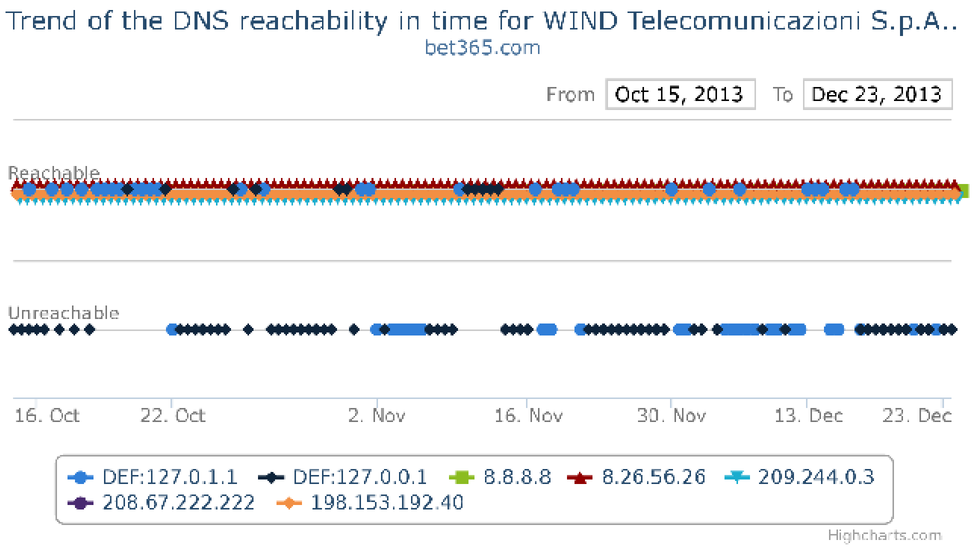


Figure 5.17: Censorship in Italy: gaming websites - DNS temporal analysis

5.3.2 Streaming and File Sharing

The second class of website censored in Italy is constituted by repositories and index directories for file sharing and multimedia streaming. As an example of this class we consider the index directory thepiratebay.sx. In Figure 5.18 the overall behavior in terms of adopted censorship techniques of different Italian ISPs can be seen. Besides the low percentages of *plausible DNS* responses for the default resolver, in this case low percentages are present also for *control* DNS servers. Moreover, differently from the case of betting websites, also the ISP connected through the Academic and Research network GARR presents low percentages (less than 50% for both default and control resolvers, and close to 40% of content availability). Other notable difference is in the result for TCP reachability: while in for the online betting website this measure scored close to 100% reachability for 3 out of 4 ISP (and more than 75% for the remaining one), in the case of the file sharing website 2 ISPs show less than 50% reachability at the TCP level.

A more in-depth inspection of the results of DNS measures, reported in Figure 5.19, shows a much more diverse scenario with respect to the case of betting websites (Figure 5.16).

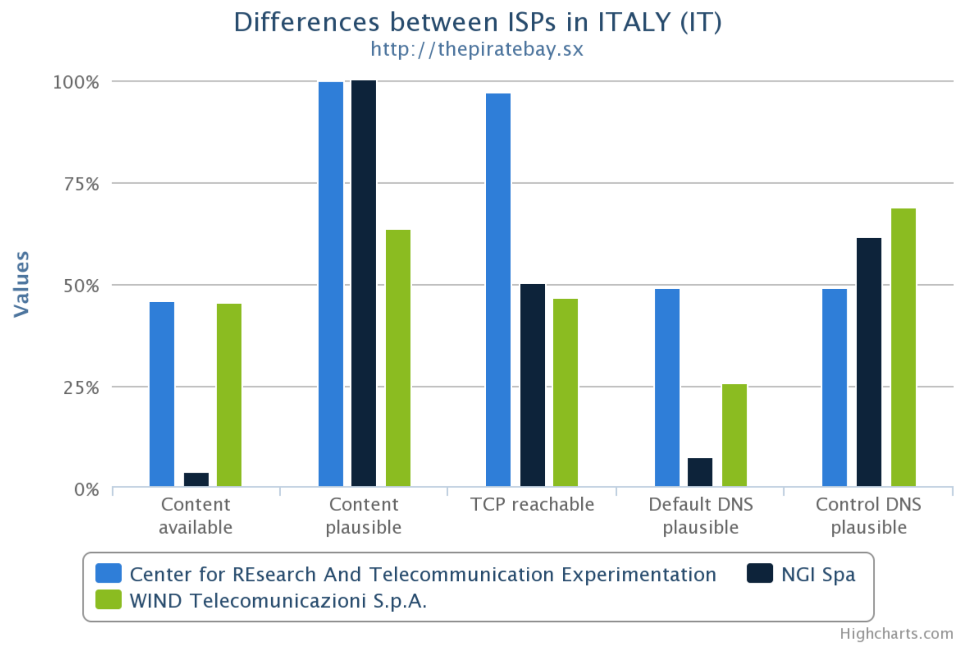


Figure 5.18: Censorship in Italy: file sharing websites - results of different techniques for thepiratebay.sx

All the ISPs show different DNS errors, both for default and control DNS servers. One ISP ("Wind Telecomunicazioni") for the default resolver shows a 65.5% responses returning a *failing IP*, namely `127.0.0.1`, and 7.7% of `NXDOMAIN` or `TIMEOUT` DNS errors. Different percentages of errors are shown by the other ISPs, each characterized by the presence of multiple symptoms of DNS unreachability in strong discordance with the case of betting websites (each ISP concentrated in one kind of DNS unreachability symptom).

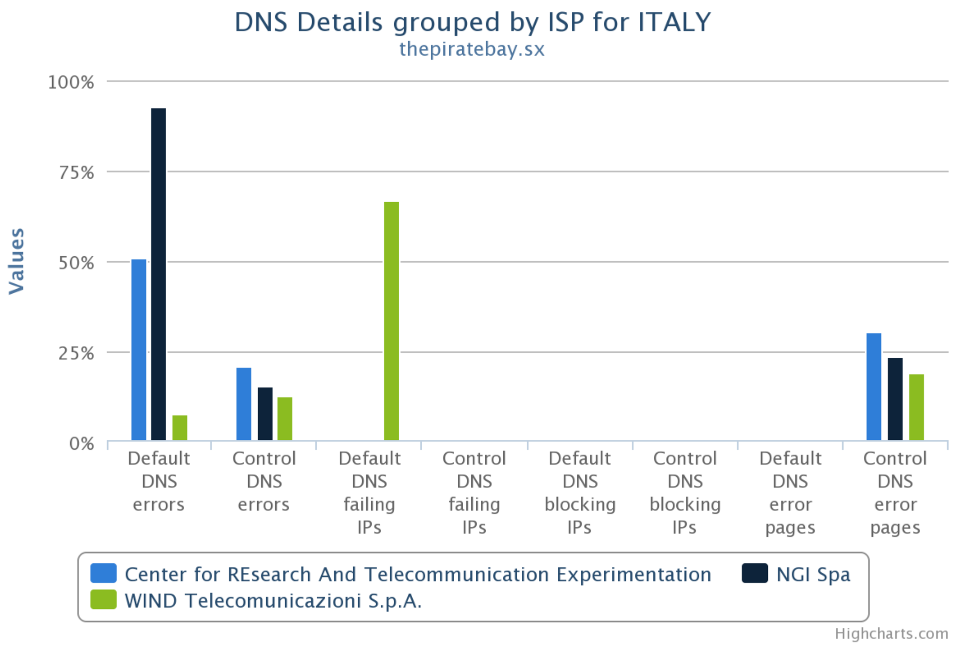


Figure 5.19: Censorship in Italy: file sharing websites - detail of DNS analysis for thepiratebay.sx

The temporal analysis of the DNS measures, represented in the time series of Figure 5.20, helps explaining such combination of results for the "Wind" ISP. In fact similarly to the case of betting websites (Figure 5.17) there is for the default resolvers an oscillation between reachability and unreachability spun over the first half of the timeline, again explainable with the lack of control over the default DNS set for the probe; in this case, however, all the resolvers no matter if default or control, report unreachability.

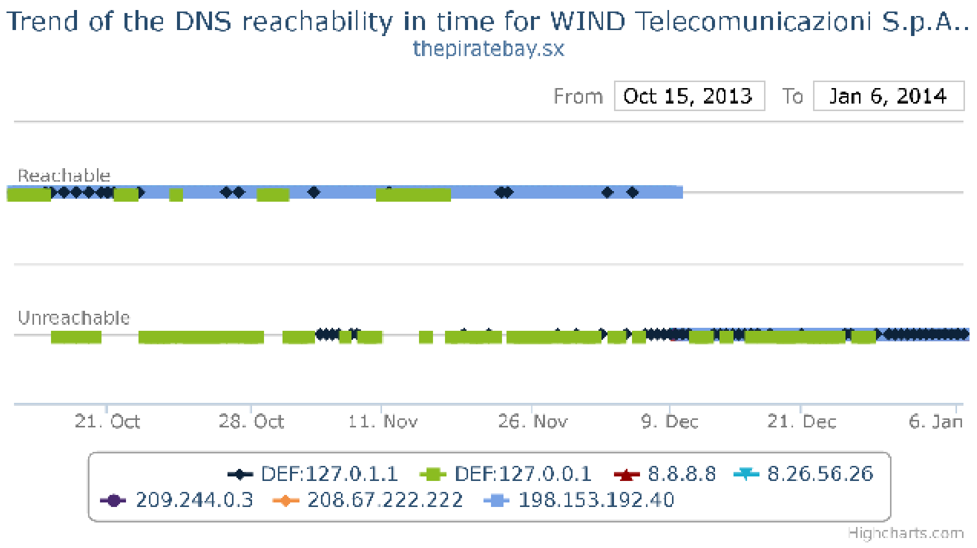


Figure 5.20: Censorship in Italy: file sharing websites - DNS temporal analysis for thepiratebay.sx and the "Wind" ISP.

The unreachability of thepiratebay.sx starting from December 10th 2013 is verified by the probes in *all* the countries, signaling a server-side event has occurred. From manual check of external information (the news section of the same website, freshly moved to another Top Level Domain: http://thepiratebay.se/blog/234) we can validate the finding of the UBICA platform: the old hostname has been dismissed on December 10th.

5.4 Censorship in Korea

The access to online content in South Korea is regulated by a government body, Korea Communications Standards Commission (KCSC) nominated by the president and in charge of the Ethics of Internet communications. The nation is reported by [ONI] as applying "selective filtering" for *Social* topics and "pervasive filtering" for the *Conflict/Security* category.

In this measurement campaign 384 *targets* have been tested, of which 26 have resulted in a censorship detection.

One single UBICA probe (of type *standalone script*) has been available for the testing, thus the results are not to be considered representative for the country, but again as an example of usage and a validation of the platform functionality.

Adult websites

A category of websites that is forbidden per order of the Ethical authority is the one showing adult content (classified among "obscenity and perversion"). The detection algorithm has signaled censorship for URLs such as hardsextube.com, pornhub.com and redtube.com, coherently with the expectations. We will consider the case of hardsextube.com in detail, as the other present analogous results.

By considering the summarized view for the different tested techniques, aggregated by country (Figure 5.21), it becomes evident the peculiar response in Korea with respect to the other tested countries. More specifically, the "content plausible" percentage of tests, result of the analysis based on the size ratio of the downloaded resource, is near 0% while other countries show near 100%, thus limiting to Korea only the issue in accessing the original content. Also no other censorship detection technique has been matched, thus excluding DNS Tampering and TCP-level filtering.

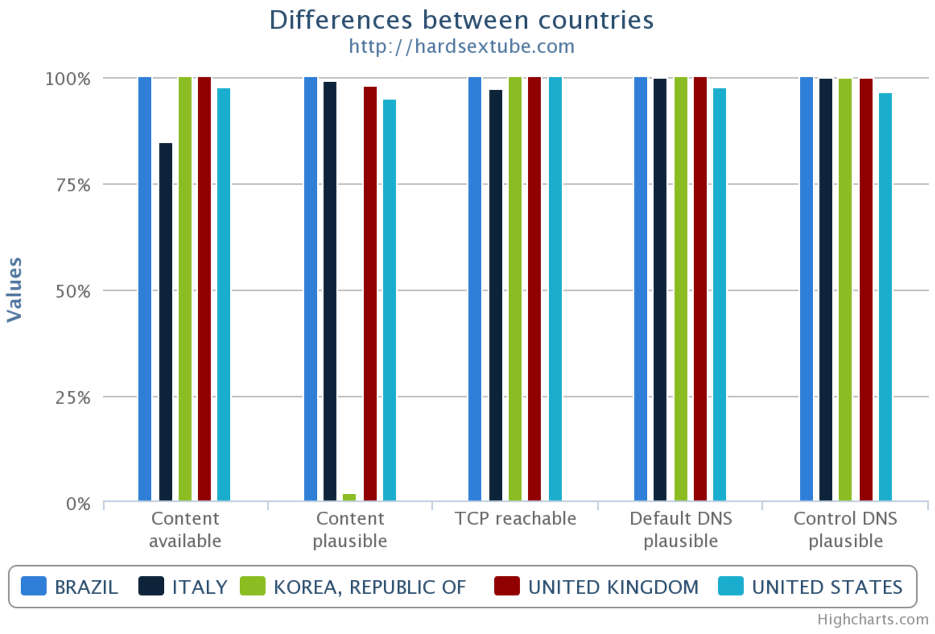


Figure 5.21: Censorship in Korea: porn websites - comparison with other countries, for different techniques.

To inspect in more detail the test that has detected censorship we refer to Figure 5.22, where the Empirical Cumulative Distribution Function is drawn of the ratio of each sample content size over the global average.

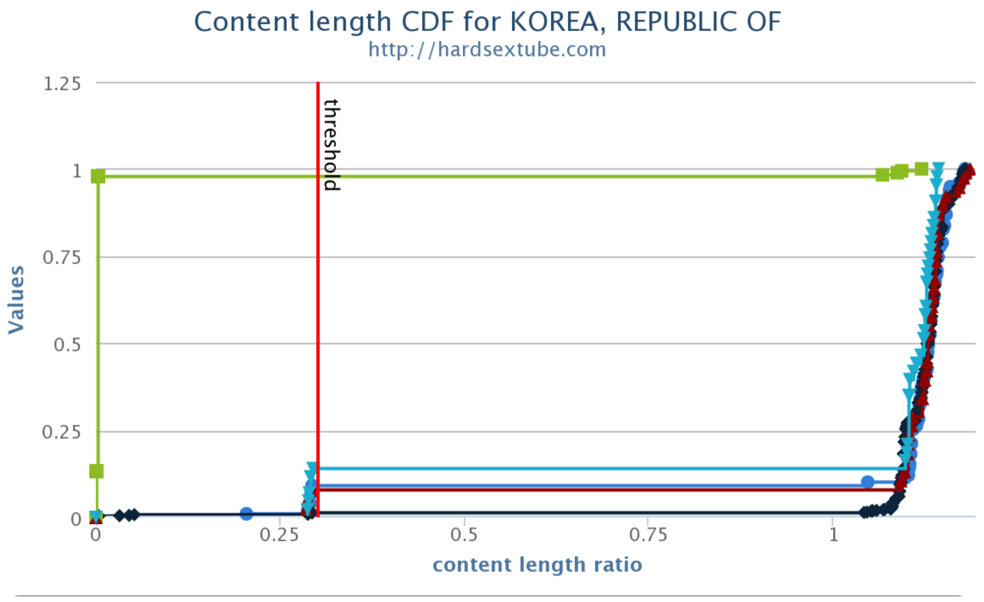


Figure 5.22: Censorship in Korea: porn - Empirical CDF of the content size ratio

It can be seen that only results for Korea (in green, close to the ordinates axis) are almost completely (0.98%) below the detection threshold (empirically set to 0.3), while all other countries have the almost totality of samples beyond 1.1, with the exception for U.K., U.S.A. and Brazil with small fraction (less than 0.16) falling just short of the threshold.

Even though these results would not have raised a censorship verdict due to the small relative percentage (pre-filtering data cleansing ignores cases that represent less than 70% of the results, see Section 4.4.1) we have manually checked the content and found that corresponds to mobile versions of the requested website. The detection algorithm based on the size ratio has proven robust to content adaptation [Md Fudzee and Abawajy, 2008] in this scenario but further research should be pursued in order to generalize this result.

In order to validate the censorship verdict, we have manually inspected the returned resource, whose content is shown in Figure 5.23. It can be seen that the returned webpage, result of the *HTTP tampering* technique (see Section 2.3.6), consists of a single JavaScript section whose effect when interpreted by the browser is to redirect (line 15, method `location.replace`) to the address specified at line 2: http://warning.or.kr, the official block page of the Korean authority for Internet censorship.

```
 1 <html><script>
 2 var arg = "http://warning.or.kr";
 3 var str = new Array();
 4 str = arg.split("&", 1);
 5 var a = new Array();
 6 a = str[0].split("=");
 7 var b = Math.floor(a[1] / 100);
 8 var c = new Array();
 9 if(b == 10){location.replace("http://www.naver.com");}
10 else if(b == 20){location.replace("http://www.daum.net")
11 ;}
12 else if(b == 30){location.replace("http://www.paran.
13 com");}
14 else{ c = a[0].split("?");
15 location.replace(c[0]);}
16 </script></html>
```

Figure 5.23: HTML code of censored webpage (hardsextube.com) in South Korea: it consists completely of a JavaScript code redirecting invariably to the official Korean block page whose address is in line 2.

5.5 Discussion

The realized prototype has proved not just functional in field tests, but actually helpful in investigating the phenomenon of censorship in its dynamism and effective in providing results about the actual enforcement of Internet censorship in several countries worldwide.

Thanks to the experimental evaluation, from the field test some shortcomings and limitations have surfaced that will be discussed hereafter.

5.5.1 Accounting for legitimate HTML-based redirection

In describing the collection of clues related with HTTP reachability (Section 4.3.2) we have highlighted that the sequence of HTTP requests does follow HTTP redirects (HTTP response code `30X`), while it does *not* process the returned content, thus stopping in case of HTML-based redirections. In case of censorship enforced by returning such a redirection in place of the requested content, the clue is the redirection code itself, that proves the censorship and its intentionality.

There are several legitimate cases of usage of HTML-based redirection, such as website migration, auto-refreshing dynamic data, splash screens, URL shortening services and varying degrees of customization of the content [Chellapilla and Maykov, 2007]. If the requested resource does legitimate use of HTML-based redirection then the actual target to be tested would be the resource to which the user would be redirected. In this case an user adopting a (JavaScript enabled-) browser would automatically try and get the linked resource, and censorship could be triggered or enforced on the latter. In the proposed design this scenario is modeled in terms of *target collection* phase: the retrieved HTML can be parsed in search of HTML redirections, that are added as new *targets*, leveraging the closed-loop control paradigm that exploits measurement results to inform the scheduling of subsequent measurements.

Strictly speaking, the monitoring platform is right in reporting that the original target resource is not censored. This procedure, though technically coherent, differs from what an user would experience if censorship is triggered or enforced on the landing page, and by checking the censorship report she could find the results counter-intuitive at best.

As one of the design goals of the platform is to provide an informative report to the citizen, this aspect has to be addressed in future work on the platform. A possible solution is, having implemented the HTML parsing in the *target collection* phase, to save the relationship between the original target resource and this secondary HTML-based-redirected resource and exploit this relationship to provide the user with results of censorship analysis that matches with her intuitive point of view.

5.5.2 Performance and scalability

The implemented prototype concentrated in a single server the roles of Management, Repository, Database, Analysis Engine and Reporting Servers. In case the load on the system were to become significant for the underlying hardware the different components of the platform could be instantiated on different machines networked - possibly on a LAN - in order to distribute the load (at the expense of transmission latency and bandwidth consumption).

This solution is not of much help if the Data Base Management System (DBMS) is the bottleneck, that could be the case when the parsing and insertion of report archives, the censorship analysis detection algorithms, and the queries for visual reporting of the results can happen concurrently.

Preliminary investigation has suggested that the parsing process has a significant impact on the computing resources: a possible solution could be the split of parsing processes over multiple distributed servers. A solution to be investigated is to split the parsing process onto multiple distributed servers. Possible drawbacks could then derive in the management of a distributed database, with related synchronization issues, and the execution of analysis algorithms on such distributed data sets. An approach better suited for this distributed scenario is provided by the "No-SQL" DBMS paradigm [Grolinger et al., 2013]; the analysis of the opportunity to move from a Relational model to this paradigm (or possibly the adoption of the “New-SQL” tools without changing the model) has been demanded to future work.

```
-- q_cc_stats_sqlite.sql
-- reachable means HTTP 200 and content size >0
-- probes are counted as different IPs
--   (todo: derive AS and ISP)

select
    r.cc as probeCC,
    p.probes ,
    r.count+u.count as tot_tests,
    r.count as test_reach,
    u.count as test_unreach,
    round ( 100 * ( cast(r.count as real) / (r.count+u.count)) ,1) as reachperc,
    round ( 100 * ( cast(u.count as real) / (r.count+u.count)) ,1) as unreachperc
from (
    select
        cc,
        count(*) as count
    from experiment
    where
        termination = 'normal'
        and contentsize is not null
        and httpcode = '200'
    group by cc
) as r
left join (
    select
        cc,
        count(*) as count
    from experiment
    where
        termination = 'normal'
        and contentsize is null
    group by cc
) as u on r.cc=u.cc
left join (
    select
        cc,
        count(distinct clientip) as probes
    from experiment
    where termination = 'normal'
    group by cc
) as p on r.cc=p.cc
order by probes desc
;
```

Figure 5.1: Presence of content analysis test: SQL code - basic analysis of *HTTP reachability* collection test.

```
IF ( DNS response for webpages ) <> ( ground truth )
OR
( content of all webpages is unreachable )
OR
( X% of Directory Authorities is not TCP-reachable )
OR
( Y% of relays is not TCP-reachable )
THEN Tor is censored
```

Figure 5.3: Pseudocode of the censorship test revealing the blocking of the Tor application. X and Y are threshold parameters to be defined empirically.

```sql
-- q_url_contentsize_PKvsUS.sql
-- given one CountryCode (opt: and a threshold ratio),
-- lists URLs whose average content size
-- divided by average of all other countries
-- is smaller than the threshold
SELECT
    pkavg.url AS URL
    , round(pkavg.avgsize,2) AS avgPK
    , round(outpkavg.avgsize,2) AS avgnotPK
    , round(pkavg.avgsize / outpkavg.avgsize,2) AS "PK vs all ratio"
FROM (
    SELECT
        url
        , avg(contentsize) AS avgsize
    FROM experiment
    WHERE
        cc="PK"
        AND contentsize > 0
    GROUP BY url
) AS pkavg
LEFT JOIN (
    SELECT
        url
        , avg(contentsize) AS avgsize
    FROM experiment
    WHERE
        cc<>"PK"
        AND contentsize > 0
    GROUP BY url
) AS outpkavg
ON    pkavg.url=outpkavg.url
ORDER BY     ( pkavg.avgsize / outpkavg.avgsize ) ASC
;
```

Figure 5.4: SQL code used to find possible censorship evidence according to average size ratio

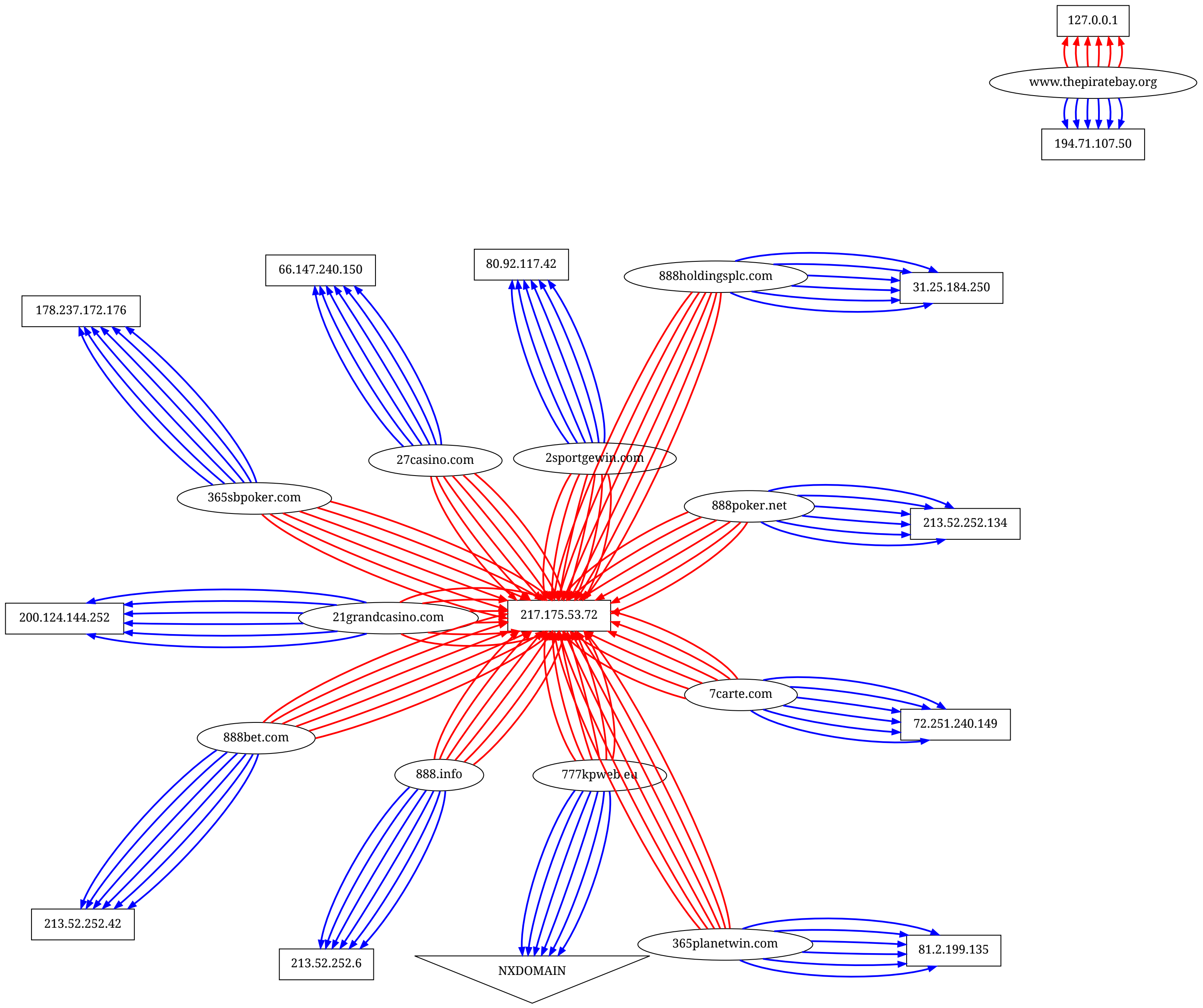


Figure 5.14: DNS hijacking in Italy: DNS resolution graph for betting websites. Ellipses contain host names, rectangles contain IP addresses, red lines are resolutions from inside Italy, using default (ISP) resolver, while blue lines are resolutions requested by a probe in USA to its default resolver.

Chapter 6

Conclusions

The awareness of the political importance of Internet as a mass medium has raised with examples such as the political campaign on Online Social Networks of the president B. Obama; the supposed role of blogging platforms during the "Arab Spring" of 2011, and -all differences considered- in Italy, with the recent trend of political communication through microblogging platforms. Governments worldwide, having acknowledged the Internet as an important channel for information, public discussion and organization of communities of interest, have exerted their control power on it. This has led to an arms race for the adoption of surveillance and censorship tools on a side, and privacy preserving and censorship circumvention tools on the other. As a consequence, different censorship techniques have been adopted over time in different countries worldwide. The actual extension of this phenomenon is not advertised by the censors, that thus become hardly accountable for it; hence the necessity of an independent and provable assessment of Internet censorship by its detection and continuous monitoring. These motivations have lead the research conducted by the candidate, described hereafter.

The available literature on Internet censorship has been found based mainly in network security (more focused on circumvention), while the techniques employed to enforce censorship derive also from traffic classification and traffic engineering. A selection of findings and studies have been analyzed adopting a network monitoring point of view, in order to extract the elements instrumental to the detection of network-based censorship.

Using web applications as a reference for client-server network applications on the Internet, a simplified model of the communication has been defined comprising the protocols involved (at the network, transport and application layers of the TCP/IP stack), the network topology, and the intermediate devices found on the path between client and server. Using the defined model, the censorship techniques have been characterized according to different elements, namely: the location of the surveillance device, the *trigger*, i.e. the element of the communication that elicits the activation of the censoring action; the localization of the *censoring* device, i.e. the component of the censoring system that applies the censoring action and the censoring action itself, i.e. the blocking or impairing of the access to the resource or service, or the mangling of the content of the communication. This has constituted an original contribution of the candidate, aimed at providing an unifying and comprehensive model of a complex phenomenon so far investigated in heterogeneous study fields.

With the same approach of analysis and with reference to the same model, the techniques and tools available for the *detection* of Internet censorship have been characterized, based on their ability to purposely generate or just receive traffic to elicit censorship, the types of *triggers* that said traffic can contain, the criteria to infer censorship and to ignore involuntary outages. The availability has been researched of *censorship monitoring platforms*, aimed at providing a quasi-real-time running report of the state of application of Internet censorship on a global scale and for year-long time scales. The properties such a platform should provide have also been analyzed and described, on the basis of the different approaches found in literature and in field usage.

From such analysis of the state of the art and definition of requirements, it has surfaced the lack of a full-fledged censorship monitoring platform performing a complete monitoring cycle, comprising the collection of the potential targets, the scheduling of the measurement experiments, the collection and analysis of the measurement results, and the publishing of the results of the analysis. In fact the only available platform that more closely matches the requirements, namely the [Herdict] project, has design properties that strongly limit the collection of evidences on which analysis should be based. Being based on user-submitted reports of reachability of web pages either chosen by the user or suggested by the platform, it is subject to limits such as: (i) report data prone to human error and possibly arbitrary judgment; (ii) choice of *targets* to be tested depends on users; (iii) frequency of testing depends on users active participation; (iv) availability of only one kind of basic reachability test; (v) little or no possibility to infer the censorship technique (vi) only raw and aggregated data are reported, with no described preprocessing (necessary to discard human errors and outages).

Two other proposals from the literature have provided a consistent coverage of the defined requirements for a censorship monitoring platform, but either are not publicly available [Sfakianakis et al., 2011] or are still partially developed and not ready for use [OONI], and both lack analysis and publishing functionalities.

In order to fill this gap and provide the research community with a platform to investigate the phenomenon of Internet Censorship for years-long periods on a global scale, and the citizen with an informative report on their own capabilities to access the Internet, the UBICA (User Based Internet Censorship Analysis) platform has been designed.

The platform is based on architectures for network performance measurement and monitoring [de Donato et al., 2014] and is composed by a management server, a limited number of helper servers and a heterogeneous set of clients.

The client has been designed to be highly portable, composed of a core measurement-related part (written in the Ash shell scripting language) and leverages standard UNIX utilities and mature network diagnostic tools. Wrapping the core measurement part there are different platform-specific implementations, including a GUI for desktop versions and a minimal command-line interface for headless versions. This design allows to execute the core in almost identical form on the different hardware and software platforms, including embedded home gateway routers [Sundaresan et al., 2012], a platform for distributed experiments system - PlanetLab [Chun et al., 2003] - and personal computers (both recent Windows and Linux-based operating systems).

The probes periodically ask the management server for experiments, if scheduled then update the measurement script, download the list of targets, perform the measurements and return the results to the server.

The management server performs a cyclic sequence of activities, including the collection of lists of possible censorship targets from external sources and by direct submission from users; the orchestration of the measurement activities performed by clients; the analysis of measurement data to (a) create the censorship reports and (b) update the scheduling of measurements.

The monitoring platform presents three different interfaces: a management interface for operators, a global censorship report view (public) and an user-centric view (for the volunteers hosting the probes). The management interface provides the most complete access to both the database and the experiment definition and scheduling; the public global view offers an aggregated view of the results of censorship analysis, at global and country level; the user-centric view provides the volunteers a quick report on the status of the censorship from their network viewpoint, offering an incentive for running the UBICA client.

A set of measurement tests have been defined and implemented to collect evidence of censorship, comprising DNS querying, TCP reachability, HTTP reachability, TLS reachability and verification, HTTP content size evaluation. These tests are based on censorship tests defined in literature, implemented leveraging the capabilities of the platform.

An algorithm for the detection of censorship and inference of the classes of censorship techniques from measurement data has been defined and applied to the results of a preliminary measurement campaign.

This algorithm constitutes an original contribution, together with the design of the platform, of the measurement tests and the reporting interfaces, and the characterization of the Internet censorship techniques based on the review of the literature.

A measurement campaign has been performed on a global scale with the help of volunteers running the probes and concurrently with the usage of the PlanetLab and BISMark [Sundaresan et al., 2012] measurement platforms. The preliminary results, validated by means of external sources of information, have verified the effectiveness of the platform and of the proposed censorship detection and analysis algorithm.

While the platform prototype has proved effective and useful in collecting evidence of censorship from a diverse set of countries and networks, it has shown possible limitations in the elaboration performance: a preliminary check suggests that the parsing and insertion in the database of the measurement results, running concurrently with censorship analysis algorithms, may suffer scalability problems. A solution to be investigated is to split the parsing process onto multiple distributed servers. Possible drawbacks could then derive in the management of a distributed database and the execution of the censorship detection algorithm on it. Such investigation has been left for future research.

The implemented measurement tests do not yet leverage the potential of the platform for an automatic tuning of testing frequencies: further research could investigate a robust scheduling algorithm optimizing the generated probe traffic and the frequency of tests.

Once UBICA had reached significant public visibility, a reaction could come from governments that would prefer not have their censorship activities exposed: possible threats for the platform functionality could go from blocking communications with the centralized management server (censoring UBICA) to whitelisting the UBICA clients, so that no censorship is detected. While the circumvention has been explicitly taken apart from the requirements for the platform, future research on this could investigate a distributed implementation of UBICA less prone to blocking. In case of whitelisting, it could be exploited the fact that users hosting an UBICA client would then get a more open access to the Internet.

Both data and the access to the platform will be provided to the scientific community for further analysis, fostering more extensive and diverse research from third parties, and to the public, to help keeping a watchful eye on the Internet Censorship.

Acknowledgments

I thank the tutor, prof. Giorgio Ventre, the former tutor prof. Giuseppe D'Elia, and the co-tutor prof. Antonio Pescapè for having accepted me as a student and for the support along the years of the doctoral course.

Special thanks go to prof. Pescapè, with whom a continuous research collaboration has endured that has had visible fruits in the publications I have co-authored with him and less visible fruits deriving from his mentor-by-example group leading style.

I wholeheartedly thank the colleagues of the *Traffic* research group lead by prof. Pescapè: Alessio - kind of a Senpai / elder colleague, though he doesn't look that old - and Walter - fellow (ab-)user of shell code automation and distributed code versioning systems. A thank you to Alberto, that is doing wonderfully abroad - it's a pity you couldn't find satisfaction here, but I'm glad to have had you as the other senior, as long as you were here. Thanks go to "hidden optimism" Pietro, also for the occasional chats planning new research - I hope soon - together and sharing the philosophical insights about writing a PhD Thesis - that we are enjoying in the same period. Thanks to the newly arrived in the group, Valerio, that kindly endures my tries to share with him all my hard earned wisdom in making bad tactical decisions: giving them a sense as a bad example for others to avoid makes me feel better, regardless of their -questionable- effective usefulness.

I would like to thank the Master students I have had the pleasure to assist - and in most cases fruitfully collaborate with - during my studies and researches: of a long list I'll thank just Luca E. and Giovanni F. as representatives for all, being the most recent graduates and the ones that most have participated in the implementation and testing of the prototype whose design I have proposed in my thesis.

I thank also Dr Saad and his students T. Ahmad and F. Awan for the assistance in testing some BISMark and UBICA client probes and the suggestion of possible probing targets.

During the two internships abroad in the U.S.A. I have had the opportunity to collaborate with expert supervisors and meet friendly colleagues: I wish to thank prof. Cedric Westphal (and his colleague prof. Onur Guleryuz) for their guidance, and the fellow interns Guillaume, Dyvia and Philipp, at DoCoMo Labs, Palo Alto, CA, and prof. Nick Feamster, his students Srikanth, Sam and Joon, and the other lab fellows [1] at Georgia

[1] A sad thought goes to Saamer, that tragically left his friends and the family on March 6, 2014. I had the opportunity to meet him at the lab, and a few parties, and his energy and vitality and clever humor make what happened to him even more incomprehensible.

Tech, Atlanta, GA. that have surrounded me with kindness, good humor and pleasant chats.

I wish to thank also people that made my staying abroad a really joyful experience instead of a burden: the community "Magic" in Palo Alto (specially the founders, and Chris), and the crew of house mates and friends - more or less in order of appearance in my journey: Talitha, Chen, Pablo, Janina, Rebecca, Thalya, Madeleine, Souhayl, Ibrahim, and some more. The worst part of the internship is to part from people that you've come to consider friends. The good part is to have met some of them again already!

The work environment at the Networked Systems Lab of the newly reborn (bigger than ever) Department of Electric and Information Technology Engineering has been a pleasant place to live research, and a constant healthy reminder that there exist a whole world of different highly specialized research topics just outside the boundaries of my interests and competencies, and they are as important, fruitful, challenging and rewarding as mine. At least for them, I mean.

Outside of the faculty, I have had little life in the last years. Except having met Pamela, that still is making me happy. So far so good. Besides her, most of the time of my non-lab life has been spent with Gianfilippo and Rocco, kind of two opposite edges in the spectrum of character and mood but both good friends to me: thanks.

Last (and first) comes my family, to whom goes my deepest gratitude. A good share of my accomplishments, big or small they may be, is because of you. Same goes for failures, naturally.

And you, reader, thanks for reading. Let's keep an eye on the freedom to be able to continue doing it.

Bibliography

Herdict. http://www.herdict.org. [Online; accessed 8-February-2014].

D. Eastlake 3rd. Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066 (Proposed Standard), January 2011. URL http://www.ietf.org/rfc/rfc6066.txt.

Giuseppe Aceto, Nick Feamster, and Antonio Pescapè. User-side approach for censorship detection: home-router and client-based platforms. In *Connaught Summer Institute on Monitoring Internet Openness and Rights*. University of Toronto, 2013.

Bernhard Ager, Wolfgang Mühlbauer, Georgios Smaragdakis, and Steve Uhlig. Comparing dns resolvers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 15–21. ACM, 2010.

E. Alexandrova. *Using New Media Effectively: An Analysis of Barack Obama's Election Campaign Aimed at Young Americans*. Fordham University, 2010. URL http://books.google.it/books?id=xygGcgAACAAJ.

Collin Anderson. Dimming the internet: Detecting throttling as a mechanism of censorship in iran, June 2013. URL http://arxiv.org/abs/1306.4361.

anonymous. The collateral damage of internet censorship by dns injection. *ACM SIGCOMM Computer Communication Review*, 42(3), 2012.

Simurgh Aryan, Homa Aryan, and J Alex Halderman. Internet censorship in iran: A first look. In *Presented as part of the 3rd USENIX Workshop on Free and Open Communications on the Internet*. USENIX, 2013.

D. Atkins and R. Austein. Threat Analysis of the Domain Name System (DNS). RFC 3833 (Informational), August 2004. URL http://www.ietf.org/rfc/rfc3833.txt.

Hitesh Ballani, Paul Francis, and Xinyang Zhang. A study of prefix hijacking and interception in the internet. *ACM SIGCOMM Computer Communication Review*, 37(4): 265–276, 2007.

D.E. Bambauer. Censorship v3.1. *Internet Computing, IEEE*, 17(3):26–33, May 2013. ISSN 1089-7801. doi: 10.1109/MIC.2013.23.

Constance Bitso, Ina Fourie, and Theo Bothma. Trends in transition from classical censorship to internet censorship: selected country overviews. *FAIFE Spotlight*, 2012. URL http://www.ifla.org/files/assets/faife/publications/spotlights/1%20Bitso_Fourie_BothmaTrendsInTransiton.pdf.

Sam Burnett and Nick Feamster. Making sense of internet censorship: a new frontier for internet measurement. *ACM SIGCOMM Computer Communication Review*, 43(3): 84–89, 2013.

Richard Carlson. Developing the Web100-based Network Diagnostic Tool (NDT). In *Passive and Active Measurement Workshop (PAM)*. Citeseer, 2003.

Rui Castro, Mark Coates, Gang Liang, Robert Nowak, and Bin Yu. Network tomography: Recent developments. *Statistical Science*, 19(3):499–517, Aug 2004. doi: 10.1214/088342304000000422. URL http://dx.doi.org/10.1214/088342304000000422.

Kumar Chellapilla and Alexey Maykov. A taxonomy of javascript redirection spam. In *Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web*, AIRWeb '07, pages 81–88, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-732-2. doi: 10.1145/1244408.1244423. URL http://doi.acm.org/10.1145/1244408.1244423.

Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.

Richard Clayton. Failures in a hybrid content blocking system. In *Privacy Enhancing Technologies*, pages 78–92. Springer, 2006.

Richard Clayton, StevenJ Murdoch, and RobertN Watson. Ignoring the great firewall of china. In George Danezis and Philippe Golle, editors, *Privacy Enhancing Technologies*, volume 4258 of *Lecture Notes in Computer Science*, chapter 2, pages 20–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-68790-0. doi: 10.1007/11957454_2. URL http://link.springer.com/chapter/10.1007/11957454_2.

M. Cotton and L. Vegoda. Special Use IPv4 Addresses. RFC 5735 (Best Current Practice), January 2010. URL http://www.ietf.org/rfc/rfc5735.txt. Obsoleted by RFC 6890, updated by RFC 6598.

Jedidiah R Crandall, Daniel Zinn, Michael Byrd, Earl T Barr, and Rich East. Conceptdoppler: a weather tracker for internet censorship. In *ACM Conference on Computer and Communications Security*, pages 352–365, 2007.

David Dagon, Manos Antonakakis, Paul Vixie, Tatuya Jinmei, and Wenke Lee. Increased dns forgery resistance through 0x20-bit encoding: security via leet queries. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 211–222. ACM, 2008.

Alberto Dainotti, Claudio Squarcella, Emile Aben, Kimberly C Claffy, Marco Chiesa, Michele Russo, and Antonio Pescapé. Analysis of country-wide internet outages caused by censorship. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 1–18. ACM, 2011.

Walter de Donato, Alessio Botta, and Antonio Pescapé. HoBBIT: a platform for monitoring broadband performance from the user network. In *Sixth International Workshop on Traffic Monitoring and Analysis (TMA) London, UK*, April 2014.

Ronald Deibert. *Access controlled: The shaping of power, rights, and rule in cyberspace.* MIT Press, 2010.

T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. URL http://www.ietf.org/rfc/rfc5246.txt. Updated by RFCs 5746, 5878, 6176.

Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

Maximillian Dornseif. Government mandated blocking of foreign web content, 2003. URL http://arxiv.org/abs/cs/0404005.

Peter Eckersley. How unique is your web browser? In *Privacy Enhancing Technologies*, pages 1–18. Springer, 2010.

T. Elahi and I. Goldberg. CORDON: A taxonomy of internet censorship resistance strategies. Technical report, Technical Report CACR 2012-33, University of Waterloo, 2012.

Guangchao C. Feng and Steve Z. Guo. Tracing the route of China's Internet censorship: An empirical study. *Telematics and Informatics*, (0), 2012. doi: 10.1016/j.tele.2012.09.002.

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. URL http://www.ietf.org/rfc/rfc2616.txt. Updated by RFCs 2817, 5785, 6266, 6585.

Arturo Filastò and Jacob Appelbaum. Ooni: Open observatory of network interference. In *Proc. 2nd USENIX Workshop on Free and Open Communications on the Internet (FOCI 2012)(August 2012)*, 2012.

GFC. The great firewall of china - greatfirewallofchina.org. http://www.greatfirewallofchina.org, 2013. [Online; accessed 16-December-2013].

Phillipa Gill, Masashi Crete-Nishihata, Jakub Dalek, Sharon Goldberg, Adam Senft, and Greg Wiseman. Characterizing censorship of web content worldwide. 2013.

F. Gont and S. Bellovin. Defending against Sequence Number Attacks. RFC 6528 (Proposed Standard), February 2012. URL http://www.ietf.org/rfc/rfc6528.txt.

Katarina Grolinger, Wilson A Higashino, Abhinav Tiwari, and Miriam AM Capretz. Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of Cloud Computing: Advances, Systems and Applications*, 2, 2013.

B. Hoehrmann. Scripting Media Types. RFC 4329 (Informational), April 2006. URL http://www.ietf.org/rfc/rfc4329.txt.

Ralph Holz, Lothar Braun, Nils Kammenhuber, and Georg Carle. The ssl landscape: A thorough analysis of the x.509 pki using active and passive measurements. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 427–444, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-1013-0. doi: 10.1145/2068816.2068856. URL http://doi.acm.org/10.1145/2068816.2068856.

Luca Invernizzi, Christopher Kruegel, and Giovanni Vigna. Message in a bottle: Sailing past censorship. *GSWC 2012*, page 3, 2012.

Mumm J. Localize or fail. *open tok Blog*, 2011.

Van Jacobson. Traceroute. ftp://ftp.ee.lbl.gov/traceroute.tar.gz, 1999.

Li Jun and S. Brooks. I-seismograph: Observing and measuring internet earthquakes. In *INFOCOM, 2011 Proceedings IEEE*, pages 2624–2632, April 2011. doi: 10.1109/INFCOM.2011.5935090.

Sheharbano Khattak, Mobin Javed, Philip D. Anderson, and Vern Paxson. Towards illuminating a censorship monitor's model to facilitate evasion. In *Presented as part of the 3rd USENIX Workshop on Free and Open Communications on the Internet*, Berkeley, CA, 2013. USENIX. URL https://www.usenix.org/conference/foci13/workshop-program/presentation/Khattak.

Jesse Kornblum. Identifying almost identical files using context triggered piecewise hashing. *Digital investigation*, 3:91–97, 2006.

Jim Kurose. Open issues and challenges in providing quality of service guarantees in high-speed networks. *ACM SIGCOMM Computer Communication Review*, 23(1):6–15, 1993.

M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. SOCKS Protocol Version 5. RFC 1928 (Proposed Standard), March 1996. URL http://www.ietf.org/rfc/rfc1928.txt.

John Leyden. Inside 'operation black tulip': Diginotar hack analysed. http://www.theregister.co.uk/2011/09/06/diginotar_audit_damning_fail/, 2011. [Online; accessed 21-February-2014].

Bingdong Li, Esra Erdin, Mehmet Hadi Gunes, George Bebis, and Todd Shipley. An overview of anonymity technology usage. *Computer Communications*, 36(12):1269–1283, 2013.

Wilson Lian, Eric Rescorla, Hovav Shacham, and Stefan Savage. Measuring the practical impact of dnssec deployment. In *Proceedings of USENIX Security*, 2013.

Matthew Luckie, Young Hyun, and Bradley Huffaker. Traceroute probe method and forward IP path inference. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 311–324. ACM, 2008.

Pietro Marchetta and Antonio Pescapé. DRAGO: Detecting, quantifying and locating hidden routers in traceroute IP paths. In *INFOCOM, 2013 Proceedings IEEE*, pages 3237–3242. IEEE, 2013.

Mohd Farhan Md Fudzee and Jemal Abawajy. A classification for content adaptation system. In *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, pages 426–429. ACM, 2008.

P. V. Mockapetris. Domain names - implementation and specification. RFC 1035 (Standard), November 1987a. URL http://www.ietf.org/rfc/rfc1035.txt.

P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034 (INTERNET STANDARD), November 1987b. URL http://www.ietf.org/rfc/rfc1034.txt. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936.

Milton L. Mueller. *Access contested: security, identity, and resistance in Asian cyberspace, ed. Ronald J. Deibert, John Palfrey, Rafal Rohozinski, and Jonathan Zittrain*, pages 177–194, 2012.

Steven J Murdoch and Ross Anderson. Tools and technology of internet filtering. *Access Denied: The Practice and Policy of Global Internet Filtering, ed. Ronald J. Deibert, John Palfrey, Rafal Rohozinski, and Jonathan Zittrain*, pages 57–72, 2008.

Zubair Nabi. The anatomy of web censorship in pakistan. In *Presented as part of the 3rd USENIX Workshop on Free and Open Communications on the Internet*, Berkeley, CA, 2013. USENIX. URL https://www.usenix.org/anatomy-web-censorship-pakistan.

Jong Chun Park and Jedidiah R Crandall. Empirical study of a national-scale distributed intrusion detection system: Backbone-level filtering of html responses in china. In *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pages 315–326. IEEE, 2010.

Mukaddim Pathan and Rajkumar Buyya. A taxonomy of CDNs. In *Content Delivery Networks*, pages 33–77. Springer, 2008.

Becker Polverini and William Pottenger. Using clustering to detect chinese censorware. In *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research*, page 30. ACM, 2011.

Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. RFC 1918 (Best Current Practice), February 1996. URL http://www.ietf.org/rfc/rfc1918.txt. Updated by RFC 6761.

Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006. URL http://www.ietf.org/rfc/rfc4271.txt. Updated by RFCs 6286, 6608, 6793.

E. Rescorla and N. Modadugu. Datagram Transport Layer Security. RFC 4347 (Proposed Standard), April 2006. URL http://www.ietf.org/rfc/rfc4347.txt. Obsoleted by RFC 6347, updated by RFC 5746.

F Baldi Risso, M Morandi, O Baldini, A Monclus, and P Lightweight. Payload-based traffic classification: An experimental evaluation. in proceeding of communications, 2008. icc'08. In *IEEE International Conference*, 2008.

David Robinson, Harlan Yu, and Anne An. Collateral Freedom: A snapshot of Chinese users circumventing censorship. Technical report, The Open Internet Tools Project, May 2013.

Keith W Ross and James F Kurose. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison-Wesley Longman Publishing Co., Inc., 1999.

A. Sfakianakis, E. Athanasopoulos, and S. Ioannidis. Censmon: A web censorship monitor. In *USENIX FOCI*, August 2011. URL https://www.usenix.org/legacy/events/foci11/tech/final_files/Sfakianakis.pdf.

Christopher Soghoian and Sid Stamm. Certified lies: Detecting and defeating government interception attacks against ssl (short paper). In *Financial Cryptography and Data Security*, pages 250–259. Springer, 2012.

Srikanth Sundaresan, Walter De Donato, Nick Feamster, Renata Teixeira, Sam Crawford, and Antonio Pescapè. Measuring home broadband performance. *Communications of the ACM*, 55(11):100–109, 2012.

Juan Pablo Timpanaro, Isabelle Chrisment, and Olivier Festor. A bird's eye view on the i2p anonymous file-sharing environment. In *Network and System Security*, pages 135–148. Springer, 2012.

John-Paul Verkamp and Minaxi Gupta. Inferring Mechanics of Web Censorship Around the World. In *Free and Open Communications on the Internet*, Bellevue, WA, USA, 2012. USENIX Association.
urlhttps://www.usenix.org/system/files/conference/foci12/foci12-final1.pdf.

Nart Villeneuve. Breaching trust: An analysis of surveillance and security practices on china's tom-skype platform. Technical report, Information Warfare Monitor / ONI Asia.

Paul Vixie. Refusing REFUSED. http://www.circleid.com/posts/20120111_refusing_refused_for_sopa_pipa/, 2012. [Online; accessed 20-January-2014].

Claire Voeux and Julien Pain. Going online in cuba. Technical report, Reporters Without Borders, October 2006.

W3C. Html 4.01 specification. http://www.w3.org/TR/html401/present/frames.html, 1999.

WASC. Url redirector abuse. http://projects.webappsec.org/w/page/13246981/URL%20Redirector%20Abuse, 2011.

Nicholas Weaver, Robin Sommer, and Vern Paxson. Detecting forged tcp reset packets. In *Network and Distributed System Security (NDSS) Symposium*, 2009.

J. Weil, V. Kuarsingh, C. Donley, C. Liljenstolpe, and M. Azinger. IANA-Reserved IPv4 Prefix for Shared Address Space. RFC 6598 (Best Current Practice), April 2012. URL http://www.ietf.org/rfc/rfc6598.txt.

Philipp Winter. Towards a censorship analyser for tor. In *Presented as part of the 3rd USENIX Workshop on Free and Open Communications on the Internet*. USENIX, 2013.

Xueyang Xu, Z. Morley Mao, and J. Alex Halderman. Internet censorship in china: Where does the filtering occur? In *Passive and Active Measurement*, pages 133–142. Springer, 2011.

Tao Zhu, David Phipps, Adam Pridgen, Jedidiah R. Crandall, and Dan S. Wallach. The velocity of censorship: High-Fidelity detection of microblog post deletions, March 2013. URL http://arxiv.org/abs/1303.0597.

Paul Zwillenberg, Dominic Field, and David Dean. Greasing the wheels of the Internet Economy. Technical report, Boston Consulting Group, Jan 2014.