Analysis of the Internet topology

Jérémie Leguay
jeremie.leguay@rp.lip6.fr

LIU[1] LIP6[2] LIAFA[3]

Supervisor: Kavé Salamatian
Examinor: Nahid Shahmehri

4th February 2004

[1]Linkoping University
[2]Laboratoire d'Informatique de Paris VI
[3]Laboratoire d'Informatique Algorithmique, Fondements et Applications - Paris

Contents

Acknowledgements

I would like to thank Kavé Salamatian from LIP6 for bringing the subject to my attention and for being the instructor of this master's thesis.

I would like to express my thanks to Timur Friedman from LIP6 and Matthieu Latapy from LIAFA and Kavé Salamatian for their ideas, their advices, their time and their support.

I am also thankful of my co-workers at LIP6 for their kindness and the pleasure I had to work with them.

Introduction

The Internet is the world wide network that everyone use everyday to use applications like web browsing or e-mail from everywhere around the world. We may think that this communication ”tool” designed by engineers in the seventies is now well known and that not secrets remain about its functioning and structure. Unfortunately, lots of open problems still exist about this fascinating network in order to understand how it works.

This master's thesis will concentrate first on one open problem that concern the Internet: the topology. We will perform a state of the art in order to see the existing means for acquiring the topology and the studies already done on it. Then, we will study in detail this topology on the data provided by an American project. And finally, still with the same data, we will try to characterize of paths and their connections with the topology in this network.

Chapter 1

State of the art

The first thing to understand is why acquire and study the Internet topology is so important. Because we don't know how this world-wide network evolve and is exactly structure it is hard to simulate some phenomena or monitor the network. Indeed, it is hard for system designers to perform some tests of their applications or protocols before deployment. We will see that some topologies generators that can be used to fill this lack. Other points that could motivate this comprehension are network management and siting. The first concern all the monitoring tasks on the network and the second deal with the localization of some phenomena in the network. Note that it can be also use to detect the result of natural disasters, wars or DoS[1] attacks. In [5], authors were able to observe a loss of network connectivity in Yugoslavia during the NATO bombing in 1999. In this chapter, we will first detail how the Internet is organized, then we will describe the way to acquire its topology and finally we will try to summarize works that have been done on the analysis of this topology.

1.1 Organization of the Internet

1.1.1 History and global structure of the Internet

The Internet is born because of the request in 1962 of the US Air force who has asked to a small research group to work on the creation of a nuclear attack resistant network. The concept of this network was based on a decentralize system in order to survive if one or several nodes were destroyed. The first version of the Internet called ARPANET was achieved in 1972. It was later partition in two parts, the military one and the public one which has become the Internet. The way it has grown up [6] explains many things of its actual structure. The Internet is now a huge network divided into domains that are connected together. Each domain is a collection of hosts interconnected via transmission and switching facilities. Domains that share their resources with

[1] Denial of Service

other domains are called network service providers (or just providers). Domains that utilize other domain's resources are called network service subscribers (or just subscribers). A given domain may act as a provider and a subscriber simultaneously. In a more precise way, domains are called autonomous systems (AS) and are composed of a network or a group of networks. Each AS is under a common administration and has its own routing policy. The relations between AS follow commercial or political agreements, which specify in particular the way packets are transmitted between them. AS interconnects them at peering points. One peering point is Parix[2], an exchange point in Paris that connects among other things Tiscali (AS 3257), Telia (AS 1299), Easynet (AS 4589), NERIM (AS 13193) and PROXAD (AS 12322). Figure 1.1 shows a map of the AS topology. This map and lots of other beautiful maps are referenced by a research project called Cyber-Geography[3].

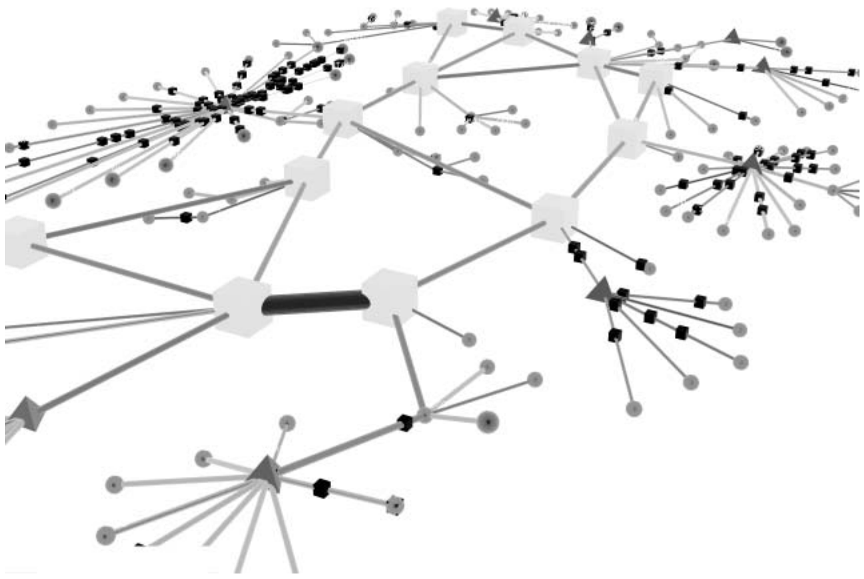

Figure 1.1: Interconnection in the Internet

1.1.2 Addresses allocation

This part will detail the way IP[4] are managed on the Internet, it is important because we will deal in section 2 and section 3 with an IP graph.

[2]http://www.parix.net/
[3]http://www.cybergeography.org/atlas/topology.html
[4]Internet Protocol

Both IPv4 and IPv6 addresses are assigned in a delegated manner. Users are assigned IP addresses by Internet service providers (ISPs). ISPs obtain allocations of IP addresses from a local Internet registry (LIR) or a national Internet registry (NIR), or from their appropriate Regional Internet Registry (RIR) like:

- APNIC (Asia Pacific Network Information Center) for the asia and the pacific region
- ARIN (American Registry for Internet Numbers) for north America and sub-Sahara Africa
- LACNIC (Regional Latin-American and Caribbean IP Address Registry) for Latin America and some Caribbean Islands
- RIPE (Réseaux IP Europens) - Europe, the Middle East, Central Asia, and African countries located north of the equator

The IANA[5]'s role is to allocate IP addresses from the pools of unallocated addresses to the RIRs according to their established needs.

We will just describe the organization of the IPv4 address space because we will not deal with the ipv6 protocol. IPv4 addresses are partitioned into 5 classes described in the RFC 1466:

Class	Net mask	Address range
A	255.0.0.0	0.0.0.0 - 127.255.255.255
B	255.255.0.0	128.0.0.0 - 191.255.255.255
C	255.255.255.0	192.0.0.0 - 223.255.255.255
D	-	224.0.0.0 - 239.255.255.255
E	-	240.0.0.0 - 247.255.255.255

Figure 1.2: IPv4 space

This system is being replaced by the Classless Inter-Domain Routing (CIDR) (RFC 1517 to RFC 1520) system to face the inefficient use of the address space. The inefficiencies are mainly in the block assignments. For example, if an institution gets a class C IP range and only use a hundred of those, there are 154 unused and unavailable addresses. Another problem that motivates the CIDR system is the overtaxing of the routing tables. If an organization handles several consecutive class C IP ranges, routers have to store one route per IP range. With the CIDR system, an IP address range might look like this 212.80.191.0/24. This means that the first 24 bits in the address are used to identify the network while the remaining 8 bits are used to identify the host.

This approach is in use nowadays and is being encouraged. Implementation of the CIDR has been vital for the growth of the Internet, allowing more organizations and users to take advantage of this increasingly global networking and

5

information resource. Those problems have been solved by the IPv6 standard but IPv4 is still dominant.

We have to note that in IPv4, some IP addresses are reserved for special purpose (see Figure 1.3).

Use	IP range
Loopback	127.0.0.1
Private networks (RFC 1918)	10.0.0.0/8, 172.16.0.0/12 and 192.168.0.0/16
Reserved for future use	218.0.0.0/8 - 223.0.0.0/8 et 240.0.0.0/8 - 255.0.0.0/8
Reserved multicast	224.0.0.0/8 - 239.0.0.0/8
broadcast	255.255.255.255

Figure 1.3: Special addressed in IPv4

1.1.3 Routing

We will use active measurements (see section 1.2),so packets send to perform this task are under the law of routing policies that is one reason to explain a bit how information is route in the Internet. The other reason is the fact that we will study in section 3 paths in the Internet that are the consequences of these same routing policies.

The main function of routers is to make packets arrive at their destination. Each router has its policy to treat information. This policy can depends on commercial agreements, on the bandwidth of adjacent links, on congestion of certain parts of the networks, etc... The policy of a router is generally partition into two parts: a static one set by an administrator and a dynamic one handle by a routing protocol. There are two families of routing protocols use on the Internet depending on the kind of routers they are installed on. Indeed, there are routers called *border routers* that handle traffic between AS and routers called *intern routers* that handle traffic in a subnet as shown on 1.4.

The Border Gateway Protocol (BGP) is use to exchange routing information between AS on *border router* while an Interior Gateway Protocol (IGP) such as RIP[6], OSPF[7] or IS-IS[8] is use to exchange routing information within AS on *internal router*.

RIP sends routing-update messages at regular intervals and when the network topology changes. When a router receives a routing update that includes changes to an entry, it updates its routing table to reflect the new route. The metric value for the path is increased by 1, and the sender is indicated as the

[6]Routing Information Protocol

[7]Open Shortest Path First

[8]Intermediate System to Intermediate System

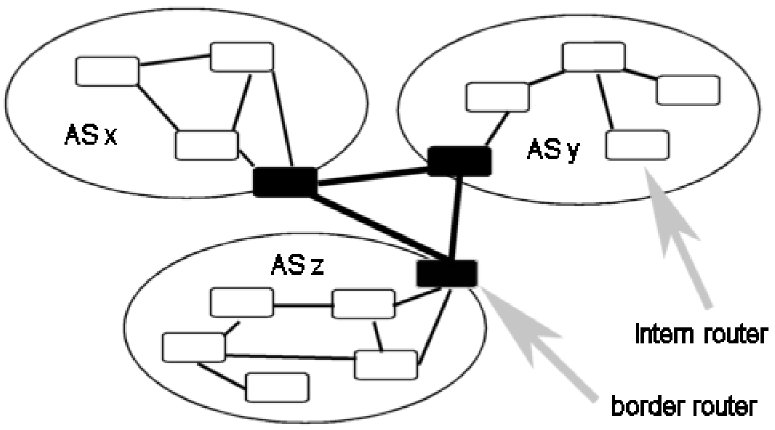

Figure 1.4: The different types of routers.

next hop. RIP routers maintain only the best route (the route with the lowest metric value) to a destination. This protocol just considers distance as the number of hops between two hosts but it does not take into account delay or bandwidth of links. This protocol is now outdated but it was one of the first.

OSPF is more efficient than RIP and starts to replace it slowly. In opposite to the RIP protocol, routers don't propagate their distances in term of hops but the state of their adjacent links. Each router is able to compute the best path by knowing a map of the network. Furthermore, less information is transmitted over the network.

The main thing to remind here is that routing protocols are design to provide the best path to packets, meaning the shortest path inside AS and the faster or most efficient between them. Remind also that there is static information that can influence routing. In particular, at the AS level, commercial agreements called *peering sessions*.

1.1.4 Levels in the topology

Since we study the physical topology of this world-wide network, let's describe levels of abstraction that are commonly studied with the help of Figure 1.5.

Indeed, the topology of Internet may be studied as a graph at three different levels. The first level concerns the way AS are interconnected. The second level is the interconnection of routers on the Internet. It represents cables, satellite or radio links, etc... This physical infrastructure is the one over which information is routed. Finally, at a kind of microscopic level, one may consider the IP graph, composed of the interfaces (of routers) which exchange information since each interface owns an IP on the Internet. All these three levels of the Internet topology are obviously strongly linked.

We will see that each level have been studied in section 1.2. In the exper-

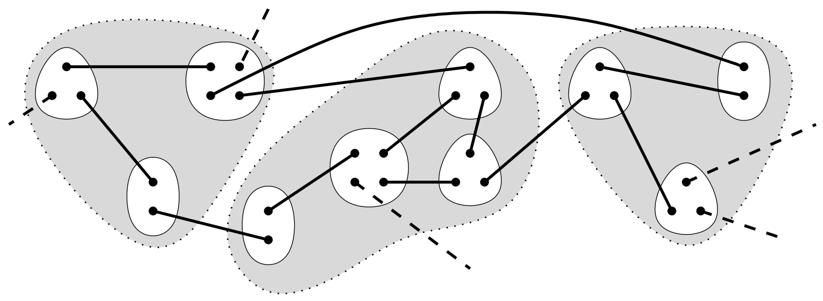

Figure 1.5: The three levels of the Internet architecture. Black dots represent interfaces, blank shapes stand for routers and shaded areas for AS. The (plain or dotted) lines correspond to physical links (always between two interfaces).

iments we have realized (see section 2 and 3), we will mainly deal with the interface level and a bit with the AS one.

1.2 Existing acquisition methods and research projects

Since each acquisition methods of the network topology acts usually at only one level this part will treat each level separately.

1.2.1 At the AS level

One of the most efforts to get snapshots of the Internet topology at this level is the Route Views project from the University of Oregon[9]. This project uses BGP information from many BGP routers to collect routes between AS. Because a BGP router sees a set of routes by receiving information from several other BGP routers. Historically, the Route Views project was originally backed by telecommunication operators in order to monitor the visibility of their prefixes and their AS space. But, Route Views data have served many other interesting projects. For example, it has been used for AS path visualization or to study IPv4 address space utilization [10]. All papers that deal with the topology of Internet at the AS level are based on those data. But note that this is not the only source of AS information. For instance, we can get AS information from RIPE.

1.2.2 At the IP level

This is at this level that the most number of techniques or tools exist. There are several methods to acquire the IP graph as [27] said:

[9]http://www.routeviews.org

- *SNMP*[10]: With this protocol, each router can provide a list of its neighbors. The main advantage of this method is that it generates a very low overhead on the network. Unfortunately, in most of case, SNMP is not available because routers don't support this feature or allow a restricted access to it.

- *Path forwarding*: This is the technique use by the *traceroute*[11] tool. The *traceroute* tool is based on the use of the TTL (Time To Live) of IP packets. Recall that the TTL is an IP header field that is designed to prevent packets from running in loops. Every router that handles a packet subtracts one from the packet's TTL. If the TTL reaches zero, the packet has expired and is discarded. *traceroute* depends on the common router practice of sending an ICMP[12] time exceeded message [19], back to the sender when this occurs. *traceroute* first sends a packet to the destination with a TTL value 1 and iterate this operation with increasing values of the TTL. This gives the list of intermediate interfaces on the way to the destination. Some implementations of this tool can be found at http://www.traceroute.org.

- *broadcast ping*: A ping is just an ICMP Echo request [19] to an host in order to know if it is alive or not. This simple tool can also be used to send a ping request to a broadcast address in order to get answer of all the alive machines in the subnet. There are some heuristics in [27] to discover subnet netmask and to get information about the topology.

- *DNS*[13] *transfer zone*: Most DNS servers respond to a Zone Transfer command by returning a list of every name in the domain.

[27] comes to the conclusion that the most efficient is the path forwarding method if it is done at a large scale. A lot of academic projects have dealt with this method:

- *Mercator*[14]: Mercator is a program from the SCAN project at the University of Southern California that uses *traceroute* like methods to infer the Internet topology. This program works on a single computer and does not need any input since it uses random address probing to explore the all IP address space. It also uses source-route capable routers wherever possible to enhance the quality of the resulting map. With this technique the sender of a packet can specify the route that a packet should take through the network. This method allows discovering cross links. Unfortunately, few routers support this feature. In [15], they found that approximately 8% of all Internet routers were capable of source-routing.

[10]Simple Netwok Management Protocol
[11]http://www.traceroute.org
[12]Internet Control Message Protocol
[13]Domain Name Server
[14]http://www.isi.edu/scan/mercator/mercator.html

- *Nec*: Nec[15] is a software developed at the University Louis Pasteur[16] by Damien Magoni. It uses public *traceroute* servers to acquire quickly a short snapshot of the topology.

- *Skitter*: Skitter is a project from CAIDA [17]. It provides to the research community a free access to the data collected. This measurement platform is composed of about thirty servers which perform traceroutes toward thousands of destinations every day. Destinations are chosen to span all the CIDR prefixes in use. The CIDR prefixes in use can be found in BGP tables (see section 1.2.1).

To our knowledge the Skitter [12] project is at the state of the art since it is the most ambitious. Our data for the studies of the section 2 and section 3 come from Skitter. In section 2.1.1, more detailed about Skitter can be found.

1.2.3 At the router level

This level of study strongly depends on the interface level because the only mean to acquire it is to start from the interface topology and to infer it with the help of some heuristics. We forgot here the possibility to ask ISP for this information because of confidentiality reasons. For example the ones described in [15]. The main aim of this technique is to merge interfaces into routers. The trick is to discover if two interfaces belong to the same router or not. Here is a brief list of the heuristics developed in [15], let's analyze two of them:

- Usually routers have an interface called the loopback interface from which they use to respond to *traceroute* probes. So, to know if two interfaces belong to the same router, it is possible to send to each of them an UDP[18] packet to an unused port and to compare the address source of the ICMP port unreachable packets. If they are the same, they belong to the same router.

- Interfaces have usually DNS names and are often called like X.1.Y.domain.com and X.2.Y.domain.com for example in the case of two interfaces. So you can guess by looking at their name if they belong or not to the same router. This method is not so accurate because DNS content is often obsolete and this convention of naming is not so used nowadays.

1.2.4 Cross-level methods

Cross layering is possible but difficult and a lot of mistakes are possible. We have seen in the previous paragraph that from the IP graph you can infer the router one by using some heuristics. Some tools have implemented such techniques

[15]http://clarinet.u-strasbg.fr/ magoni/nec/
[16]Strasbourg - France - http://www-ulp.u-strasbg.fr
[17]the Cooperative Association for Internet Data Analysis - www.caida.org
[18]User Datagram Protocol

[11] but they don't obtain accurate results. Mercator and Nec have also this feature. In the same way, it is possible to go from the IP level to the AS graph with the help of whois databases like RADB [19] or a continental registry like RIPE [20] for Europe but it is a difficult task. The IP graph and the information of these databases have to be acquired at the same period and many problems appear like the fact that some IP don't belong to any AS and some routers have different interfaces belonging to different AS. Furthermore, note that obsolete data appear in whois databases.

1.3 Limits of acquisition methods

1.3.1 Dynamical aspects

As [25] said, the Internet is *an immense moving target* so measurements are hard to perform. In addition to the size of this large network, this phenomena is emphasized by its structure. Indeed, the Internet has not been built in a centralized way, it allows heterogeneous networks with very different administrative policies to interoperate. Consequently, researchers have resigned to believe that it is impossible to have a complete snapshot of the network at a certain moment. This is a quite fresh research subject and it does not exist a solid solution yet. Let's see some ways to analyze this dynamic and complex system:

- By collecting a huge amount of data on a certain period, the shortest as possible, like Skitter does. Then one can start to study those data and assume that they are representative of reality as we did in our study (see section 2). The entire problem here is to choose the period. Nobody has the answer. One can say that the graph is quite stable during a month so you can aggregate data collected during a month, another can argue that very important changes can occurs suddenly like new links or new routing policy that change radically the graph.

- By studying the dynamic of this graph and its evolution on a small data collection by making the assumption that they are representative. Nec is an example of a tool use to study the evolution of the core of the network for instance.

1.3.2 Quality of the data

All the methods previously described in section 1.2 related to the acquisition of the Internet topology are disturbed by some distortion phenomena. At the AS level, many papers point out problems with instabilities of BGP [16], at the Interface level the path forwarding method suffers also of various issues and finally at the router level the heuristics used don't work in all the case. Because all

[19] Routing Assets Database

[20] Réseaux IP Européens

the data studied in this thesis are the result of millions of *traceroute* executions, let's concentrate on problems inferred by the path forwarding method.

As seen in section 1.2.2, the path forwarding method triggers ICMP errors along a path to discover it by playing[32] with the TTL IP field. Unfortunately, many tricky problems go against this like:

- The desire to keep confidential the topology or other reasons that motivate administrators to configure routers to not answer to *traceroute* probes.
- Route changes or parallel load balanced links that make packets of a *traceroute* probe take different paths. Leading to the creation of bad links in the IP graph as shown in Figure 1.6.

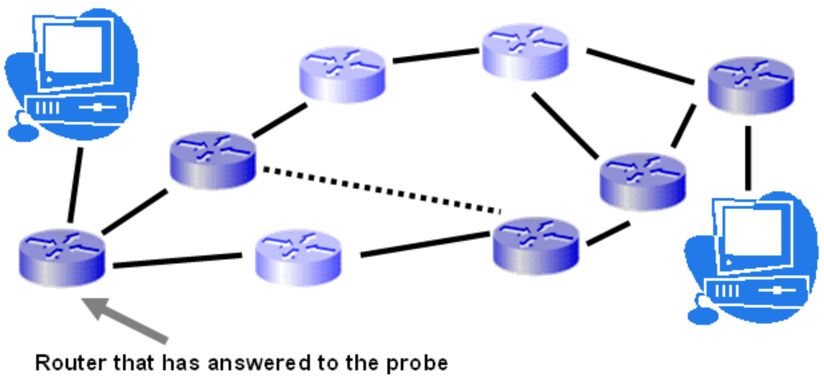

Figure 1.6: Fake links that can be seen with *traceroute*.

- Intrusion detection systems triggered alarms because of the traffic generated. Indeed, the traffic generated may looks like to port scan probes if UDP is use. The solution is to use only ICMP.
- Bad answers to *traceroute* probes are sent. Special IPs appear in the traces. Due to the fact that:
  - Sometimes NAT[21] is not well performed for the sender address in IP packets. As a consequence, packets circulate in the Internet with a private IP for the sender. Packets like this generally reach their destination because routers don't take care about the sender address.
  - Some administrators configure routers to reply with their own private IP because they want it to be visible as a hop in a route but not to be reachable from the outside of the subnet.
- The fact that routers contain bugs. Some examples of *traceroute* related problems can be found in appendix 3.3.

[21]Network Address Translation

Finally, other facts that have an impact on the quality of data but at a more structural level because this method study the topology at the IP level:

- Figure 1.7 shows that the logical topology at the IP level is a bit different than the topology at the router level. The paper of Renata Teixeira[26] shows this.

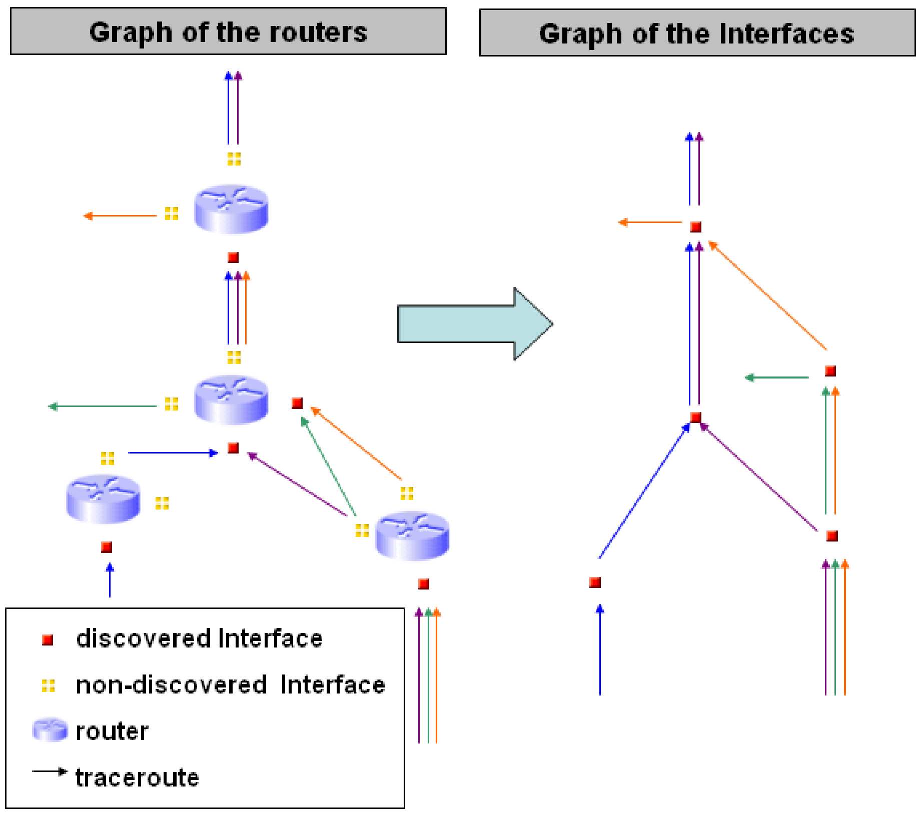

Figure 1.7: The difference between the physical topology and the logical one at the IP level.

- Packets sometimes enter ATM[22] or MPLS[23] networks and are encapsulated. It does not allow the path forwarding method to discover intermediate hops. It seems that it is a reason for the existence of nodes with very high degrees in the IP graph (see chapter 2).

[22]Asynchronous Transfer Mode
[23]Multiprotocol Label Switching

1.4 Existing analysis of the Internet topology

1.4.1 Properties already observed

Since this topology is a graph, all the analysis made on this graph deal with graph theory.

The first thing the community has pointed out is the existence of power laws between properties of this graph. This was introduced by the Faloutsos brothers in [9] and impressed the network community. The first power-law relationship they found is between the out-degree of a node v (d_v) and its rank (its index in order of decreasing outdegree) r_v: $d_v \alpha^{24}$ r_v^R. It reflects the way domain connects. The second power-law relationship is between the outdegree d and its frequency f_d: $f_d \alpha d^O$. This shown that the degree distribution on the Internet is not arbitrary, it also shows that there is a huge number of nodes with a very small outdegree and some nodes with a high degree. Finally, the third one they found links the eigen value λ_i and the order i: $\lambda_i \alpha i^k$. This last law allows to distinguish the differences between graph families.

There are several remarks on this. First, the computation of the exponent is subject to large error [4]: small variation on the distribution has a huge impact on the exponent computation. This is why focusing on the exact measured exponent does not seem to be a reasonable aim for model designers. Second, this power-law seems to exist because of the way the graph has been explored [24].

The clustering coefficient has also been studied. The clustering coefficient of a node gives information about connectivity between neighbors of a node. It is given by the following formula:

$$C(u) = \frac{2 * E_u}{k_u * (k_u - 1)}.$$

Here, k_u is the number of neighbors of the node u and E_u is the number of existing links between the neighbors of u. If this coefficient is equal to 1, it means that all possible links between neighbors of a node exist. Meaning that the fewer neighbors are connected together, the less is the clustering coefficient. Some studies like [21] have shown that at the AS level the average of the clustering coefficient is quite high (it is at least higher than for traditional random graphs).

This phenomena and the fact that there is a degree distribution in power-law may be an explanation for the small-world phenomena observed on the graph of Internet [21]. Intuitively, having a graph that fits a small-world phenomena means that the length of a path between two nodes chosen randomly is quite short, in the same order of $log(N)$ if the graph has N vertices. This is the case in many complex networks [22] like social networks [20] or in the human language [18]. Small-world graphs are usually obtained by rewiring edges from a regular graph. [31] provides a small JAVA[25] applet that illustrates this phenomena.

[24]"'Proportional to"'

[25]http://java.sun.com

In order to understand why this graph is very specific, let's compare it to a random graph:

Property	Random graph	the Internet graph
Degree distribution	Poisson	Power law
Clustering	low	high

Figure 1.8: Comparison of the graph of the Internet and a random graph

1.4.2 Existing models of generators

There are several topology generators available to the research community. Some of them mainly aim to generate random topologies, others aim to imitate the hierarchical properties of the Internet, others aim to reproduce degree-related properties of the Internet and some use some engineering or economic related constraints.

Random graph generators

Erdos-Renyi [8] developed one of the first topology generators. It creates a random graph by assigning a uniform probability for creating a link between any pair of nodes. Later, Waxman [34] improved the Erdos-Renyi random graph model by including network-specific characteristics such as placing the nodes on a plane and using a probability function to interconnect two nodes which is dependant on the distance that separates them in the plane.

Structural generators

In this category of generators, models focus on reproducing the hierarchical structure of the topology of the Internet.

The Transit-Stub [3] model is one the most famous. The background idea is that the Internet can be viewed as a collection of interconnected routing domains. Each routing domain in the Internet can be classified as either as stub domain or as transit domain. A domain is a stub domain if the path connecting any two nodes u and v goes through that domain only if either u or v is in that domain. A transit domain does not have this restriction. So, in the Transit-Stub domain, a connected random graph is first generated using the Waxman method. Then, each node in the graph represents an entire Transit domain. Each transit domain node is expanded (see Figure 1.9) to form another connected random graph, representing the backbone topology of that transit domain. Next, for each node in each transit domain, a number of random graphs are generated representing Stub domains that are attached to that node. Finally, some extra connectivity is added, in the form of *back-door* links between pairs of nodes, where a pair of nodes consists of a node from a transit domain and another from a stub domain, or one node from each of two different stub domains.

Another generator that implements models trying to imitate the structure of the Internet is Tiers [7]. The generation model of Tiers is based on a three-level hierarchy aimed at reproducing the differentiation between Wide-Area, Metropolitan-Area and Local-Area networks comprising the Internet.

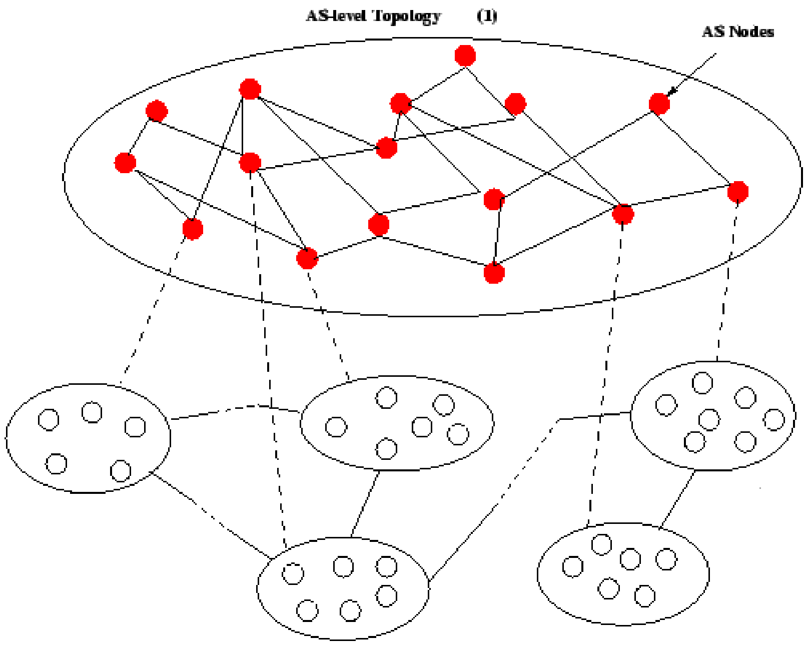

Figure 1.9: The Transit-Stub generator.

Degree based generators

PLRG[26] [28] is a generator which tries to reproduce the connectivity properties of Internet topologies as reported in [9]. This generator initially assigns node degrees from a power-law distribution and then proceeds to interconnect them using different rules.

The Albert and Barabasi model

This model [1] builds a graph by adding nodes one by one: when inserted, the node is linked to a fixed number of already present nodes chosen randomly, proportionally to their current degrees. This is the *rich get richer* or the *preferential attachment* principle.

Others models that take into account the correlation between the connectivity of the network and the density of the population have been established in [23].

[26] Power-Law Random Graph generator

Here are two sofwares that implements the previous algotithms :

- *GT-ITM*: the Georgia Tech Internetwork Topology Models generator from the Georgia institute of technology [30].
- *BRITE*: the Boston University Representative Internet Topology gEnerator [13].

The main question that remains open is how to validate a model of the Internet? One way may be to ask why are we looking for models of the Internet? The answer to this question will certainly help to determine proper parameters to measure.

Chapter 2

Analysis of the Skitter graph

In this chapter, we will detail the observations we made on the data provided by the Skitter project of the CAIDA group after a presentation of the methodology adopted.

2.1 Methodology

We have seen in section 1.3.1 that there are two methods to analyze data collected with the path forwarding method. A quasi-static one consists to consider data on a short period and a dynamic one considering data on a large period. We have concentrated ourselves with the static method by taking data collecting by Skitter during a day. We will describe in this part how Skitter works and what we can expect of its data and we will give information about the tools we have used to analyze these data. Finally, we will end with some explanations about implementation of some algorithms.

2.1.1 Skitter

We have seen in section 1.2.2 that Skitter performs millions of traceroutes per day from their servers. CAIDA currently maintains thirties Skitter hosts all over the world. However, not all Skitter monitors are running the full destination set at all times. So we have chosen a day that contains data for 23 servers. Each server performs the measurements described in 2.1.1 by targeting a destinations defined in 2.1.1 using the path forwarding method with ICMP[1].

Servers' location

2.1 and 2.2 show the location of the servers.

[1] http://www.caida.org/tools/measurement/skitter/packets/

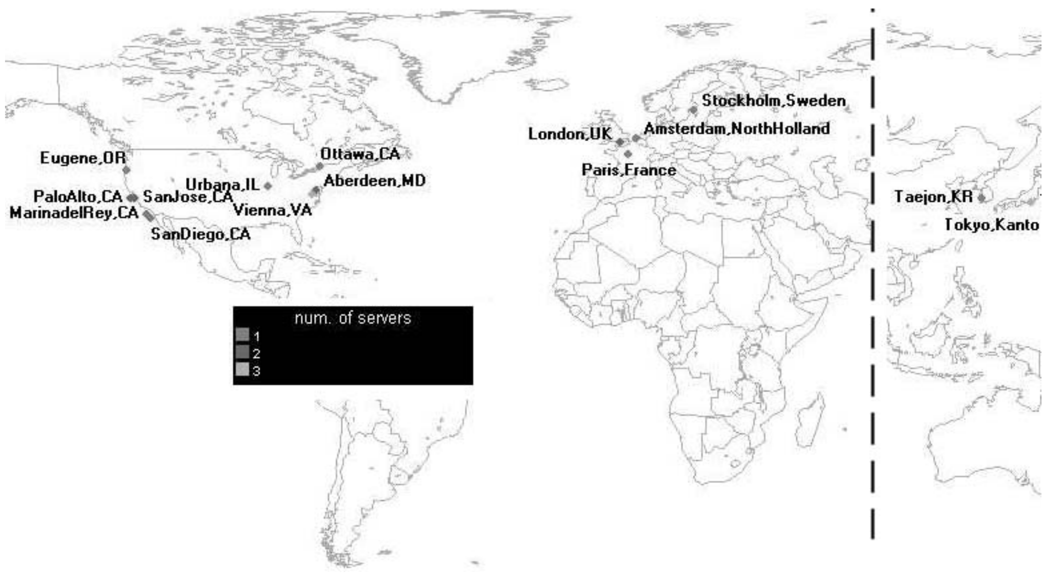

Figure 2.1: Positions of Skitter servers

Name	Town		Name	Town	
a-root	Herdon,VA	US	apan-jp	Tokyo	Jp
b-root	Marina del Re	US	cdg-rssac	Paris	Fr
champagne	Urbana,IL	US	d-root	College Park,MD	US
e-root	Moffett Field,CA	US	f-root	Palo Alto,CA	US
g-root	Vienna,CA	US	h-root	Aberdeen,MD	US
i-root	Stockholm	S	iad	Washington,DC	US
k-peer	Amsterdam	Nl	k-root	London	UK
kaist	Taejon	Kr	lhr	London	UK
m-root	Tokyo	Jp	mwest	San Jose,CA	US
nrt	Tokyo	Jp	riesling	SanDiego, CA	US
sjc	San Jose, CA	US	uoregon	Eugene, OR	US
yto	Ottawa	Ca			

Figure 2.2: Lists of Skitter' servers use for our studies.

Measurements

Here is a list given by [12] of the measurements done by Skitter:

- *Forward IP Paths*: Skitter records each hop from a source to many destinations. By incrementing the "time to live" (TTL) of each IP packet header and recording replies from each router (or hop) leading to the destination host.

- *Round Trip Time*: Skitter collects round trip time (RTT) along with path (hop) data. skitter uses ICMP echo requests as probes to a list of IP destinations.
- *Persistent Routing Changes*: Skitter data can provide indications of low-frequency persistent routing changes. Correlations between RTT and time of day may reveal a change in either forward or reverse path routing.
- *Visualization of the Network Connectivity*: By probing the paths to many destinations IP addresses spread throughout the IPv4 address space, skitter data can be used to visualize the directed graph from a source to much of the Internet

Targets

The pool of replying destinations is constantly changing and decreasing due to firewalls, changing IP addresses and other reasons. They need to refresh the destinations lists every 8 to 12 months. Currently, CAIDA use two kinds of destination lists[2]: DNS list and IPv4 list. The two families of list in 2.3 have two different goals:

- *IPv4 list*: Provide a representative coverage of the routable IPv4 space for topology measurements. Ideally, this list should have an IP address in each populated /24 prefix of the global Internet space.
- *DNS list*: Provide a representative coverage of clients querying DNS root name servers.

Family	Date	Size	Monitors
IPv4 list	02/25/2003	147,016	champagne, kaist, riesling
IPv4 list	02/25/2003	365,605	apan-jp, iad, lhr, nrt, sjc, yto
IPv4 list	02/25/2003	814,356	mwest, uoregon
DNS Clients list	01/13/2003	147,943	a-root, b-root, d-root, e-root, f-root, g-root, h-root, i-root, k-root, m-root, k-peer, cdg-rssac

Figure 2.3: List of destinations provided by Skitter (fall 2003).

Format of the traces

Traces available to the research community provided by Skitter are stored in a binary format called Arts++[3]. This format defined by the CAIDA group

[2]http://www.caida.org/analysis/topology/macroscopic/list.html
[3]http://www.caida.org/tools/utilities/arts/

is dedicated for the storage of traces from active measurements on networks. CAIDA has provided a C++ class library to access data in this format and an example [4] that use it. This small example is entirely sufficient to extract basic information (see section 3.4) collected in the traceroute fashion by Skitter. Please note that Skitter affects a key for each trace which determines its quality. Here are its possible values:

- *N*: no reply was received from the destination although a partial path may have been recorded. The RTT has no meaning in this case.
- *I*: Skitter got a reply from the destination, but did not receive a reply from every intermediate hop on the path. The RTT to the destination is valid. *I* stands for *incomplete*.
- *C*: The destination and all intermediate hops in the path all replied. The RTT to the destination is valid. *C* stands for *complete*.

Activity

The activity of Skitter is irregular. It can append that a monitor goes down for a month and then goes up back again. For our studies we have chosen a day during which 23 monitors have worked. The *2th* of July 2003. All monitors of 2.2 have performed about 13 millions of traceroutes where 8 millions are of type *C* toward about 600 000 destinations.

We have done some reverse engineering on the data in order to know how skitter proceed. We will not give all the details of this study in this report but just an example for the monitor called *a-root*.

Figure 2.4 shows the number of traceroutes by type this server have done. For our study we have only keep traces of type *C* because it was simpler to deal with.

Nb total of traceroutes 478983	
where:	
Type N	134049
Type I	66325
Type C	278609
Type C & q	158

Figure 2.4: Consistency of the data collected by *a-root*.

Monitors generally target a same destination several times in a day. Figure 2.5 shows that they usually perform 3 traceroutes toward a destination. This figure also shows the number of different routes discovered when there are several routes in the data set between a Skitter server and a destination. We can observe that in many case, there are more than one route toward a destination discovered per day.

[4] http://www.caida.org/tools/measurement/skitter/sample_code/

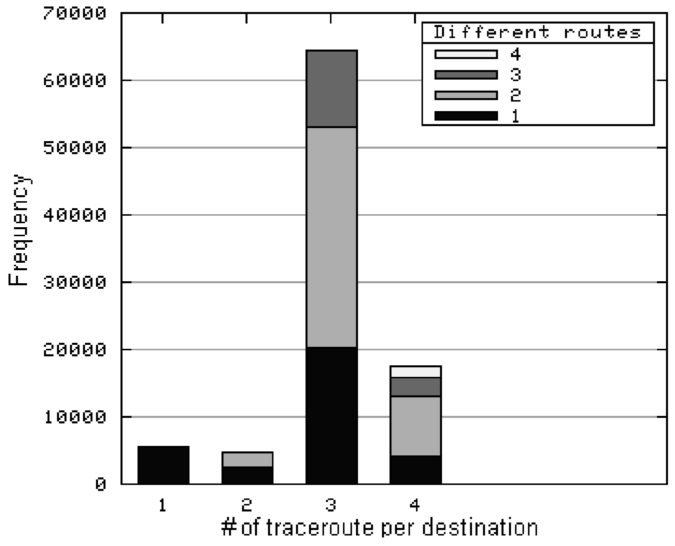

Figure 2.5: Frequency and stability of traceroutes done by *a-root*.

2.1.2 Tools used

BOOST

The BOOST[5] library is a set of portable C++ libraries. The library we have used is the BOOST Graph Library (BGL) developed by Jeremy Siek. It provides generic components and algorithms for manipulations of graphs and offers genericity like the STL[6] and optimizations. Algorithms and data structures are generic, meaning that they can be adapted to any kind of problem. Here are some algorithms from graph theory that this library implements:

- Breadth first search,depth first search and uniform cost search
- Dijkstra's shortest paths, Bellman-Ford and Johnson's shortest paths
- Kruskal's Minimum Spanning Tree, Prim's Minimum Spanning Tree
- Connected Components, Strongly Connected Components
- Topological Sort

DOT

The DOT[7] format is a very powerful way to represent graphs because it is simple and it has lots of extend that can handles layout related information.

[5]www.boost.org
[6]Standard Template Library
[7]http://www.research.att.com/ erg/graphviz/info/lang.html

Documentation on the language is maintained by AT&T within the framework of the project GraphViz[8] which provides a set of tools for graph visualization (see Figure 2.6). We have chosen this format because of its simplicity and also because the BOOST Graph Library has an interface that allows to load graph store in DOT.

Name	**Function**
dot	makes hierarchical layouts of directed graphs
neato	makes "spring" model layouts of undirected graphs
dotty & tcldot	two customizable interfaces
libgraph	the base library for graph tools

Figure 2.6: Content of the Graphviz package.

Here is an Hello World example in DOT:

```
digraph g{
    "202.188.126.141"->"210.187.15.253
    "202.188.126.141"->"202.188.128.14
    "202.188.128.144"->"210.187.15.253
}
```

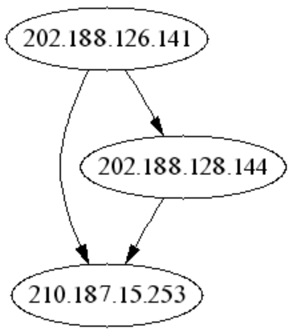

Figure 2.7: Example of a graph drawn with Graphviz.

Shell Scripts

In addition to the use of C++ programs we have use shell scripts because it is often more efficient to use small utilities already developed and tested. Existing tools are often better than the ones you intend to make. So, we have chosen to implement our tools as a mix of gnu tools like *sort*, *awk* or *gnuplot* and C++ programs embedded in Tcsh[9] shell scripts.

[8]http://www.research.att.com/sw/tools/graphviz/
[9]http://www.tcsh.org

2.1.3 Implementations

Creation of the Skitter graph

The goal here was to take the entire traces offer by Skitter and to build a graph in DOT. The question has been regarded before our study. An algorithm developed at LIP6 before this study in C++ was available. Taking all traceroutes and adding links to a graph structure one by one. Unfortunately, this algorithm was too slow, it would have taken more than one month to obtain the graph. So, we have come with a very simple and efficient solution that builds the graph in DOT in one hour:

1. For each trace (with *awk*)
    - print each link into a file
2. sort uniquely this file using the *sort*
3. make some post processing and arrangement to fit the DOT format

This solution which consists to manipulate text files appeared to be faster that the previous approach.

We have seen in section 1.3.2 that this graph contains special IPs that should not be seen on the Internet. We need to remove them because they create fake nodes with high degree. This operation is a part of the post processing.

Manipulations of the Skitter graph

All operations that only concern the Skitter graph have been realized within the C++ programs such as:

- degree distribution
- clustering distribution
- shortest paths computation

The graph is first loaded with the ReadGraphviz function in a directed or in an undirected fashion. Then, algorithms are developed simply with BOOST functionalities and data structures. In term of performance, our program needs about 300 Mo of RAM memory to use the Skitter graph. It is able to load it and to run algorithms in few minutes.

Manipulations of the Skitter data

Shell scripts are used to perform manipulations of the Skitters data because it allows calling all the efficient tools that already exist and that can solve our problems. For some of our studies, the algorithms use both shell scripts and the C++ program.

A complete list of the functionalities developed is available in section 3.5.

2.2 Observations

2.2.1 Data consistency

The Skitter graph freshly build from the set of traceroutes provided by Skitter contains special IPs (see section 1.1.2) that should not appear in this graph as shown in section 1.3.2. Indeed, there are

- 12077 (1.3%) IPs are private ones.
- 42390 (3.2%) links in the graph imply at least one private IP.
- 237 links imply the 127.0.0.0/8 addresses.
- 97 links imply the 0.0.0.0 address.
- 210 links imply broadcast IPs.

After purification, the Skitter graph is composed of:

- Vertices: 885 435
- Edges: 1 259 039

All the following studies have been made on this graph.

2.2.2 Degree distribution

As we analyze a graph, a common and useful distribution which characterize it is its degree distribution. We have seen in section 1.4.1, that on the graph of the Internet, this distribution follows a power-law $f_d\alpha\ d^O$ where f_d is the frequency of the degree d and O a constant around -2. This observation made in 1999 by the Faloutsos brothers is shown on Figure 2.8 on a directed IP graph.

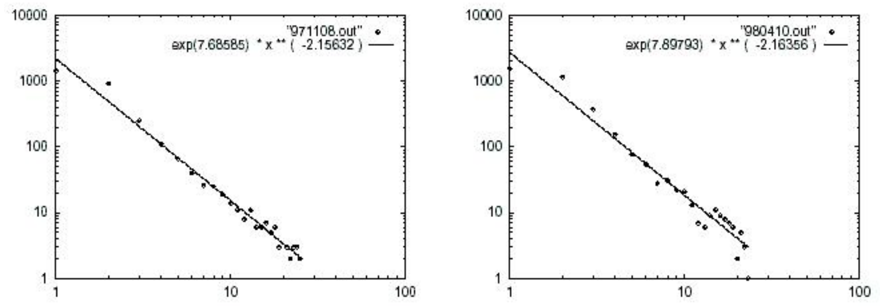

Figure 2.8: Two observations of the degree distribution done by the faloutso brothers in [9].

We have recomputed this statistic on Skitter data considering our graph in a directed fashion and also in an undirected fashion. The question of the orientation of the graph will be discuss in section 2.2.6. The exponents of the power-law found are: -1.9 for the study on the directed graph and -2.2 for the study on the undirected graph. Our distributions show a distribution with a heavy tailed. This was pointed out by [4].

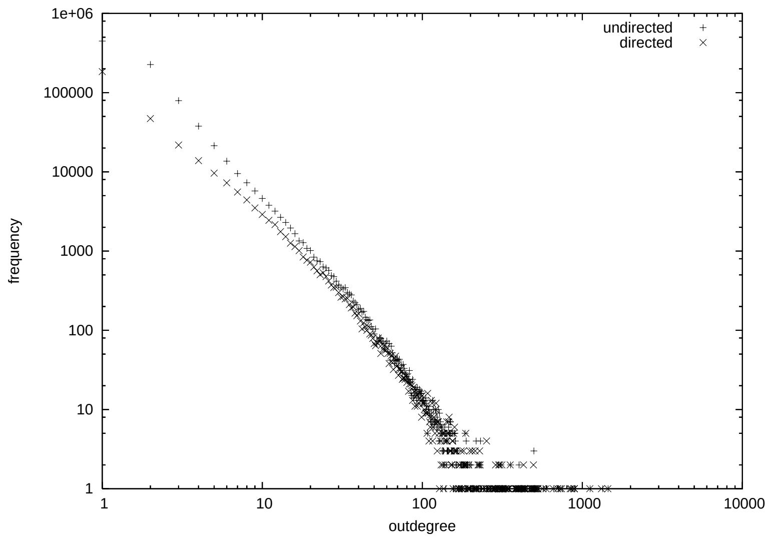

Figure 2.9: Out-degree distribution of the Skitter graph.

2.2.3 Clustering coefficient

The clustering coefficient is also a tool generally use to describe graph. The formula is given in section 1.4.1. The value of the clustering itself is not sufficient to compare two graphs because it is influenced by the number of vertices and the number links of the graphs. Therefore, it is useful to compare two graph of the same size. That is why in this part we have computed the average clustering coefficient for the Skitter graph and we have compared it to the one for a random graph of the same size generated with the Erdos-Renyi model (see section 1.4.2). We have found a value of 0.0347379 for the Skitter graph and $1.29822e-06$ for the random graph. This experiment has shown that the Skitter graph is highly clustered. Meaning that there is a strong connectivity in this graph.

2.2.4 Length of paths

The length of shortest paths between nodes in a graph is also an important parameter. We have again made this study both in a directed way and in undirected. The source of paths correspond here to the Skitter' monitors and the destinations to destinations of these servers because we want to know what is the typical distance between the extremities of the graph. Figure 2.10 shows us that shortest paths are quite small. We meet again here the small-world behavior described in section 1.4.1. Note that the difference between the distribution done in directed and in undirected is not so important. We will discuss of the orientation in section 2.2.6. Lengths of shortest paths in the directed graph are obviously longer in average than in the undirected graph because some detours are taken.

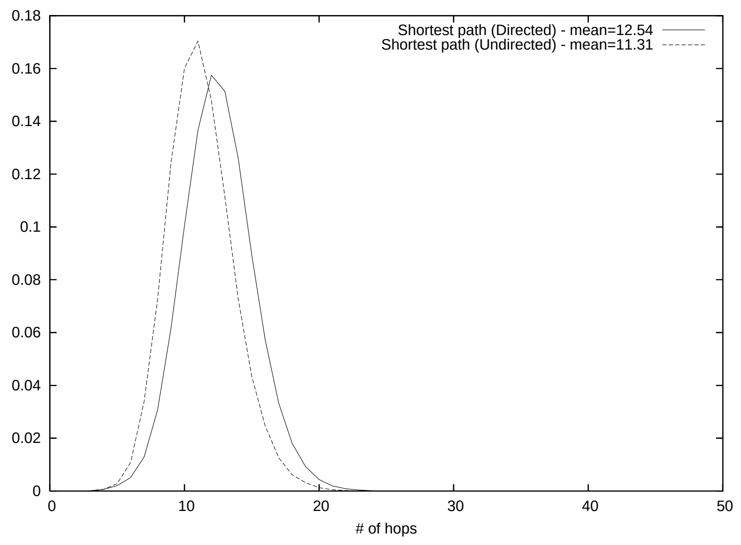

Figure 2.10: Probability distribution function of the shortest path lengths.

2.2.5 Core of the Internet

Here is just a trial to analyze the core of the Internet. Our hope was to extract the core of Internet by different manners and to analyze them by doing a degree distribution. We intend to not find a power law. Figure 2.11 shows the results for the different methods explained in the legend. You can see that the power-law stands again.

2.2.6 Orientation of the graph

This subject has been the center of many discussions during this work and is not simple. Shall we consider our graph as oriented or not? Shall we perform our analysis or experimentations on a directed graph or not? We have begun to look at works already done [2] and we have seen that they always consider the Skitter graph or the IP graph as directed. We come to the conclusion that since it is the IP graph we are dealing with, the only possibility is to take it as directed. But we can make some assumptions depending what we are studying that allow to consider this graph as undirected. We will do that for our study in

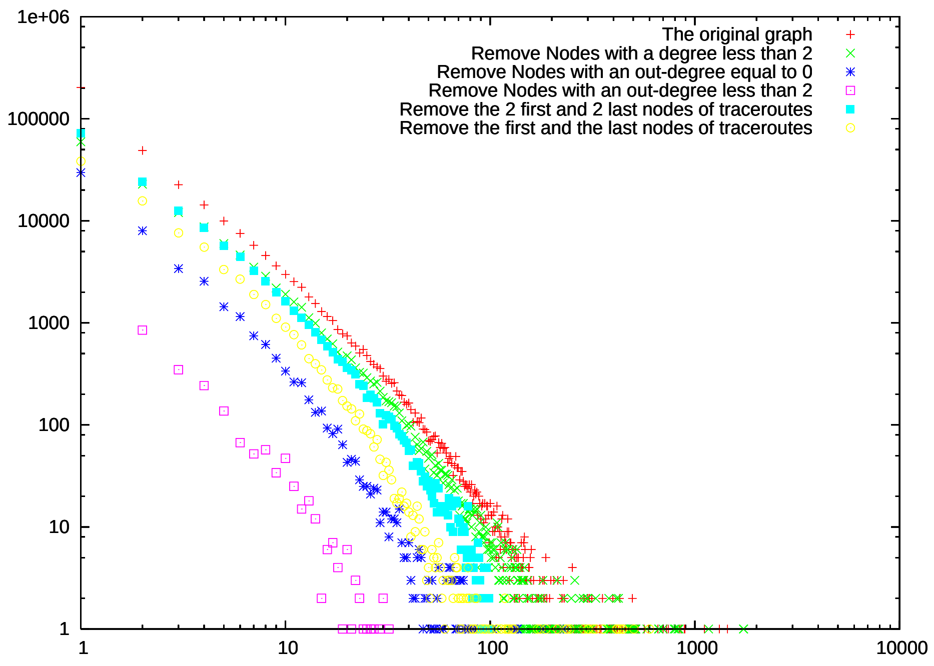

Figure 2.11: Degree distributions on the different versions of the core of Internet extracted.

chapter 3 on paths in the Internet Indeed, we can assume that if a link between two interfaces exists in a way it should exists in the way back but it has not been discovered. So, it is not crazy to consider the Skitter graph as undirected since properties like distances in term of number of hops are conserved.

Chapter 3

What is a route on the Internet?

3.1 Observations

Let's recall that we are working on the Skitter graph which has been construct from the merging of millions of traces from traceroute. We have performed some statistical analysis to describe the Skitter graph in chapter 2. Furthermore, we have performed other statistical analysis to described routes in this graph.

3.1.1 Length of routes

We plot in Figure 3.1 the length distributions of both routes and shortest paths, together with the distribution of their difference, that we now call δ. These curves have been computed both on the directed and undirected graphs.

Through the δ distribution, Figure 3.1 shows that routes have not always the same length the shortest paths. We have computed the shortest path and the δ distributions both in the directed and the undirected fashion because the two manners are interesting. So, experimentally, we bring the evidence that routing protocols does their best effort to choose routes closer as possible to shortest paths but fail at least in more than 80 percent of the cases. This value is given by the δ curve done with on a directed version of the Skitter graph, it certainly gives a lower bound for this percentage. According to the discussion in section 2.2.6 where we come to the conclusion that shortest paths in the undirected graph are more realistic than in the directed one, it is also interesting to look at the undirected version of the δ distribution because it may give a value close to the real value of the previous percentage. Meaning that shortest path are in 95% of the cases longer than in shortest path. Furthermore, because lots of edges are certainly missing in our graph, this percentage may be higher in reality. This curve shows that routes are generally 3 hops longer than shortest paths.

Intuitively, the value of δ is not independent of the length of the shortest

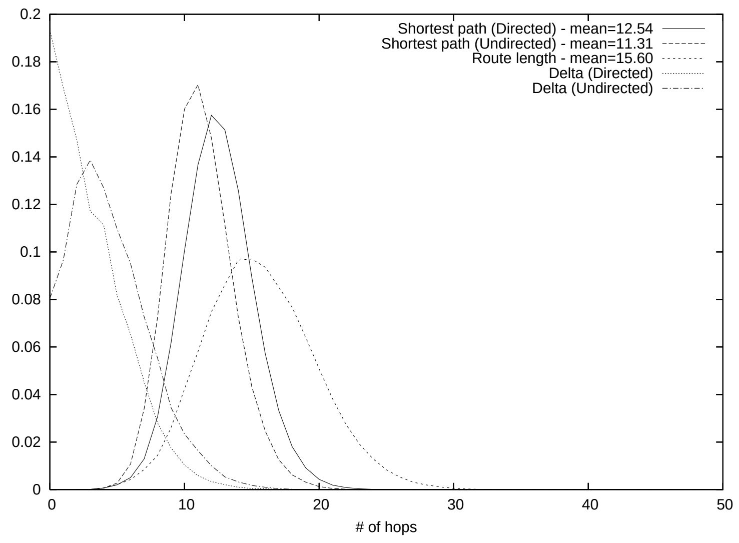

Figure 3.1: Length distribution of routes, shortest paths and their difference (δ) both in the directed and the undirected fashion.

path length because the values of δ for shortest paths of 5 hops may be shorter than the values of δ for shortest paths of 15 hops. Actually, as the Figure 3.2 shows, we have observed that the average of δ arraises slightly with the shortest paths lengths . This figure was plot for shortest path lengths between 9 and 16 which represents more than 85 percents of the cases. We have also plotted quantiles in order to back this result, they show that values are mainly close to the mean.

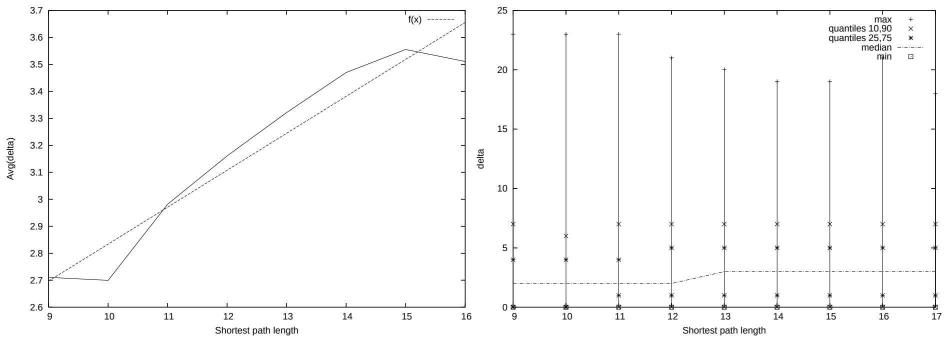

Figure 3.2: δ function of shortest path lengths.

The analysis of routes in terms of shortest paths in the network assumes a global knowledge of the network. In the *real* network, on the contrary, a route is discovered step after step, each router taking is own decision, without a global knowledge of the whole Internet topology. This may reveal special local properties of routes, like correlations between nodes along a route, we have studies in next sections.

3.1.2 Degree evolution along routes

In the willing to extract local properties of routes we have come to the following question: how does the degree evolve along a route? To answer this question, we plot the evolution of the degree along routes in Figure 3.4. To this question, before doing the experiments, we came with the expectations of Figure 3.3.

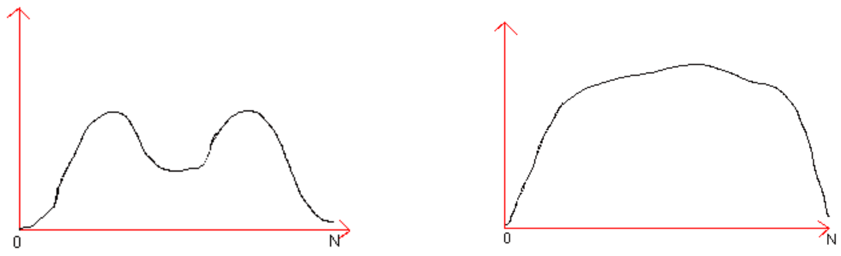

Figure 3.3: Our expectations before doing experiments for the degree evolution along traceroutes of length N+1

The curve in Figure 3.4 presents the experiments we have made for routes that have a length of 15. This example has been chosen because 15 is one of the most common route length in our data set and because the shape and the values shown in this Figure are very similar to the ones observed for length between 9 and 20. The median value of the out-degree is surprisingly constant in the core, around 10. Theses results break the intuitive idea that nodes' out-degree increase and decrease with a peak at the middle of the route. It also appears that nodes close to the border but not at the extremities of routes seem to have slightly a higher out-degree.

In Figure 3.5, we plot the evolution of the average degree which is mainly disturbed by sparse values.

3.1.3 Local properties of routes

We try to answer here the following question: do routers choose their best-connected neighbor as the next step on the route? To answer this, we have for each nodes with an out-degree N classified decreasingly its out-edges function of the targeted node' out-degree in order to attribute a rank to each of them. Thus, the neighbor that has a rank equal to 1 has the highest out-degree. Then, we have tried to see if routes usually follow low rank edges rather than high rank edges. In the case of several neighbors have the same out-degree, we consider that they have the same importance.

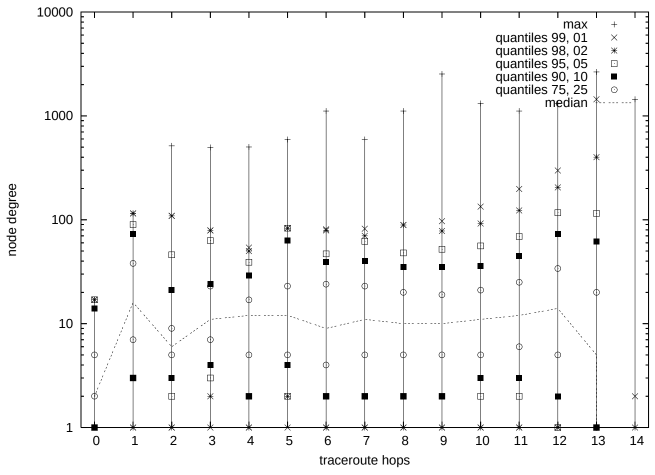

Figure 3.4: Evolution of the degree along routes.

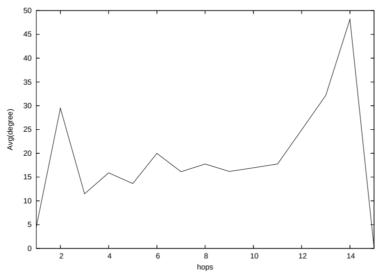

Figure 3.5: Evolution of the average degree along routes.

We finally observed in Figure 3.6 that there is a slight trend for routes to use links that leads to the neighbor that have the highest degree. Therefore, from an out-degree of 8, the curve still is a bit different. Routes follow preferably edges that lead to a neighbor that has one of an high out-degree compare to the others which is not always the highest out-degree neighbor as shown in Figure 3.7.

3.1.4 Properties of links

We have seen in section 3.1.1 that routes are in most of the cases longer than shortest paths and we have tried to reveal some characteristics that could explained this phenomena. In addition, we compute statistics shown on Figure 3.8 to see in which proportion links in routes go away from the server, approach toward the server or remain at a constant distance from the server. This experience has been made on a sample of routes randomly chosen from the Skitter data set.

Statistics on Figure 3.8 show that a non negligible proportion of links in

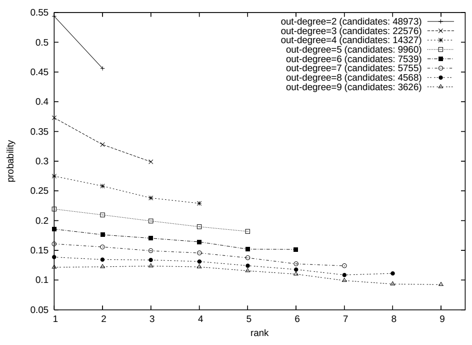

Figure 3.6: Choice of each router with respect to its neighbor's degrees.

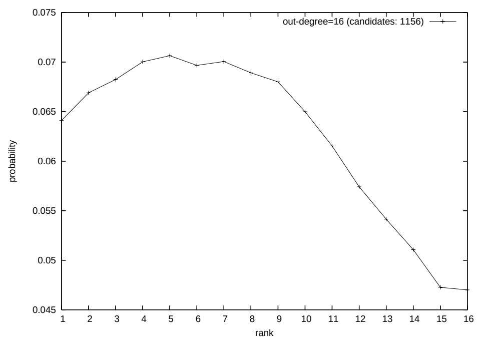

Figure 3.7: Choice at router having an out-degree of 8.

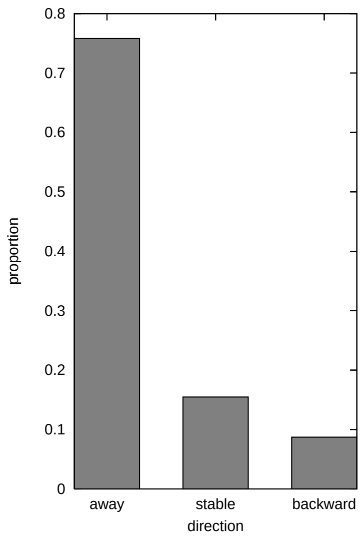

Figure 3.8: Direction of links in a sample of routes.

routes go backward, meaning that they go back to the server, and that a quite important part of them remain stable. This is an explanation of the fact that routes are longer than shortest paths. Routes may certainly follow some detours. This phenomena has been pointed out at the AS level in [29].

Figure 3.9 shows the evolution of these proportion function of the distance from the server in routes of length 15. We can observed that it is in the middle of routes that there are more floating behaviors. We can suppose that middle of routes correspond to the core of the network. This shows that it is in the core of the network that detours appear.

Note that going away from the server is not the same as going toward the

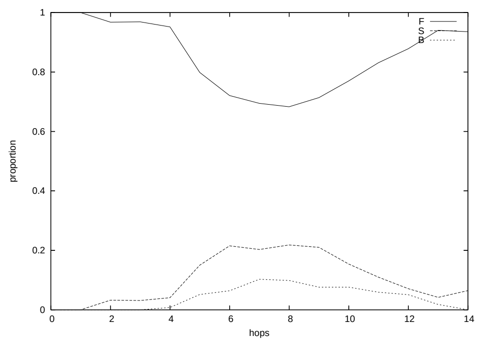

Figure 3.9: Direction of links in a sample of routes of length 15.

destination. Unfortunately, it was not possible to study the proportion of links that go toward destinations because of computation problems. Indeed, in the Skitter data set, from the point of view of a server, you can treat more traceroutes (thus links) at a time than from the point of view of a destination.

3.1.5 Dynamics of routes

We just have made preliminary studies. We realized for each Skitter' server statistics like the ones of Figure 2.5 showing that if several traceroutes have been perform between a server and a destination, in most of case there are more than one route observed. We did not go deeper in this study and the way these routes are different have to be characterized. Furthermore, studies concerning length evolution of routes are also necessary.

3.2 How to model routes?

On the light of statistics pointed out in the previous section, we have tried to define models to design paths on the Internet. Until now, paths in this world wide network are often modeled by shortest paths because no precise studies have described what a route looks like on the Internet.

One possibility could have been to complicate a lot our models but we have taken the position to make it the simple as possible in order to provide efficient solutions. Let's described the three methods we have designed to build fake routes and the results of simulations.

3.2.1 Methods

Random perturbation based

This method tends to generate routes with a random based solution. The shortest path is compute between the starting point and the destination. Then, at each hop we set a probability that a perturbation appears. If it appears, a routes going through a neighbor of the current node not in the shortest path is searched. The algorithm also avoids two-loops.

Length based

This one tried to concentrate on the length of the path given information plotted in Figure 3.1. The shortest path is compute between the starting point and the destination. Then, a δ is randomly chosen by following the distribution of Figure 3.1 and is added to the shortest path length. A path of the previous length is searched.

Local properties based

Finally, the last one is a bit more complicated. Assume that for all nodes, predecessors are their highest degree neighbor. From the source and from the destination as well, we replace the current node by its predecessor to find the piece 1 and 3 (see Figure 3.10) of the path until a loop is reached. At the end, a shortest path is compute between the two tops of the trees as shown in Figure 3.10, it corresponds to the part 2.

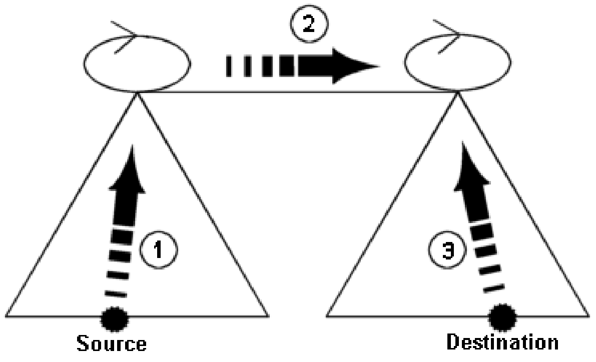

Figure 3.10: Arbre.

3.2.2 Simulation results

To evaluate our models, we have try to see if routes generated have properties that are close to the ones studied in section 3.1. We had identify four properties:

- the lengths of routes compare to shortest paths.
- the evolution of node' degrees along routes.
- the slight trend in routes to go to the highest degree neihbor.
- the proportion of type of links: forward, backward and stable.

We have generated fake routes on the Skitter graph with the three methods previously seen. We will see the results of the experiments. Unfortunatly, this is just preliminary results.

First, the method based on length was too greedy in computation power, so we are not able to present any results. Indeed, the algorithm stops on some blocking cases like this one: the shortest path between a source and a destination was equal to 14 hops, the value of δ randomly choosen was equal to 5, the algorithm took several days to find that there exist 35968166 paths of length 19! One can imagine some tricks to avoid these blocking cases but it has a great influence on the model and a loss of control on what is exactly generated can appear.

Then, the method based on random perturbations and the one based on local properties have worked well to reproduce length properties of routes. Results are available in Figure 3.11 and Figure 3.13. We found an average traceroute length of 16.77 hops for the random based method and of 14.66 for the local properties based. Concerning the results on the degree evolution along routes plotted in Figure 3.12 and Figure 3.14 for both methods. A preliminary qualitative result is that the method based on local properties gives good results because we find again the result of Figure 3.4.

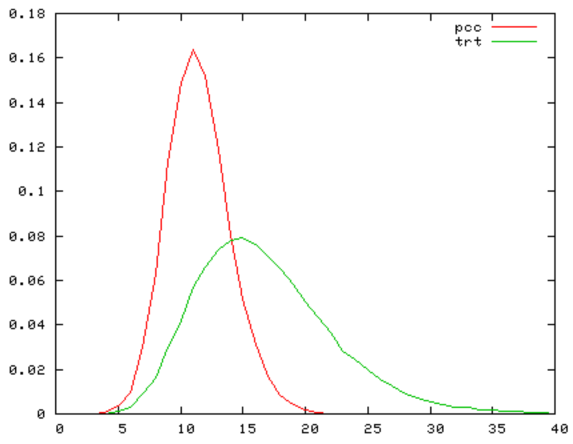

Figure 3.11: Probability distribution function of lengths for the random based model.

From the results obtained experimentally, the best model seems to be the one based on local properties but this need to be confirm.

Naturally, these results and simulation are on going work because we need more statistics on more generated data like the ones on the type of links. We have also begun to generate fake routes on other graphs real and simulated. The real graphs was obtain using the *Mercator* software and public traceroute servers. Generated ones was build with the Erdos-Renyi and the Albert and Barabasi models.

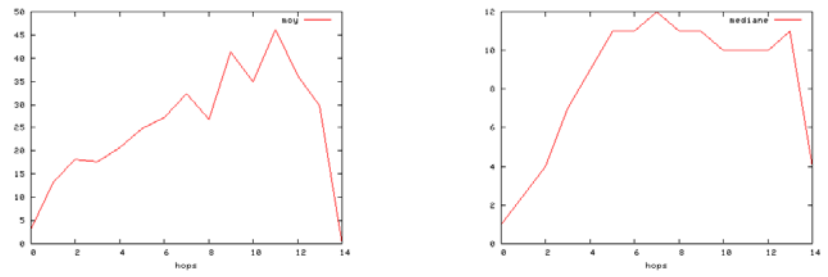

Figure 3.12: Degree evolution along routes for the random based model.

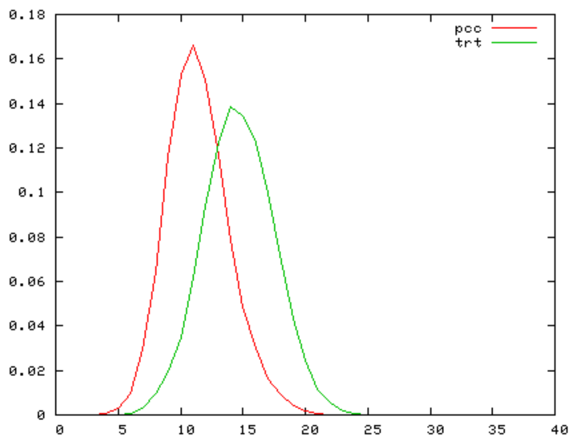

Figure 3.13: Probability distribution function of lengths for the local properties based model.

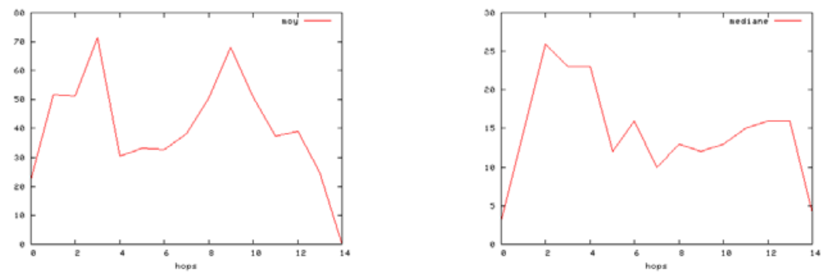

Figure 3.14: Degree evolution along routes for the local properties based model.

Conclusion and future works

In this work, after doing a state of the art in topology analysis of the Internet, we have performed some new analysis on the Skitter graph which is a pseudo graph of the Internet, then we have realize a deep study on paths in this network. Our main contribution in this work was to give ideas on what a route on the Internet is and to provide a simple model to improve simulations accuracy. We have also provided tools to manipulate huge IP graphs and to perform analysis on this kind of graph.

For future works, we will try to finish the simulations already started and also to focus on dynamics of routes to answer the following question: If different routes exist between two hosts, do packets usually take the same? If yes, why?

Furthermore, we will try to find some solutions to improve the way of grabbing data. The Skitter platform which is at the state of the art must be improved. Indeed, more monitors (*e.g* points of view) are needed to get accurate data and intelligent ways of doing distributed traceroutes have to be found in order to enlarge such measurement platforms and to reduce their overhead on the network.

Bibliography

[1] Albert-Laszlo Barabasi, Erzsebet Ravasz, and Tamas Vicsek. Deterministic scale-free networks. *Physica A 299, (3-4)*, pages 559–564, 2001.

[2] A. Broido and kc claffy. connectivity of ip graphs. 2001.

[3] Kenneth L. Calvert, Matthew B. Doar, and Ellen W. Zegura. Modeling internet topology. *IEEE Communications Magazine*, 35(6):160–163, June 1997.

[4] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. The origin of power laws in internet topologies revisited.

[5] Bill Cheswick, Hal Burch, and Steve Branigan. Mapping and visualizing the internet. pages 1–12.

[6] K. G. Coffman and A. M. Odlyzko. Growth of the internet.

[7] M. Doar. A better model for generating test networks, 1996.

[8] P. Erdos and A. Renyi. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959.

[9] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.

[10] The National Laboratory for Applied Network Research (NLANR). `http://moat.nlanr.net/IPaddrocc/`.

[11] Iffinder from CAIDA. `http://www.caida.org/tools/measurement/iffinder/`.

[12] Skitter from the Cooperative Association for Internet Data Analysis. `http://www.caida.org/tools/measurement/skitter/`.

[13] Boston University Representative Internet Topology gEnerator. `http://www.cs.bu.edu/brite/`.

[14] Ramesh Govindan and Anoop Reddy. An analysis of internet inter-domain topology and route stability. In *INFOCOM (2)*, pages 850–857, 1997.

[15] Ramesh Govindan and Hongsuda Tangmunarunkit. Heuristics for internet map discovery. In *IEEE INFOCOM 2000*, pages 1371–1380, Tel Aviv, Israel, March 2000. IEEE.

[16] Y. Hyun, A. Broido, and k claffy. Traceroute and bgp as path incongruities.

[17] Young Hyun, Andre Broido, and kc claffy. On third-party addresses in traceroute paths. In *Passive and Active Measurement Workshop 2003*, La Jolla, CA, Apr 2003.

[18] Ramon Ferrer i Cancho and Ricard V. Sol. The small-world of human language. Technical report, Santa Fe Working paper 01-03-016, 2001.

[19] The Internet Control Message Protocol (ICMP). `http://www.freesoft.org/CIE/RFC/792/`.

[20] Emily M. Jin, Michelle Girvan, and M. E. J. Newman. The structure of growing social networks. *Phys. Rev. E 64*, (046132), 2001.

[21] S. Jin and A. Bestavros. Small-world internet topologies: Possible causes and implications on scalability of end-system multicast. Technical Report BUCS-2002-004, Boston University, 2002.

[22] Rajesh Kasturirangan. Multiple scales in small-world networks. Technical Report AIM-1663, 1999.

[23] A. Lakhina, J. Byers, M. Crovella, and I. Matta. the geographic location of internet resources, 2002.

[24] A. Lakhina, J. Byers, M. Crovella, and P. Xie. Sampling biases in ip topology measurements, 2002.

[25] Vern Paxson and Sally Floyd. Why we don't know how to simulate the internet. In *Winter Simulation Conference*, pages 1037–1044, 1997.

[26] R.Sharma R.Siamwalla and S.Keshav. Discovering internet topology. 1999.

[27] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network topology generators: Degree-based vs structural, 2002.

[28] Hongsuda Tangmunarunkit, Ramesh Govindan, Scott Shenker, and Deborah Estrin. The impact of routing policy on internet paths. In *INFOCOM*, pages 736–742, 2001.

[29] Renata Teixeira, Keith Marzullo, Stefan Savage, and Geoffrey M. Voelker. In search of path diversity in isp networks. In *IMC*, 2003.

[30] the Georgia Tech internetwork topology models generator. `http://www.cc.gatech.edu/projects/gtitm/`.

[31] Angelos Stavrou. Java Applet to create small-world graph according to (Watts and Strogatz model). `http://comet.ctr.columbia.edu/~angelos/smallworld.html`.

[32] Tutorial: Using Traceroute. `http://www.exit109.com/~jeremy/news/providers/traceroute.html`.

[33] D. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998.

[34] B. M. Waxman. Routing of multipoint connections. *IEEE Journal of Selected Areas in Communications*, pages 1617–1622, 1988.

[35] Y. Zhang, V. Paxson, and S. Shenker. The stationarity of internet path properties: Routing, loss, and throughput. *ACIRI Technical Report*, 2000.

Appendix

3.3 Some Traceroute related problems

This is an extract of the man page of traceroute.

```
* Note that lines 2 \& 3 are the same.  This is due to a buggy
* kernel on the 2nd hop system -- lbl-csam.arpa -- that forwards
* packets with a zero ttl.
*
* A more interesting example is:
*
*     [yak 72]% traceroute allspice.lcs.mit.edu.
*     traceroute to allspice.lcs.mit.edu (18.26.0.115), 30 hops max
*      1  helios.ee.lbl.gov (128.3.112.1)  0 ms  0 ms  0 ms
*      2  lilac-dmc.Berkeley.EDU (128.32.216.1)  19 ms  19 ms  19 ms
*      3  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  19 ms  19 ms
*      4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  19 ms  39 ms  39 ms
*      5  ccn-nerif22.Berkeley.EDU (128.32.168.22)  20 ms  39 ms  39 ms
*      6  128.32.197.4 (128.32.197.4)  59 ms  119 ms  39 ms
*      7  131.119.2.5 (131.119.2.5)  59 ms  59 ms  39 ms
*      8  129.140.70.13 (129.140.70.13)  80 ms  79 ms  99 ms
*      9  129.140.71.6 (129.140.71.6)  139 ms  139 ms  159 ms
*     10  129.140.81.7 (129.140.81.7)  199 ms  180 ms  300 ms
*     11  129.140.72.17 (129.140.72.17)  300 ms  239 ms  239 ms
*     12  * * *
*     13  128.121.54.72 (128.121.54.72)  259 ms  499 ms  279 ms
*     14  * * *
*     15  * * *
*     16  * * *
*     17  * * *
*     18  ALLSPICE.LCS.MIT.EDU (18.26.0.115)  339 ms  279 ms  279 ms
*
* (I start to see why I'm having so much trouble with mail to
* MIT.)  Note that the gateways 12, 14, 15, 16 \& 17 hops away
* either don't send ICMP "time exceeded" messages or send them
* with a ttl too small to reach us.  14 - 17 are running the
* MIT C Gateway code that doesn't send "time exceeded"s.  God
* only knows what's going on with 12.
*
```

```
* The silent gateway 12 in the above may be the result of a bug in
* the 4.[23]BSD network code (and its derivatives):  4.x (x <= 3)
* sends an unreachable message using whatever ttl remains in the
* original datagram.  Since, for gateways, the remaining ttl is
* zero, the icmp "time exceeded" is guaranteed to not make it back
* to us.  The behavior of this bug is slightly more interesting
* when it appears on the destination system:
*
*      1  helios.ee.lbl.gov (128.3.112.1)  0 ms  0 ms  0 ms
*      2  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  19 ms  39 ms
*      3  lilac-dmc.Berkeley.EDU (128.32.216.1)  19 ms  39 ms  19 ms
*      4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  39 ms  40 ms  19 ms
*      5  ccn-nerif35.Berkeley.EDU (128.32.168.35)  39 ms  39 ms  39 ms
*      6  csgw.Berkeley.EDU (128.32.133.254)  39 ms  59 ms  39 ms
*      7  * * *
*      8  * * *
*      9  * * *
*     10  * * *
*     11  * * *
*     12  * * *
*     13  rip.Berkeley.EDU (128.32.131.22)  59 ms !  39 ms !  39 ms !
*
* Notice that there are 12 "gateways" (13 is the final
* destination) and exactly the last half of them are "missing".
* What's really happening is that rip (a Sun-3 running Sun OS3.5)
* is using the ttl from our arriving datagram as the ttl in its
* icmp reply.  So, the reply will time out on the return path
* (with no notice sent to anyone since icmp's aren't sent for
* icmp's) until we probe with a ttl that's at least twice the path
* length.  I.e., rip is really only 7 hops away.  A reply that
* returns with a ttl of 1 is a clue this problem exists.
* Traceroute prints a "!" after the time if the ttl is <= 1.
* Since vendors ship a lot of obsolete (DEC's Ultrix, Sun 3.x) or
* non-standard (HPUX) software, expect to see this problem
* frequently and/or take care picking the target host of your
* probes.
*
* Other possible annotations after the time are !H, !N, !P (got a host,
* network or protocol unreachable, respectively), !S or !F (source
* route failed or fragmentation needed -- neither of these should
* ever occur and the associated gateway is busted if you see one).  If
* almost all the probes result in some kind of unreachable, traceroute
* will give up and exit.
*
```

3.4 The format used for Skitter data

```
==========================================================================
    sample output (default version)
==========================================================================

Key  Source         Destination    Time       RTT    Count  hop1  hop2 ...
------------------------------------------------------------------------------
C  192.172.226.24  194.225.70.80  978307205  709.657  26 192.172.226.1

Key - parameter characterizing the quality of a trace
-------
  no reply
    N - Noreply
    no reply was received from the destination although a partial
    path may have been recorded. The RTT has no meaning in this case.

  replied
    I - Incomplete
    skitter got a reply from the destination, but did not receive a reply
    from every intermediate hop on the path. The RTT to the destination
    is valid.
    C - complete
    The destination and all intermediate hops in the path all replied.
    The RTT to the destination is valid.

Source and Destination -  IP addresses in standard dotted octet notation.
----------------------
    Source - the IP address of the skitter monitor. The first IP address
    in all paths measured by this monitor.
    Destination - the IP address of the final destination to which the
    packets were sent in a given trace. The last IP address of the
    measured IP path.

Time - UNIX timestamp
-------
    The meaning of this parameter is different in different versions of skitter.
    version 0-9-a3 or earlier (files collected before 2001/02/06):
        In I and C type traces - time when the reply was received
        from the destination
        In N type traces - time when the current cycle (one pass over
            all monitored addresses) of probing started. Same value for all
        N-type traces in a cycle.
    version caida-1.1 (files collected after 2001/02/07):
         In all types of traces - time when the skitter send the first
         probing packet to this destination.

RTT (Round Trip Time) - in milliseconds
---------------------
    The amount of time elapsed between the packet leaving the skitter
    source and the reply from the destination arriving back at the
    source. Skitter will always take the first such reply from the
    destination; later replies are thrown away.

Count
--------
    The Time To Live (TTL) of the first probe packet for which a reply
    was received from the destination. Since the packet had to travel
    through that many routers to reach the destination, this number
    represents the distance from the source to the destination measured
    as ‘the number of hops’.

hop1 hop2 ... - IP addresses in standard dotted octet notation or ‘q’.
---------------
    These are the IP addresses that replied with the ‘TTL expired’
    message at each TTL from the source to the destination.
    A ‘q’ is printed if no IP address replied at a given TTL.
    Sometimes, multiple IPs reply at a given probe TTL. They all are
    printed in the output, separated by commas. In this case, the number
    of intermediate IP addresses will be larger than the value of ‘Count’.
    It is also possible for a IP address to reply to TTL expired
    message at the same value as the count.
```

3.5 Functionalities implemented

For Analysis of traces

Program name	**Type**	**Function**
WhereAreTheSpecialIPs	Script	Performs some statistics on placement of special IPs.
Stats	Script	This script performs some statistical analysis on Skitter's data and generates an HTML report for each server.
CreateDotFile	Script	Creates the IP graph from the traces in DOT format, performs some statistics and generates an HTML report.
Bidirectionnalroutes	Script	Extract from Skitter data, traceroutes that exist in one way and also the way back.
ConvertGraphIPToGraphAS	Script	Converts the AS graph from the IP graph, computes some statistics and generates an HTML report.
CreateDotFileWithTroncatedTrt	Script	Creates an IP graph from traces in DOT by removing beginnings of traceroutes and the ends.
Downloads	Script	Looks at the Skitter data available on CAIDA's servers and tells between to dates for a list of servers when full data are available.
DynRoutes	Script	For a server, it follows the evolution of traceroute length toward a set of destination on a long period.

For Analysis of IP graphs

Program name	**Type**	**Function**
SkitterAnalysis	C++	Degree Distribution
SkitterAnalysis	C++	Print all degrees of nodes
SkitterAnalysis	C++	Print all clustering coefficients of the nodes
SkitterAnalysis	C++	Several methods to extract the core of the Internet
SkitterAnalysis	C++	Give the proportion of bidirectional links
SkitterAnalysis	C++	Give all the shortest path lengths from a server

For Analysis of traces and IP graphs

Program name	**Type**	**Function**
DegreesAlongTrt SkitterAnalysis	Script C++	study the evolution of degrees of nodes crossed by the traceroutes.
AnalyzeLengths SkitterAnalysis	Script C++	Study traceroute lengths versus shortest path lengths
TraficVsDegree SkitterAnalysis	Script C++	Study the correlation between degree of nodes and the traffic
TypeOfLinks SkitterAnalysis	Script C++	Study types of links in traceroutes: Forward, Stable and Backward.
DoesTrtFollowDegree SkitterAnalysis	Script C++	Study the trend for traceroute to choose at a node the highest degree neighbor for the choice of the next node on a route.

For Generation of fake routes

Program name	**Type**	**Function**
SkitterAnalysis	C++	Generate traceroutes with the model seen in section 3.2.1
SkitterAnalysis	C++	Generate traceroutes with the model seen in section 3.2.1
SkitterAnalysis	C++	Generate traceroutes with the model seen in section 3.2.1