A Survey on Internet Topology Inference

Mohammad Malli, Chadi Barakat, Walid Dabbous

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Survey on Internet Topology Inference

Mohammad Malli, Chadi Barakat, and Walid Dabbous

N° 5439

December 2004

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--5439--FR+ENG

INRIA
SOPHIA ANTIPOLIS

A Survey on Internet Topology Inference*

Mohammad Malli, Chadi Barakat, and Walid Dabbous

Thème COM — Systèmes communicants
Projet Planète

Rapport de recherche n° 5439 — December 2004 — 44 pages

Abstract: Inferring the topology of the Internet is not a trivial task due to the immense scale of the Internet, its continuous evolution, and its distributed administration. At the same time, the inference of this topology is of particular importance for network operators and users. It guarantees an efficient operation of the network and a better tuning of its protocols and applications. For this reason, we survey in this work the body of the literature that deals with Internet topology inference. We classify the works based on their belonging to either the active or passive approach and we provide a summary of the main measurement tools in this field.

Key-words: Internet topology, measurement tools, active approaches, passive approaches

* This work is supported by Alcatel under grant numbered ISR/3.04.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Etat-de-l'Art de l'inférence de la topologie de l'Internet

Résumé : Découvrir la topologie de l'Internet n'est pas une tâche évidante grâce à l'immense taille de l'Internet, sa évolution continue, et sa administration distribuée. En même temps, l'inférence de cette topologie est d'importance particulière pour des opérateurs et des utilisateurs de réseau. Elle garantit une opération efficace du réseau et une meilleur optimisation de ses protocoles et applications. Pour cette raison, nous examinons dans ce travail le corps de la littérature qui traite l'inférence de topologie d'Internet. Nous classifions ces travaux en deux principaux categories les approches active et les approches passive et nous fournissons un sommaire des principaux outils de mesure dans ce domaine.

Mots-clés : topologie de l'Internet, outils de mesure, approche actif, approche passive

Contents

1 Introduction

The worldwide increasing number of Internet users and networks makes the topology of the Internet more and more complex. A huge number of routers is installed in the backbone and edges of the Internet in order to provide a high degree of connectivity between the different nodes spread over the globe (e.g., end hosts, servers). Besides, digital contents become shared between the Internet users and stored in replicated servers distributed across the Internet in order to improve the QoS offered to the users and to provide a high availability of data. The replication of data over different machines makes the choice of its location a challenging problem that requires a knowledge of the topology to make it optimal.

In this survey, our aim is to underline the main approaches proposed in the literature to infer the topology of the Internet along with the tools used for measurements. We discuss the utility of these approaches (for the end users, ISPs, etc.) and their impact on the network.

The network entities used in the following discussions are: (i) Autonomous System (AS), which is a collection of routers under a single administrative authority, using a common Interior Gateway Protocol for routing packets. (ii) Internet Service Provider (ISP), is the network manager and administrator who provides the Internet access to a part or all the end users of a certain AS, (iii) Peer-to-Peer network (e.g., Kazaa[1]), where peers behave as clients and servers, (iv) Content Distribution Network (CDN), where client requests are forwarded by request redirectors, and where the contents are stored in mirror servers geographically distributed over the Internet. Many companies, like Akamai[2], provide CDNs to content providers, (v) Overlay network which is a peer-to-peer network or a CDN network.

Inferring the Internet topology aims to construct a network map that identifies the connectivity between certain network entities and to characterize the performance on the paths among the entities of the constructed map. A network map describes the network: (i) at the router level for a particular ISP or for all the Internet, (ii) at the Autonomous System (AS) level where the purpose is to identify the connectivity between the different ASs and to characterize the performance on the paths among them, (iii) at the overlay node level where the purpose is to cluster the overlay nodes or to characterize the paths that connect the different nodes of an overlay (e.g., proxies, end host, server, etc).

Nodes of an Internet map are classified by their proximity to each other. The load on the nodes (e.g., server load, router congested, etc.) can be considered as well. Links between the pair of nodes need to be characterized by their delay, available bandwidth, loss rate, etc.

[1] `http://www.kazaa.com`
[2] `http://www.akamai.com`

There are two main approaches for inferring the Internet topology: the active approach and the passive approach. The active approach is based on sending messages (e.g., ICMP echo messages) among nodes (e.g., end hosts, proxies, etc.) in order to infer the topology of a network. Active probing can be used to infer the topology of all the Internet, of a particular ISP or of a particular AS. It can also be used for characterizing the paths among one overlay nodes as well as the performance of these paths. The passive approach provides network topology inference without injecting new packets (probing packets) into the network. It consists of inferring the topology by observing the data traffic circulating in the Internet, by looking at the routing tables (e.g., BGP routing tables), or by observing the control messages circulating between routers.

The rest of this survey is organized as follows. In the next section, we present the motivations that are behind inferring the topology. We overview in Section 3.2 the different works in the literature that fit within the active approach for inferring the topology and in Section 4.2 we overview those that fit within the passive approach. Finally, we present the conclusions in Section 5.

2 Motivations

Knowing the topology of the Internet is interesting from different standpoints: (i) An ISP can use this information for a better control of its network and a better connectivity to others ISPs (e.g., QoS issues), (ii) A digital content service provider can use this information for a better service to its users, (iii) Internet users can profit from the topology of the Internet for a better lookup for resources, (iv) researchers can also profit from this information for studying the evolution of the Internet and for obtaining more realistic scenarios in their research studies and try to solve the actual problems such as those concerning the network protocols behavior.

Inferring the topology provides many hints for an ISP, such as: (i) Analyzing routing protocols and Internet robustness and resilience. As an example for this first case, we notice the profit from an efficient network-layer restoration: to switch sending packets from one interface to another based on certain criteria such as a link panic, or a QoS degradation, (ii) better connections with the other ISPs, and setting BGP routing tables, (iii) knowing if the QoS offered by a neighbor ISPs to a given ISP is satisfied, (iv) discovering the prefixes which are responsible of a nefarious behavior such as a pattern attack, anomalies, etc.; along with knowing the locations and the ISPs of these prefixes, (v) for traffic engineering

purpose inside an ISP's network. As an example for this last case, we notice the profit from locating unpleasant points and avoid them by re-routing traffic through another path inside the network.

Also, inferring the topology may provide many hints for a digital content service provider, such as: (i) better distribution of its servers across the Internet, given the geographical distribution of clients who are requesting the service, (ii) better distribution of web proxies and better policies to push documents and update the content of caches, (iii) adjust the structuring of an overlay network to enhance the performance of the transmission, (iv) better redirection of clients to the best server handling the requested information.

Besides, inferring the topology may provide many hints for an Internet user, such as: (i) identify the best server to download a content in point-to-point or the best certain set of servers to download the content in parallel, (ii) in Peer-to-Peer network, nodes can profit from the topology to cluster themselves in an efficient way that minimizes the lookup time for a content, (iii) identify a client position in an Internet coordinate system, (iv) identify the best Internet access when there are many ones (multi-homed machine), (v) optimize the performance achieved by end-to-end protocols (e.g., data collection protocols) by considering the topology and the characteristics of the paths in the network.

3 Topology inference using active measures

To describe the topology of a network, we need to invent an inference approach which incorporates a network measurement tool along with many other engines (e.g., an engine for manipulating the obtained traces) depending on the designed inferring architecture (see Figure 1). In this section, we are interested by the approaches which are based on active measures for inferring the network topology.

First, we begin with an overview on the active measurement tools existing in the literature to have an idea on the characteristics that can be measured by the active tools and the network measurement methodologies. Then, we describe the main active approaches for inferring the network topology which depend on the described tools.

3.1 Active measurement tools

An active tool is a software module which aims to describe the characteristics of a network entity by injecting stimulus packets (probing packets) into the network and then measure

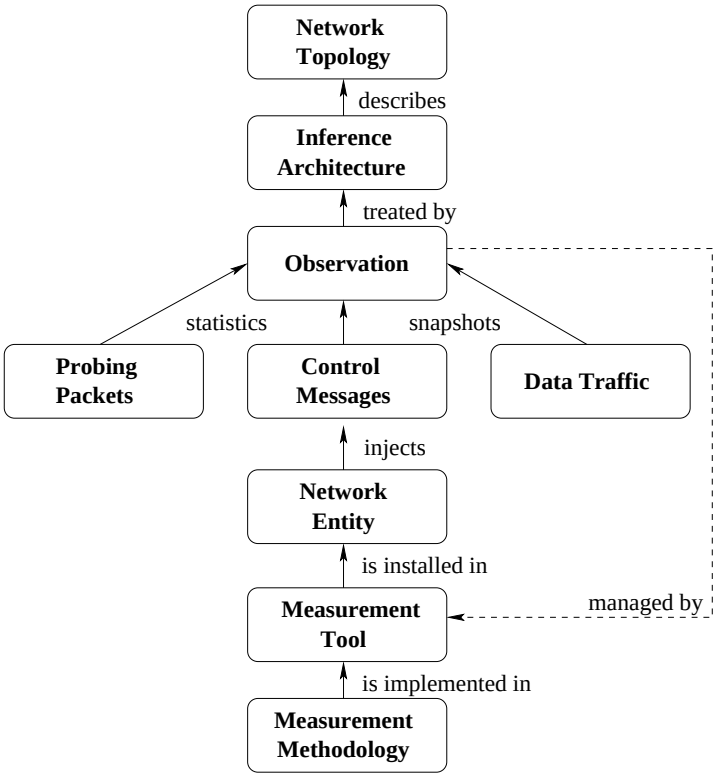

Figure 1: Paradigm of Internet Topology Inference

the response. Many tools are designed and implemented to achieve this purpose such as the basic tools ping and traceroute. The characteristics measured on the path between two nodes can be the bandwidth capacity, the available bandwidth, the achievable bandwidth, the round trip time delay, the loss rate, and the hop list. We notice that characterizing the performance on the paths between the network entities may give important hints for inferring the network topology.

Table 1 classify the mostly used tools according to the characteristics measured. These characteristics are measured per path by some tools and per hop (refers to the IP level) by anothers as described in Table 2.

BW capacity	Available BW	Achievable BW	Round Trip Time	Loss	Hoplist
bing	cprobe	Iperf	bing	bing	traceroute
bprobe	netest	netest	clink	netest	
clink	pathload	Netperf	netest	pathchar	
Nettimer	pipechar	ttcp	pathchar	pchar	
pathchar	TReno		pchar	pipechar	
pathrate	ABwE		pipechar	ping	
pchar			ping		
sprobe					

Table 1: Mostly used tools with their characteristics measured

Per-Path	Per-Hop
bing, bprobe, cprobe, ttcp	clink
Iperf, netest, Netperf, TReno	pathchar
Nettimer, pathload, pathrate	pchar
ping, sprobe, traceroute	pipechar

Table 2: Tools classified according to their measurements: Per-Path or Per-Hop

Measurement methodologies vary from one active tool to another according to their concepts which depend on such basics encountered: variable packet size, packet pair, packet train, packet pair with tailgating, path flooding, SLOPS (Self-Loading Periodic Streams), ICMP echo, varied TTL, UDP packet with port unreachable, and TCP simulation.

An interesting tool, called Traceroute, can provide the unidirectional IP path from its location to a destination and the round trip time of each hop on the path. A unidirectional IP path is defined as the IP devices traversed by IP packets traveling from a source to a destination at a single point in time. Traceroute consists of setting the IP TTL field from

1 to n until the ultimate destination is reached. This behavior leads to sending a packet to each router along a path without actually knowing the path. Upon receiving a packet with an expired (0) TTL, the hop generates an ICMP Time Exceeded response back to the source, thus identifying the hop and its round trip delay. Each UDP packet is sent to a highly numbered port (probably-unused), so when the destination receives the packet it responds with ICMP Port Unreachable.

Many active tools exist in the literature for measuring the bottleneck bandwidth [1, 5, 9, 24] and the available bandwidth [16, 27, 33, 34, 36] along a path. One of these tools is {b&c}probe [5] which is designed to measure the bandwidth capacity and the available bandwidth on the path of a connection. Such informations can be useful for many purposes such as localizing the best server among a set of replicated servers. Actually, the static approaches for server selection that are based on the geographical proximity and the number of hops are not efficient for choosing the best server as we will see in the next section. Therefore, we need in some cases to estimate the bandwidth on the paths between the network entities as a critical weight for the inferred network topology.

bprobe is implemented to measure the speed of a bottleneck link by sending a set of packet pairs. Traveling together, two packets are queued as a pair at the bottleneck link. Assuming that there are not another packets intervening in the pair, so the inter-packet spacing will be proportional to the processing time of the second packet of the pair at the bottleneck router. Given a packet of size m and an inter-arrival time gap of a packet pair, the receiver can measure the base bandwidth of the bottleneck link as follows:

$$B_{bls} = \frac{m}{gap} \tag{1}$$

To enhance the accuracy of the measurement, the bandwidth can be estimated as the average value over many measurement repetitions. This is necessary to avoid abnormal events such as: (i) interfering with competing traffic at the bottleneck router, (ii) queuing failure where it may happen that the packets are not sent too fast to be queued in the bottleneck router. This can be avoided by sending a large number of packet pairs with increasing the size of packet pairs.

cprobe is designed to measure the available bandwidth by sending a number N of streams of ICMP echo packets. By marking the receiving time of the first packet (T_1) and of the last packet (T_2) of a stream, the sender calculates the receiving rate of the stream by dividing the

number of stream's bytes being sent by the inter-arrival time $T_2 - T_1$. Then, the available bandwidth is evaluated as the average value over the N receiving rates already calculated.

The active tool Pathload [17] is also designed to measure the available bandwidth on the path between two network entities S and R by running a process at S and a process at R. Briefly, Pathload is based on the following key idea: When a process at S sends a periodic stream of UDP packet at a rate R_s higher than the available bandwidth in the path, the relative one-way packet delays show an increasing trend and the next fleet rate is lower than R_s. When the stream rate is lower than the available bandwidth, the relative one-way packet delays show no consistent trend and the next fleet rate is higher than R_s. The iterative estimation algorithm of Pathload terminates when the rate of two successive fleet is less than a user-specified resolution,

3.2 Active Approaches

An active approach provides a specific network topology information using an inference architecture which depends on an active tool (described in the previous section). In this section, we describe briefly the famous active approaches existing in the literature. First, we describe how to infer the topology of the Internet using the answers of traceroute probes. Then, we explain the Landmark approach for locating the machines in the Internet based on some reference points. Finally, we describe some interesting active schemes proposed in the literature for selecting the best server.

3.2.1 Using traceroute probes

Many approaches are designed and implemented to infer the topology of the Internet using the basic idea of the traceroute tool (described in section 3.1). The major problems of inferring the topology of the Internet using traceroute probes are: (i) we need a large number of machines geographically distributed to be the sources of traceroute probes, (ii) we need to know the placement of these machines and the identities of the machine pairs (source-destination) so as to maximize the amount of new information we would obtain, (iii) we need to identify the network interfaces which belongs to the same IP device (alias resolution). Next, we describe some solutions which have investigated these problems partially or entirely.

Skitter [8] is a widely used tool deployed by CAIDA[3] for actively probing the Internet in order to analyze topology and performance. It uses BGP tables and a database of Web

[3] `http://www.caida.org`

servers to find destination prefixes. These prefixes are probed from about 20 skitter monitors geographically distributed. Similarly to traceroute, skitter determines the unidirectional IP path from its location (the probing source) to a destination. Figure 2 shows the skitter output packets. In the payload, 12 bytes are reserved for kernel timestamping. This value is obtained from a kernel module added to the operating system on which skitter runs in order to improve the accuracy of the round trip time measurements. When not using this kernel timestamping, the next 8 bytes are used in the same manner as ping: they hold a timeval representing the sending time of the packet. The next 4 bytes hold the destination address. They add this field to resolve the alias resolution problem. Basically, a router may transmit an echo reply via an interface that is different than the interface to which the echo request was sent, and may use the IP address of the transmitting interface as the source address in the echo reply. In this case, we can recognize that the source address and the address indicated in this last field belong to the same IP device.

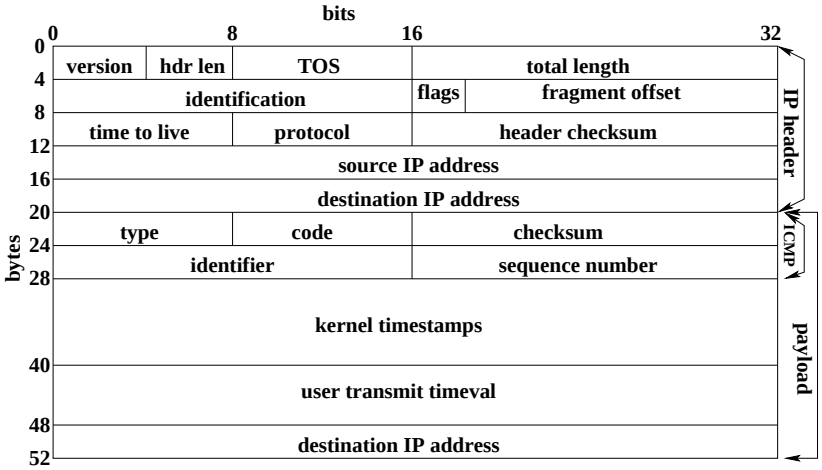

Figure 2: Skitter output packet format

Another solution, called Rocketfuel [35], infers the topology using traceroute probes, BGP routing tables, and DNS information. The key concept of this work is to infer an ISP map which is more complete than that obtained by the other mapping methods along with using as few measurements as possible. The architecture of Rocketfuel (described in Figure 3) aims mainly to find out the useful prefixes to be probed for the discovery of an ISP

topology, then to collect the aliases which are belong to the same router. Next, we describe the main functionalities of this architecture.

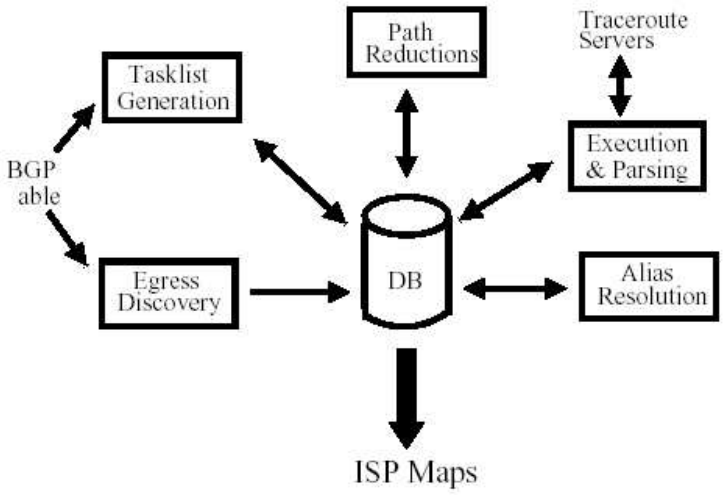

Figure 3: Architecture of Rocketfuel. The Database (DB) becomes the inter-process communication substrate.

Using BGP tables, they pick up the prefixes where their traceroutes will transit the ISP being mapped. A BGP table maps destination IP address prefixes to a set of ASes that can be traversed to reach that destination. Based on BGP tables, Rocketfuel defines three classes of traceroutes that should transit the ISP network: (i) Traceroutes to dependent prefixes which are originated by the ISP or one of its customers. All traceroute probes to these prefixes from any vantage point should transit the ISP. (ii) Traceroutes from insiders which are traceroute servers located in a dependent prefix. Traceroutes from insiders to any prefix should transit the ISP. (iii) Traceroutes that are likely to transit the ISP based on the AS-path indicated in the BGP table.

Then, Rocketfuel filters these prefixes by suppressing those are likely to follow redundant paths through the ISP network. Three techniques are used to reduce the redundant measurements: (i) Ingress Reduction. When traceroutes from two different vantage points to the same destination enter the ISP at the same point, the path through the ISP is likely to be the same as shown in Figure 4a. In this case, only one probing is needed either from T1 or T2. (ii) Egress Reduction. Similarly, traces from the same ingress to any prefix behind

the same egress router should traverse the same path as shown in Figure 4b. Thus, only one prefix is needed for being probed either P1 or P2. (iii) Nest-hop AS Reduction. The path through an ISP usually depends only on the next-hop AS, not on the specific destination prefix (see Figure 4c). Thus, only one trace from ingress router to next-hop AS is needed so as by probing either P1 or P2. Using the described techniques, Rocketfuel find that it can reduce the number of traces required to map an ISP by three order of magnitude compared to the all-to-all approach.

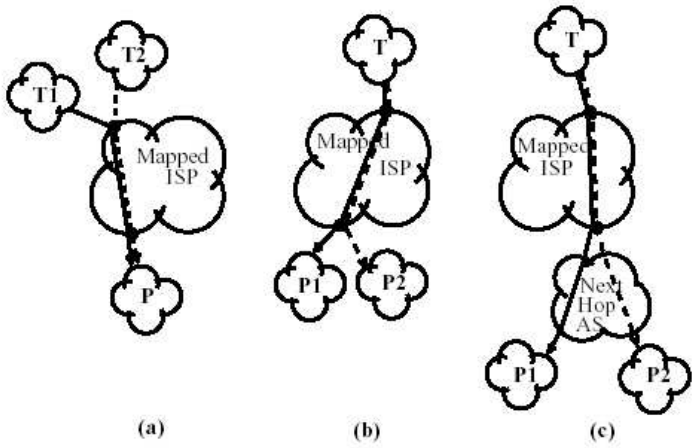

Figure 4: Path reductions

Also, Rocketfuel considers the alias resolution problem for assembling the IP interfaces listed in a traceroute into their corresponding routers. Rocketfuel's alias resolution tool, called Ally, depends on Mercator technique [13]. It consists of sending traceroute probe to the potentially aliased IP addresses with a highly numbered UDP port, a TTL of 255, and consecutive IP identifiers x and y. We remind that an IP identifier is designed to uniquely identify the portions of a packet for reassembly after fragmentation. If both aliases belong to the same router, the router will respond by two "UDP port unreachable" messages with the address of the outgoing interface as the source address and including the identifiers x and y respectively. Then, Ally sends a third message to the address that responded first with an identifier z as shown in Figure 5. Basically, if $x < y < z$, and z - x is small, the addresses are likely aliases.

Finally, Rocketfuel relies on the DNS informations to check up if an obtained router address belongs to the ISP being mapped. Also, these informations serve to identify the

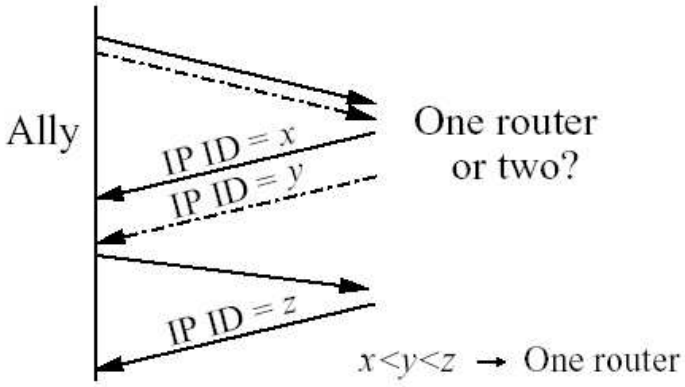

Figure 5: Alias resolution by IP identifiers. A solid arrow represents messages from an IP to an alias, the dotted arrow to another alias.

router role and location. Basically, each ISP has a naming convention for its routers as for example, sl-bb11-nyc-3-0.sprintlink.net is a Sprint backbone (bb11) router in New York City (nyc).

3.2.2 Using Landmark approach

Landmark approach aims to localize the machines in the Internet based on some reference machines called *landmarks*. A number N of landmarks is spread over the Internet with well-known addresses. By measuring the delay to each landmark, a host i obtains a vector of delays $X^i(d_1^i, d_2^i, ..., d_N^i)$. The closeness between the network nodes are based on the vectors X^i.

The advantage of this approach is to estimate the closeness between network nodes without the need to know and probe each other. The major problems of this approach are specifying: (i) the number of landmarks and their distribution, (ii) the metric to use for estimating the closeness degree among pairs of nodes, (iii) the accuracy of the estimation.

The most popular metrics used in the literature for estimating the closeness degree between nodes are: (i) Cartesian distance, (ii) Hotz metric, (iii) Binning method. Assuming that the network is an N-dimensional cartesian space and the N landmarks are the axis of this space, so the delay vector X^i is the coordinates of the node i in this virtual space. Thus, the distance between two nodes i and j can be expressed as the cartesian distance between their coordinates as the following:

$$D = \sum_{l=1}^{l=N} (d_l^i - d_l^j)^2 \tag{2}$$

Hotz [15] estimates the distance between two nodes based on the delay vectors of these nodes. It consists of bounding the distance d between two nodes i and j by two values which are the difference distance $|d^i - d^j|$ and the summation distance $|d^i + d^j|$ as shown in the following equation:

$$|d^i - d^j| < d < |d^i + d^j| \tag{3}$$

where d^i and d^j are the distances between a landmark l with nodes i and j respectively. Thus, if there are N landmark points then the delay between two nodes i and j is lower bounded by $max|d_l^i - d_l^j|$ and upper bounded by $min|d_l^i - d_l^j|$ where l vary from 1 to N as shown in the following equation:

$$max_{l=1...N}|d^i - d^j| < d < min_{l=1...N}|d^i + d^j|. \tag{4}$$

Then, the distance between the nodes i and j is estimated as the average value of both bounds as shown in the following equation:

$$d = \frac{max_{l=1...N}|d^i - d^j| + min_{l=1...N}|d^i + d^j|}{2}. \tag{5}$$

Another method for comparing the positions of machines is the binning method. This method consists of clustering the nodes into bins where a bin is identified by a specific increasing order of closeness to landmark points. Thus, each ordering of landmarks represents a bin (e.g., for $N = 4$, l_2, l_4, l_1, l_3 represents a bin). A node measures its RTT to each landmark and ranks the landmarks in an increasing order of RTT. Then, each node belongs to the bin which has the same ordering of landmarks. Thus, the machines which belong to the same bin have the same order and it is expected that these nodes are relatively closer to one another than to nodes not in their bin. Moreover, it is expected that the more the order is different the farther are the machines.

The representation of a bin can be refined to use not only the ordering of landmarks but also the absolute value of the RTT measurements. These values can be indicated by dividing the range of possible latency values into a number of levels. For example, the range of possible latency values can be divided into the three following levels: (i) level 0

for latencies in the range [0,100]ms, level 1 for latencies in the range [100,200]ms, and (iii) level 2 for latencies greater than 200ms. Thus, a bin can be identified by this notation: $l_i l_j l_k : a_1 a_1 a_3$ where $l_i l_j l_k$ is the relative ordering of landmarks and $a_1 a_1 a_3$ is the latency levels of each ordering respectively.

One potential application that can profit from the landmark approach is the *topologically-aware construction of overlay networks*. Based on the binning strategy, [31] constructs the overlay network CAN [30] (Content-Addressable Network) to be congruent with the underlying IP topology. CAN is an application-level network forming a virtual d-dimensional Cartesian coordinate space. This space is partitioned among all the nodes in a manner that each node is responsible of a space portion. Obviously, routing in CAN consists of sending packets along the direct line path through the cartesian space from the source to the destination coordinates as shown in the example drawn in Figure 6.

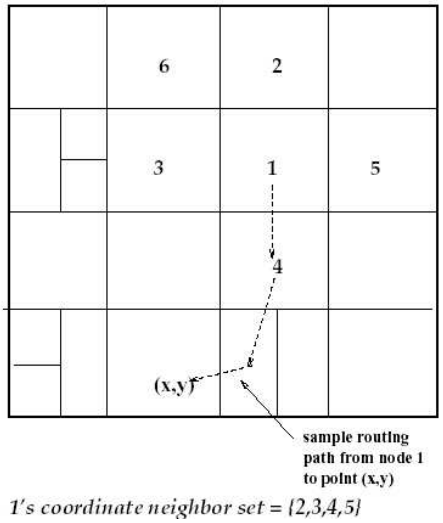

Figure 6: Example on routing in 2-d space

The binning scheme consists of dividing the space into the number of possible orderings of landmarks. Thus, there are m! equal sized portions when the number of landmarks is equal to m where each portion belongs to a single ordering. The partitioning of the space is done in a cyclical ordering of the dimensions (e.g., xyzxyzx...). First, the space is divided into m portions along the first dimension. Then, along the second dimension, each portion is divided into m - 1 portions each of which is divided to m - 2 portions and so on. After

that a CAN node determines its bin based on its RTT measurements to the landmarks, the node joins the CAN in a random point of the correspondent portion of the coordinate space which is suitable with its landmark ordering. When a node joins a portion which is already assigned to another node, the portion is divided equally between them. This behavior may achieve an unequal dividing of the space among the overlay nodes. Therefore, the load balancing between the nodes may be affected as well as the average number of hops on the path between two nodes in the CAN space decreases.

A metric called *latency stretch* is used to evaluate the congruence of the CAN topology constructed using the binning strategy with the underlying IP topology. This metric is calculated as the following: first, we express the CAN path latency by $(\#hops)X$(latency of each hop). After obtaining the average value of the CAN path latency, we divide it by the average number of hops in order to not take advantage of the reduced number of hops caused by the binning-based construction of CAN. Then, the obtained average per-hop latency is multiplied by the average number of hops obtained if the space is evenly partitioned. The obtained value is the path latency for a CAN constructed using the binning strategy without taking advantage of the unequal space partitioning. Finally, this value is divided by the path latency on the underlying IP network to obtain the latency stretch. Obtaining small values of latency stretch means that the construction of CAN based on the binning strategy is congruent with the IP topology.

Figure 7 shows the CAN latency stretch as a function of the number of CAN nodes tested using TS-1k topology. TS-1k is a transient-stub [39] topology (with 1000 nodes) which model networks as a 2-level hierarchical graphs with small domains called stubs connected by a backbone of links called transient. Delays are assigned as follows: 20ms for backbone links, 5ms for stub-transient links, and 2ms for intra-stub domain links. CAN size is scaled by adding CAN nodes to the stub (leaf) nodes in the underlying topology. The delay of the link from the end-host node to the stub node is set to 1ms. Thus, when the number of CAN node increases, the density of the graph increases without scaling the backbone domain. Figure 7 compares the latency stretch for the randomly constructed CAN (where nodes joins the CAN at a random point in the space, as described in [30]) to the stretch obtained with the binning-based CAN construction for different values of landmarks. As shown in the figure, the stretch decreases when CAN construction is based on the binning strategy. Moreover, this strategy is more accurate with more landmarks.

Figure 8 shows the same test for topology PLRG (Power-Law Random Graphs [37]) with 1779 nodes. For each link in the topology is assigned a random delay between 5 and 100ms.

Finally, the NLANR data set is used to evaluate the latency stretch. The NLANR is the National Laboratory for Applied Network Research where a distributed network of over 100 active monitors are used to systematically perform scheduled measurements between each other. Table 3 notes the stretch for a 100 node CAN using the NLANR data set. In Figure 8 and Table 3, we observe as in Figure 7 that the binning-based construction of CAN improves the stretch latency especially when the number of landmark increases.

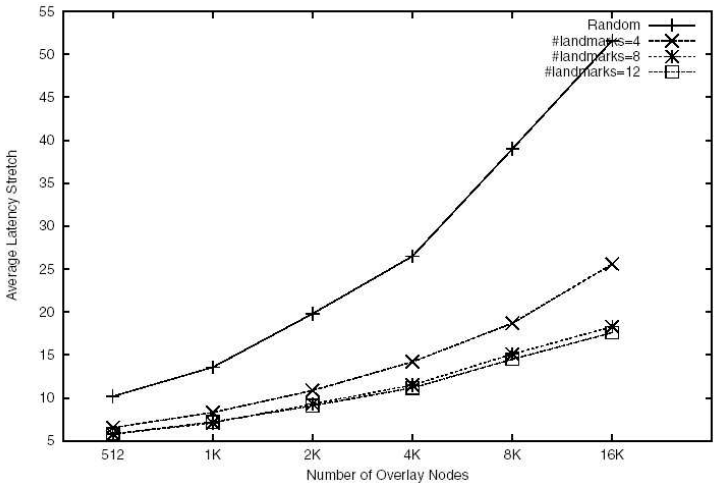

Figure 7: Stretch for a 2-dimensional CAN; topology TS-1K

Construction	Latency Stretch
Random	4.44
Bin, #landmarks=2	4.33
Bin, #landmarks=3	3.9
Bin, #landmarks=4	3.2
Bin, #landmarks=5	3.1

Table 3: Stretch on a 2-dimensional CAN using NLANR

Many other works [11, 14, 15, 29] have also investigate this problem of estimating the network distance between arbitrary hosts without direct measurements between these hosts. In [23], the authors propose a solution for representing the locations of Internet hosts in a Cartesian coordinate system to facilitate the estimation of the network distance between two arbitrary Internet hosts. Basically, as each landmark node defines an axis in the distance

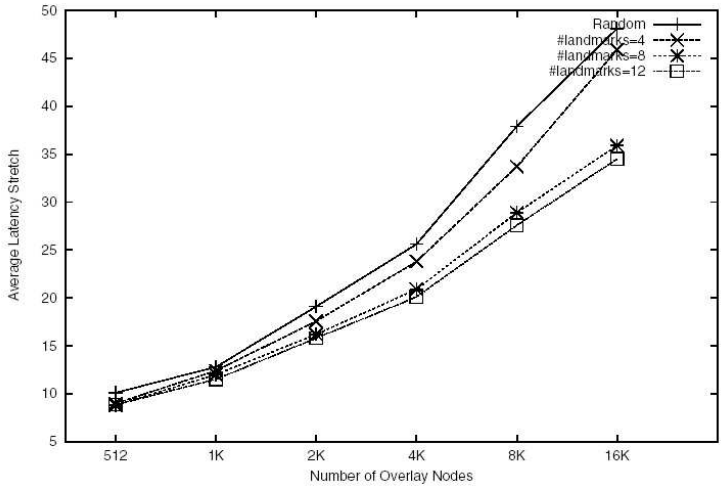

Figure 8: Stretch for a 2-dimensional CAN; topology PLRG

data space, the bases may be correlated. So, the authors propose to project the distance data space into a new, uncorrelated and orthogonal Cartesian coordinate system of much smaller dimensions. The linear transformation essentially extracts topology information from delay measurements between landmark nodes and retains it in a new coordinate system. Then, the coordinate of a host is determined by multiplying its original delay vector to the landmarks nodes with the obtained linear transformation matrix.

3.2.3 Server selection approaches

Service replication is a scalable solution for the distribution of digital content over the Internet. The need for this replication is caused by the increasing number of Internet users and by the desire to improve the QoS. Also, it is important for achieving a high availability of data. Many overlay networks are proposed and installed to realize this replication such as Content Distributed Networks (CDN) and Peer-to-peer networks. The most interesting approaches proposed in the literature for localizing the best server are dynamic and based on active measures due to the dynamic performance in the network and on the servers. In this section, we mention some interesting active approaches (for localizing the best server) that take into account one or many parameters having an impact on the content transfer time.

After proving that the geographical locations and the number of hops are poor predictors of latency, the authors in [6] prove the need of a dynamic approach for server selection. First, they compare between these server selection policies: (i) static, based on geographical distance, (ii) static, based on the number of network hops between client and server, (iii) dynamic, based on random selection of a server, (iv) dynamic, based on the mean of 1,2,3,4, or 5 RTT measurements. The comparison between these policies for server selection is done in the objective of minimizing the content transfer time. The optimal and pessimal policies, which are the minimum and maximum transfer times observed, are also included in the comparison. Figure 9 shows the average time to fetch an object for each of the file sizes (5,10,15, and 20 kbytes). This figure shows clearly the benefit of a dynamic policy especially when the document size increases.

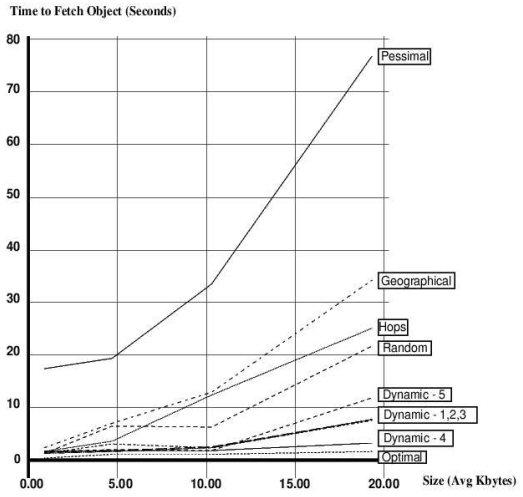

Figure 9: Fetch Times of Static and Dynamic policies plotted against document sizes

Then, they study the profit of considering the available bandwidth. Also, they obtain that the content fetching time is decreased especially when the document size increases. Their developed tool, CPROBE, measures the available bandwidth using a default probing overhead approximately equal to $30,000 bytes$ and up to 4 seconds of measurement time. Therefore, they recommend that the dynamic approach for server selection must avoid injecting a high amount of probing bytes to the network. Thus, they finish by proposing the OnePercent policy for best server selection. This policy consists of choosing the server that

have the minimum round trip time on its path with the requested client. They measure the round trip time as pursue: files under 10,000 bytes require a single ping, files under 20,000 bytes require the mean of two pings, etc until 50,000 bytes, and all the larger documents require the mean of 5 pings. So, they propose to use at most one ping per 10 Kbytes to be transferred. While a policy like the OnePercent can be efficient in term of reducing the probing cost, measuring RTT can not provide always an accurate best server selection as obtained in [25].

In [10], the authors use a technique which combines server push and client probe approaches to identify the best server. The server push technique consists in the server sending to clients or to DNS servers his current performance information every time there is a considerable change in this performance. The client probe technique consists in the client measuring the path between it and a server as for example the measurement of the available bandwidth. This hybrid solution requires installation of proxies that act as probing agents where a probing agent is a cluster's delegate. Each probing agent characterizes its paths with the replicated servers, instead of achieving this task by the clients, along with receiving permanently the performance of the replicated servers. Table 4 compares between this scheme, the nearest server scheme (based on the number of hops), and the random selection scheme. The average and the standard deviation of the response time are reported in the table to show how this hybrid scheme outperforms the nearest server and the random selection schemes.

Server selection Algorithm	Average Response Time (sec.)	Standard Deviation (sec.)
Hybrid scheme	0.49	0.69
Nearest Server	1.12	2.47
Random	2.13	6.96

Table 4: Performance of server selection schemes

To be scalable, a proxy needs to be installed for each nearby set of clients. Based on the measured characteristics of the path and the server performance, the proxy estimates the transfer time from a server and considers this estimate as equal to what the clients located behind it would have obtained. But, the proxy estimate of the transfer time does not necessarily correspond to the actual transfer time to the client, even though the proxy and the client can be nearby located. A mismatch can appear in cases when a bottleneck link, which is highly congested or has a high packet loss rate (e.g., a wireless link), exists on the server - client path while it does not exist on the server - proxy path.

In [31], the authors identify the best server using a set of landmark machines which are geographically distributed. This proposal requires that the client and servers determine their delay vectors by measuring their RTT to a set of landmark points. By knowing the bins of the client and servers (based on their delay vectors), the DNS server can classify the servers (from the best to the worst) based on the distance between their bins and the client's one. Similarly, using the Hotz metric (described in Section 3.2.2) or the cartesian distance, the selected server is that its estimated distance with the client is the minimum. Thus, using the landmark approach the client does not measure the performance of the paths that link it to the servers but rather of those that link it to the landmark points. The drawback of this behavior can be observed in the case where a client and a server have a short delay path to the landmark points while not having such a short delay path between them. Moreover, having a short delay may not be enough to obtain a good transfer time, since other parameters can impact the performance as well, as the available bandwidth and the server load.

Finally, the authors in [25] define a metric that corresponds to a prediction of the content transfer time from a server to the client using TCP. Based on this metric, the servers are ranked from the best one to the worst one regarding the requested client. The originality of this metric is in the fact that it considers the critical characteristics of the paths between servers and client together with the server load. This prediction function incorporates the round trip time, the packet loss rate, and the available bandwidth on the path as well as the maximum receiving window limitation. The experimental traces verify the accuracy of the transfer time prediction for various size contents and the necessity of each used parameter as well.

Figure 10 shows the accuracy of the transfer time prediction for 10 small size contents (between $5KB$ and $400KB$) gathered in Sophia Antipolis from 4 ftp servers located in Canada, Poland, Italy, and Netherlands. In this figure, each point is the average result of 6 times file transfer which are very probably achieved in the slow start phase.

To prove the weakness of the prediction based on the geographical proximity, number of hops (hops) and even RTT, the authors present the following scenario: a client (in Sophia Antipolis) downloads a file of size $S = 75MB$ from two servers S_{berlin} (in Berlin) and S_{paris} (in Paris) during a congested period. As shown in Table 5, S_{paris} is closer to the client than S_{berlin} in term of geographical proximity, number of hops and RTT. While the best server selection based on these parameters must be S_{paris}, the selection based on their PTT (Predicted Transfer Time) function is S_{berlin} which is the correct choice verified by the real

transfer time (ReTT). They observe (in both downloads) that the download rate obtained after dividing the file size by the transfer time, is limited by A (available bandwidth). Thus, the good performance of their prediction in this scenario is caused by the fact that the PTT function considers the limitation of the available bandwidth.

Then, the authors prove that considering the available bandwidth is not sufficient if W_{max} (maximum TCP window size imposed by the sender or the receiver) is not taken into account in the prediction function. Table 6 shows this case, where the transfer is achieved in the congestion avoidance phase for the different large document sizes collected from S_{berlin} (in Berlin) to the client (in Sophia Antipolis). During these connections, the measured parameters have the following values: A is equal to $24.34Mbps$, packet loss rate (P) is equal to zero, and W_{max} is equal to $65535bytes$. They observe that the download rate obtained, after dividing each file size by its transfer time, is limited by $W_{max} * (messagesize)/RTT$ (equal to $12.2Mbps$) even though there is more available bandwidth on the path and a negligible packet loss ratio. This is caused by the sending window limitation which is taken into account in the prediction function. Thus, the window size limits the transmission rate to much less than the measured bandwidth on the path bottleneck, which is equal to $98.25Mbps$.

After showing the critical impact of the available bandwidth (in Table 5) and the maximum sending window size (in Table 6) limitations on the transfer time prediction, the

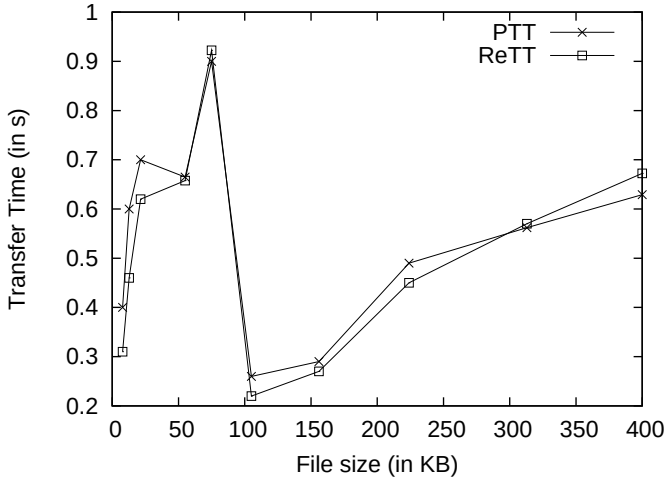

Figure 10: Transfer time prediction for small size contents

	S_{berlin}	S_{paris}
hops	13	10
RTT	41 ms	23 ms
A	10 Mbps	8 Mbps
P	0	0
PTT	60.30 s	75.12 s
ReTT	63 s	77 s
S/ReTT	9.523 Mbps	7.792 Mbps

Table 5: Transfer time prediction when A limits the download rate

	PTT (in s)	ReTT (in s)	S/ReTT (in Mbps)
5.4 MB	3.85	4	10.8
14.63 MB	9.90	11	10.64
25.26 MB	16.88	18	11.23
66 MB	43.60	45	11.73
95.81 MB	63.16	64	11.98
102.24 MB	67.63	68	12.03

Table 6: Transfer time prediction when W_{max} limits the download rate

S_{hkong}					
S	9.75MB	RTT	381.839ms	P	0.03
A	5.77Mbps	W_{max}	45	R_p	115.93kbps
PTT	688s	ReTT	701s	S/ReTT	113.87kbps
S_{poland}					
S	10.64MB	RTT	96 ms	P	0.04
A	6.6Mbps	W_{max}	45	R_p	370.156kbps
PTT	235.334s	ReTT	239s	S/ReTT	364.789kbps

Table 7: Transfer time prediction when P limits the download rate

authors present in Table 7 a trace where the server's maximum sending rate is reduced due to a non-negligible packet loss rate. This trace is the result of 2 files transfer from 2 FTP servers (one located in Hong-Kong and another in Poland) to the end host in Sophia Antipolis. During these connections, the value of the packet loss rate is significant. It is clear in Table 7 that the download rate obtained, after dividing each file size by its transfer time, is limited by R_p (considered as a factor of the TCP average throughput in the congestion avoidance phase) which is less than the available bandwidth on the path and the limited rate imposed by W_{max}. Thus, the sending rate limitation is caused by the packet loss rate, as it is considered in the prediction function.

But, the accuracy of the transfer time prediction requires the measurement of available bandwidth, RTT, and packet loss rate. Measuring the available bandwidth accurately may have an expansive cost for applications of short duration. Thus, the proposed solution can be more useful for applications of long duration such as downloading large files.

4 Topology inference using passive measures

To describe the topology of a network passively, we need a passive tool to monitor the existing traffic circulating in the network along with an inference architecture (see Figure 1) which may involve many other engines.

First, we begin with an overview on the passive measurement tools existing in the literature to have an idea on the existing passive measurement methodologies and the measured characteristics. Then, we describe the main passive approaches which infer the network topology using the described tools as shown in Figure 11.

4.1 Passive measurement tools

A passive tool analyzes network traffic by collecting packets traversing a network link, or a switch/router device. The measurement can be used for showing network traffic in real-time or for analyzing the traffic off-line after saving the measured data. Observing and analyzing the network traffic are important for inferring the network topology and optimizing the communication protocols. There are two main kinds of passive tools: (i) tools that are used for monitoring packets traversing network interfaces such as Tcpdump, IPMON tools, and multiQ, (ii) tools that are used for providing router/switch traffic statistics such as SNMP and Netflow tools.

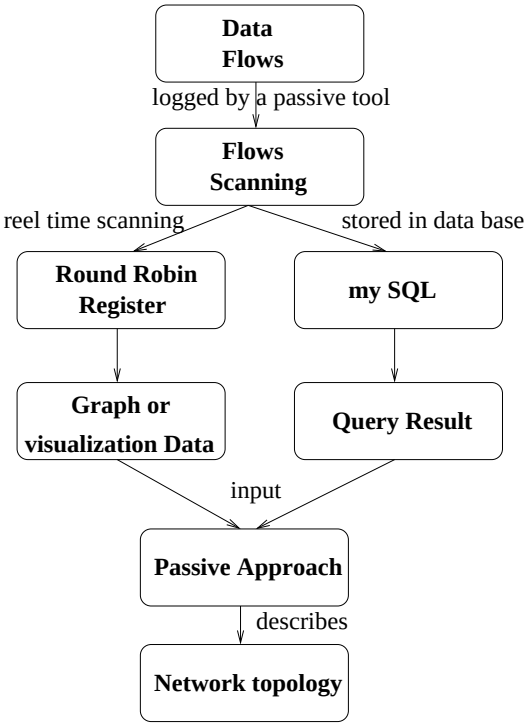

Figure 11: Paradigm of Passive Approach

Tcpdump [4] is a powerful tool that allows us to sniff network packets and make some statistical analysis out of those dumps. Once the statistics are obtained, they can be logged to be used later. Basically, it consists of putting a network interface (i.e., Ethernet, Fiber Distributed Data Interface (FDDI), token-ring, and loopback interfaces) to promiscuous mode and capture selective frames passing by it as specified by the user. Figure 12 shows the way that tcpdump runs using BSD Packet Filter (BPF) which is the driver of receiving copies (of the sent and received packets) from the ethernet device driver. The user can set a filter so only interesting packets go through the Kernel up to the user process. SUN OS uses Network Interface Tap (NIT) which only allows to capture packets received from the interface but no packets sent by the host. SUN OS tcpdump does the filtering at the user process level which means that more data goes through the kernel.

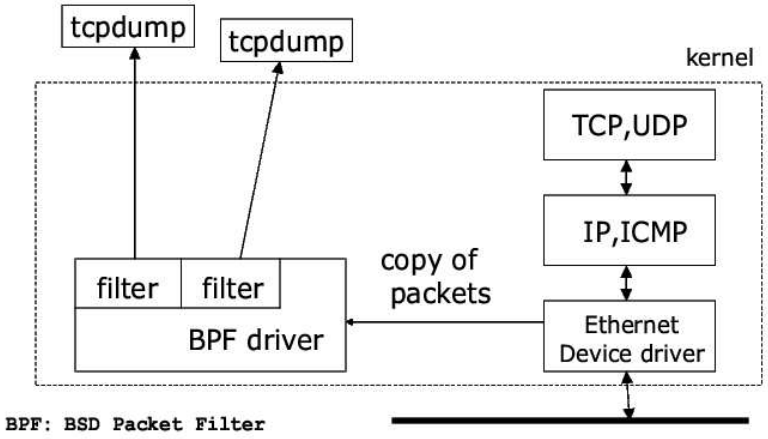

Figure 12: Packet filtering using BPF

Tcpdump has a rich expression syntax that allows the user to select only those packets necessary to troubleshoot a problem. Especially on the firewall, where possibly thousands of packets per second are traversing, it's critical to limit what tcpdump displays. If no expression parameter is given, all packets passing the network interface will be dumped. The Expression parameter consists of one or more primitives. Primitives usually consist of an id (name or number) preceded by one or more qualifiers. There are three types of qualifier: (i) *type*: specifies the kind of the id name or number. Possible types are *host*, *net*, and *port*. If there is no type qualifier, *host* is assumed. (ii) *dir*: specifies a particular

[4] http://www.tcpdump.org/

transfer direction to or from id. Possible directions are *src*, *dst*, *src or dst*, and *src and dst*. If there is no dir qualifier, *src or dst* is assumed. (iii) *proto*: restricts the match to a particular protocol. Possible proto qualifiers are: *ether*, *ip*, *arp*, *rarp*, *tcp*, and *udp*. If there is no proto qualifier, all protocols consistent with the type are assumed.

We show in the following some samples concerning the tcpdump expressions that can be commanded by a user:

1. To print all packets arriving from or departure to www.yahoo.com:
   tcpdump host www.yahoo.com

2. To print all traffic between local hosts and hosts at Berkeley: *tcpdump net ucb-ether*

3. To print traffic neither sourced from nor destined for local hosts: *tcpdump ip and not net localnet*

4. To print the start and end packets (the SYN and FIN packets) of each TCP conversation that involves a non-local host: tcpdump $'tcp[13] \& 3 != 0$ and not src and dst net localnet'

5. To print all ICMP packets that are not echo requests/replies (i.e., not ping packets): tcpdump $'icmp[0] != 8$ and $icmp[0] != 0'$

The output of the tcpdump command varies from one protocol to another. By default, all output lines are preceded by a timestamp. The timestamp is the current clock time in the form *hh:mm:ss.frac*.

We describe briefly in the following the tcpdump output format when the filtered packets are TCP segments. The general output format in this case is:
src > dst: flags data-seqno ack win urg options.

The indication of each field is presented in Table 8. While the fields src, dst and flags are always present, the other fields depend on the contents of the packet's TCP protocol header and are output only if appropriate.

We show in the following example the opening portion of the rlogin (remote login) command from host garfield.inria.fr to host shadow.inria.fr:
$*tcpdump -c7 -S host shadow and tcp and port 1023*
16:28:27.349495 garfield.inria.fr.1023 > shadow.inria.fr.login: S 4266992336:4266992336(0) win 5840 <mss 1460,sackOK,timestamp 60255292[|tcp]> (DF)
16:28:27.349674 shadow.inria.fr.login > garfield.inria.fr.1023: S 3656135324:3656135324(0)

ack 4266992337 win 5792 <mss 1460,sackOK,timestamp 35277822[|tcp]> (DF)
16:28:27.349689 garfield.inria.fr.1023 > shadow.inria.fr.login: . ack 3656135325 win 5840 <nop,nop,timestamp 60255292 35277822> (DF)
16:28:27.349707 garfield.inria.fr.1023 > shadow.inria.fr.login: P 4266992337:4266992338(1) ack 3656135325 win 5840 <nop,nop,timestamp 60255292 35277822> (DF)
16:28:27.349921 shadow.inria.fr.login > garfield.inria.fr.1023: . ack 4266992338 win 5792 <nop,nop,timestamp 35277822 60255292> (DF)
16:28:27.349931 garfield.inria.fr.1023 > shadow.inria.fr.login: P 4266992338:4266992364(26) ack 3656135325 win 5840 <nop,nop,timestamp 60255292 35277822> (DF)
16:28:27.350170 shadow.inria.fr.login > garfield.inria.fr.1023: . ack 4266992364 win 5792 <nop,nop,timestamp 35277822 60255292> (DF)

The first line says that TCP port 1023 on host garfield sent a packet to the login port on host shadow. The S indicates that the SYN flag was set. The packet sequence number was 275192024 and it contained no data. The notion first:last(n) means sequence numbers first up to n bytes of user data. There was no ack field, the available receive field win was 5840 bytes and there was a max-segment-size(mss) option requesting an mss of 1460 bytes. Similarly, host shadow replies with a packet which includes an ack field for host garfield's SYN. Then, host garfield acknowledges host shadow's SYN. The · (period) means no flags were set. The packet contains no data so there is no data sequence number. On the sixth output line, host garfield sends host devo 26 bytes of data. The P (PUSH) flag is set in the

Output Field	Specification
src	the source (host) address and port
dst	the destination address and port
flags	S (SYN), F (FIN), P (PUSH) or R (RST) or a single · (period) to indicate no flags
data-seqno	the portion of sequence space covered by the data in this packet
ack	the sequence number of the next data expected from the other direction on this connection (acknowledgement)
win	the number of bytes of receive buffer space available from the other direction on this connection
urg	urgent data in the packet
options	TCP options enclosed in angle brackets

Table 8: Tcpdump output Fields when the observed packets are TCP segments

packet. On the seventh line, host shadow says it received data sent by host garfield up to but not including byte 26.

A performance evaluation can be achieved using the tcpdump traces, such as: (i) the end-to-end delay: it can be estimated using the clock indication, or the timestamp and timestamp echo, (ii) the packet loss rate, packet reordering, packet size, (iii) application mixtures and protocol usage. TCP is one important protocol that can be analyzed using the tcpdump traces. Tcptrace[5] is one tool that uses the measurements collected by tcpdump to analyze tcp connections.

The IP Monitoring Project (IPMON [28]) built a monitoring system which is a measurement and analysis infrastructure capable of collecting packet traces and routing information, and analyzing terabytes of data. There are 60th monitoring systems deployed at 4 PoPs (Points of Presence) of the SprintLink IP backbone. They have collected more than 20TB of data from OC3 (155 Mbps), OC12 (622 Mbps) and OC48 (2.5 Gbps) links.

The *multiQ* tool [21] uses equally-spaced mode gaps in TCP flows packet inter-arrival time distributions to detect multiple bottleneck capacities and their relative order. Unlike other passive tools, multiQ can discover up to three bottlenecks from the trace of a single flow, and can work with acknowledgment as well as data inter-arrivals. MultiQ works in concert with *mystery* which is a TCP loss and RTT analyzer. This passive tool can also be useful to track the evolution of the Internet over time by using Internet traces collected over the years. Using this tool, the results obtained by the authors show that multiQ is highly accurate, and through passive, achieves the same accuracy as Pathrate [9], which is active. Compared to the passive tool Nettimer [22], the authors found that multiQ is more accurate when tools have access only to receiver side traces. They also found that Nettimer can be as accurate as multiQ only when it is given access to both ends of the measured path.

The Simple Network Management Protocol (SNMP [7]) is an application layer protocol that facilitates the exchange of management information between network devices. SNMP enables network administrators to manage network performance, find and solve network problems, and plan for network growth.

As shown in Figure 13, an SNMP-managed network consists of three key components: managed devices, agents, and network-management systems NMSs (which holds the managing entity). Next, we define each of these components.

A managed device is a network node that contains an SNMP agent and that resides on a managed network. Managed devices collect and store management information and

[5] `http://www.tcptrace.org`

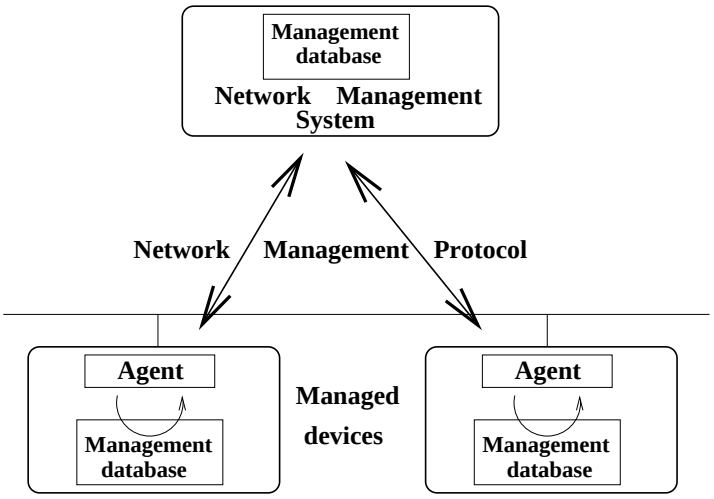

Figure 13: Network Management Infrastructure

make this information available to NMSs using SNMP. Managed devices, sometimes called network elements, can be routers and access servers, switches and bridges, hubs, computer hosts, or printers.

An agent is a network-management software module that resides in a managed device. An agent has local knowledge of management information and translates that information into a form compatible with SNMP.

An NMS executes applications that monitor and control managed devices. NMSs provide the bulk of the processing and memory resources required for network management. One or more NMSs must exist on any managed network.

Netflow[6] is an SNMP-like tool which aims to provide network administrators with access to IP flow information from their data networks. Exported NetFlow data can be used for a variety of purposes, including network management and planning, enterprise accounting and departmental chargebacks, ISP billing, data warehousing, and data mining for marketing purposes. One benefit for Netflow over SNMP management information base statistics is that the SNMP counters provide per-link statistics, such as per source destination statistics or per traffic-class statistics, while NetFlow counters provide flow statistics.

[6] `http://www.cisco.com/en/US/tech/tk812/tech_protocol_home.html`

4.2 Passive Approaches

A passive approach provides a specific network topology information using an inferring architecture which depends on a passive tool (described in the previous section). In this section, we describe how to infer the topology using BGP (Border Gateway Protocol) and then using TCP traffic.

4.2.1 Using BGP

Interdomain routing in the Internet is managed by the Border Gateway Protocol (BGP [32]). BGP is commonly used between ISPs and it allows the border routers to share routing information. Thus, BGP routing tables contain informations on many prefixes in the Internet, such as the identity of the AS a prefix belong to, and the list of the ASs to follow to reach a prefix. Actually, AS operators usually have the use to aggregate their IP network prefixes in order to hide the details of their domain from other operators. Therefore, by looking at a snapshot of the BGP routing tables, one can infer the network topology as a tree that connects ASs, with the list of network prefixes per AS.

The authors in [4] propose a method of inferring logical relation-ships between network prefixes within an AS using only passive monitoring of BGP update messages. Basically, BGP updates its routing information when there is a change. An update is any BGP routing message that is specific to a prefix, such as an announcement, or withdrawal. Each update contains a timestamp indicating the second at which it was received, and the prefix that was affected. Thus, while two network prefixes close to each other have a high probability to be updated at the same time, two network prefixes far from each other have a small probability to be updated at the same time. So, the idea is to study the correlation of update messages of all pairs of prefixes and classify these pairs of prefixes in a binary tree based on the correlation of their update messages.

Thus, the distance metric used to estimate the nearness between two network prefixes is based on the correlation between the update vectors belonging to these prefixes. To compute this, the authors take the discretized update groups, and determine the update vector $u_p^{(t)}$ for each prefix p as the following:

$$u_p^{(t)} = \begin{cases} 1\text{if } p \text{ updated during interval } t \\ 0\text{otherwise} \end{cases} \quad (6)$$

Figure 14 shows an example of update vector creation. The distance metric between two prefixes (p_1, p_2) is evaluated as the correlation coefficient between their corresponding update vectors across n time intervals:

$$C(p_1, p_2) = \frac{\frac{1}{n}\sum_{t=1}^{n}\left(u_{p1}^{(t)} - \overline{u_{p1}}\right)\left(u_{p2}^{(t)} - \overline{u_{p2}}\right)}{\sqrt{\sigma_{p1}^2}\cdot\sqrt{\sigma_{p2}^2}} \quad (7)$$

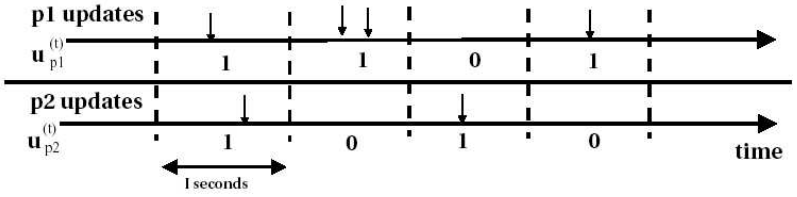

Figure 14: Generating the update vectors, $u_p^{(t)}1$ and $u_p^{(t)}2$ for prefixes (p1,p2), from a stream of BGP updates. Time is discretized into I-second bins.

We notice that $\overline{u_p}$ is the average of u_p, and σ_p^2 is its variance. Then, the authors use for clustering the *Simple-linkage* method [41]. This method computes the $O(n^2)$ pairwise distances between entities and stores them in ranked order. Then, it iterates through the prefix pairs from closest to farthest. When it encounters a new node in a pair, the single-linkage method joins the node to its neighbor, or its neighbor's cluster if the neighbors already been clustered. If both prefixes are in the same cluster, it does nothing. If the prefixes are in different clusters, the two are merged. Figure 15 shows an example of the single-linkage clustering method. The authors considers for evaluating the clustering accuracy three network metrics which are the IP address similarity, the ratio of shared to unshared traceroute path length, and the DNS-based point of presence (PoP) comparison. We notice that a POP is a physical location where the ISP places a collection of routers.

They examine the distance between netblocks to see if the clustering groups netblocks that are close numerically. They count the number of IP addresses that exist between adjacent netblocks. Two adjacent netblocks have a distance of zero. Two netblocks separated by a "/24 class C" netblock have a distance of 2^8, and so on. Figure 16 shows these distances for AT&T and UUNET. The first few clustering decisions cluster prefixes that are numerically close to each other; the next 1/4 or so cluster prefixes that are closer than

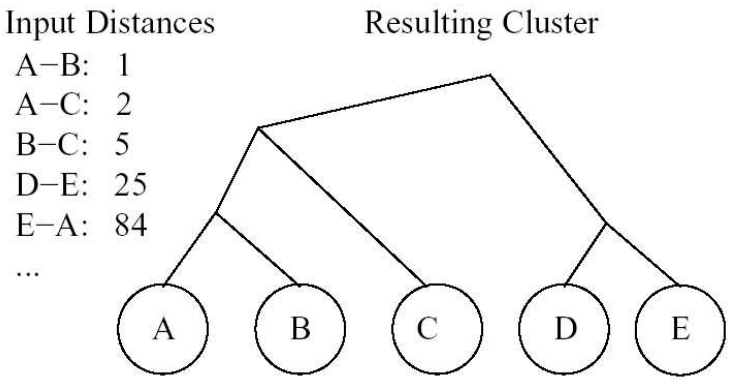

Figure 15: An example of the single-linkage clustering method. The list on the left shows the 5 most closely related items, in order, and the graph on the right shows the clustering that results from this ordering.

average. This is intuitively compatible with the idea that providers allocate IP addresses in a logical, hierarchical way.

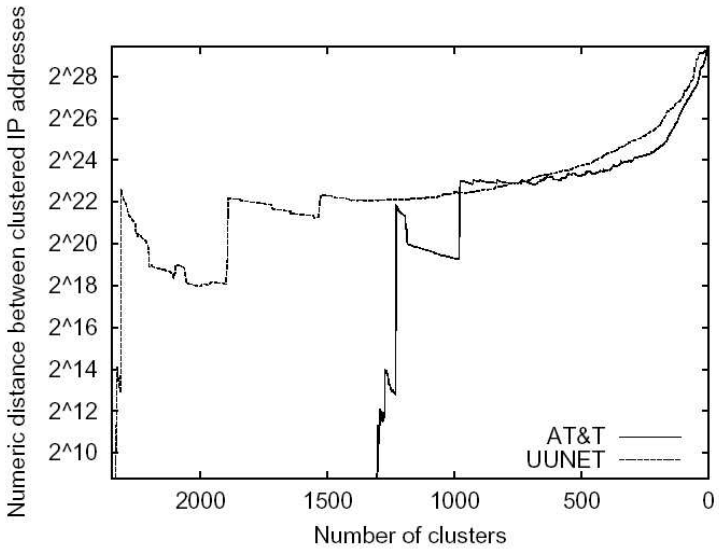

Figure 16: IP address space distance

Figure 17 shows the number of shared traceroute distances between pairs as clustering progresses. This figure shows that temporal clustering produces a grouping of prefixes that is accurate to several more network hops than a random grouping would be especially for the first few hundred clusters. Finally, Figure 18 shows that correlation-based clustering groups the 2338 UUNET prefixes into about 1200 distinct clusters while retaining over 95% PoP-level accuracy. The 1310 AT&T prefixes is grouped into 900 clusters with about 97% accuracy.

Thus, the prefixes linked by this algorithm often share administrative or topological features. Therefore, such temporal clustering can provide an aid to other Internet mapping techniques by guiding the choice of paths on which to traceroute. Better destination selection can increase the information gained from fewer traceroute probes.

The connectivity between ASs alone cannot fully characterize the structural properties of the Internet due to the various policies used between ASs. These routing policies (e.g., selecting routes, propagating reachability information) are constrained by agreements between administrative domains. In [12], the author infers the AS graph with a more detailed representation that classifies AS relationships into customer-to-provider, peer-to-peer, and sibling-to-sibling relationships based on BGP routing tables.

Its algorithm for inferring AS relationships is based on the fact that ASs set up their export policies according to these relationships and on the resulting patterns on BGP routing table entries. Basically, in exchanging routing information: (i) with a provider, an AS can export its routes and the routes of its customers and siblings, but can not export routes learned from other providers or peers, (ii) with a customer, an AS can export its routes and the route of its customers and siblings, as well as routes learned from its providers and peers, (iii) with a peer, an AS can export its routes and the routes of its customers and siblings, but can not export the routes learned from other providers or peers, (iv) with a sibling, an AS can export its routes and routes of its customers and sibling, as well as routes learned from its providers and peers.

This selective export rule indicates that a BGP routing table entry should have a certain pattern. Therefore, an AS path must be valley-free which means: (i) a provider-to-customer edge can be followed by only provider-to-customer or sibling-to-sibling edges, (ii) a peer-to-peer edge can be followed by only provider-to-customer or sibling-to-sibling edges. Thus, the selective export rule ensures that BGP routing table entries contain only valley-free AS paths. As shown in the example of Figure 19, AS paths (1,2,3) and (1,2,6,3) are valley-free while AS-path (1,4,3) and (1,4,5,3) are not valley-free.

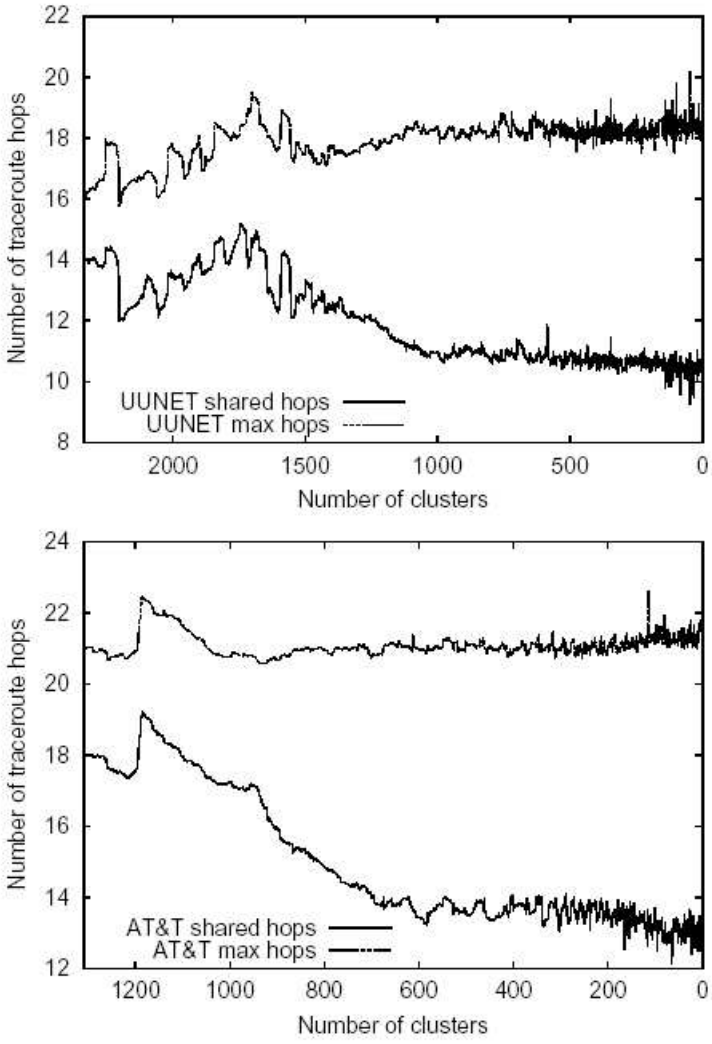

Figure 17: Shared traceroute distance

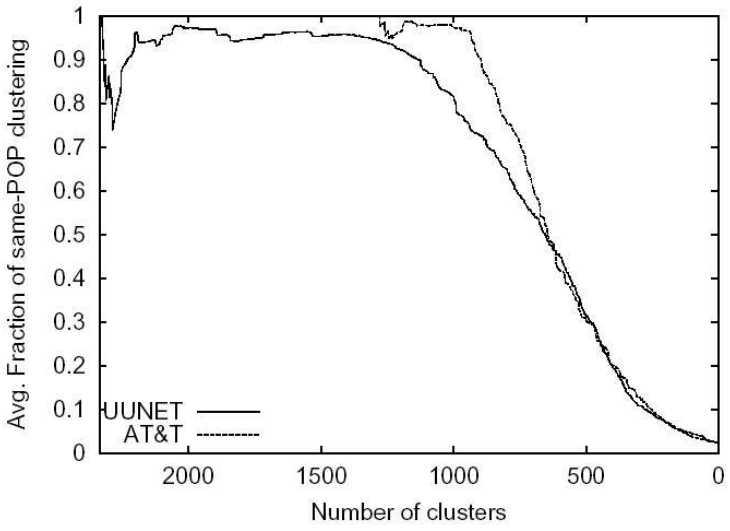

Figure 18: Assignment to the same PoP

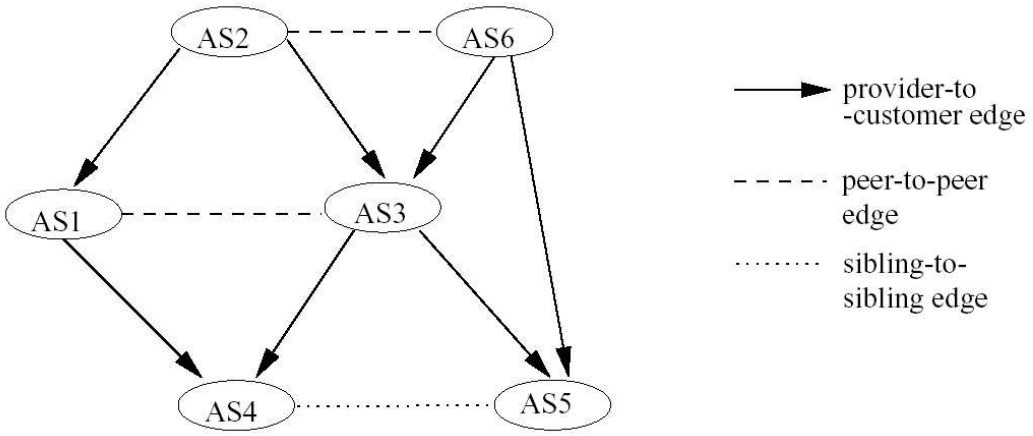

Figure 19: An annotated AS graph

Therefore, an AS path of a BGP routing table entry has one of the following patterns: (i) an uphill path, (ii) a downhill path, (iii) an uphill path followed by a downhill path, (iii) an uphill path followed by a peer-to-peer edge, (iv) a peer-to-peer edge followed by a downhill path, or (v) an uphill path followed by a peer-to-peer edge, which is followed by a downhill path. We notice that a downhill path is a sequence of edges that are either provider-to-customer or sibling-to-sibling edges, and an uphill path is a sequence that are either customer-to-provider or sibling-to-sibling edges.

Their basic algorithm goes through the AS path of each routing table entry. It finds the highest degree AS and lets the AS be the top provider of the AS path. Knowing the top provider, the authors say that consecutive AS pairs on the left of the top provider have a customer-to-provider or sibling-to-sibling relationship, and consecutive AS pairs on the right of the top provider have a provider-to-customer or sibling-to-sibling relationship. Moreover, a peer-to-peer edge can be only between a top provider and one of its neighboring ASes in an AS path. Thus, an AS pair (u1,u2) has: (i) a sibling-to-sibling relationship if and only if the pair provides transit services for each other, (ii) a provider-to-customer (customer-to-provider) relationship if and only if u1 (u2) provides transit services for u2 (u1), (iii) a peer-to-peer relationship if one AS of the pair is a top provider, the pair do not have any relationship among the first two cases, and the pair has a comparable degrees.

Using this algorithm, the author infers that more than 90.5% of the connected AS pairs have customer-to-provider relationships, and less than 1.5% of the connected AS pairs have sibling-to-sibling relationships, and less than 8% of the connected AS pairs have peer-to-peer relationships.

Finally, we notice that monitoring BGP update messages can be useful to predict network failures for reactive routing in overlay networks. In [3], the authors study the correlation between the network failures and the BGP routing instability. They found that failures appearing in the network core correlate better with BGP instability than failures appearing close to end hosts. On average, they observed that most failures precede BGP messages by about four minutes. Also, they observed that passive observations of BGP routing messages could be used to predict about 20% of impending failures, allowing re-routing around faulty paths. Motivated by the fact that BGP-4 detects and recovers from a fault in several minutes, RON (Resilient Overlay Network [2]) is based on active probing for outage detection. RON's routing mechanism (on top of IP) is able to detect, recover, and route around the failures (according to application-specified routing metrics) in less than twenty seconds on average.

Besides, the authors found that forwarding packets via at most one intermediate RON node is sufficient to overcome faults and improve performance in most cases.

4.2.2 Using TCP traffic

Monitoring TCP connections is important from different standpoints: (i) characterizing the end-to-end performance perceived by clients, (ii) correlating the behavior of different TCP connections, to identify common patterns, which may guide to clusterize prefixes, (iii) identifying the ASs that contribute significantly to connection loss, and thus driving down connection throughput.

Many measurement studies have inferred the characteristics of TCP connections in the Internet through passive measures. In [38], the authors infers passively the characteristics of a TCP connection by estimating its aggregate properties (e.g., size, duration, throughput). In [40], the authors monitor TCP connections passively and develop heuristics that are used to classify connections according to the factor(s) that limit their throughput. In [20, 26], the authors considered the problem of estimating the RTT of a connection through passive measurements. These works compute one RTT sample per TCP connection, either during the triple-handshake or during the slow-start phase.

In [19], the authors propose a passive measurement methodology to infer and keep track of the values of two important variables associated with a TCP connection: the sender's congestion window (cwnd) and the connection round trip time (RTT). Knowing these two values, many important characteristics of the sender, receiver, and the network path that connects them can be determined. For example, by comparing cwnd with the amount of data actually sent, one can determine when a TCP connection is starved for application-level data (i.e., that the connection could have a higher transfer rate, if more data were available); by observing the manner in which cwnd changes in response to loss, one can identify the particular flavor of TCP (e.g., Tahoe, Reno, New Reno); by monitoring RTTs, one can characterize RTT variability within and among flows, and determine the extent to which application-level adaptivity is needed to cope with variable network delays. The used passive measurement consists of observing TCP segments passing through a measurement point.

The authors infer a sender's congestion window based on the observed receiver-to-sender ACKs, which (if received at the sender) would cause the sender to change state. Transitions are also caused by detecting a timeout event at the sender. These timeouts are manifested in the form out-of-sequence sender-to-receiver retransmissions, which are detected using

the passive measurement techniques from [18]. But, the three flavors of TCP can respond differently to loss events. To identify the sender's flavor, they are based on these two hypothesis: (i) a TCP sender can never have more outstanding unacknowledged packets than its usable window size, (ii) a sender's usable window size is the smaller of cwnd and the window advertised by the receiver. For every data packet sent by the sender, they check whether this packet is allowed by the current estimate of cwnd for each particular flavor. Given a flavor, if the packet is not allowed, then the observed data packets represents a violation which is a restricted event. By maintaining a count of the number of such violations for each of the candidate flavors, the sender flavor is inferred to be that flavor with the minimum number of violations, and at any time during the life of a connection, the sender's congestion window is the value of cwnd as estimated for this flavor.

The basic idea behind their RTT estimation technique is illustrated in Figure 20. Since they are not able to directly measure the sender's RTT sample shown in the left of the figure, they instead measure: (i) the round trip delay from the measurement point to the receiver and then back to the measurement point (labeled d1 in the figure), and (ii) the round trip delay between the measurement point, the sender and back to the measurement point (labeled d2 in the figure). The sum of these two delays $d1 + d2$ is their RTT estimate as shown in the figure.

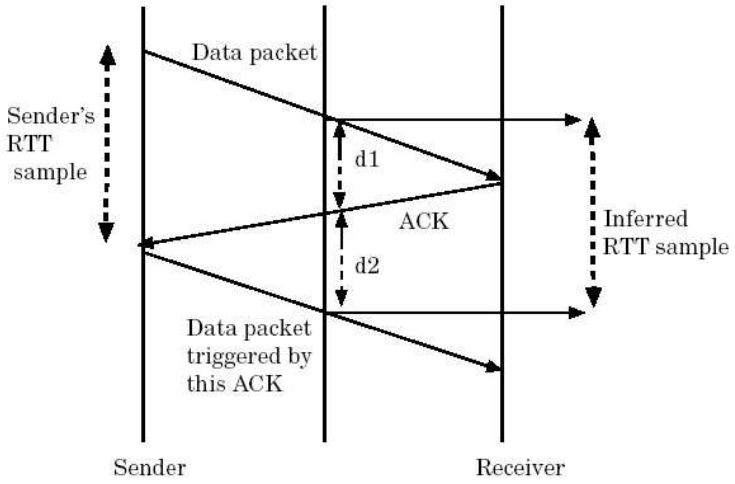

Figure 20: RTT estimation technique

The main results incurred from this work are: (i) the sender throughput is often limited by lack of data to sent, rather than by network congestion, (ii) in the few cases where TCP flavor is distinguishable, it appears that NewReno is the dominant congestion control algorithm implemented, (iii) connections do not generally experience large RTT variations in their lifetime. The ratio between the 95th and 5th percentile of the RTT is less than 3 for approximately $80 - 85\%$ of the connections.

5 Conclusion

Inferring the topology of the Internet is a wide research domain which is of particular interest for network operators and users. It is necessary for controlling the network and for better tuning of its protocols and applications. Therefore, we present in this report the state-of-the-art of Internet topology inference. We overview the existing measurement tools (e.g., traceroute, tcpdump) and then we study the schemes which use these tools for providing network topology informations. We are interested by the schemes that identify the connectivity between certain network entities (e.g., routers, overlay nodes) along with those that characterize the performance on the paths among the network entities. We classify these works into two main categories which are aliased by active and passive. While active approaches (e.g., Rocketfuel [35]) infer a network topology through active measures, passive approaches (e.g., Gao [12]) use for inferring the topology passive monitoring of network's traffic and the routing tables as well.

However, still the continuous evolution of the Internet and its distributed administration make the inference of its topology more and more complex. Moreover, the emerging IPv6 clouds in the Internet does not simplify the task but we will have a mosaic in the new hybrid IP topology (IPv4/IPv6). Once the transition from IPv4 to IPv6 is completed (if this happens), our problem becomes more simple since the IPv6 address scheme is assigned in a hierarchical way.

References

[1] A. Akella, S. Seshan, and A. Shaikh. An empirical evaluation of wide-area internet bottlenecks. *IMC*, October 2003.

[2] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. *ACM SOPS*, 2001.

[3] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Measuring the effects of internet path faults on reactive routing. *Sigmetrics*, 2003.

[4] D. Andersen, N. Feamster, S. Bauer, and H. Balakrishnan. Topology inference from bgp routing dynamics. *ACM Internet Measurement Workshop*, 2002.

[5] R. Carter and M. Crovella. Dynamic server selection using bandwidth probing in wide-area networks. *Technical Report, Boston University*, March 1996.

[6] R. Carter and M. Crovella. Server selection using dynamic path characterization in wide-area networks. *IEEE Infocom*, April 1997.

[7] J. Case, M. Fedor, M. Shoffstall, and J. Davin. Simple network management protocol. *RFC 1157*, May 1990.

[8] K. Claffy, T. Monk, and D. McRobb. Internet tomography. *In Nature*, January 1999.

[9] C. Dovrolis, P. Ramanathan, and D. Moore. What do packet dispersion techniques measure? estimation. *IEEE Infocom*, 2001.

[10] Z. Fei, S. Battacharjee, E. Zegura, and M. Ammar. A novel server selection technique for improving the response time of a replicated service. *IEEE Infocom*, March 1998.

[11] P. Francis, S. Jamin, L. Zhang, D. Gryniewicz, and Y. Jin. An architecture for a global internet host distance estimation service. *IEEE Infocom*, 1999.

[12] L. Gao. On inferring autonomous system relationships in the internet. *Transactions on Networking*, December 2001.

[13] R. Govindan and H. Tangmunarunkit. Heuristics for internet map discovery. *IEEE Infocom*, 2000.

[14] J. Guyton and M. Schwartz. Locating nearby copies of replicated internet servers. *Sigcomm*, August 1995.

[15] S. Hotz. Routing information organization with heterogeous path requirements. *ph.d. thesis*, 1994.

[16] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth techniques. *IEEE JSAC*, 2003.

[17] M. Jain and C. Dovrolis. Pathload: A measurement tool for end-to-end available bandwidth. *PAM*, March 2002.

[18] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Measurement and classification of out-of-sequence packets in a tier-1 ip backbone. *IEEE Infocom*, 2003.

[19] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Inferring tcp connection characteristics through passive measurements. *IEEE Infocom*, 2004.

[20] H. Jiang and C. Dovrolis. Passive estimation of tcp round-trip times. *Technical report*, July 2001.

[21] S. Katti, D. Katabi, C. Blake, E. Kohler, and J. Strauss. A passive toolkit for measuring, tracking, and correlating path characteristics. *under submission*, 2004.

[22] K. Lai and M. Baker. Nettimer: A tool for measuring bottleneck link bandwidth. *Usenix*, 2001.

[23] H. Lim, J. Hou, and C. Choi. Constructing internet coordinate system based on delay measurement. *ACM IMC*, October 2003.

[24] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. User level internet path diagnosis. *ACM SOSP*, October 2003.

[25] M. Malli, C. Barakat, and W. Dabbous. An efficient approach for content delivery in overlay networks. *IEEE CCNC*, 2005.

[26] H. Martin, A. McGregor, and J. Cleary. Analysis of internet delay times. *PAM*, 2000.

[27] B. Melander, M. Bjorkman, and P. Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. *Global Internet Symposium*, 2000.

[28] S. Moon and T. Roscoe. Metadata management of terabyte datasets from an ip backbone network: Experience and challenges. *Workshop on Network-Related Data Management*, 2001.

[29] E. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. *IEEE Infocom*, 2002.

[30] S. Ratnasamy, P. Francis, M. Handly, R. Karp, and S. Shenker. A scalable content-addressable network. *Sigcomm*, August 2001.

[31] S. Ratnasamy, M. Handly, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. *IEEE Infocom*, June 2002.

[32] Y. Rekhter and T. Li. A border gateway protocol 4. *RFC 1771*, March 1995.

[33] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, and R. Baraniuk. Multifractal cross traffic estimation. *ITC*, September 2000.

[34] V. Ribeiro, R. Riedi, and R. Baraniuk. pathchirp: Efficient available bandwidth estimation for network paths. *PAM*, 2003.

[35] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. *Sigcomm*, August 2002.

[36] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. *IMC*, 2003.

[37] H. Tangmunarunkit, R. Govidan, S. Jamin, S. Shenker, and W. Willinger. Network topologies, power laws and hierarchy. *Technical Report, University of Southern California*, 2001.

[38] K. Thompson, G. Miller, and R. Wilder. Wide-area internet traffic patterns and characteristics. *IEEE Network Magazine*, November 1997.

[39] E. Zegura, K.Calvert, and S. Bhattacharjee. How to model an internetwork. *IEEE Infocom*, 1996.

[40] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the characteristics and origins of internet flow rates. *ACM Sigcomm*, 2002.

[41] Jure Zupan. Clustering of large data sets. *John Wiley and Sons*, 1982.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399