THE CONTROL PLANE OF THE PORTOLAN INTERNET TOPOLOGY MEASUREMENT SYSTEM, BASED ON SMARTPHONE CROWDSOURCING: ARCHITECTURE, PROTOCOLS AND IMPLEMENTATION

Relatori:
Prof. Luciano Lenzini
Ing. Alessio Vecchio
Ing. Enrico Gregori

Candidato:
Valerio Luconi

Anno Accademico 2011/2012
to Alice, “je serais là, pour toujours, suspendu à tes lèvres”

to Mom
The Internet actually connects around two billion users through a global communication infrastructure, emerged from an early nucleus of academic and government networks and constantly evolving. An accurate description of its structure, at multiple levels of abstraction, is important for several purposes, such as designing routing protocols, detecting network failures, or planning Internet Service Providers (ISPs) business relationships. As ISPs operate as commercial entities, they are reluctant to publicly reveal their network structure. As a consequence, in the last 10-15 years a number of measurement systems aiming to discover the Internet structure and properties have been deployed. However, despite significant efforts, the Internet structure has not yet been fully discovered, as measurements were carried out following a top-down approach, from the core of the Internet down to its edges, not being able of detecting its peripheral structure. The PORTOLAN Internet Topology Measurement System suggests a new approach in measuring the Internet, based on smartphone crowdsourcing, which will allow the discovery of the Internet peripheral structure, still in large part invisible to current measurement systems: a new bottom-up and bottom-to-bottom measurement paradigm. This thesis contributions are the design and the implementation of a scalable and efficient control plane architecture for the PORTOLAN Internet Topology Measurement System. The proposed architecture provides an interface for specifying measurement tasks and coordinates the smartphones and the tasks execution performed by the smartphones themselves. Moreover, a measurement campaign to validate and evaluate the implemented framework has been conducted, and led to the discovery of previously unknown links at the AS-level of abstraction.
Acknowledgements

First and foremost, I am grateful to my supervisor Prof. Luciano Lenzini for giving me the opportunity of working on such an interesting and innovative subject as the Internet topology discovery, and for offering me constant support and guidance.

Deepest gratitude is also due to my other two supervisors, Ing. Alessio Vecchio and Ing. Enrico Gregori, without whose knowledge and assistance this study would not have been successful.

I would like to thank the network administrator of Registro.it Lorenzo Rossi for providing the technical support to carry out the testing phase of this thesis.

Special thanks also to Alessandro Improta and Luca Sani for their precious advises and for offering me a room and free food during the testing phase at CNR.

I wish to express love and gratitude to my mother, for her endless patience through the duration of all my studies, and to my whole family and friends for their understanding and love.

Last but not least, Alice, for always being at my side, for always advising me right and always supporting me when things are difficult. Without her nothing would ever have been possible.
Contents

1 Introduction ........................................... 1
   1.1 The Internet: origins and structure ................. 1
       1.1.1 A brief history ............................. 1
       1.1.2 The actual Internet ......................... 2
   1.2 Measuring the Internet ............................ 4
       1.2.1 Motivations .................................. 5
       1.2.2 Techniques .................................. 5
       1.2.3 Limitations .................................. 8
   1.3 The Portolan Internet Topology Measurement System . 8
   1.4 Objectives and contents of this thesis .............. 10

2 Measurement methods .................................. 12
   2.1 Active measurements ............................... 12
       2.1.1 Traceroute .................................. 13
       2.1.2 Traceroute limitations ....................... 14
       2.1.3 Paris Traceroute and Multipath Detection Algorithm (MDA) . 17
       2.1.4 Monothonic ID-based Alias Resolution (MIDAR) .... 17
       2.1.5 Research projects ............................ 19
   2.2 Passive measurements ............................. 21
       2.2.1 Border Gateway Protocol (BGP) routing informations 21
       2.2.2 Routing registry informations .................. 22
       2.2.3 Research projects ............................ 23

3 System design ......................................... 25
   3.1 Portolan measurement system: a bottom-up approach ... 25
   3.2 System overview ................................... 27
CONTENTS

3.3 System’s interface ............................................. 27
  3.3.1 Sensor Observation Service (SOS) ....................... 28
  3.3.2 Sensor Planning Service (SPS) .......................... 29
3.4 Task specification ............................................. 30
  3.4.1 Operation .................................................. 31
  3.4.2 Source identification .................................. 31
  3.4.3 Destination identification ............................. 32
  3.4.4 Duration .................................................. 33
  3.4.5 Operation specific parameters ......................... 33
  3.4.6 Urgent flag .............................................. 35
  3.4.7 Task examples .......................................... 36
  3.4.8 Task translation: from task to microtask ............ 37
3.5 System architecture ........................................... 39
  3.5.1 Limitations .............................................. 40
  3.5.2 Portolan architecture .................................. 41
  3.5.3 Enhancements ............................................ 44
  3.5.4 Examples of execution .................................. 47
4 Implementation ............................................... 53
  4.1 System components overview .............................. 53
  4.2 Sensor Planning Service ................................. 55
    4.2.1 Task Handler .......................................... 57
    4.2.2 Microtask Builder .................................... 59
    4.2.3 Task Sender .......................................... 59
    4.2.4 Microtask Queue ...................................... 60
    4.2.5 Mail Sender .......................................... 61
    4.2.6 SPS Plugin execution flows ......................... 61
  4.3 Proxy Assigner ............................................. 64
  4.4 Proxy .......................................................... 67
    4.4.1 ProxySPSServlet ...................................... 68
    4.4.2 ProxyMDServlet ....................................... 70
  4.5 Mobile application features ............................. 74
    4.5.1 Measurement failures ................................ 75
5 Experimentation ............................................... 76
  5.1 Experiment setup .......................................... 76
  5.2 Experiment results ......................................... 83
List of Figures

1.1 Internet multi-tier architecture ............................... 3
1.2 Inter-ISP connectivity example ............................... 4
1.3 IP interface level and router level topologies ............... 6
1.4 Three different levels of the Internet topology ............... 7

2.1 Standard traceroute example .................................. 13
2.2 Per-flow load balancing example .............................. 15
2.3 Per-packet load balancing example ............................ 16
2.4 Per-destination load balancing example ....................... 16
2.5 Two interfaces of the same router are considered as two different nodes in the IP-level topology graph ............... 18

3.1 Monitor placement ............................................. 26
3.2 OGC SWE framework architecture ............................ 29
3.3 Task translation ............................................... 38
3.4 Basic architecture ............................................ 39
3.5 Portolan architecture ......................................... 42
3.6 Proxy – mobile devices interaction, via GCM service ........ 45
3.7 Interaction between human user, SPS and proxy ............. 48
3.8 Interaction between mobile device, proxy assigner and proxy .... 49
3.9 Urgent task notification via GCM service ..................... 50
3.10 Poll rate change notification via GCM service ............... 51

4.1 SPS plugin architecture ....................................... 56
4.2 Microtask Queue structure .................................... 60
4.3 GetFeasibilityRequest execution flow ......................... 62
4.4 SubmitRequest execution flow ................................ 63
4.5 CancelRequest execution flow ................................ 65
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.6</td>
<td>Proxy Assigner architecture</td>
<td>66</td>
</tr>
<tr>
<td>4.7</td>
<td>Proxy architecture</td>
<td>67</td>
</tr>
<tr>
<td>4.8</td>
<td>ProxySPSServlet architecture</td>
<td>68</td>
</tr>
<tr>
<td>4.9</td>
<td>ProxyMDServlet architecture</td>
<td>71</td>
</tr>
<tr>
<td>5.1</td>
<td>Connection between two ASes via MIX IXP</td>
<td>81</td>
</tr>
<tr>
<td>5.2</td>
<td>Direct connection between two ASes</td>
<td>81</td>
</tr>
<tr>
<td>5.3</td>
<td>Different subnets within the same AS may be reached via different ASes</td>
<td>85</td>
</tr>
</tbody>
</table>
# List of Tables

3.1 Source identification section ........................................ 32
3.2 Destination identification section ................................. 33
3.3 Operation specific parameters section .............................. 35
3.4 Implicitly specified targets example ............................... 36
3.5 Explicitly specified targets example ............................... 37

4.1 Proxy Assigner responses ........................................... 67
4.2 ProxySPSServlet responses .......................................... 70
4.3 ProxyMDServlet responses ........................................... 73

5.1 Target Autonomous Systems ......................................... 80
5.2 Submitted task ....................................................... 82
5.3 ASes connected to AS2597 via MIX IXP ............................ 84
5.4 ASes directly connected to AS137 ................................ 86
5.5 ASes connected to AS137 via VSIX NAP ............................ 86
5.6 ASes connected to AS137 via NaMeX .............................. 86
5.7 ASes connected to AS137 via MIX ................................. 87
5.8 ASes reached via Level 3 (AS3356) ................................. 88
5.9 Isolario’s AS2597 links ............................................. 90
5.10 Previously unknown AS2597 connections ......................... 91
5.11 AS137 links in Isolario, CAIDA and DIMES datasets ........ 92
5.12 Previously unknown AS137 connections .......................... 93
Chapter 1

Introduction

1.1 The Internet: origins and structure

The Internet is a worldwide system of computer networks that serves billions of users worldwide. Initially formed by a nucleus of academic and government networks, it is nowadays a network of networks that consists of millions of private, public, academic, business, and government networks, of local to global scope, that are linked by a broad array of electronic, wireless and optical networking technologies.

1.1.1 A brief history

The origins of the Internet reach back to research of the 1960s, commissioned by the United States government in collaboration with private commercial interests, to build robust, fault-tolerant, and distributed computer networks. The first two nodes of that would become the ARPANET (Advanced Research Projects Agency Network), were interconnected in September 1969. Later on two more nodes were added and, by the end of 1969, four host computers were connected together into the early ARPANET. ARPANET was initially funded by the Advanced Research Projects Agency (ARPA, later DARPA) within the U.S. Department of Defense for use by its projects at universities and research laboratories in the U.S., and was the world’s first operational packet switching network and the progenitor of what was to become the global Internet.

The funding of a new U.S. backbone by the National Science Foundation in the 1980s, as well as private funding for other commercial backbones, led to
worldwide participation in the development of new networking technologies, and the merger of many networks.

Commercial Internet Service Providers (ISPs) began to emerge in the late 1980s and early 1990s, and in 1995 the removing of the last restrictions to carry commercial traffic led to the Internet commercialization.

Since the beginning of the 1990s, the Internet has undergone impressive growth, which can be appreciated in terms of the equipment, such as routers and links, that has been added, as well as in the numbers of users and the value of commerce that it supports. This growth led to the current Internet, which connects around two billion users (nearly a third of the Earth’s population) through a global communication infrastructure that consists of thousands of service providers of different business types such as regional or international transit providers, content providers, enterprise and academic networks, access providers, and content distribution networks.

Interested readers may find a detailed description of the Internet’s history in [1].

1.1.2 The actual Internet

The complex Internet infrastructure plays a crucial role in determining end-to-end experiences of the users. The core of the actual Internet is a multi-tier hierarchy of IP (Internet protocol [2]) transit providers. About ten tier-1 transit providers form a mutually interconnected clique with global presence. Regional tier-2 ISPs are customers of tier-1 transit providers, i.e., a regional ISP pays a tier-1 transit provider for universal Internet reachability. Residential and small-business access (i.e., tier-3) providers are typically customers of tier-2 transit providers. The Internet ecosystem also involves major traffic sources – such as content providers and enterprise customers – that are typically customers of tier-1 or tier-2 transit providers. The transit path from a content provider (or an enterprise customer) to a tier-3 ISP through the hierarchical Internet commonly traverses from two to four transit providers. Figure 1.1 summarizes the Internet multi-tier structure.

The Inter-ISP connectivity has important technical, social, and economic implications for Internet users. There exist two main ways for ISPs to interconnect. The most common one is IP transit where an ISP buys connectivity from a transit provider that becomes responsible for delivery of the ISP’s traffic to/from global Internet users. The second way to interconnect
is via a peering agreement, that is, a reciprocal relationship where two ISPs maintain a direct interconnection to exchange their own customers’ traffic. To peer, two ISPs have to physically connect their networks in one of the following two ways:

- Private peering: the two ISPs connect their networks directly with a physical link and split the costs of operating this link.

- Public peering: both ISPs connect a physical link to the same IXP (Internet eXchange Point). An IXP is a physical location that allows multiple ISPs to exchange their traffic through its switching infrastructure. ISP members of an IXP typically share the IXP operation costs. The primary role of an IXP is to keep local Internet traffic within the local infrastructure and reduce the costs associated with the local traffic delivery.

An Inter-ISP connectivity example is shown in figure 1.2, where the B and C ISPs are A customers, D and E are B customers and F is C customer.
Moreover, A is providerless while D, E and F are customerless. Finally, B and C share a private peering connection and E and F publicly peer via an IXP.

Further on, we will refer to Autonomous Systems (ASes) rather than to ISPs. As stated in [3, Section 3], an AS is a connected group of one or more IP prefixes run by one or more network operators which has a single and clearly defined routing policy. Theoretically, there is no one to one correspondence between ISPs and ASes, as an ISP might own more than one AS. However, for the purposes of our analysis we can refer to ASes and ISPs as if they were synonyms.

1.2 Measuring the Internet

As previously said, ISPs typically operate as commercial entities and are reluctant to publicly reveal their network structure and properties. Therefore, over the past decade the networking research community has shown a
CHAPTER 1. INTRODUCTION

growing interest in discovering and analyzing the Internet topology.

1.2.1 Motivations

There are several reasons for wanting to discover the Internet topology. The main ones are:

- Building a formal graph of the Internet for studying its characteristics, such as the average degree, the degree distribution, the clustering coefficient or the betweenness centrality. Such a graph may influence the engineering of future topologies, strategies for repair in the face of failures, and the understanding of fundamental properties of existing networks.

- Drawing a map of the Internet, which could be useful to monitor its connectivity.

- Helping network management, for example in deciding where to add new routers and in figuring out whether current hardware is correctly configured.

- Designing new protocols or improving efficiency and scalability of existing ones.

- Planning ISP business strategies.

- Detecting and correcting network anomalies, and containing the spread of viruses or worms.

As shown, gathering information about the Internet topology is a mandatory task for modeling the network but also for monitoring the network. Hence, a number of measurement infrastructures – also known as measurement systems – have been developed with the goal of discovering the Internet structure and properties indirectly.

1.2.2 Techniques

Measurement methods are typically classified as passive or active. Passive methods use observation points – also known as monitors or vantage points – to extract Internet reachability information from captured BGP (Border Gateway Protocol [4]) path announcements. Complementary to the passive
measurements, active methods generally refer to techniques that inject specially crafted probes into the Internet in order to infer its topology. The most commonly used active measurement method is traceroute. Further informations on passive and active measurement methods and their main advantages and drawbacks are provided in chapter 2.

The Internet topology may itself be seen at four different levels of abstraction:

- the IP interface level,
- the Router level,
- the Point-of-Presence (PoP) level,
- the Autonomous System (AS) level.

The first level, the IP interface level topology, considers IP interfaces of routers and end-systems, and it is usually obtained by using data collected with a probing tool such as traceroute.

The second level is the router level topology, which considers each router as a node in the topology graph. It can be obtained by aggregating IP interfaces through a technique called alias resolution. Figure 1.3 shows a IP interface level topology and the router level topology obtained after resolving aliases.

The third level, the PoP level topology, can be obtained by further aggregating the routers, or directly aggregating the IP interfaces that are identified
Figure 1.4: Three different levels of the Internet topology

as being geographically co-located (a PoP is a physical location that houses servers, routers and switches).

Finally, the fourth level is the AS *level* topology, which provides information about the connectivity of ASes. Typically this information is drawn from passive measurements, such as inter-domain routing information and address databases.

Figure 1.4 summarizes three levels of abstraction: the IP interface level, the router level, and the AS level. The black dots represent the IP interfaces of routers, the shaded shapes stand for the routers themselves and the blank clouds represent the ASes. The plain and dotted lines correspond to links. The IP interface level is illustrated by the black dots. The router level is obtained when all interfaces of a router are grouped in a single identifier. Finally, the AS level is obtained when we look only at ASes and the links between them.
1.2.3 Limitations

The common approach followed by previous research projects for discovering the Internet topology is a top-down approach: measurements are carried out from the Internet core down to the edges of the network. However, despite the significant measurement efforts, the Internet structure has not yet been fully discovered. Furthermore, either number or location of missing Internet links remains unknown, as highlighted by [5–7], and, finally, the provided data are not fully reliable.

The previous projects failed to provide correct Internet maps because of two main reasons:

- the measurement campaigns are carried out with a small number of dedicated observation points, which is three or four order of magnitude less than the Internet device population;
- these monitors are often statically located closer to the Internet core than to the Internet edges, which often prevents the monitors from observing the large part of the Internet near a vast majority of Internet users.

Thus, up to now, the quantity and quality of the data gathered by top-down methods have been insufficient.

These considerations led to the conception of the Portolan Internet Topology Measurement System, which proposes a new approach of Internet topology discovery, based on smartphone crowdsourcing.

1.3 The Portolan Internet Topology Measurement System

As described in section 1.2.3, the current measurement systems use a top-down approach in measuring the Internet, that is, from within the Internet down to its edges.

The Portolan Internet Topology Measurement System reverses the top-down approach by proposing an innovative bottom-up and bottom-to-bottom approach, that is, from the edges of the net up to its core or between end-users.
CHAPTER 1. INTRODUCTION

To remove many drawbacks of the current Internet measurement systems method, the Portolan system proposes a mobile, user-centric, bottom-up, energy efficient, and scalable active measurement framework. The key underlying idea of the Portolan system is to run measurement tools in the ever increasing number of mobile smartphones.

Connected to their wireless access networks, mobile monitors can probe the Internet from the periphery (which is the most difficult portion of the Internet to be detected by the existing measurement systems) up to the Internet core (vertical probing) and from end-user to end-user (horizontal probing).

Relying on a huge number of cooperative users distributed all over the world, the Portolan system will succeed in discovering the Internet periphery structure, and additionally in shedding light in the actual performance that end-users experience at the fringes of the Internet. This information is of paramount importance to end-users, researchers and ISPs.

The basic components of the Portolan Internet Topology Measurement System are a mobile application which runs on the mobile devices and a server infrastructure which is responsible of:

• managing the mobile monitors,
• managing and supervising the measurement campaigns,
• participating actively in measurement campaigns (e.g., running alias resolution algorithms),
• storing the collected measurements.

The Portolan system components may also be considered on two separate planes:

• an execution plane, which includes all the components employed for performing measurements and storing results,
• a control plane, which includes all the components employed for submitting the measurement campaigns, managing and coordinating the mobile monitors and managing and supervising the measurement campaigns.

In section 1.4 we will describe which of these components were designed, developed and implemented in this thesis.
1.4 Objectives and contents of this thesis

As shown in section 1.3, the Portolan Internet Topology Measurement System is made of two separate planes, the execution plane and the control plane.

The execution plane of the Portolan system has been implemented and successfully tested in [8, 9]. A mobile application for the Android framework which performs active measurements on the Internet and the server infrastructure component which stores collected data has already been developed.

Hence, this thesis main objectives are the design of a scalable and efficient control plane architecture, the implementation of the designed architecture and its testing.

Specifically, the designed and implemented control plane architecture is responsible for:

- providing to researchers a way to specify measurement campaigns,
- notifying researchers about campaigns completion,
- translating the measurement campaign into smaller jobs to be assigned to mobile devices,
- assigning measurement jobs to mobile devices,
- managing and coordinating mobile devices themselves.

The testing is carried out by performing an experimentation on a real network scenario, in order to demonstrate that the designed and implemented architecture succeeds in the previously specified tasks. Thus, the experimentation results are first validated by comparison with a ground truth and finally evaluated by confrontation with results achieved by other measurement systems.

The thesis is structured into five more chapters. Chapter 2 gives an overview on measurement methods and the main research projects which perform measurements to discover the Internet topology.

Chapter 3 describes the design process of the control plane architecture, which includes the identification and solution of the design issues, the integration with previously implemented components and of components developed by third party and, finally, the conception of a scalable and efficient
architecture for accomplishing all the control plane’s previously identified assignments.

Chapter 4 shows the implementation of every single component of the designed architecture, of their sub-components and of the communication protocols between them.

Chapter 5 describes the experiments carried out for testing the implemented architecture and the validation and evaluation of the achieved results. We will show that the control plane of the Portolan system succeeds in managing all the phases of a measurement campaign, in order to get results that agree with a well known topology. Moreover, we show that the discovered topology contains links previously unknown to other popular measurement systems.

Finally, chapter 6 concludes this thesis and gives an overview on some of the possible future developments.
Chapter 2

Measurement methods

In this chapter we provide an overview of the Internet measurements state of the art. As previously discussed in section 1.2.2, there are currently two Internet topology measurement methods: active and passive methods. Active measurement methods inject specific probes into the network in order to infer its topology. Passive measurement methods use monitors acting as observation points for gathering informations about Internet topology, for example by capturing BGP path announcements. The chapter is structured into two sections. In the first section we fully describe the active measurement methods and the most relevant projects that make use of them. Since the Portolan Internet Topology Measurement System makes use of active measurement methods, our attention will be mainly focused on them. Then, in the last section, we briefly describe passive measurement methods and their most important research projects.

2.1 Active measurements

The most commonly spread active measurement method is traceroute. Traceroute is a networking tool, created by Van Jacobson in 1989, which allows to discover the path a data packet takes to go from a source machine (the monitor) to a destination machine (the target). Traceroute allows the discovery of the topology at the IP interface level. To obtain the router level topology alias resolution techniques are to be used for merging together IP addresses owned by the same router.

In the next sections we show how traceroute and alias resolution techniques
work. Moreover, we give an overview on the most relevant projects making use of active measurements.

2.1.1 Traceroute

The standard version of traceroute works as follows. The monitor (source) sends multiple User Datagram Protocol (UDP [10]) probes into the network with increasing time-to-live (TTL) values. Each time a packet enters a router, the router decrements the TTL. When the TTL value is one, the router determines that the packet has consumed sufficient resources in the network, drops it, and informs the source of the packet by sending back an Internet Control Message Protocol (ICMP [11]) Time Exceeded message. By looking at the IP source address of the ICMP message, the monitor can learn one of the IP addresses of the router at which the probe packet stopped. When, eventually, a probe reaches the destination, the destination is supposed to reply with an ICMP Destination Unreachable message with the code Port Unreachable. This works if the UDP packet specifies a high, and presumably unused, port number, i.e., above 1024. An example of a traceroute execution is shown in figure 2.1.

The traceroute behaviour just explained is the ideal case. The real case presents some issues. Routers along the path might not have the ICMP protocol enabled, thus they might not reply to probes, or they could have some ICMP rate limiting activated. Therefore, timeout is used in order to
avoid waiting an infinite time for a ICMP reply. If timeout expires, then the router is said as *non-responding* or *anonymous*.

Another issue occurs when the destination is non-responding, for example as a consequence of firewall restrictions. In such a case it would be impossible to know that the destination was reached. In order to avoid inferring less path, an upper bound on the number of successive non-responding machines is used. The default setting in Van Jacobson traceroute is 30 hops total.

As previously described, standard traceroute uses UDP probes. However two variants exist, a ICMP based one and a Transport Control Protocol (TCP [12]) based one.

The ICMP based version sends ICMP *Echo Request* messages and the destination is supposed to reply with a ICMP *Echo Reply* message.

The TCP based variant aims to bypass most common firewall filters at the destination by sending TCP SYN packets, assuming that firewalls will permit inbound TCP packets to specific ports, such as port 80 (Hyper Text Transfer Protocol – HTTP [13]). The behavior of TCP traceroute for the intermediate routers is the same as in standard traceroute.

### 2.1.2 Traceroute limitations

Traceroute measurements have several key advantages, as providing the ability to infer the data paths that packets take when traversing the Internet. Because of their nature of active measurements, traceroutes can be designed to potentially cover every corner of the Internet given sufficient numbers of measurement monitors optimally spread in the Internet, for example in locations with distinct network views.

Despite their advantages, active measurements, and traceroute in particular, present the following main limitations.

A first limitation is the fact that traceroute probes only discover forward paths towards a given destination, since reverse paths may differ, as routing prefix-based policies can cause asymmetry. This asymmetry reflects also on Round-Trip Time (RTT) comparison between two consecutive hops, because different return paths from the two routers could exist.

A second limitation can occur when traceroute probes traverse an AS that internally makes use of Multi-Protocol Label Switching (MPLS [14]), which may hide the underlying topology, suspending the TTL mechanism used by traceroute. Routers using MPLS may be configured either to decrement the
TTL, as traceroute requires, or to ignore the TTL field because, the switched path of MPLS being configured to have no loops, the IP TTL is not needed. The MPLS specification, however, recommends the TTL to be decremented where possible. Note that the version of traceroute that makes use of ICMP extensions, can be aware of MPLS nodes along the path [15, 16].

A third and most important issue occurs when path towards a destination encounters a certain type of router called load balancer. Load balancers are routers configured to split traffic among different paths, according to pre-defined load balancing policies. There are three types of load balancing:

- *per-flow* load balancing,
- *per-packet* load balancing,
- *per-destination* load balancing.

*Per-flow* load balancers use flow informations in packet headers to split traffic among different paths. A flow is commonly identified by a *five-tuple*, which contains the following fields of IP and TCP/UDP packet headers:

- source IP address,
- destination IP address,
- source port,
- destination port,
- transport layer protocol type (TCP/UDP).
CHAPTER 2. MEASUREMENT METHODS

16

Figure 2.3: Per-packet load balancing example

Figure 2.4: Per-destination load balancing example

Additionally, also the Type of Service field in IP header, the ICMP Checksum field and the ICMP Code field may be used to identify a flow. An example of per-flow load balancing is shown in figure 2.2, where L is a per-flow load balancer.

Per-packet load balancers simply split traffic among different paths in order to keep a uniform load distribution on each outgoing interface. Figure 2.3 illustrates an example of per-packet load balancing.

Finally, per-destination load balancers use the destination IP address to distribute traffic among the outgoing interfaces. An example of per-destination load balancing is shown in figure 2.4.

Load balancing may lead to an incorrect resulting topology, where nodes and links could be missing, as well as false links could be introduced. In order to circumvent malfunctions due to load balancing, a new traceroute tool named Paris Traceroute [17] was introduced in 2006.
CHAPTER 2. MEASUREMENT METHODS

2.1.3 Paris Traceroute and Multipath Detection Algorithm (MDA)

Paris Traceroute is a new version of traceroute able to detect the presence of load balancers by implementing intelligent heuristics to recover from their effects. As classic traceroute, three versions are available: UDP, ICMP and TCP. Unlike classic traceroute, Paris Traceroute keeps constant the packets header fields that identify a flow, in order to recover from the effects of per-flow load balancers. Hence, the path discovered by Paris Traceroute is correct and not affected by anomalies due to load balancing. However, Paris Traceroute discovers only one of the available paths from a source towards a destination.

In order to discover all the available paths towards a destination, Paris Traceroute is enhanced by the Multipath Detection Algorithm (MDA [18]).

The Multipath Detection Algorithm is a stochastic algorithm that sends multiple probes at each hop, changing the flow identifying packet header fields for each probe, in order to discover the entire set of interfaces for that hop. As stated in [18], the complete set of IP interfaces for one hop can be discovered, with a 95% degree of confidence, by sending 96 probes (i.e., six probes for each interface after the first, as load balancers split traffic on at most sixteen different paths).

Moreover, MDA provides a mechanism for handling non-responding routers and for identifying per-packet load balancers.

For their characteristics of reliability, Paris Traceroute and MDA are the traceroute algorithms implemented within the Android mobile application used in the Portolan Internet Topology Measurement System [8,9].

2.1.4 Monothonic ID-based Alias Resolution (MIDAR)

The topology inferred by active measurements like traceroute is at the IP interface level. Thus, two different traceroutes could discover two IP interfaces of the same router, considering them as two separate nodes in the topology graph, as highlighted by figure 2.5. Hence, to aggregate the nodes corresponding to the same router (aliases) in order to obtain the router-level topology, several techniques (alias resolution techniques) have been developed.

The most recent and advanced algorithm of alias resolution is the Monothonic ID-based Alias Resolution (MIDAR [19]) algorithm, proposed in 2011.
The MIDAR algorithm is based on the IP ID field of IP packet header. The IP ID field is a 16 bit number involved in the identification and re-assembly of fragmented IP packets. The key idea of MIDAR relies upon two facts:

- routers themselves can generate packets (e.g., responses to ping or traceroute or by running BGP);
- the vast majority of routers share an IP ID counter between all of their interfaces.

Thus, if two addresses share a counter, then they are conclusively aliases,
and their IP ID time series, a sequence of IP ID values collected over time, will have similar values in a given measurement period and will form a monotonically increasing IP ID sequence when merged together.

MIDAR algorithm considers a sequence of IP ID values collected from the tested IP addresses, and executes a monotonicity test on them (Monotonic Bound Test – MTB). The IP ID values that result monotonically ordered in time are considered as generated by the same router and the interfaces originating that packets are aggregated into the same router.

MIDAR is the algorithm used by the Portolan Internet Topology Measurement System for alias resolution.

### 2.1.5 Research projects

In this section we present the most relevant projects of Internet topology discovery using active measurements: CAIDA Archipelago and DIMES.

#### CAIDA Archipelago

The Cooperative Association for Internet Data Analysis (CAIDA [20]) is a collaborative undertaking among organizations with a strong interest in keeping primary Internet capacity and usage efficiency in line with ever-increasing demand. Participants come from the commercial, government, and research sectors.

In 1998 CAIDA started the skitter¹ project, with the goal of tracking global IP level connectivity by sending probe packets from a set of source monitors to hundreds of thousands of destinations stratifying the current IPv4 address space as well as the Earth. Since 2008, skitter project evolved in the Archipelago Measurement Infrastructure (Ark)², which as of September 2012 counts of 61 measurement monitors probing the Internet with active measurements.

Measurements are performed using Scamper³, an active measurement tool supporting IPv4, IPv6, traceroute, and ping, developed by the WAND Network Research Group of the Computer Science Department of the University of Waikato, New Zealand. Scamper supports TCP-, UDP-, and ICMP-based measurements and Paris Traceroute variations.

---

³ [http://www.wand.net.nz/scamper](http://www.wand.net.nz/scamper)
CAIDA distributes the results of Ark measurements within four separate publicly available datasets:

- the **IPv4 Routed /24 Topology Dataset**\(^4\),
- the **IPv4 Routed /24 AS Links Dataset**\(^5\), which contains AS links derived from the IP paths of the Topology Dataset,
- the **Macrosopic Internet Topology Data Kit (ITDK)**\(^6\), which contains the router level topology,
- the **IPv6 Topology Dataset**\(^7\).

**DIMES**

The Distributed Internet Measurements and Simulations (DIMES [21, 22]) system is a measurement infrastructure that achieves a large scale by following the model of SETI@home [23] project. DIMES provides a publicly downloadable route tracing tool, released as a daemon in September 2004, which performs Internet measurements such as traceroute and ping at a low rate, consuming at peak 1KB/sec. The project obtains a portion of the computing power of the users’ computers, and in turn the users are rewarded by the knowledge that they are participating in a collective research effort, by attractive visualisations of the data, and by having their contributions publicly acknowledged. At the time of writing this thesis, DIMES counts more than 25000 agents scattered over the five continents.

DIMES distributes the results of its measurements within four datasets:

- **ASNodes** dataset, which is a set of found AS level nodes,
- **ASEdges** dataset, which is a set of AS level edges,
- **Nodes** dataset, which is a set of IP level nodes,
- **Edges** dataset, which is a set of IP level edges.

The datasets are publicly available at [http://www.netdimes.org/new/?q=node/65](http://www.netdimes.org/new/?q=node/65).

---

\(^4\)http://www.caida.org/data/active/ipv4_routed_topology_aslinks_dataset.xml
\(^5\)http://www.caida.org/data/active/ipv4_routed_topology_aslinks_dataset.xml
\(^6\)http://www.caida.org/data/active/internet-topology-data-kit/
\(^7\)http://www.caida.org/data/active/ipv6_allpref_topology_dataset.xml
2.2 Passive measurements

Passive measurement methods are commonly used to discover the AS-level topology of the Internet. There are two main sources of AS-level topology data: BGP routing informations and Internet Registries.

The next sections describe the two types of passive measurements with their pros and cons and the most popular active research projects which make use of them.

2.2.1 Border Gateway Protocol (BGP) routing informations

Unlike link-state protocols such as Open Shortest Path First (OSPF [24]), BGP does not maintain any unified view of the network. Each BGP router chooses its best path for a specific destination which is propagated to its neighbours, leading to an individual view of the network for each router. This distributed nature calls for the use of information gathering methods in order to obtain the most complete common view of the topology.

Common sources of BGP data are looking glasses and route servers. A looking glass is a web interface to a BGP router which usually allows BGP data querying and limited use of debugging tools such as ping and traceroute. A route server is a BGP router offering interactive login access permitting to run most non-privileged router commands. Both are usually made public to help network operators in their debugging tasks, but they can also provide BGP information to properly crafted network discovery tools. A list of looking glasses and route servers is maintained at [25].

A second source of BGP information is BGP dumps. Route collectors are deployed in various locations and peer with BGP routers from multiple ASes. They then periodically save snapshots of their state, known as table dumps, along with all routing updates received between the preceding and current snapshot, known as update traces.

We consider now advantages and drawbacks of using BGP routing information to discover AS level topology.

The first advantage is the fact that there is no need to deploy an infrastructure for exploring the network as data have been gathered and are available at specific places. Another advantage is that provided information by BGP corresponds to the actual state of the network, even though it only provides local views of it. Finally, BGP update traces allows dynamic
behavior analysis such as backup link detection.

However, using BGP routing information has several drawbacks. First, BGP does not provide complete information due to missing AS relationships that include both provider-customer and peering relationships. Further, unlike active measurement methods such as node-probing, the publicly available BGP paths do not cover the entire Internet due to issues such as visibility constraints, route aggregation, hidden sub-optimal paths and policy filtering.

### 2.2.2 Routing registry informations

There are two publicly-available registries that share information about the Internet and its topology: Regional Internet Registries (RIR) and the Internet Routing Registry (IRR).

**Regional Internet Registries (RIR)** are organizations responsible for allocating AS numbers and IP address blocks, all of which are accessible using the Whois [26] protocol.

The **Internet Routing Registry (IRR)** is another group of databases maintained by several organizations and containing documented routing policies available through the Whois protocol and expressed in the Routing Policy Specification Language (RPSL [27]).

The use of Internet registry information has some key advantages in the topology discovery. First, the implementation of the access is simpler and more efficient than the one of active method probing. Second, there is no need of exploring the network to obtain the topology, as the information is grouped at specific locations. Finally, they provide high-level information such as routing policies which are otherwise more difficult to obtain.

However, the information source has also several limitations, that are the reason why current work has tended to focus on other information sources for topology discovery at the AS level. First, the informations are often incomplete for various reasons such as confidentiality and administrative overhead. Second, registry data quality is questionable and often inconsistent: the same entry in one registry may overlap and sometimes even contradict information in other registries. Finally, these registries are not able to precisely reflect the actual state of routing in the network. For instance, one cannot determine whether portions of the internet are reachable or not, or whether backup links exist and are used.
2.2.3 Research projects

In this section the most popular research projects that make use of passive data gathering for Internet topology discovery are shown: RIPE NCC RIS, RouteViews and PCH.

RIPE NCC RIS

The Réseaux IP Européens (RIPE) is a collaborative forum open to all parties interested in wide area IP networks in Europe and beyond, which hosts the RIPE Network Coordination Center (RIPE NCC [28]), one of the five Regional Internet Registries (RIRs). RIPE NCC provides Internet resource allocations, registration services and coordination activities that support the operation of the Internet globally.

Among its projects RIPE NCC maintains the Routing Information Service (RIS)\(^8\), that collects and stores Internet routing data from 17 Remote Route Collectors (RRCs) located around the globe. Moreover, RIS offers tools that bring this data to the Internet community in various structured formats as well as raw data public access at https://www.ripe.net/data-tools/stats/ris/ris-raw-data.

RouteViews

The RouteViews [29] project is founded by the Advanced Network Technology Center (ANTC) at the University of Oregon. It was originally conceived as a tool for Internet operators to obtain real-time information about the global routing system from the perspectives of several different backbones and locations around the Internet.

While the RouteViews project was originally motivated by interest on the part of operators in determining how the global routing system viewed their prefixes and/or AS space, there have been many other projects that made use of the Route Views collected data to infer the AS-level topology of the Internet, for example several projects by CAIDA.

RouteViews allows Internet users to view global BGP routing information from the perspective of other locations around the Internet. Nowadays, RouteViews owns 10 route collectors that collect BGP routing informations by peering directly with other BGP routers, typically at large IXPs. The

\(^8\)http://www.ripe.net/data-tools/stats/ris
collected data is available in Multi-Threaded Routing Toolkit (MRT [30]) format at http://archive.routeviews.org/.

PCH

The Packet Clearing House (PCH [31]) is a non-profit research institute that supports operations and analysis in the areas of Internet traffic exchange, routing economics and global network development.

Since July 2010, PCH makes available on its website\(^9\) BGP data in MRT format collected by 51 route collectors, each located on a distinct IXP.

\(^9\)http://www-01.pch.net/resources/data/routing-tables/mrt-bgp-updates/
Chapter 3

System design

In this chapter we provide a comprehensive description of Portolan measurement system: a new distributed Internet measurement system based on traceroute, that aims at measuring the Internet through smartphone-based crowdsourcing [8]. The chapter is structured into five parts. Firstly, we discuss the approach that characterize the system, and show how it differs from previous works approaches. Secondly, we give an overview on the system and show which parts of it were already implemented or deployed before this work. Thirdly, we show system’s interface with human users. Fourthly, we describe in detail a measurement campaign structure and how it is translated into a set of smaller jobs that can be executed by mobile devices. Finally, we provide a detailed description of the system architecture designed in this thesis work.

3.1 Portolan measurement system: a bottom-up approach

Internet structure and topology can be measured in at least two ways:

- From within the Internet core, following a top-down approach;
- From the Internet edges, following a bottom-up approach.

Previous research works, such as CAIDA Archipelago described in section 2.1.5, RIPE NCC RIS and RouteViews described in section 2.2.3, and many others, followed a top-down approach, with a small number of monitors (compared to the cardinality of the Internet) placed in proximity of the
CHAPTER 3. SYSTEM DESIGN

Internet core. This kind of approach was not fully successful and a large part of Internet is still unknown to researchers, mainly because of the small number of observation points and of their not optimal placement strategy, as stated in [5–7].

On the other hand, Portolan measurement system follows a bottom-up approach, from the edge to the core of the Internet. Measurement tools are run from a large number of mobile smartphones, acting as monitors. This approach will allow to discover the Internet peripheral structure, which is in large part invisible to top-down measurement systems, and a significant number of the missing links, especially the peer-to-peer links between ASes. Consider the example in figure 3.1, where AS2 and AS3 are customers of AS1 and moreover share a peering link. If we place our monitors next to or into AS1 (top-down approach), we will not be able to find the peer-to-peer link between AS2 and AS3. With the bottom-up approach followed by Portolan system, monitors would be located next to or into AS2 or AS3 instead, allowing to discover the peer-to-peer link between AS2 and AS3.
3.2 System overview

The system is composed by a large number of mobile devices performing measurements, coordinated by a central unit which assigns tasks to every mobile device, collects results and performs some data processing, such as dealiasing of IP interfaces discovered by mobile devices. The central unit is also responsible for providing to human users an interface for submitting measurement campaigns of interest and translating them into smaller jobs which will be executed by mobile devices. What has already been implemented and developed in previous works [9] is a mobile application for Android system which runs a traceroute tool similar to Paris traceroute [17] using the MDA algorithm for multipath detection [18], and a special purpose server for collecting measurements and performing the dealiasing of detected IP interfaces with MIDAR (Monotonic ID-based Alias Resolution) [19] tool by CAIDA. Hence, the design issues addressed in this thesis work are:

- providing to human users an interface for submitting measurement campaigns,
- translating the submitted campaigns into smaller jobs to be performed by mobile devices,
- designing a central unit system architecture for coordinating and assigning jobs to mobile devices which satisfies requirements of efficiency and scalability.

In the next sections design solutions we found to these issues are described in detail.

3.3 System’s interface

A human user willing to retrieve data on the Internet structure can view the set of mobile devices performing measurements as a network of wireless mobile sensors, and the central unit as a base station. One can specify a measurement campaign and collect results as if she was querying the sensor network. Interfaces for tasking sensors and collecting results has been standardized by Open Geospacial Consortium (OGC) [32] in the Sensor Web Enablement (SWE) framework [33]. OGC’s SWE framework includes XML standards for:
• describing the process within sensors and observation processing systems (Sensor Model Language, SensorML [34]),

• encoding observations and measurements (Observation and Measurements, O&M [35]);

and XML web interfaces for:

• obtaining observations and sensor and platform descriptions from one or more sensors (Sensor Observation Service, SOS [36]),

• determining the feasibility of collecting data from one or more sensors or models and submitting collection requests (Sensor Planning Service, SPS [37]).

Thus, for providing a standard interface to human users we decided to use the SOS and SPS services specified by OGC and implemented by 52North [38], which is an open source initiative located in Münster, Germany whose main purpose is to foster innovation in the field of Geoinformatics through a collaborative research and development process. 52North is an open network of partners from research, industry and public administration, including the Institute for Geoinformatics [39], the International Institute for Geoinformation Science and Earth Observation [40] and many others. We now give a panoramic over the two services: SOS and SPS (in figure 3.2 is shown the basic architecture of a system based on the two services).

### 3.3.1 Sensor Observation Service (SOS)

The Sensor Observation Service aggregates readings from live, in-situ and remote sensors. The service provides an XML interface to make sensors and sensor data archives accessible via an interoperable web based interface. SOS is primarily designed to provide access to observations, which are modeled following the O&M standard. In Portolan system every mobile device registers at SOS for sending measurements data. SOS then stores data into a database. Each measurement is uniquely identified by mobile device unique identifier (that is, the IMEI for GSM and the MEID for CDMA phones). Measurement data are organized by SOS in Observation Offerings, which are groups of observations that cannot overlap for time or for subject with
CHAPTER 3. SYSTEM DESIGN

Figure 3.2: OGC SWE framework architecture

other groups. 52North’s SOS module was deployed and configured in a previous thesis work [9], to which interested readers may refer for an exhaustive description.

3.3.2 Sensor Planning Service (SPS)

The Sensor Planning Service provides an XML interface for handling and tasking sensors or sensor networks. Among others, SPS interface includes operations for:

- submitting tasks to sensors or sensor networks,
- checking feasibility of a task,
- cancelling a previously submitted task,
- checking a previously submitted task status,
- request service metadata (that is, a description of the abilities of the specific server implementation).

In Portolan system a 52North’s implementation of SPS module has been deployed. In this thesis work we implemented SPS functionalities for:

- providing a general way of specifying measurements campaign, including a traceroute measurement campaign, that can be easily extended to future areas of interest,
CHAPTER 3. SYSTEM DESIGN

- translating measurements campaigns specified by human users into smaller jobs that can be executed by mobile devices (up to now only traceroute campaigns are available),
- sending jobs to mobile devices for execution.

In section 3.4 we will see how a human user can specify a measurement campaign, and how the campaign is translated into smaller tasks to be executed by mobile devices.

3.4 Task specification

The Portolan measurement system has been initially conceived for executing traceroute campaigns in a distributed environment, in order to gather data to improve the current Internet topology graph. However, other data of potential interest could be collected from the Internet (e.g., bandwidth between two points of the Internet). Therefore, we designed the task specification as general as possible, so that it can be extended by adding other types of tasks as soon as they will be available. From now on the term task will be used to mean a measurement campaign specified by a human user of Portolan system. The term microtask will be used to specify the part of a task that is assigned to a single mobile device. A task is made of six sections:

- An operation to be performed. That is, the desired type of measurements (e.g., traceroute or bandwidth measurement).
- A source identification part. That is, information needed to identify which mobile devices can perform requested measurements.
- A destination identification part. That is, information needed to identify measurements targets (e.g., for traceroute, target IP interfaces to be probed).
- A duration. That is, the maximum time that should pass before a task can be considered finished (even if the execution of all microtasks has not been completed). As we will see this parameter is necessary because a task could be very huge and its completion could require a lot of time.
CHAPTER 3. SYSTEM DESIGN

- **Operation specific parameters.** That is, additional informations for a specific operation (e.g., hop limit for traceroute operation).

- **Urgent flag.** For tasks to which must be given precedence.

In the next sections we fully describe each part of a task.

### 3.4.1 Operation

As previously said, this is the section that indicates which measurements should be committed to mobile devices.

In this thesis work we focused our attention on traceroute operation, which is the only available operation in Portolan measurement system, up to now. So, there is only one possible value for this field: *traceroute*.

### 3.4.2 Source identification

Source identification informations identify which mobile devices are allowed to perform specified task. This section is common to every task and its fields are:

- **Source country.** Expressed as an ISO 3166–1 country code. Mobile devices allowed to perform the task must reside in the specified country.

- **Source geographic area.** Expressed as a triple of real numbers indicating latitude and longitude of the centre of a circular area, and its radius (e.g., \{43.721025, 10.389698, 2\} is a geographic area centred on the faculty of engineering in Pisa, with a radius of 2 km). Mobile devices allowed to perform the task must reside in the specified area.

- **Source autonomous system.** Expressed as an integer number, the AS number. Mobile devices allowed to perform the task must have an IP address of the address space of the specified AS.

- **Source network type.** Permitted values are *wifi* or *mobile*, indicating that mobile devices allowed to perform the task must be connected to the Internet via wi-fi connection or via mobile connection (e.g., GPRS or UMTS).

- **Source provider name.** Expressed as a string of characters. Mobile devices allowed to perform the task must have an account with specified provider (if source network type is *mobile*).
3.4.3 Destination identification

Destination identification informations are used to retrieve measurements targets. Of course this section of the task is operation dependent, as targets could differ depending on the type of requested measurements. For traceroute operation targets are IP addresses, and they can be identified in two ways: explicitly or implicitly. In explicit mode targets are specified as a list of host names or IP addresses, named destination target list. Mobile devices will send probes to the addresses in the list. On the other hand, in implicit mode targets are identified by a few fields:

- **Destination country.** Expressed as an ISO 3166–1 country code. It indicates that probes must be sent to target IP addresses located in the specified country.

- **Destination geographic area.** Expressed as a triple of real numbers indicating latitude and longitude of the centre of a circular area, and its radius. It indicates that probes must be sent to target IP addresses located in the specified area.
### Mode Parameter Type

<table>
<thead>
<tr>
<th>Mode</th>
<th>Parameter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Target List</td>
<td>IP addresses</td>
</tr>
<tr>
<td>Implicit</td>
<td>Destination Country</td>
<td>ISO 3166–1 country code</td>
</tr>
<tr>
<td></td>
<td>Destination Geographic Area</td>
<td>Latitude, Longitude, Radius(km)</td>
</tr>
<tr>
<td></td>
<td>Destination AS List</td>
<td>AS numbers</td>
</tr>
</tbody>
</table>

Table 3.2: Destination identification section

- **Destination autonomous system list.** Expressed as one or more strings of numbers. It indicates that probes must be sent to target IP addresses that belong to the specified ASes address space.

None of the three fields is mandatory: if none of them is specified, targets are all IP addresses of the world. Table 3.2 summarizes destination identification section.

#### 3.4.4 Duration

Duration information is used to set a finishing time to a task. Portolan system’s human user has no control over the mobile devices that should perform tasks. So, when a task is specified, human user knows neither how many mobile devices would perform the task, nor if there will be some that could perform it. A duration is hence needed to specify a deadline to task execution. Duration is expressed as an integer that indicates the maximum number of days a task might remain in execution before being terminated. If enough mobile devices are available, the task could be terminated before its deadline. This section is common to every task. Default value is seven days.

#### 3.4.5 Operation specific parameters

Obviously this section is operation specific. For traceroute operation there are two types of parameters: parameters that affect probes and parameters that affect targets. Parameters that affect probes are useful to tune measurements settings allowing for example to stop the sending of probes under certain conditions. The parameters are:

- **Hop limit.** Expressed as an integer. It indicates the maximum TTL
value reached by traceroute tool before the measurement is stopped. Default value is 64.

- **Black List.** Expressed as a list of IP addresses. It indicates that measurements should stop when one of the specified IP addresses is discovered.

- **PF only flag.** Expressed as a boolean value. If true it indicates that only per-flow load balancing is to be considered when executing measurements. Default value is true.

- **Probe Limit.** Expressed as an integer. It indicates the maximum number of probes to be sent for next-hop interfaces discovery. Default value is 96.

- **Exploration probe.** Expressed as an integer. It indicates the number of probes sent for each newly discovered interface. Default value is 6.

More informations about parameters that affect probes are available in a previous thesis work [9].

Parameters that affect targets are useful to reduce a task complexity or to increase measurement replications in order to obtain the best configuration to satisfy the needs of a human user submitting a task. The parameters are:

- **Detail level.** Expressed as an integer. It indicates at which subnet granularity the target IP addresses should be chosen. With destination identification informations the set of target IP addresses is identified. However, this set could be too big to be explored completely in reasonable time, as it could contain millions of IP addresses. Thus, a way to choose a subset of IP addresses is needed. Allowed values go from 8 to 24, indicating that it will be chosen an address for each subnet of the specified dimension (e.g., with a value of 24 it will be chosen an address for each /24 subnet in the given set of IP addresses). Default value is 24. The parameter is ignored if target IP addresses are specified explicitly as a target list.

- **Measurement replication.** Expressed as an integer. It indicates how many replicas of a single measurement should be performed. Let $n$ be the desired number of replicas. If target IP addresses are specified
CHAPTER 3. SYSTEM DESIGN

<table>
<thead>
<tr>
<th>Mode</th>
<th>Parameter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affecting Probes</td>
<td>Hop Limit</td>
<td>Integer [1, ∞)</td>
</tr>
<tr>
<td></td>
<td>Black List</td>
<td>IP addresses</td>
</tr>
<tr>
<td></td>
<td>PF Only Flag</td>
<td>Boolean</td>
</tr>
<tr>
<td></td>
<td>Probe Limit</td>
<td>Integer [1, ∞)</td>
</tr>
<tr>
<td></td>
<td>Exploration Probe</td>
<td>Integer [1, ∞)</td>
</tr>
<tr>
<td>Affecting Targets</td>
<td>Detail Level</td>
<td>Integer [8, 24]</td>
</tr>
<tr>
<td></td>
<td>Measurement Replication</td>
<td>Integer [1, ∞)</td>
</tr>
<tr>
<td></td>
<td>Target Number</td>
<td>Integer [50, 200]</td>
</tr>
</tbody>
</table>

Table 3.3: Operation specific parameters section

implicitly in the destination identification section, the set of IP addresses is divided in subnets of dimension specified by the detail level parameter. For each of these subnets, \( n \) IP addresses will be chosen as targets, and assigned to \( n \) different microtasks. If targets are specified explicitly, every address is replicated \( n \) times. Default value is 1.

- **Target number.** Expressed as an integer. It indicates how many target IP addresses should be assigned to a microtask performed by mobile devices. In [8] is stated that with 141 targets the battery consumption on a mobile device was never greater than 1%. In order to keep a low battery consumption when executing a microtask, the number of targets assigned to it should not be much greater. So, allowed values for this parameter are between 50 and 200. Default value is 100.

Table 3.3 summarizes Operation specific parameters section.

### 3.4.6 Urgent flag

Urgent flag is a boolean value. If true it indicates that precedence should be given to that task. If human user submits a task to her more important than the others in execution, urgent flag allows to have that task executed faster than the others (if mobile devices satisfying source identification are available). Default value is false.
3.4.7 Task examples

In this section we give some task examples, which hopefully will help to understand a task’s structure. Consider the example with implicitly specified targets of table 3.4. The requested operation is traceroute. Mobile terminals allowed to perform task are those located in Italy, in a geographic area centred in a point having latitude of 43.72 and longitude of 10.38, and with a radius of 2 km. Moreover, mobile devices must be connected to the Internet with a wi-fi connection and must have an IP address owned by AS3269 (Telecom Italia). The duration of the task is of seven days, and the task is not urgent. Target IP addresses of probes are one IP address for each /24 subnet owned by AS1267 or AS12874 (respectively Infostrada and Fastweb) and located in Italy. Finally, targets assigned to each microtask are 150, and every single traceroute must be stopped when it reaches the 10th hop.

The second example is a task with explicitly specified targets and is shown in table 3.5. Again, requested operation is traceroute. Every mobile terminal located in the USA and connected to internet with a mobile connection is allowed to perform the task. Targets are the host www.google.com and IP address 212.34.6.7. Each target will be replicated for ten times, one for each of ten different microtasks. The duration is of ten days and the task is
### Table 3.5: Explicitly specified targets example

<table>
<thead>
<tr>
<th>Section</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>—</td>
<td>traceroute</td>
</tr>
<tr>
<td>Source Identification</td>
<td>Source Country</td>
<td>US</td>
</tr>
<tr>
<td></td>
<td>Source Network Type</td>
<td>mobile</td>
</tr>
<tr>
<td>Destination Identification</td>
<td>Destination Target List</td>
<td><a href="http://www.google.com">www.google.com</a>, 212.34.6.7</td>
</tr>
<tr>
<td>Duration</td>
<td>—</td>
<td>10</td>
</tr>
<tr>
<td>Operation Specific Parameters</td>
<td>Measurement Number</td>
<td>10</td>
</tr>
<tr>
<td>Urgent Flag</td>
<td>—</td>
<td>true</td>
</tr>
</tbody>
</table>

3.4.8 Task translation: from task to microtask

Task translation is the process of converting a task specification into a set of microtasks that can be executed by mobile devices. For traceroute operation every microtask is composed by six sections:

- **Operation** section. This section is the same as in task specification.
- **Source identification** section. This section is the same as in task specification. As we will see in section 3.5.2, mobile devices allowed to execute certain microtasks are selected dynamically.
- **Target list** section. A list of IP addresses to which mobile device will send probes. The list is done by combining task’s sections destination identification and parameters that affect targets in operation specific parameters.
- **Duration** section. This section is the same as in task specification.
- **Parameters affecting probes** section. This section is the same as in task specification in operation specific parameters section.
- **Urgent flag** section. This section is the same as in task specification.

The translation process is illustrated in figure 3.3. Operation, Source identification, Urgent flag, Duration and parameters that affect probes from
CHAPTER 3. SYSTEM DESIGN

Figure 3.3: Task translation

Operation specific parameters sections are simply copied from task to each microtask. The target list section is obtained combining Destination identification and parameters that affect targets from Operation specific parameters sections.

If targets are specified explicitly, let $T$ be the value of target number field, $M$ the value of measure number field and $N$ the number of specified targets. First, targets are distributed in $\left\lfloor \frac{N}{T} \right\rfloor$ microtasks, $T$ for each microtask (if $N \mod T \neq 0$, the last microtask will have $N \mod T$ targets). Finally, each microtask is replicated $M$ times.

If targets are specified implicitly, let $D$ be the value of detail level field, $M$ the value of measure number field and $T$ the value of target number field. First, system retrieves all IP addresses identified by destination country, destination geographic area and destination autonomous systems list. The
IP addresses are searched in a database that associates to each IP address its location expressed in latitude and longitude coordinates and its AS of origin. This database is obtained by merging data from MaxMind GeoLite Country and City databases [41] and Isolario data, which pairs IP subnet prefixes to ASes [42]. Isolario data is obtained by combining BGP data collected by RouteViews [29] and RIPE NCC RIS [28] projects, publicly available on the web. The choice of MaxMind database is driven by the fact that it is publicly available on the web, and according to [43] its accuracy at the country level is of 99.8%. At the city level accuracy varies, so, when specifying a task, this fact must be taken into account, especially when specifying destination geographic area field. Then, when all IP addresses has been retrieved, the entire set is divided into ranges of $2^{32-D}$ consecutive addresses (or less, if some range of consecutive addresses is smaller than $2^{32-D}$). For each range, $M$ addresses are chosen randomly and are assigned to $M$ different microtasks. Finally, when a microtask contains $T$ target IP addresses, $M$ new microtasks are created and filled. If no target IP addresses are available any more, the translation process is completed. In section 3.5 we will see how microtasks are assigned to mobile devices.

### 3.5 System architecture

In section 3.3 we described system’s interface with human users and its main modules: Sensor Observation Service for collecting measurements from mo-
bile devices and making them available to human users, and Sensor Planning Service for allowing human users to task the system, translating tasks into microtasks and assigning microtasks to mobile devices. Basic architecture of such a system is illustrated in figure 3.4. As shown, SPS and SOS act as a frontend, hiding system complexity to human user and providing a standard way to interact with the system.

In this section we describe the way microtasks are assigned to mobile devices. We will see that additional logic is required to coordinate mobile devices efficiently and to pass through limitations imposed by the chosen framework and by the nature of mobile devices.

3.5.1 Limitations

Sensor Planning Service specification by Open Geospatial Consortium is mainly designed for dealing with single sensors or few easily reachable sensors, not with potentially millions of mobile sensors that can come and go without notice. The specification requires that every single sensor is registered to SPS with a unique identification, and when submitting a task, the ID of sensor to be tasked must be specified. Obviously, this approach could not work in a context of a large number of mobile devices executing small pieces of bigger tasks and whose behaviour is unpredictable. There are three main issues that must be addressed in designing an efficient architecture for coordinating task execution by mobile devices: availability, NAT reachability and scalability.

Availability

Mobile devices, by nature, are not always available: they can reside in an area without connectivity, they can be switched off or simply Portolan application can be closed. Thus, when a task is submitted, there could be no available mobile device that satisfies source identification requirements, but it is likely that in the future some will arrive. In addition, a task may be divided into a large number of microtasks which could be executed by multiple mobile devices. For this reasons a task should be submitted without knowing either which mobile devices are going to execute it or when it is going to be executed. Therefore, mobile devices should not be registered to SPS as sensors, and a mechanism is to be set up in order to allow the system
and not human users) to choose which mobile device is the best candidate to execute a microtask at any given instant.

**NAT Reachability**

Mobile devices often reside in private networks, owning private IP addresses, and have access to the Internet via a gateway. On the gateway, a NAT server maps multiple private IP addresses to one public IP address, so a mobile device is unreachable from outside the private network, unless a persistent connection is maintained between server and device. Maintaining such a connection with every mobile device participating in Portolan system would determine a consumption of energy and resources on device as on server, because the number of devices is potentially very high. Therefore, a mechanism to bypass this form of unreachability is to be set up.

**Scalability**

The number of mobile devices participating in Portolan system could be very large, too large to be handled by a single central unit. Thus, in order to avoid system performance degradation and inefficiencies, the logic of coordinating mobile devices and assigning microtask should be distributed over multiple decentralized units. Each unit would be responsible for handling a small subset of mobile devices (compared to the entire set), allowing to have a scalable system.

### 3.5.2 Portolan architecture

In order to pass through limitations imposed by the nature of mobile devices and by the Sensor Planning Service, we designed a new system architecture that differs from the basic architecture shown in figure 3.4. The new architecture is illustrated in figure 3.5.

The only entity registered at SPS as a sensor is the whole system. This solution allows to hide system complexity to human users, which will submit tasks to the system without knowing either how tasks will be executed or by whom. The system is now fully responsible of choosing the most suitable mobile device for executing a specific microtask, when it will become available.
Figure 3.5: Portolan architecture
In order to make the system scalable we added a set of new modules: *proxies*. A proxy is responsible for handling mobile devices on a geographic area basis. In particular a proxy can handle one or more countries. For example, a proxy will handle only one country if in that country there is a large number of mobile devices participating in Portolan system. On the other hand, a proxy will handle more than one country having each one a small number of mobile devices participating to Portolan system. As discussed in section 3.4.2, task *source identification* section must contain *source country* parameter. After the translation of a task into microtasks, this parameter is used by SPS to identify the proxy responsible for that country and to deliver to that proxy the microtasks just translated. That proxy from now on is responsible for selecting the mobile devices which will execute the task. Proxies make the system scalable by providing a distributed fashion for coordinating mobile devices and assigning tasks. Each proxy handles a subset of the entire set of mobile devices, and the central server handles proxies. This hierarchy allows each single unit (central server and proxies) not to be too much overloaded. This way the overall performance of the system is guaranteed to be always acceptable.

The remaining issues to be addressed are availability and NAT reachability. These issues are solved by introducing a polling mechanism, from mobile device to proxy. A mobile device residing in a certain country polls the proxy which is controlling that country at regular intervals. At each poll the mobile device sends to proxy some data that allow proxy to choose if the device is suitable or not for executing some microtask. If so, then proxy assigns the microtask to mobile device. Else, proxy communicates mobile device that there are no microtasks available at the moment. With such a mechanism availability is solved, as only mobile devices that are available will poll proxies and will execute tasks. If a mobile device is unavailable for some reason, it simply do not poll proxies, and the system acts as if that mobile device no longer participates to Portolan system. The mechanism of polling also solves NAT reachability, as it’s up to the mobile device to connect to proxy for receiving a microtask to execute. Proxy will use the same connection for assigning microtask to mobile device, then the connection is closed until the next poll.

There is only one remaining problem: how does a mobile device know which proxy does it have to poll? Another module has been added to cen-
tral server, in order to find a solution to this issue: proxy assigner. When Portolan application on mobile device starts, it connects to central server, in particular to proxy assigner module, and communicates the country in which it resides. Proxy assigner returns then the URL of the proxy that controls that country, and mobile device starts to poll that proxy.

In conclusion, Portolan system acts as follows:

1. Human user specifies a task, with syntax described in section 3.4, and submits it to SPS module.

2. SPS module translates task into microtasks and delivers them to the proxy identified by the source country parameter.

3. Proxy waits for mobile devices polls, and assigns microtasks to those mobile devices that match microtasks source identification parameters.

4. When execution of a microtask is terminated, mobile device sends results to SOS module and notifies proxy of microtask completion.

5. When execution of a task is terminated (because of completion of all of its microtasks, or because of deadline reached), proxy notifies human user of task completion.

In section 3.5.3 we will see some improvements that allow a faster execution of urgent tasks and a dynamic tuning of polling interval. In section 3.5.4 we will show some example of execution flows between the entities involved in the system. Finally, in chapter 4 we will describe in detail the main components of each module and their interaction with other components and with components of other modules.

### 3.5.3 Enhancements

In section 3.4 we showed that a task may be tagged as urgent with a specific flag, in order to give it precedence over non urgent tasks. We now show a mechanism that allows to speed up urgent task execution. The mechanism uses a service provided by Google: Google Cloud Messaging for Android (GCM) [44]. The service provides a way to send small text messages from servers to applications on Android devices, even if the application is not running. For a server to send a message the following things must be in place:
CHAPTER 3. SYSTEM DESIGN

Figure 3.6: Proxy – mobile devices interaction, via GCM service
The application has a GCM registration ID that allows to receive messages for a particular device.

The device running the application must be a 2.2 or higher Android device.

If the device running the application is running an Android version lower than 4.0.4, it must have at least one logged in Google account.

The server has stored an API key for using GCM service.

The server has stored application GCM registration ID.

When a proxy receives an urgent task, it sends via GCM service a short message to all mobile devices it handles, notifying the arrival of an urgent task. All mobile devices that receive the message immediately poll the proxy, which assigns them microtasks from the urgent task, if they satisfy task requirements. This mechanism allows to keep a low polling frequency when tasks to execute are not urgent, and, on the other hand, when an urgent task is submitted, to force polling from mobile devices, in order to assign urgent microtasks as quickly as possible.

GCM service is also used to dynamically tune polling interval. When a proxy is polled by a large number of mobile devices, keeping a high poll rate could lead to a decrease in performance, because proxy should handle a lot of requests at the same time. Instead, when a small number of mobile devices are polling for microtasks, it would be desirable to increase poll frequency, in order to keep the system responsive to task submission. This is made possible through GCM service. When mobile devices number grows over a certain amount or decreases under a certain amount, proxy sends a message via GCM service to all mobile devices it handles, communicating the new poll rate. From that moment on, all mobile devices receiving the message will poll proxy at that rate.

As previously stated, the server must have stored the application GCM registration ID, in order to send messages through GCM. Thus, mobile devices provide their GCM registration ID to proxies at every poll.

GCM service could have also been used for assigning tasks to mobile devices, instead of using polling mechanism. We decided not to use GCM service for this purpose, in order not to rely on third party services for core functionality of Portolan system. If for some reason GCM service goes down
for some period, or ceases activity, Portolan system would stop working until service goes back on in the first case, or forever in the second case. Therefore, core functionality of the system should rely only on the system itself, in order to ensure correct functioning in every situation. Figure 3.6 shows the interaction between proxies and mobile devices, via GCM service.

### 3.5.4 Examples of execution

In this section we provide examples of execution flows between entities involved in Portolan system. First of all we describe interaction between human user, SPS and proxies. Then, we provide an example of interaction between proxy, mobile device and proxy assigner. Finally we show examples of interaction between proxy and mobile device via GCM service. We do not provide examples of interaction between mobile devices and SOS and between human user and SOS as exhaustively described in another thesis work [9].

In figure 3.7 is shown an example of task submission. Human user submits a task to SPS. SPS translates a task into several microtasks which then delivers to proxy. Proxy is identified by `source country` parameter in `source identification` task’s section. Then, when microtasks are all executed by mobile devices, or task deadline has been reached, proxy notifies human user about task completion. Human user can then retrieve data from SOS.

In figure 3.8 we describe an example of normal interaction between proxy assigner, mobile device and proxy. When application is started, mobile device queries proxy assigner to retrieve URL of a proxy to poll. Proxy assigner returns the URL of the proxy that is handling the country in which mobile device resides. Mobile device starts then to poll its proxy. If mobile device satisfies a task requirements, proxy assigns it a microtask. Mobile device executes microtask, sends results to SOS and finally sends a notification of microtask completion to proxy.

In figure 3.9 we provide an example of urgent task notification via GCM service. When a task tagged urgent is delivered to proxy by SPS, proxy sends a short text message, addressed to all mobile devices, to GCM service. Then, GCM service delivers the message to all mobile devices. At the receiving of the message, mobile devices immediately poll proxy, which assigns them microtasks from the urgent task.

The last example, in figure 3.10, describes a poll rate change notification
Figure 3.7: Interaction between human user, SPS and proxy
Figure 3.8: Interaction between mobile device, proxy assigner and proxy
Figure 3.9: Urgent task notification via GCM service
Figure 3.10: Poll rate change notification via GCM service
via GCM service. When a proxy notices that a large number of mobile devices is sending polls at a high rate, it sends a short text message to all mobile devices via GCM service, containing the new poll rate. At the receiving of the message, mobile devices immediately change poll rate to the new one.
Chapter 4

Implementation

In chapter 3 we discussed the Portolan system design, from the interface with human users to the architecture of the measurement system. We gave an overview over the main components of the system and their interactions at a high level. In this chapter we provide a fully detailed description of the single blocks of the system, of their building components, and of the communication protocols between them and between their components. In the first section of this chapter we summarize system modules and their functionalities, and show which of them were implemented in this thesis work. Then, in the following four sections, we explain in detail each one of the modules implemented in this thesis work, in terms of their main components and of the interactions between them.

4.1 System components overview

As shown in section 3.5.2, and highlighted by figure 3.5, Portolan measurement system is made of five main components:

- a Sensor Planning Service (SPS),
- a Sensor Observation Service (SOS),
- a Proxy Assigner,
- one or more Proxies,
- a large number of Mobile Devices.
The SPS, SOS and Proxy Assigner modules form the central unit of the architecture. SPS and SOS act as a frontend: the interface between human users and the rest of the system. They provide to human users standard operations to submit tasks and retrieve measurements data. On the other side, they deliver tasks to proxies (which are responsible of microtasks assignment to mobile devices and mobile devices coordination) and collect measurements from mobile devices.

The Proxy Assigner is responsible of assigning each mobile device to the correct proxy. Since mobile devices coordination is done on a geographic basis, as seen in section 3.5.2, mobile devices must be assigned to the proxy that is controlling the country in which they reside. Otherwise, mobile devices would not be able of executing any microtask, as a consequence of not matching any microtask’s requirements.

Proxies are decentralized units that coordinate mobile devices and allow the system to achieve scalability by handling a subset of the entire set of mobile devices. They also provide solution to availability and NAT reachability issues through the mechanism of polling, described in section 3.5.2. As previously said, they receive microtasks (generated from tasks) from SPS module and assign them to mobile devices. When a task is completed, or when a task deadline is reached, whichever comes first, they notify human users about task completion. They also control mobile devices poll rate, in order not to overload the entire system.

Mobile devices are responsible of the execution of tasks. Each one of them will execute a small part of a task, called microtask. They poll their assigned proxy for receiving a microtask to execute. When execution is finished, they send data to SOS and, finally, they notify proxy about task completion.

In this thesis work we designed and implemented:

- SPS functionality to receive tasks, translate them into microtasks and send them to proxies.
- Proxy Assigner entire functionality.
- Proxy entire functionality.
- Mobile Android application functionality to interact with proxies (including the interaction via GCM service).
- All the communication protocols between these entities.
In the next sections we will provide a comprehensive description of the implemented modules and functionalities. We will not provide description either of the entire mobile application for the Android system or its interaction with SOS, or of SOS functionalities, as not part of this thesis work. An exhaustive description of these topics can be found in [9].

4.2 Sensor Planning Service

As seen in section 3.3, an Open Geospatial Consortium Sensor Planning Service 1.0.0 specification has been implemented by 52North. This implementation provides all the operations standardized by OGC in [37]. However, implementation of the functionalities to interact with specific sensors is left to the user. 52North implemented SPS as a Java [45] Web application. In particular, 52North SPS is implemented as a Servlet. Servlets are the Java platform technology for extending and enhancing Web servers. Servlets provide a component-based, platform-independent method for building Web-based applications that are server-independent. Basically, a Servlet is a Java class that extends functionalities of a Web server, allowing to execute user defined applications. More informations about Servlets might be found at [46].

In Portolan system, the SPS module is responsible for receiving tasks submitted by human users, translating them into a set of microtasks and delivering the generated microtasks to a specific proxy, identified by source country task field, as seen in section 3.5.2. In order to add this functionalities to 52North SPS implementation, a plugin for SPS is to be defined and deployed. A SPS plugin is a Java class that extends the SPSSensor class defined by 52North and that performs the following operations from the OGC SPS standard:

- *GetFeasibilityRequest*. Used by human users to know if a specified task is feasible. Note that this operation does not submit any task to the system, it only checks if the specified task could be performed by the system.

- *SubmitRequest*. Used by human users to submit a task to the system.

- *UpdateRequest*. Used by human users to update a task specification of a previously submitted task.
FIGURE 4.1: SPS PLUGIN ARCHITECTURE

- **GetStatusRequest.** Used by human users to check the status of a previously submitted task.
- **CancelRequest.** Used by human users to cancel a previously submitted task.

Only the **SubmitRequest** operation is mandatory, **GetFeasibilityRequest**, **UpdateRequest**, **GetStatusRequest** and **CancelRequest** operations are optional. In this thesis work we implemented the **SubmitRequest**, **GetFeasibilityRequest** and **CancelRequest** operations. The other operations from the OGC SPS standard are handled directly by the SPS, so there is no need to implement them.

The architecture of implemented SPS plugin is shown in figure 4.1. An SPS plugin is made of five different components, each of them implementing a different functionality:

- **Task Handler,**
- **Microtask Builder,**
- **Task Sender,**
- **Microtask Queue,**
Mail Sender.

Each component of the SPS plugin is a Java object. In the next sections the five modules are described in detail.

4.2.1 Task Handler

Task Handler object handles the SubmitRequest, GetFeasibilityRequest and CancelRequest operations. First, it performs a syntactical check of submitted operation, in order to check if SPS standard syntax is observed. Then, it extracts the operation type and performs operation-specific operations.

GetFeasibilityRequest

When a human user submits a GetFeasibilityRequest operation, she must provide a task specification for which feasibility is to be tested. Task Handler extracts the task specification from the operation body, then it checks for its syntactical correctness. If task is syntactically correct, then the task is marked feasible, Task Handler assigns to it a unique feasibility ID, and stores it for a while in its data structures. Then, Task Handler notifies to the human user that the task is feasible, providing the feasibility ID. The Human user can submit the SubmitRequest operation by providing the feasibility ID, instead of the whole task specification. On the contrary, if task is not syntactically correct, then Task Handler returns an error message containing the error cause.

SubmitRequest

There are two ways to submit a SubmitRequest operation:

- by specifying a task as shown in section 3.4,
- by specifying a previously obtained feasibility ID, with a GetFeasibilityRequest operation.

If task specification is provided, then the Task Handler checks if it is syntactically correct, according to rules explained in section 3.4. As previously said, the only measurement available up to now is traceroute. If task specification is violated, then Task Handler returns to human user an error message containing the error cause.
If a feasibility ID is provided, then Task handler checks for its validity. If the feasibility ID is not valid, then an error message is returned. If the feasibility ID is valid, Task Handler retrieves the task identified by that ID, which it has already been checked for syntactical correctness.

Then, Task Handler creates an instance of Microtask Builder, Task Sender and Microtask Queue. It sends the destination identification parameters to Microtask Builder for retrieving the target IP addresses and for building the microtasks to be delivered to Proxy. The remaining task parameters are passed to Task Sender, which is responsible of sending microtasks to Proxy. The communication protocol between Task Sender and Proxy is explained in section 4.4.

As target IP addresses to be retrieved and to be sent to proxy might be in large number, Microtask Builder and Task Sender execution could require a long time. Moreover, the amount of memory required to store all target IP addresses could be very large, much larger than Java Virtual Machine heap size. In order to minimize waiting time for human users and occupied memory on the JVM, Microtask Builder and Task Sender are implemented as Java threads executing concurrently, exchanging data via Microtask Queue. Task Handler starts the two threads, then notifies human user that the submitted task has been accepted, and a notification of task retrieval completion or of error occurrence will be sent via e-mail. For this reason, human users must provide also a valid e-mail address to which notification will be sent. Microtask Builder, Task Sender and Microtask queue behaviours are shown in detail in sections 4.2.2, 4.2.3, 4.2.4, respectively.

In Portolan system each task is identified by a unique ID termed task ID, assigned by Task Handler, and communicated to human users via e-mail, if and only if the task is successfully translated into microtasks and successfully delivered to proxy. The task ID must also be provided when cancelling a task.

**CancelRequest**

When a CancelRequest operation is submitted, a valid task ID is to be provided. Task Handler maintains a data structure that, for each task in execution, maps its task ID to the proxy to which the task has been delivered. If the submitted task ID appears in the data structure, Task Handler communicates to the Proxy that such task is to be aborted. If no error occurs, Task
CHAPTER 4. IMPLEMENTATION

Handler sends notification of correct execution to the human user. Else, if the submitted task ID is not valid or if an error occurs, Task Handler returns to the human user an error message containing the error cause.

4.2.2 Microtask Builder

As introduced in section 4.2.1, Microtask Builder is a Java thread instantiated and started by Task Handler. Microtask Builder is responsible for translating the destination identification section and the parameters that affect targets of the operation specific parameters of the task into a set of target lists, and to assign each one of them to a microtask. Target IP addresses identified by destination identification section are retrieved from a database that associates to each IP address its geographic location and its AS of origin. The process of translating tasks into microtasks and the structure of database are described in detail in section 3.4.8. Each microtask is then stored in the Microtask Queue data structure, from which Task Sender collects the microtasks that are ready to be sent to proxy. If an error occurs or if the set of IP addresses identified by destination identification is empty, Microtask Builder sets an error flag in Microtask Queue and ends its execution.

4.2.3 Task Sender

Task Sender is a Java thread instantiated and started by Task Handler. Task Handler passes to Task Sender all the fields of a task specification, except for the destination identification section and the parameters that affect targets of the operation specific parameters. Task Sender is responsible for delivering microtasks to the Proxy identified by source country field of source identification section. As all the fields of a microtask, except for the target list, are common to other microtasks of the same task, in order to minimize the amount of traffic on the network and transfer time, Task Sender sends all the common fields only once, then sends all the target lists belonging to each separate microtask. As long as target lists are made available by Microtask Builder Task Sender collects them from the Microtask Queue data structure and sends them to Proxy. If no error occurs in the task-to-microtask translation and in the transfer process, Task Sender notifies the Mail Sender module of correct execution, otherwise it notifies Mail Sender that an er-
Figure 4.2: Microtask Queue structure

ror has occurred. Of course, it also communicates to Mail Sender module the e-mail of the human user that submitted the task. The communication protocol between Proxy and Task Sender will be shown in section 4.4.

4.2.4 Microtask Queue

Microtask Queue is a Java class that implements a FIFO queue of microtasks. Microtask Builder inserts microtasks in the queue and Task Sender collects previously inserted microtasks. The interaction between Microtask Builder and Task Sender is an example of the producer-consumer pattern. Microtask Builder produces data (i.e., microtasks) which are consumed by Task Sender. As Microtask Builder and Task Sender are threads executing concurrently, the Microtask Queue must provide a way for synchronizing the two threads. For this purpose, we implemented the Microtask Queue as a monitor: an object intended to be used safely by more than one thread [47, 48]. The defining characteristic of a monitor is that its methods are executed with mutual exclusion. That is, at each point in time, at most one thread may be executing any of its methods. Internally, Microtask Queue is implemented as a circular array of user-defined size. When the array is full, Microtask Builder must wait for Task Sender to consume at least one microtask. When
the array is empty, Task Sender will wait until Microtask Builder inserts at least one microtask. Microtask Queue also provides methods to communicate when there are no more microtasks and if an error has occurred. Figure 4.2 summarizes the structure of Microtask Queue.

4.2.5 Mail Sender

Mail Sender is responsible for notifying to human user if a previously submitted task has been accepted by the system or if an error has occurred within the task translation process. Mail Sender uses the JavaMail API provided by Oracle [49]. It receives from Task Sender the e-mail address of human user to be notified and the result of task translation and delivery. If no error has occurred, Mail sender sends an e-mail message to human user containing the task ID (which could be used, for example, for cancelling the task). Otherwise, if an error has occurred, Mail Sender notifies to human user of error occurrence and its cause.

4.2.6 SPS Plugin execution flows

After showing the functionalities of all the components of SPS plugin separately, we now show the execution flows of the three supported operations.

The execution flow of GetFeasibilityRequest operation is described in figure 4.3. As seen in section 4.2.1, when a human user submits a GetFeasibilityRequest, the request contains a task specification, which is passed from SPS to Task Handler. Task Handler performs a syntactical check of task specification, and returns the result to the human user. If the task specification is syntactically correct, Task Handler returns to the human user a feasibility ID, that may be used to submit the specified task. Otherwise, Task Handler returns an error message containing the error cause.

The execution flow of SubmitRequest is shown in figure 4.4. As described in section 4.2.1, there are two types of SubmitRequest:

- SubmitRequest containing a task specification,
- SubmitRequest containing a feasibility ID, previously obtained with a GetFeasibilityRequest.

In figure 4.4, the first case is shown. Human user submits a SubmitRequest containing a task specification to SPS. SPS passes the request to Task Han-
 CHAPTER 4. IMPLEMENTATION

![Diagram](image)

Figure 4.3: GetFeasibilityRequest execution flow

dler, which performs a syntactical check of the task. If the task is correct, Task Handler notifies the human user that the task has been accepted and is going to be processed. Then, Task Handler sends destination identification parameters and parameters that affect targets of Operation specific parameters to Microtask Builder, and all the other parameters to Task Sender. Microtask Builder performs the retrieval of target IP addresses and sends them to Task sender via Microtask Queue. Meanwhile, Task Sender sends the parameters that are common to all microtasks to Proxy and, on targets arrival from Microtask Queue, sends the target lists to Proxy. When the transfer is complete, and if no error has occurred, Task Sender communicates to Mail Sender that the process is complete, and Mail Sender notifies the human user that task translation was successful and the task will be executed. Mail Sender delivers to the human user a task ID, which can be used to eventually cancel the task with a CancelRequest. If an error occurs in the process of translating the task and sending it to Proxy, Mail sender notifies
Figure 4.4: SubmitRequest execution flow
the human user of error occurrence and its cause.

In the case of *SubmitRequest* containing a feasibility ID, the behaviour is identical except for the Task Handler part. Task Handler extracts the feasibility ID from the request, and checks if it is valid. That is, if a task specification corresponding to that feasibility ID is stored in Task Handler’s data structures. If so, then the execution flow continues as in the first case.

If in the first case the task specification is syntactically incorrect, or if in the second case the feasibility ID is invalid, then Task Handler communicates to human user that the *SubmitRequest* is not accepted, and stops the execution.

In figure 4.5, the execution flow of *CancelRequest* operation is described. SPS passes the *CancelRequest* issued by human user to Task Handler. Task Handler extracts task ID from the request, and checks for its validity. If task ID matches a task in execution, then Task Handler retrieves the Proxy that is handling that task, and communicates it to stop task execution and cancel the task. Proxy performs the requested operation and returns the result to Task Handler, which returns it to human user.

If task ID is invalid an error message containing the error cause is returned to human user.

### 4.3 Proxy Assigner

Proxy Assigner is a module that accepts requests by mobile devices willing to participate in Portolan system, in which they specify the country code of the country they reside in, and returns the URL of the Proxy that controls that country, to which devices must send polls for requesting microtasks to be performed. Proxy Assigner module is implemented as a Web application. The reason of this choice is that implementations of the mobile device application are planned to be realized in the near future for Apple iOS [50] and MS Windows Phone [51] systems. Thus, a system-independent communication protocol is needed, hence, we decided to use the HTTP protocol [13]. Therefore, Proxy Assigner is a Java servlet which accepts HTTP GET and POST requests. Mobile device sends its country code as a query string attached to the URL if the request method is GET, or in the request body if the request method is POST. A query string is a sequence of fields in the form of `key=value` pairs, separated by the character &. Further details are
Figure 4.5: CancelRequest execution flow
provided in [52, Section 3]. Proxy Assigner accepts only one field, whose key is \textit{ccode}, and its values are ISO 3166–1 country codes. As an example, a valid query string could be \texttt{ccode=IT}, meaning that the mobile device formulating the request resides in Italy.

Proxy Assigner structure is shown in figure 4.6. Its main components are:

- \textit{Request Handler}.

- \textit{Country Database}.

When an HTTP GET or POST request arrives from a mobile device, Request Handler module checks if it is a valid request. A valid request must contain only the \textit{ccode} field and an existing country code in ISO 3166–1 format. If the request is not valid, Request Handler returns a HTTP 400 Bad Request response. If request is valid, Request Handler queries the Country Database for the Proxy URL corresponding to the given country code. Country Database returns the correct Proxy URL, which is returned to mobile device by Request Handler, within the body of a HTTP 200 OK response. If an error occurs (e.g., connection with database fails), Request Handler re-
<table>
<thead>
<tr>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid request</td>
<td>HTTP 200 OK and Proxy URL</td>
</tr>
<tr>
<td>Invalid request</td>
<td>HTTP 400 Bad Request</td>
</tr>
<tr>
<td>Error occurrence</td>
<td>HTTP 500 Internal Server Error</td>
</tr>
</tbody>
</table>

Table 4.1: Proxy Assigner responses

Figure 4.7: Proxy architecture

turns a HTTP 500 Internal Server Error response. Proxy Assigner behaviour is summarized in table 4.1.

4.4 Proxy

As stated in section 3.5.2, Proxy is the Portolan system component that handles a subset of the entire set of mobile devices, in order to provide a solution to scalability and, through polling mechanism, availability and NAT reachability issues. A Proxy needs to communicate on one side with SPS module (in particular with SPS Plugin), on the other side with mobile devices. For the same reasons explained in section 4.3 for Proxy Assigner, Proxy is implemented as a Web application, composed of two Java Servlets and a common database:

- *ProxySPSServlet*, which handles interaction between Proxy and SPS Plugin,
- *ProxyMDServlet*, which handles interaction between Proxy and mobile devices,
- *Proxy Database*, in which microtasks and other informations about task
execution are stored.

Figure 4.7 shows proxy architecture at a high level of abstraction. In the next sections we will describe the architecture of the two servlets, their behaviour and the communication protocols between them and the other components of the system.

4.4.1 ProxySPSServlet

ProxySPSServlet is responsible for handling interaction between Proxy and SPS Plugin. That is, receiving tasks and receiving requests of task cancellation from SPS Plugin. ProxySPSServlet architecture is shown in figure 4.8. ProxySPSServlet is made of two modules:

- **Task Receiver.** Receives tasks from SPS and stores them into Proxy Database. Notifies mobile devices of urgent tasks via GCM service.
- **Task Canceller.** Handles task cancellation requests received from SPS.
CHAPTER 4. IMPLEMENTATION

The two modules are described in detail in the next sections.

**Task Receiver**

As previously said, Task Receiver receives tasks from SPS and stores them into Proxy Database. A task is sent from SPS in the request body of an HTTP POST request. The request body of such a request is structured as follows:

1. The *Task ID*, followed by a line break,

2. Task common parameters (that is, *operation*, *source identification*, *duration*, *urgent flag*, and parameters that affect probes from the *operation specific parameters*), each one on a separate line, and ended with an empty line,

3. A target list for each microtask in the task, each one ended with an empty line (each target in the list stands on a separate line).

Task Receiver checks if each one of the fields is present in the correct order. If so, Task Receiver stores the task into Proxy Database. In order to minimize storage usage, task common parameters are stored only once, and for each microtask only the target list is stored. Task Receiver also assigns to each microtask a microtask ID, which allows to recognize which microtasks are assigned to mobile devices. If database operations are succesful, then Task Receiver returns to SPS a HTTP 200 OK response. Else, if an error occurs, then Task Receiver returns a HTTP 500 Internal Server Error response. If request syntax is wrong, then Task Receiver returns a HTTP 400 Bad Request response.

If received task is urgent, Task Receiver is also responsible of notifying mobile devices via Google Cloud Messaging service that an urgent task has arrived and they must poll Proxy for speeding up the task execution, as illustrated in section 3.5.3.

**Task Canceller**

A request of task cancellation is submitted by SPS to Task Canceller as a HTTP GET request, with request parameters passed as a query string. A valid query string has the following parameters:
CHAPTER 4. IMPLEMENTATION

Table 4.2: ProxySPSServlet responses

<table>
<thead>
<tr>
<th>Request</th>
<th>Request Validity</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancel Task</td>
<td>Valid</td>
<td>HTTP 200 OK and Cancel Task</td>
</tr>
<tr>
<td></td>
<td>Invalid</td>
<td>HTTP 400 Bad Request</td>
</tr>
<tr>
<td></td>
<td>Invalid Task ID</td>
<td>HTTP 404 Not Found</td>
</tr>
<tr>
<td>Send Task</td>
<td>Valid</td>
<td>HTTP 200 OK and Store Task</td>
</tr>
<tr>
<td></td>
<td>Invalid</td>
<td>HTTP 400 Bad Request</td>
</tr>
<tr>
<td>Error occurrence</td>
<td></td>
<td>HTTP 500 Internal Server Error</td>
</tr>
</tbody>
</table>

- *op*: it specifies the operation that has to be performed. In this case its value is *cancel*.
- *taskid*: it specifies the task ID of the task that has to be cancelled.

As an example, a valid query string could be the following: `op=cancel&taskid=TID13142345346354142`.

First, Task Canceller checks for request validity. If the request is valid, then Task Canceller performs task cancellation on Proxy Database and returns to SPS a HTTP 200 OK response. If request is not valid, then Task Canceller returns a HTTP 400 Bad Request response. Else, if an error occurs, then Task Canceller returns a HTTP 500 Internal Server Error response. If the given task ID does not match any task stored in Proxy Database, then Task Canceller returns a HTTP 404 Not Found response.

Table 4.2 summarizes the behaviour of ProxySPSServlet.

4.4.2 ProxyMDServlet

ProxyMDServlet handles communication between Proxy and Mobile Devices. Its architecture is shown in figure 4.9. ProxyMDServlet is responsible for receiving polls from mobile devices, assigning microtasks, and receiving notifications of microtasks correct execution. ProxyMDServlet is also responsible of notifying human users of task completion.

The main components of ProxyMDServlet are:

- *Poll Handler*. It manages all poll operations between proxy and mobile device.
CHAPTER 4. IMPLEMENTATION

Figure 4.9: ProxyMDServlet architecture

- **Notification Handler.** It receives from mobile devices notifications of microtask completion and cancels from Proxy Database the completed microtask.

- **Database Cleaner.** It performs several operations on Proxy Database, such as checking if a task is completed, or checking for database inconsistencies.

- **Mail Sender.** It receives from Database Cleaner the e-mail address of human users to be notified of task completion and sends them a notification e-mail.

**Poll Handler**

Poll Handler is responsible for handling poll requests from mobile devices. A poll request is a HTTP GET or POST request with poll data passed as a
CHAPTER 4. IMPLEMENTATION

query string attached to URL (if GET request) or stored in request body (if POST request). Request parameters are:

- **op**: the operation to be performed. For poll request the value is *poll*.
- **moblat**: the latitude coordinate of the mobile device, expressed in degrees.
- **moblon**: the longitude coordinate of the mobile device, expressed in degrees.
- **mobnettype**: the network type to which is connected mobile device. Its values are *mobile* or *wifi*.
- **mobprov**: the provider name to which is attached mobile device.
- **mobid**: mobile device unique identifier (IMEI or MEID).
- **mobgcmid**: the GCM identifier of mobile device.

These parameters are used from Poll Handler to check if the mobile device matches some task specification. Reader should notice that mobile device public IP address, which is used from Poll Handler to retrieve the AS in which mobile device resides, is missing. As Java servlets provide methods to retrieve this information from connection data, this information is not passed as a parameter of the request.

When Poll Handler receives a request, first it checks if all parameters are present and if their value is correct. If so, then it queries Proxy Database to find a microtask which mobile device can execute. If such a microtask is found, Poll Handler stores it into a queue of scheduled microtasks, so the microtask cannot be assigned to another mobile device. Then, the microtask is sent to mobile device for execution within the body of a HTTP 200 OK response. Moreover, Poll Handler stores mobile device GCM identifier into Proxy Database, so that ProxySPSServlet can use it to notify mobile device for urgent tasks. If Proxy Handler could not find any microtask that matches request parameters, a HTTP 204 No Content response is sent.

If request was invalid due to parameters incorrectness, then a HTTP 400 Bad Request response is sent. Else, if an error occurs in processing the request (e.g., an error in connecting to Proxy Database), then a HTTP 500 Internal Server Error response is sent.
CHAPTER 4. IMPLEMENTATION

### Table 4.3: ProxyMDServlet responses

<table>
<thead>
<tr>
<th>Request</th>
<th>Request Validity</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poll</td>
<td>Valid</td>
<td>HTTP 200 OK and Microtask</td>
</tr>
<tr>
<td></td>
<td>Invalid</td>
<td>HTTP 204 No Content</td>
</tr>
<tr>
<td>Notify Finished</td>
<td>Valid</td>
<td>HTTP 200 OK and Update Database</td>
</tr>
<tr>
<td></td>
<td>Invalid</td>
<td>HTTP 400 Bad Request</td>
</tr>
<tr>
<td>Error occurrence</td>
<td></td>
<td>HTTP 500 Internal Server Error</td>
</tr>
</tbody>
</table>

### Notification Handler

Notification Handler is responsible for receiving notification of microtask completion from mobile devices and for updating Proxy Database. A notification is a HTTP GET or POST request with data passed as a query string attached to URL (if request is GET) or stored in request body (if request is POST). Request parameters are:

- **op**: the operation to be performed. For notifications the value is *notify*.
- **taskid**: the ID of the task that owns the microtask being notified.
- **microid**: the ID of the microtask being notified.

When Notification Handler receives a request, first it checks if passed request parameters are correct. If so, then it updates Proxy Database. It cancels the finished microtask from the queue of scheduled microtasks and updates task status, decreasing by one the number of remaining microtasks in the task. If the update is successful Notification Handler returns an empty HTTP 200 OK response. On the contrary, if the request is invalid (that is, some parameters are missing or incorrect, or *taskid* or *microid* does not exist), then it returns a HTTP 400 Bad Request response. If an error occurs in processing the request (e.g., an error in connecting to Proxy Database), then a HTTP 500 Internal Server Error response is returned.

Table 4.3 summarizes interaction between ProxyMDServlet and mobile devices.
CHAPTER 4. IMPLEMENTATION

Database Cleaner

Database Cleaner is responsible of periodically querying Proxy Database, in order to find if some tasks are completed or if there is some inconsistency.

A task is said to be completed if all of its microtasks are completed or if its deadline has expired. If either of the two conditions is satisfied, Database Cleaner updates Proxy Database cancelling the completed task, and communicates to Mail Sender the task completion and the e-mail address of human user to be notified.

As mobile devices could fail in executing an assigned microtask or in notifying Proxy of microtask completion, a microtask could remain in the scheduled queue forever. Thus, Database Cleaner also checks scheduled queue for old microtasks. If a microtask stays in the scheduled queue for more than two hours, then it is said to be old, and we assume that some failure occurred within its execution. Database Cleaner moves back old microtasks from scheduled queue to database, so they can be assigned to other mobile devices.

Moreover, Database Cleaner checks for database inconsistencies, that could be consequence of some previous running error or failure, and repairs them.

Mail Sender

Mail Sender behaviour is basically the same described in section 4.2.5. It receives from Database Cleaner the task ID of a completed task and the e-mail address of a human user, and sends an e-mail to that user for notifying of task completion.

4.5 Mobile application features

The Android application implemented in [9] provides functionalities for executing a traceroute microtask and storing measurement results into SPS. The mechanism of polling was already implemented by a Coordinator object, which runs in background at regular intervals (the polling interval) and executes a microtask when assigned by Proxy. What has been implemented in this thesis work is a module that handles the communication via GCM service.
At application start, the module checks if the GCM identifier has already been obtained, else it performs a registration with the service, in order to obtain the identifier. At each poll the identifier is passed to ProxyMDServlet, as described in section 4.4.2. When an urgent task is submitted from human user, Proxy notifies mobile applications via GCM. It’s up to GCM service to correctly deliver the message to applications. As the application receives the message, it immediately polls its Proxy. The communication then follows the protocols shown in section 4.4.2.

In order to reduce battery consumption on mobile device, we decided to limit the number of microtasks per day that a mobile device can execute. So, the application polls at regular intervals its Proxy, until a microtask is assigned. After performing the microtask and sending results, the application stops polling until the next day.

4.5.1 Measurement failures

In addition to failures due to hardware, software or connection problems, there could be cases in which measurements should be aborted or collected data should be flagged as invalid.

As measurement monitors are mobile, their conditions may vary while a measurement is performed. The change of conditions may invalidate an entire measurement. Thus, a traceroute is aborted and collected data is cancelled in the following cases:

- If measurements are performed from a mobile connection, then if the mobile device moves from one cell to another, the traceroute is aborted. As the new cell could be served by a different router, path to destination could change while traceroute is executed, invalidating the whole measurement.

- If measurements are performed from a Wi-Fi connection, then if mobile device switches network, the traceroute is aborted. Again, the new network could be served by a different router, so path to destination could change.
Chapter 5

Experimentation

In this chapter we show the experiments set up to validate and evaluate the performance of the Portolan system architecture and implementation, described in chapters 3 and 4. The chapter is divided into four sections. The first section describes the experiment setup and the motivations that led to the choices made. The second section shows the achieved experimental results. The third section describes how the validation was carried out, in order to show if the implemented system measurement results are consistent with the reality. Finally, in the fourth section we compare our measurement results with data provided by other Internet measurement projects, in order to determine whether or not the Portolan system is able to supply a sensible contribution to Internet measurements. Moreover, we determined the performance of the system in terms of task execution time.

5.1 Experiment setup

The experiment was conducted in July 2012 and consisted in discovering the links of an autonomous system to its neighbours, via the Milan Internet eXchange (MIX, [53]). We chose an AS within its participants, placed a mobile device connected to it, and sent probes to a set of other ASes connected to the MIX.

The experiment was carried out from the autonomous system of Registro.it, which is the organization responsible of assigning the country code top level domain (ccTLD) .it, whose AS number is 2597 (from now on we will refer to it as AS2597). Registro.it operates inside the Informatics and
Telematics Institute (IIT, [54]) of CNR, the Italian National Research Council [55]. For our measurements we used a mobile device connected to a Wi-Fi network attached to AS2597, and having a public IP address of AS2597 address space. The mobile device resided in network 192.12.193.0/24. The set of ASes to which probes were sent is shown in table 5.1.

<table>
<thead>
<tr>
<th>AS Name</th>
<th>AS Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abilene</td>
<td>31611</td>
</tr>
<tr>
<td>Acantho</td>
<td>21309</td>
</tr>
<tr>
<td>Active Network</td>
<td>197075</td>
</tr>
<tr>
<td>Aria</td>
<td>48291</td>
</tr>
<tr>
<td>Aruba</td>
<td>31034</td>
</tr>
<tr>
<td>Asdasd</td>
<td>28929</td>
</tr>
<tr>
<td>Brennercom</td>
<td>20811</td>
</tr>
<tr>
<td>BT Italia</td>
<td>8968</td>
</tr>
<tr>
<td>BT Italia / I.Net</td>
<td>3313</td>
</tr>
<tr>
<td>CDLan</td>
<td>20836</td>
</tr>
<tr>
<td>Clio</td>
<td>9104</td>
</tr>
<tr>
<td>Cloud Italia</td>
<td>15589</td>
</tr>
<tr>
<td>Club Nautilus / Maki</td>
<td>8980</td>
</tr>
<tr>
<td>Comeser</td>
<td>39657</td>
</tr>
<tr>
<td>Consortium GARR</td>
<td>137</td>
</tr>
<tr>
<td>D.T.S. - ReteIVO</td>
<td>49605</td>
</tr>
<tr>
<td>Dada</td>
<td>39729</td>
</tr>
<tr>
<td>Digitel Italia</td>
<td>50809</td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>AS Name</th>
<th>AS Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4A</td>
<td>34695</td>
</tr>
<tr>
<td>Easynet Italia</td>
<td>4589</td>
</tr>
<tr>
<td>Engineering IT</td>
<td>21176</td>
</tr>
<tr>
<td>Enter</td>
<td>12850</td>
</tr>
<tr>
<td>ePress Information Services</td>
<td>24880</td>
</tr>
<tr>
<td>Estracom</td>
<td>31319</td>
</tr>
<tr>
<td>Eurocall</td>
<td>196761</td>
</tr>
<tr>
<td>Fastnet</td>
<td>8265</td>
</tr>
<tr>
<td>Fastweb</td>
<td>12874</td>
</tr>
<tr>
<td>FUB - Fondazione Ugo Bordoni</td>
<td>50112</td>
</tr>
<tr>
<td>H3G</td>
<td>24608</td>
</tr>
<tr>
<td>ICT Valle Umbra</td>
<td>15605</td>
</tr>
<tr>
<td>ICTeam</td>
<td>20924</td>
</tr>
<tr>
<td>IFOM</td>
<td>35193</td>
</tr>
<tr>
<td>Infracom Italia</td>
<td>3302</td>
</tr>
<tr>
<td>Interactive Network</td>
<td>41497</td>
</tr>
<tr>
<td>Intercom</td>
<td>8224</td>
</tr>
<tr>
<td>Internet One</td>
<td>44160</td>
</tr>
<tr>
<td>Interoute</td>
<td>8928</td>
</tr>
<tr>
<td>Itelsi</td>
<td>6760</td>
</tr>
<tr>
<td>ITGate Network</td>
<td>12779</td>
</tr>
<tr>
<td>KPNQwest Italia</td>
<td>5602</td>
</tr>
<tr>
<td>Level3 Communications</td>
<td>3549</td>
</tr>
<tr>
<td>Level IP Italia / InternetFR</td>
<td>8527</td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>AS Name</th>
<th>AS Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lottomatica / SED Multitel</td>
<td>35574</td>
</tr>
<tr>
<td>Mainsoft</td>
<td>51185</td>
</tr>
<tr>
<td>Mandarin Wimax Sicilia</td>
<td>47408</td>
</tr>
<tr>
<td>Mc-Link</td>
<td>5396</td>
</tr>
<tr>
<td>Mediaset</td>
<td>48634</td>
</tr>
<tr>
<td>Metrolink</td>
<td>8816</td>
</tr>
<tr>
<td>Nexin Technologies</td>
<td>31076</td>
</tr>
<tr>
<td>NGI</td>
<td>35612</td>
</tr>
<tr>
<td>NTRnet</td>
<td>47358</td>
</tr>
<tr>
<td>OKCom / Teleunit</td>
<td>24915</td>
</tr>
<tr>
<td>Optima Italia</td>
<td>44513</td>
</tr>
<tr>
<td>Planetel</td>
<td>47217</td>
</tr>
<tr>
<td>Postecom</td>
<td>15720</td>
</tr>
<tr>
<td>RAI - Radio Televisione Italiana</td>
<td>8234</td>
</tr>
<tr>
<td>Retelit / e-via</td>
<td>28716</td>
</tr>
<tr>
<td>Seeweb</td>
<td>12637</td>
</tr>
<tr>
<td>Seflow</td>
<td>49367</td>
</tr>
<tr>
<td>Siportal</td>
<td>28999</td>
</tr>
<tr>
<td>Spin</td>
<td>6734</td>
</tr>
<tr>
<td>T.Net</td>
<td>8922</td>
</tr>
<tr>
<td>Telecity Group Italia</td>
<td>15830</td>
</tr>
<tr>
<td>Telecom Italia</td>
<td>3269</td>
</tr>
<tr>
<td>TELEimpianti</td>
<td>50316</td>
</tr>
<tr>
<td>Telnet</td>
<td>5392</td>
</tr>
</tbody>
</table>

*continued on next page*
Thus, we submitted to the Portolan system the task shown in table 5.2. The mobile devices allowed to perform the task must reside in Italy, and must be connected to a Wi-Fi network with an address belonging to AS2597 address space. Targets IP addresses were chosen within the address spaces of the ASes shown in table 5.1. The Operation specific parameters are left to their default values, described in section 3.4.5. Since experiment was performed by a single mobile device, duration was set to 30 days, in order to be sure to complete the entire task. The task was translated into 1556 microtasks, containing approximately 100 targets each. In order to speed up the experiment execution, when a connection, via the MIX IXP, between AS2597 and another AS was found, the targets belonging to the latter were cancelled.

In order to check if AS2597 is connected to another AS via the MIX IXP, the found route must contain the following three consecutive hops, in this order:

<table>
<thead>
<tr>
<th>AS Name</th>
<th>AS Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tiscali Italia</td>
<td>8612</td>
</tr>
<tr>
<td>TopneT Telecomunicazioni</td>
<td>29449</td>
</tr>
<tr>
<td>Trentino Network</td>
<td>12835</td>
</tr>
<tr>
<td>TWT</td>
<td>30848</td>
</tr>
<tr>
<td>Unidata</td>
<td>5394</td>
</tr>
<tr>
<td>Uno Communications</td>
<td>9137</td>
</tr>
<tr>
<td>Utility Line Italia</td>
<td>9026</td>
</tr>
<tr>
<td>Verizon Italia</td>
<td>702</td>
</tr>
<tr>
<td>Welcome Italia</td>
<td>21056</td>
</tr>
<tr>
<td>Wifiweb</td>
<td>47927</td>
</tr>
<tr>
<td>Wind / It.Net</td>
<td>1267</td>
</tr>
<tr>
<td>Wolnext</td>
<td>49524</td>
</tr>
</tbody>
</table>

Table 5.1: Target Autonomous Systems
1. one IP address belonging to AS2597 address space,
2. one IP address belonging to MIX IXP address space (MIX AS number is 16004),
3. one IP address belonging to the target AS address space.

In other words, an IP address belonging to AS16004 address space is to be found between two IP addresses belonging to the two connected ASes, as shown in figure 5.1. If the address belonging to AS16004 is not found, then the two ASes are directly connected, that is, connected without passing via the MIX IXP (figure 5.2).
## Table 5.2: Submitted task

<table>
<thead>
<tr>
<th>Section</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>—</td>
<td>traceroute</td>
</tr>
<tr>
<td>Source Identification</td>
<td>Source Country</td>
<td>IT</td>
</tr>
<tr>
<td></td>
<td>Source AS</td>
<td>2597</td>
</tr>
<tr>
<td></td>
<td>Source Network Type</td>
<td>wifi</td>
</tr>
<tr>
<td>Destination Identification</td>
<td>Destination Country</td>
<td>31611, 21309, 197075, 48291, 31034, 28929, 20811, 8968, 3313, 20836, 9104, 15589, 8980, 39657, 137, 49605, 39729, 50809, 34695, 4589, 21176, 12850, 24880, 31319, 196761, 8265, 12874, 50112, 24608, 15605, 20924, 35193, 3302, 41497, 8224, 44160, 8928, 6760, 12779, 5602, 3549, 8527, 35574, 51185, 47408, 5396, 48634, 8816, 31076, 35612, 47358, 24915, 44513, 47217, 15720, 8234, 2597, 28716, 12637, 49367, 28999, 6734, 8922, 15830, 3269, 50316, 5392, 8612, 29449, 12835, 30848, 5394, 9137, 9026, 702, 21056, 47927, 1267, 49524</td>
</tr>
<tr>
<td></td>
<td>Destination AS List</td>
<td>31076, 35612, 47358, 24915, 44513, 47217, 15720, 8234, 2597, 28716, 12637, 49367, 28999, 6734, 8922, 15830, 3269, 50316, 5392, 8612, 29449, 12835, 30848, 5394, 9137, 9026, 702, 21056, 47927, 1267, 49524</td>
</tr>
<tr>
<td>Duration</td>
<td>—</td>
<td>30</td>
</tr>
<tr>
<td>Urgent Flag</td>
<td>—</td>
<td>false</td>
</tr>
</tbody>
</table>
CHAPTER 5. EXPERIMENTATION

Since the Android application implementing the traceroute tool provides only the list of traversed IP interfaces, but not the ASes to which they belong, a way to map the IP addresses to their AS had to be found. As stated in section 3.4.8, we used Isolario [42] data that provide an IP subnet prefixes to AS mapping. However, in this dataset not all entries are fully reliable, thus one could find some IP subnet prefixes which are mapped to multiple ASes. In such cases we used the whois command which implements the Whois protocol [26] to retrieve the correct AS to which an IP address belongs.

5.2 Experiment results

At the end of the experiment we found that all the ASes except one could be reached by our probes. The AS that could not be reached is Eurocall, which has AS number 196761 (from now on we will refer to it as AS196761). AS196761 announces only one subnet, 188.93.136.0/21, to which no route is available in AS2597 routing tables, as confirmed us by AS2597 network administrator. Therefore, every packet addressed to an IP address belonging to AS196761 was discarded and never delivered to destination.

Measurements show that the probed ASes are reached through three distinct paths:

- via MIX IXP (AS16004),
- via Consortium GARR network [56] (AS137), which is the Italian network that connects universities and research centers,
- via a Level 3 network [57] (AS3356), which is a big tier-1 ISP.

So, we can state that AS137 and AS3356 are directly connected to AS2597, not passing via MIX IXP. In the next sections we examine in detail the three previously mentioned cases.

5.2.1 Connections via MIX IXP

From the entire set of probed ASes, the 22 of them that are connected to AS2597 via MIX IXP are shown in table 5.3, which in the first two columns contains the ASes connected to AS2597 via MIX, and in the third and fourth
<table>
<thead>
<tr>
<th>AS Name</th>
<th>AS Number</th>
<th>Name</th>
<th>ASes reached</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acantho</td>
<td>21309</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Aruba</td>
<td>31034</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Brennercom</td>
<td>20811</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>BT Italia / I.Net</td>
<td>3313</td>
<td>Dada</td>
<td>39729</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NGI</td>
<td>35612</td>
<td></td>
</tr>
<tr>
<td>Digitel Italia</td>
<td>50809</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>E4A</td>
<td>34695</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Easynet Italia</td>
<td>4589</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ePress</td>
<td>24880</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Fastweb</td>
<td>12874</td>
<td>Active Network</td>
<td>197075</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clio</td>
<td>9104</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Engineering.it</td>
<td>21176</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lottomatica / SEDMultitel</td>
<td>35574</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Postecom</td>
<td>15720</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RAI</td>
<td>8234</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TWT</td>
<td>30848</td>
<td></td>
</tr>
<tr>
<td>Consortium GARR</td>
<td>137</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Interactive Network</td>
<td>41497</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Intercom</td>
<td>8224</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ITCgate Network</td>
<td>12779</td>
<td>Engineering.it</td>
<td>21176</td>
<td></td>
</tr>
<tr>
<td>KPNQwest Italia</td>
<td>5602</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Mainsoft</td>
<td>51185</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Me-Link</td>
<td>5396</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Seeweb</td>
<td>12637</td>
<td>Nexin Technologies</td>
<td>31076</td>
<td></td>
</tr>
<tr>
<td>Seflow</td>
<td>49367</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Telecom Italia</td>
<td>3269</td>
<td>D.T.S. - ReteIVO</td>
<td>49605</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mediaset</td>
<td>48634</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Postecom</td>
<td>15720</td>
<td></td>
</tr>
<tr>
<td>Telnet</td>
<td>5392</td>
<td>TopneT Telecomunicazioni</td>
<td>29449</td>
<td></td>
</tr>
<tr>
<td>Utility Line Italia</td>
<td>9026</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Welcome Italia</td>
<td>21056</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Wind / It.Net</td>
<td>1267</td>
<td>Clio</td>
<td>9104</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ICT Valle Umbra</td>
<td>15605</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mandarin Wimax Sicilia</td>
<td>47408</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3: ASes connected to AS2597 via MIX IXP
columns contains the ASes that could be reached by traversing the AS specified in the first two columns, if any. Notice that some of the ASes in the third and fourth column can be reached passing via more than one AS (e.g., AS21176 Engineering.it is reached either via AS12779 ITGate Network or via AS12784 Fastweb). This is possible because different subnets of the same AS may be reached via different ASes, as exemplified by figure 5.3.

Also notice that table 5.3 contains AS137. We found that for only one subnet (143.225.0.0/16, owned by Napoli University Federico II) traffic addressed to AS137 passes via MIX IXP instead of being directly delivered to AS137.

5.2.2 Connections via Consortium GARR

We found that 33 ASes of the submitted set can be reached via Consortium GARR network (AS137). Among these, some are directly connected to AS137, some are connected to AS137 via MIX IXP, and other ones are connected to AS137 via two more IXPs, VSIX North East Neutral Access Point (VSIX NAP [58]), whose AS number is 2593, and Nautilus Mediterranean eXchange Point (NaMeX [59]), whose AS number is 24796. Tables 5.4, 5.5, 5.6 and 5.7 show the ASes reached via AS137, respectively with a direct connection, via VSIX NAP IXP, via NaMeX IXP and via MIX IXP. Again, different subnets belonging to the same AS can be reached by multiple paths, traversing different IXPs.
5.2.3 Connections via Level 3

We found that 11 ASes of the submitted set can be reached via Level 3 network (AS3356). Some of them are directly connected to AS3356, others are reached by a more complex path, which traverses various ASes including some big transit networks, such as Cogent (AS174, [60]) and Tinet Spa (AS3257, [61]), and some tier-1 networks, such as TeliaNet (AS1299, [62]) and Telecom Italia Sparkle (AS6762, [63]). These paths are not presented in this thesis work, because we focused our attention on finding and showing the direct connections of AS2597, as this was the experimentation goal. However, we gave relevance also to the links of AS137 network because we found a huge number of them, and because the large majority of them are established via an IXP. Since links established via IXPs are in large part unknown, for the reasons explained in section 3.1, and since their discovery is one of the main reasons that led to the realization of the Portolan mea-

<table>
<thead>
<tr>
<th>AS Name</th>
<th>AS Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUB - Fondazione Ugo Bordoni</td>
<td>50112</td>
</tr>
<tr>
<td>Trentino Network</td>
<td>12835</td>
</tr>
</tbody>
</table>

Table 5.4: ASes directly connected to AS137

<table>
<thead>
<tr>
<th>AS Name</th>
<th>AS Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abilene</td>
<td>31611</td>
</tr>
<tr>
<td>Asdasd</td>
<td>28929</td>
</tr>
<tr>
<td>TELEmpianti</td>
<td>50316</td>
</tr>
</tbody>
</table>

Table 5.5: ASes connected to AS137 via VSIX NAP

<table>
<thead>
<tr>
<th>AS Name</th>
<th>AS Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>BT Italia</td>
<td>8968</td>
</tr>
<tr>
<td>Cloud Italia</td>
<td>15589</td>
</tr>
<tr>
<td>Fastnet</td>
<td>8265</td>
</tr>
<tr>
<td>Postecom</td>
<td>15720</td>
</tr>
<tr>
<td>Unidata</td>
<td>5394</td>
</tr>
</tbody>
</table>

Table 5.6: ASes connected to AS137 via NaMeX
<table>
<thead>
<tr>
<th><strong>AS Name</strong></th>
<th><strong>AS Number</strong></th>
<th><strong>ASes reached</strong></th>
<th><strong>Name</strong></th>
<th><strong>Number</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Asdasd</td>
<td>28929</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDLan</td>
<td>20836</td>
<td>Optima Italia</td>
<td>44513</td>
<td></td>
</tr>
<tr>
<td>Club Nautilus / Maki</td>
<td>8980</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comeser</td>
<td>39657</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fastnet</td>
<td>8265</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H3G</td>
<td>24608</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICTeam</td>
<td>20924</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Infracom Italia</td>
<td>3302</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interoute</td>
<td>8928</td>
<td>Aria</td>
<td>48291</td>
<td></td>
</tr>
<tr>
<td>Internet One</td>
<td>44160</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Itelsi</td>
<td>6760</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level IP Italia / Internet FR</td>
<td>8527</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level3 Communications</td>
<td>3549</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metrolink</td>
<td>28816</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NTRnet</td>
<td>47358</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OKCom / Teleunit</td>
<td>24915</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Planetel</td>
<td>47217</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retelit / e-via</td>
<td>28716</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Siportal</td>
<td>28999</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spin</td>
<td>6734</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Telecity Group Italia</td>
<td>15830</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T.Net</td>
<td>8922</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uno Communications</td>
<td>9137</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wifiweb</td>
<td>47927</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wolnext</td>
<td>49524</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.7: ASes connected to AS137 via MIX
 CHAPTER 5. EXPERIMENTATION

<table>
<thead>
<tr>
<th>AS Name</th>
<th>AS Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDLan</td>
<td>20836</td>
</tr>
<tr>
<td>Digitel Italia</td>
<td>50809</td>
</tr>
<tr>
<td>Enter</td>
<td>12850</td>
</tr>
<tr>
<td>Estracom</td>
<td>31319</td>
</tr>
<tr>
<td>IFOM</td>
<td>35193</td>
</tr>
<tr>
<td>Mandarin Wimax Sicilia</td>
<td>47408</td>
</tr>
<tr>
<td>Retelit / e-via</td>
<td>28716</td>
</tr>
<tr>
<td>Tiscali Italia</td>
<td>8612</td>
</tr>
<tr>
<td>Telecity Group Italia</td>
<td>15830</td>
</tr>
<tr>
<td>TWT</td>
<td>30848</td>
</tr>
<tr>
<td>Verizon Italia</td>
<td>702</td>
</tr>
</tbody>
</table>

Table 5.8: ASes reached via Level 3 (AS3356)

surement system, we preferred to give more emphasis on them rather than on links between ASes that form the core of the Internet.

Finally, as previously said, some of the ASes reached via AS3356 can also be reached via different paths traversing other ASes, depending on the probed subnet. Table 5.8 shows the ASes reached via AS3356.

5.3 Validation

In this thesis work only the validation of the implemented architecture shown in chapters 3 and 4 has been carried out. The validation and evaluation of the Android mobile application is exhaustively discussed in [8] and [9].

Validation was carried out by comparing the list of connections of AS2597 found in our experiment with the list of actual connections provided to us by the AS2597 network administrator.

The comparison confirmed us that, within the set of ASes submitted to the Portolan system, all the existing links were found. Therefore, we can conclude that Portolan system succeeds in translating the submitted tasks into microtasks containing sets of addresses that allow to produce a correct representation of the real topology of networks traversed by traceroute probes.
5.4 Evaluation

We carried out experimental evaluation as follows. First, we conducted an evaluation of the dataset found in our experiment by comparing the achieved results with datasets provided by the following research projects:

- CAIDA’s *AS Rank* dataset, which combines topological data collected by CAIDA’s Archipelago Measurement Infrastructure and Border Gateway Protocol (BGP) routing data collected by the Route Views Project and RIPE NCC.
- Isolario *Topology* dataset, which combines BGP routing data collected by RouteViews Project, RIPE NCC and PCH.
- DIMES *ASEdges* dataset.

Finally, we evaluated the Portolan system’s performance by analyzing the number of microtasks needed to carry out measurements and trying to make an estimate of the amount of time necessary to complete a task, depending on the number of available mobile devices participating actively in the Portolan system.

5.4.1 Dataset evaluation

For carrying out our evaluation we considered the following releases of the three previously mentioned datasets:

- June 2012 CAIDA’s *ASRank* dataset\(^1\),
- July 2012 Isolario *Topology* dataset\(^2\),
- April 2012 DIMES *ASEdges* dataset\(^3\).

All datasets are publicly available on the Web and are as close as possible to the period in which measurements were performed.

The compared links can be grouped in two sets, which were evaluated separately:

\(^1\)http://as-rank.caida.org/
\(^2\)http://www.isolario.it/results/Topologies/2012_02/RouteViews_RIS_PCH/Global/Topology.txt.bz2
\(^3\)http://www.netdimes.org/PublicData/csv/ASEdges4_2012.csv.gz
AS Name | AS Number
---|---
Consortium GARR | 137
E4A | 34695
Fastweb | 12874
ITGate Network | 12779
Level 3 Communications | 3356
Mainsoft | 51185
Seeweb | 12637
Telnet | 5392

Table 5.9: Isolario’s AS2597 links

- AS2597 links,
- AS137 links.

**AS2597 links evaluation**

First we intersected the set of probed ASes with the sets of AS2597 links provided by Isolario, CAIDA and DIMES. This operation allows to discard any AS that might be present in the publicly available datasets, but not probed in our measurement campaign.

Isolario *Topology* dataset has the largest intersection (highlighted by table 5.9), while CAIDA’s *ASRank* misses the Telnet (AS5392) link and DIMES includes only the Consortium GARR (AS137) and Level 3 (AS3356) links.

By comparing our dataset with Isolario *Topology* dataset (which is the most complete), we can state that our measurements could find all the Isolario links and, in addition, 16 previously unknown links, which are summarized by table 5.10.

**AS137 links evaluation**

In this case, first we took out the set of AS2597 from the set of probed ASes links. Then, we intersected the resulting set with the sets of AS137 links provided by Isolario, CAIDA and DIMES (the result is summarized in table 5.11, ordered by dataset). The first operation was done to ensure that the AS2597 links we found were not considered in this evaluation. As probes are sent from within AS2597, it would be impossible to find in the AS137 links
set one or more links present in the AS2597 links set, even if the link existed for both ASes.

By comparing our dataset with the three chosen datasets we found that our measurement system has been able to discover all the Isolario, CAIDA and DIMES links, except for AS29449 (TopneT Telecomunicazioni), AS9104 (Clio) and AS31076 (Nexin Technology) which were reached by other paths, and, in addition, 24 previously unknown links.

Dataset evaluation conclusion

By considering the new links discovered within our experimentation, we can conclude that the Portolan measurement system is able to find out links that are invisible to other measurement systems, as a consequence of performing measurements directly from within the networks of interest. The reason why the three other research projects did not discover the links shown in tables 5.10 and 5.12 is the same reason that led to Portolan system conception and

<table>
<thead>
<tr>
<th>AS Name</th>
<th>AS Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acantho</td>
<td>21309</td>
</tr>
<tr>
<td>Aruba</td>
<td>31034</td>
</tr>
<tr>
<td>Brennercom</td>
<td>20811</td>
</tr>
<tr>
<td>BT Italia / I.Net</td>
<td>3313</td>
</tr>
<tr>
<td>Digitel Italia</td>
<td>50809</td>
</tr>
<tr>
<td>Easynet Italia</td>
<td>4589</td>
</tr>
<tr>
<td>ePress</td>
<td>24880</td>
</tr>
<tr>
<td>Interactive Network</td>
<td>41497</td>
</tr>
<tr>
<td>Intercom</td>
<td>8224</td>
</tr>
<tr>
<td>KPNQwest Italia</td>
<td>5602</td>
</tr>
<tr>
<td>Mc-Link</td>
<td>5396</td>
</tr>
<tr>
<td>Seflow</td>
<td>49367</td>
</tr>
<tr>
<td>Telecom Italia</td>
<td>3269</td>
</tr>
<tr>
<td>Utility Line Italia</td>
<td>9026</td>
</tr>
<tr>
<td>Welcome Italia</td>
<td>21056</td>
</tr>
<tr>
<td>Wind / It.Net</td>
<td>1267</td>
</tr>
</tbody>
</table>

Table 5.10: Previously unknown AS2597 connections
<table>
<thead>
<tr>
<th>Dataset</th>
<th>AS Name</th>
<th>AS Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isolario</td>
<td>Asdasd</td>
<td>28929</td>
</tr>
<tr>
<td></td>
<td>FUB - Fondazione Ugo Bordoni</td>
<td>50112</td>
</tr>
<tr>
<td></td>
<td>Interoute</td>
<td>8928</td>
</tr>
<tr>
<td></td>
<td>Level3 Communications</td>
<td>3549</td>
</tr>
<tr>
<td></td>
<td>Registro.it</td>
<td>2597</td>
</tr>
<tr>
<td></td>
<td>TopneT Telecomunicazioni</td>
<td>29449</td>
</tr>
<tr>
<td></td>
<td>Trentino Network</td>
<td>12835</td>
</tr>
<tr>
<td>CAIDA</td>
<td>Asdasd</td>
<td>28929</td>
</tr>
<tr>
<td></td>
<td>FUB - Fondazione Ugo Bordoni</td>
<td>50112</td>
</tr>
<tr>
<td></td>
<td>Interoute</td>
<td>8928</td>
</tr>
<tr>
<td></td>
<td>Level3 Communications</td>
<td>3549</td>
</tr>
<tr>
<td></td>
<td>OKCom / Teleunit</td>
<td>24915</td>
</tr>
<tr>
<td></td>
<td>Registro.it</td>
<td>2597</td>
</tr>
<tr>
<td>DIMES</td>
<td>Asdasd</td>
<td>28929</td>
</tr>
<tr>
<td></td>
<td>CDLan</td>
<td>20836</td>
</tr>
<tr>
<td></td>
<td>Clio</td>
<td>9104</td>
</tr>
<tr>
<td></td>
<td>Interoute</td>
<td>8928</td>
</tr>
<tr>
<td></td>
<td>Level3 Communications</td>
<td>3549</td>
</tr>
<tr>
<td></td>
<td>OKCom / Teleunit</td>
<td>24915</td>
</tr>
<tr>
<td></td>
<td>Nexin Technologies</td>
<td>31076</td>
</tr>
<tr>
<td></td>
<td>Registro.it</td>
<td>2597</td>
</tr>
<tr>
<td></td>
<td>Telecity Group Italia</td>
<td>15830</td>
</tr>
<tr>
<td></td>
<td>Trentino Network</td>
<td>12835</td>
</tr>
<tr>
<td></td>
<td>Unidata</td>
<td>5394</td>
</tr>
</tbody>
</table>

Table 5.11: AS137 links in Isolario, CAIDA and DIMES datasets
### Table 5.12: Previously unknown AS137 connections

<table>
<thead>
<tr>
<th>AS Name</th>
<th>AS Number</th>
<th>Connection Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abilene</td>
<td>31611</td>
<td>via VSIX NAP</td>
</tr>
<tr>
<td>BTItalia</td>
<td>8968</td>
<td>via NaMeX</td>
</tr>
<tr>
<td>Cloud Italia</td>
<td>15589</td>
<td>via NaMeX</td>
</tr>
<tr>
<td>Club Nautilus / Maki</td>
<td>8980</td>
<td>via MIX</td>
</tr>
<tr>
<td>Comeser</td>
<td>39657</td>
<td>via MIX</td>
</tr>
<tr>
<td>Fastnet</td>
<td>8265</td>
<td>via MIX &amp; NaMeX</td>
</tr>
<tr>
<td>H3G</td>
<td>24608</td>
<td>via MIX</td>
</tr>
<tr>
<td>ICTeam</td>
<td>20924</td>
<td>via MIX</td>
</tr>
<tr>
<td>Infracom Italia</td>
<td>3302</td>
<td>via MIX</td>
</tr>
<tr>
<td>Internet One</td>
<td>44160</td>
<td>via MIX</td>
</tr>
<tr>
<td>Itelsi</td>
<td>6760</td>
<td>via MIX</td>
</tr>
<tr>
<td>Level IP Italia / Internet FR</td>
<td>8527</td>
<td>via MIX</td>
</tr>
<tr>
<td>Metrolink</td>
<td>28816</td>
<td>via MIX</td>
</tr>
<tr>
<td>NTRnet</td>
<td>47358</td>
<td>via MIX</td>
</tr>
<tr>
<td>Planetel</td>
<td>47217</td>
<td>via MIX</td>
</tr>
<tr>
<td>Postecom</td>
<td>15720</td>
<td>via NaMeX</td>
</tr>
<tr>
<td>Retelit / e-via</td>
<td>28716</td>
<td>via MIX</td>
</tr>
<tr>
<td>Siportal</td>
<td>28999</td>
<td>via MIX</td>
</tr>
<tr>
<td>Spin</td>
<td>6734</td>
<td>via MIX</td>
</tr>
<tr>
<td>TELEimpianti</td>
<td>50316</td>
<td>via VSIX NAP</td>
</tr>
<tr>
<td>T.Net</td>
<td>8922</td>
<td>via MIX</td>
</tr>
<tr>
<td>Uno Communications</td>
<td>9137</td>
<td>via MIX</td>
</tr>
<tr>
<td>Wifiweb</td>
<td>47927</td>
<td>via MIX</td>
</tr>
<tr>
<td>Wolnext</td>
<td>49524</td>
<td>via MIX</td>
</tr>
</tbody>
</table>
realization: as measurements are performed from fixed monitors close to the Internet core, they can not show the complete topology of the periphery of the Internet. Probing the network from the edges of the Internet (that is, from within the networks of interest) allows to acquire a more complete description of the Internet itself.

5.4.2 Performance evaluation

The performance in terms of a task’s execution time is hard to estimate, as many factors play a significant role in the execution of a task:

- the number of microtasks in which a task is translated,
- the number of targets per microtask,
- the number of mobile devices available for task execution,
- the aims of the human user when submitting a task.

The task submitted within this experimentation was translated into 1556 microtasks of 100 targets each. The execution of a single traceroute by mobile application takes approximately from 5 to 8 seconds. Thus, a microtask is completed in approximately 10 or 15 minutes. Therefore, the task completion would require, in the best case, 10 days of non stop execution performed by a single mobile device. However, since energy saving is a main concern to us, we decided to limit the executions of microtask performed by a single mobile device to one or two per day (as stated in [8], the execution of a single microtask reduces the battery level of approximately 1%). Hence, the primary need is to have multiple mobile devices participating in a task execution (e.g., if 1000 mobile devices were available the task could have been completed in two or three days). Then, optimizations could be done for reducing the task execution time, however they are strictly dependant on the aims of the human user when she submits a task.

In our case, our aim was to discover the links between two ASes, so when a connection to an AS was discovered, the target IP addresses belonging to that AS were cancelled. However, if no link between the two ASes was discovered the whole target IP addresses set had to be probed.
Chapter 6

Conclusion and future work

In this thesis work we addressed the problem of designing and implementing an efficient and scalable architecture for the Control Plane of the Portolan Internet Topology Measurement System. The Portolan system is a new Internet topology measurement infrastructure whose aim is to discover the Internet structure and properties by probing the network with a traceroute-based tool run by a huge number of mobile devices. The desired architecture needs to satisfy the following requirements. First, it should be able to coordinate a great number of mobile devices in a scalable fashion and to assign them Internet topology measurement tasks. Further, it should provide to researchers an interface for specifying measurement campaigns to be performed by mobile devices.

The task has been accomplished in four separate phases. Firstly, we designed a general interface for specifying measurement tasks to be executed by mobile devices. This interface allows a human user to identify the mobile devices that are authorized to perform the measurements, as well as the targets of the measurement itself. Moreover, optional parameters depending on the specific measurement type have been added. Secondly, we focused our attention on traceroute measurements and formalized the specification of a traceroute measurement task, according to the previously designed interface. Thirdly, we designed and implemented the architecture of a server infrastructure for accomplishing two separate tasks. From the human user’s perspective the server infrastructure acts as a medium for specifying measurement campaigns and being notified of a campaign completion. From the mobile devices point of view the implemented architecture act as a co-
ordinating and managing unit, assigning them tasks to be performed and collecting results. The core of the architecture contains the logic for translating the campaigns specified by human users into small jobs that mobile devices can execute in a transparent fashion for device's owner. At this stage, the issues of scalability and mobile devices availability and NAT reachability were addressed and solved with the introduction of decentralized coordinating units named *proxies* as well as of a polling mechanism. The designed architecture has been integrated with third-party components standardized by the Open Geospatial Consortium: the Sensor Planning Service (SPS), which allows a human user to specify measurement tasks and the Sensor Observation Service (SOS), which collects data and provides them to human user. Both the two services are implemented by 52North. Finally, we conducted an experimental measurement campaign with the aim of validating and evaluating the implemented framework. Achieved results showed that the system succeeds in collecting data that reflect a previously obtained ground truth. By comparing collected data with publicly available datasets provided by popular research projects, we could verify the effectiveness of the Portolan Internet Topology Measurement System innovative *bottom-up* and *bottom-to-bottom* approaches: the conducted experimentation was able to discover previously unknown links at the AS-level topology.

For the forthcoming future an intensive system’s robustness testing phase is scheduled, in order to obtain a stable version of the implemented framework, including the Android application, ready for the release. After the test phase, a distribution of the application within the Italian scientific community is planned and a measurement campaign for discovering the Italian Internet infrastructure will start. In the meanwhile, versions of the mobile application for the most popular mobile platforms (e.g., iOS or Windows Phone) will be developed. Moreover, new measurement types, either on the Internet (e.g., bandwidth measurement between to end points) or on the mobile infrastructure (e.g., received signal strength), are in a feasibility study phase.
Bibliography


97
BIBLIOGRAPHY


[34] M. Botts, “OpenGIS Sensor Model Language (SensorML) 1.0.0.”


[40] “International Institute for Geo-Information Science and Earth Observation.”
http://www.itc.nl/.


