Université catholique de Louvain
Faculté des Sciences Appliquées
Département d'Ingénierie Informatique

Unleashing Traffic Engineering for IPv6 Multihomed Sites

Thèse soumise par

Cédric de Launois

en vue de
l'obtention du grade de Docteur en Sciences Appliquées

Composition du Jury :

Prof. Olivier **Bonaventure** (Co-promoteur)	Université catholique de Louvain
Prof. Marc **Lobelle** (Co-promoteur)	Université catholique de Louvain
Prof. Guy **Leduc**	Université de Liège
Prof. Francis **Dupont**	ENST Bretagne
Prof. Timur **Friedman**	Université Pierre et Marie Curie
Prof. Yves **Deville** (Président)	Université catholique de Louvain

– Septembre 2005 –

Table of Contents

List of Figures

List of Tables

Unleashing Traffic Engineering for IPv6 Multihomed Sites

Cédric de Launois
Université catholique de Louvain, 2005

Supervisors : Prof. Olivier Bonaventure and Prof. Marc Lobelle

Abstract

Internet connectivity takes a strategic importance for a growing number of companies. Therefore, for reliability and performance reasons, many Internet service providers and corporate networks connect to at least two providers, a practice called multihoming. However, the current multihoming mechanism contributes to the explosive growth of the Internet routing tables. This growth has major implications for routers on storage requirements, protocol overhead and stability, and forwarding performance. As a consequence, the traditional way to be multihomed in IPv4 is prevented in the next generation IPv6 Internet. Many approaches for IPv6 multihoming were proposed, with little consideration for traffic engineering aspects. The aim of the thesis is to bridge this gap.

The thesis investigates the way to best provide traffic engineering for IPv6 multihomed sites. It first demonstrates that Host-Centric multihoming, the foreseen approach for IPv6 multihoming, is the most promising in terms of fault-tolerance and traffic engineering capabilities. Compared to traditional multihoming approaches, our simulation results show that Host-Centric IPv6 multihomed sites are able to obtain lower delays by leveraging the path diversity that underlies the Internet. Unfortunately, no traffic engineering mechanism is available for this multihoming approach. Therefore, this thesis next presents a technique to effectively use the multiple interdomain paths that exist between multihomed sites. The proposed mechanism allows the multihomed sites to control how their flows are distributed over the links with their providers. The mechanism is able to take into account complex and very dynamic routing policies. Finally, the thesis proposes the use of synthetic coordinates as a scalable and efficient way to help hosts in selecting the interdomain paths with the lowest delays. Experimental results with real measurements show that this mechanism allows sites to avoid all paths with really bad delays, and to most often select the lowest delay path.

Preface

Overview of the Thesis

The Internet connects today more than 18000 *Autonomous Systems* (AS) [112]. An autonomous system can be defined as a set of networks operated by the same technical administration. The large majority of autonomous systems do not allow external domains to use their infrastructure, except to reach them. These domains are named *stub* ASes. Autonomous systems that provide transit services to other ASes are called *transit* ASes. The Border Gateway Protocol (BGP) is used to distribute routing information among routers that interconnect ASes.

Internet connectivity takes a strategic importance for a growing number of companies. Therefore, many ISPs and corporate networks wish to be connected through at least two providers to the Internet, primarily to enhance their reliability in the event of a failure in a provider network, but also to increase their network performance such as network latency. *Site Multihoming* refers to those stub networks that connect to at least two different network service providers, while *ISP Multihoming* refers to transit providers that are multihomed. In today's IPv4 Internet, at least 60% of stub domains are multihomed to two or more providers [3], and this number is growing. Many sites are expected to also require to be multihomed in IPv6.

Unfortunately, according to a work about the growth of the routing tables [37], the current multihoming mechanism is responsible for approximately 20 to 30% more prefixes in the rapidly growing BGP routing tables of the Internet. The current size of those tables causes operational issues for some Internet Service Providers, as it can decrease the packet forwarding speed and demands large memory space [110]. Moreover, several experts are concerned about the increasing risk of instability of BGP [15].

In order to preserve the size of those BGP routing tables, the IPv4 multihoming mechanism is prevented in IPv6 by operational procedures [173]. Hence, a new multihoming solution that ensures route aggregation is required for IPv6 [15]. This is described in Chapter 3.

Over 40 solutions for IPv6 multihoming have been proposed during the last few years [22]. Chapter 4 reviews the most relevant ones. It methodically compares the solutions according to their mechanisms, benefits and drawbacks, and outlines the major steps that have led to a new multihoming architecture for IPv6. Three representative approaches are distinguished : *Routing*, *Middle-Box*, and *Host-Centric*. Middle-box approaches have many concerns, and do not appear to be the best solutions for IPv6 multihoming. In Routing approaches, hosts use a single IPv6 address, transport-layer survivability being provided by routing mechanisms. In Host-Centric approaches, multihomed sites receive multiple provider-aggregatable (PA) prefixes, one per provider. Hosts within those multihomed sites use several IPv6 addresses, and use them interchangeably during the lifetime of a flow in order to survive outages affecting any of those addresses. Both Routing and Host-Centric approaches allow route aggregation. Routing approaches do not require the hosts be updated, but provide fault-tolerance only for the links with the providers. On the other hand, Host-Centric approaches provide complete fault-tolerance, but require modifications to the hosts.

This thesis focuses on the traffic engineering aspects of IPv6 multihoming. The term "traffic engineering" is defined as [18, 19] :

> *Internet traffic engineering is defined as that aspect of Internet network engineering dealing with the issue of performance evaluation and performance optimisation of operational IP networks. Traffic Engineering encompasses the application of technology and scientific principles to the measurement, characterisation, modelling, and control of Internet traffic.*

At the start of this thesis in 2002, the IETF was still actively developing multihoming architectures, with few considerations for traffic engineering aspects. The purpose of this thesis is to answer the following questions :

> *Which IPv6 multihoming architecture is the most promising, in terms of fault-tolerance and network performance ?*
>
> *How to engineer the traffic in Host-Centric environments, taking into account complex and dynamic load sharing policies, in an efficient and scalable way ?*
>
> *In Host-Centric environments, how to find the low delay paths towards any given destination, without prior communication with this destination, without actively probing all paths, and without adding QoS attributes to BGP ?*

Chapter 5 shows that Host-Centric approaches bring significant advantages in terms of fault-tolerance and traffic engineering capabilities. It shows how stubs that use multiple PA prefixes can exploit paths that are otherwise unavailable. In other words, it explains how the use of such prefixes increases the number of concurrent paths, i.e. the AS-level path diversity. By use of simulations, we show that a dual-homed stub that uses several prefixes has a better Internet path diversity than any multihomed stub that uses a single PI prefix, whatever its number of providers. We also show in Chapter 5 that lower delays can often be found among the new paths introduced by the use of multiple PA prefixes. Simulations suggest that a delay improvement is observed for approximately 60% of the stub-stub AS pairs, and that the delay improvement could be higher in the actual Internet.

However, the Host-Centric approach for IPv6 multihoming has implications on traffic engineering methods, as the source address selected by a host determines the upstream provider used. Therefore, the outgoing traffic is entirely determined by how hosts select their source addresses. No traffic engineering mechanism has ever been developed for such Host-Centric environment, despite the large opportunity for leveraging the interdomain path diversity. Chapter 6 presents a technique to effectively use the multiple interdomain paths that exist between multihomed sites. It focuses on a new mechanism, named *NAROS*, to allow a site to engineer its traffic. Thanks to this mechanism, Host-Centric multihomed sites are able to share their traffic load without manipulating the BGP attributes. The mechanism is scalable, and is able to take into account complex and very dynamic routing policies.

The next chapter, Chapter 7, focuses on how to identify low delay paths, in order to better exploit the path diversity. Chapter 7 presents a solution to find the low delay paths for any given destination, without prior communication with this destination, without actively probing all paths, and without adding QoS attributes to BGP. The solution is based on the use of synthetic network coordinates, assigned to each prefix of a multihomed site, and stored in the Domain Name System (DNS).

In short, Chapter 5 shows that many interdomain paths exist for Host-Centric IPv6 multihomed sites, and that some of the paths have low delays. Chapter 6 presents a mechanism to enable those IPv6 multihomed sites to effectively control which interdomain path is used to send and receive traffic. Finally, Chapter 7 explains how to identify low delay paths, in order to better exploit the path diversity.

Structure of the Thesis

In Part I of this thesis, we present the challenges of finding a suitable multihoming solution for an IPv6 Internet. Chapter 1 provides the necessary background on interdomain routing and on multihoming. Chapter 2 provides some background on

IPv6. In Chapter 3, the IPv6 multihoming problem is stated. Chapter 4 presents and reviews the major solutions proposed for IPv6 multihoming.

In Part II, we focus on the way to best provide traffic engineering capabilities to IPv6 multihomed sites. In Chapter 5, we identify the key advantages of the Host-Centric approach to IPv6 multihoming. In Chapter 6, we propose a mechanism that allows such multihomed sites to effectively engineer their traffic. In Chapter 7, we propose and evaluate a mechanism, based on the usage of synthetic network coordinates, to identify the low delay interdomain paths for any given destination. In Chapter 8, we envision the future state of traffic engineering in Host-Centric IPv6 multihomed sites. We present the future works, and finally conclude the thesis.

Further detailed informations about the NAROS protocol are given in Appendix A. In Appendix B, we present a new DNS resource record to store synthetic coordinates in the Domain Name System.

Thesis Statement

> *The Host-Centric multihoming approach is the most promising approach in terms of traffic engineering capabilities, because it allows sites to exploit interdomain paths that are otherwise unavailable. This approach does not jeopardise the scalability of the interdomain routing.*
>
> *There exists a mechanism to engineer the traffic in such Host-Centric environments. This mechanism is able to take into account complex and dynamic load sharing policies, in an efficient and scalable way.*
>
> *In Host-Centric environments, there also exists a mechanism to find the low delay paths towards any given destination, without prior communication with this destination, without actively probing all paths, and without modifying BGP.*

Publications made during the Thesis

During the thesis, the following papers related to IPv6 multihoming were written :

- C. de Launois, O. Bonaventure and M. Lobelle, *The NAROS Approach for IPv6 Multi-homing with Traffic Engineering.* In Proc. QoFIS 2003, LNCS 2811, pp. 112-121, October 2003, Sockholm, Sweden.

- C. de Launois, B. Quoitin and O. Bonaventure, *Leveraging Network Performance with IPv6 Multihoming and Multiple Provider-Dependent Aggregatable Prefixes*. In Proc. QoSIP 2005, LNCS 3375, pp. 339-352, February 2005, Catania, Italy.
- C. de Launois, S. Uhlig and O. Bonaventure, *Scalable Route Selection for IPv6 Multihomed Sites*. In Proc. Networking 2005, LNCS 3462, pp. 1357-1361, May 2005, Waterloo, Canada.
- C. de Launois, B. Quoitin and O. Bonaventure, *Leveraging Network Performance with IPv6 Multihoming and Multiple Provider-Dependent Aggregatable Prefixes* (Extended version). To appear in Elsevier Computer Networks, April 2005.
- C. de Launois and M. Bagnulo, *The Paths Towards IPv6 Multihoming*. Submitted to IEEE Network Magazine, April 2005.
- M. Bagnulo, A. García Martínez, A. Azcorra and C. de Launois, *An Incremental Approach to IPv6 Multihoming*. Elsevier Computer Communications, July 2005.

The following other documents, also related to IPv6 multihoming, were written :

- C. de Launois and O. Bonaventure, *NAROS : Host-Centric IPv6 Multihoming with Traffic Engineering*. Internet-Draft, <draft-de-launois-multi6-naros-00.txt> (expired), work in progress, May 2003.
- C. de Launois, B. Quoitin and O. Bonaventure, *Leveraging Internet Path Diversity and Network Performance with IPv6 Multihoming*. Research Report RR 2004-06, August 2004.
- C. de Launois, S. Uhlig and O. Bonaventure, *A Stable and Distributed Network Coordinate System*. Technical Report, December 2004.

Additionally, two papers, not related to IPv6 multihoming were written during the thesis. We choose not to include them in this work in order to maintain the consistency of this thesis. These papers are :

- C. de Launois, A. Bonnet and M. Lobelle, *Connection of Extruded Subnets : a Solution Based on RSIP*. In Proc. Networking 2002, LNCS 2345, pp. 685–696, May 2002, Pisa, Italy.
- C. de Launois, A. Bonnet and M. Lobelle, *Connection of Extruded Subnets : a Solution Based on RSIP* (Extended version). In IEEE Communications Magazine, vol. 40, no. 9, September 2002.

Acknowledgements

The realisation of this thesis would not come to an end without the support of many persons. Acknowledging every person who had an impact on it is a difficult task in such a limited space.

Let me first thank Olivier Bonaventure, my advisor, who is the very first person responsible for the achievement of this thesis. He dedicated considerable time reading all my drafts, reports and papers. He encouraged me by providing numerous comments, advises and suggestions. I thank him for his time and confidence. Without his tireless support, you would not be reading this document.

I am also indebted to Marc Lobelle, my second advisor, who initially pushed me to start a thesis. He is the person who first made me appreciate computer networking. He co-authored [66] and [67] and provided many advises. I also thank him for his financial support.

Let me also thank Guy Leduc, for having accepted to follow me for two years now, as a member of my thesis committee. He also provided suggestions for my thesis, and manifested his interest in my work.

I am also grateful to the two other members of my jury, Francis Dupont and Timur Friedman, as well as Yves Deville, the president of my jury, for reading my thesis and for their valuable feedback.

Bruno Quoitin from UCL helped me in several ways. He provided the two-level topology with delays used in Chapter 5, and other simulation datas. More importantly, he coded the C-BGP simulator which plays a key role in the simulation presented in Chapter 5 and Chapter 7. He co-authored [69].

Steve Uhlig, from the UCL, co-authored [70]. I had many lengthy discussions with him on the usage of coordinates for detecting the low-delay paths, and on the Vivaldi algorithm. He contributed many suggestions and comments on NAROS [64]. He also provided several traffic traces.

Marcelo Bagnulo from Universidad Carlos III de Madrid has been a friendly paper co-author to work and discuss with. We met only once at the 57th IETF meeting in Vienna, where we had a fruitful discussion on IPv6 multihoming. Since

then, our contacts unfortunately occur through e-mails. He co-authored [63] and I co-authored his paper [22]. He dedicated time for reading my Internet drafts and for providing valuable comments on my work. Iljitsch van Beijnum also dedicated time in reading some of my papers. He also provided valuable feedback.

I also thank Aurélien Bonnet, who co-authored [66] and [67]. A special thanks goes to Damien Sandras, who shared my office during two years, and helped me to not despair in finishing my thesis. This document proves he was right.

Thanks to Cristel Pelsser, who is trying to get something valuable from my SVivaldi coordinates, and to Sébastien Tandel, Xavier Brouckaert and Pierre François. They all contribute, or have contributed, to the nice atmosphere that reigns in the office we share. I am also thankful to all my other colleagues and friends from the INGI department who have made these years of research a very enjoyable experience. I further would like to acknowledge the technical staff of the department for providing the computing environment, and for their technical support.

Of course, I cannot close this acknowledgement section without heartily thank my wife Vinciane de Wouters, for her love and tireless support during my whole thesis. Living by my side during these years was probably not easy. My thanks also goes to my little girl Marie, whose smiley in the frame on my desk says endlessly : "Come on daddy, finish that work !". Many thanks to my parents Ghislaine de Rosen and Baudouin de Launois, who, by allowing me to make these exciting studies, allowed this work to find its way up to here.

Finally, I express my gratitude to the *Fonds pour la Formation à la recherche dans l'Industrie et dans l'Agriculture (F.R.I.A.)*, which provided the main financial support for this thesis. Thanks for their confidence. I also acknowledge the *Université catholique de Louvain* for having hosted me for the duration of my thesis.

All remaining omissions, inaccuracies and errors are my entire responsibility. May the reader be kind enough to contact me to provide any correction to the present document.

Cédric de Launois

Université catholique de Louvain
Septembre 2005

Part I

Introduction to IPv6 Multihoming

Chapter 1

Internet Routing and Multihoming Background

This chapter presents some background on the routing in the Internet, and various other elements related to some or all multihoming techniques, such as route aggregation, NAT, or BGP route filtering. In section 1.1, we detail the IP addressing architecture, the IP address allocation process, and the two address allocation types. In section 1.2, we present how BGP handles the routing in the Internet. The main features of BGP, the route filtering, and its decision process are also described in this section. In section 1.3, the term *multihoming* and its variants are defined. Finally, the current IPv4 multihoming practices are explained in section 1.4.

1.1 IP Addressing

The Internet is today composed of thousands of interconnected networks, disseminated around the world. A network can be made of a single computer, or composed of several thousands of individual computers. Each computer connected to the Internet has an Internet Protocol (IP) address. This address both identifies the machine and describes where it can be found. To send a packet to a server, a host simply addresses it to the IP address of the server.

1.1.1 Prefixes and Masks

In the current version of the Internet Protocol (IPv4), hosts and routers are identified with 32-bit numbers, which brings a total of 2^{32}, i.e. more than 4 billion possible IPv4 addresses. A set of contiguous IPv4 addresses can be aggregated into a single block, called a *network prefix*. Each network connected to the Internet has one or more network prefixes. An IPv4 network prefix consists of a 32-bit IP address and a *mask* or *prefix length*. This mask length, e.g. 8, specifies how many bits from the beginning of the address correspond to the network part. For instance, the network prefix `4.0.0.0/8` represents a block of 2^{24} IPv4 addresses, all beginning with 4 as the first byte in the address value, i.e. IPv4 addresses between 4.0.0.0 and 4.255.255.255. Before the standardisation of Classless Interdomain Routing (CIDR) [85] in 1993, the mask length of a prefix could only be 8, 16 or 24 which represent Classes A, B or C respectively. Now, with CIDR, a prefix can be of any length.

1.1.2 The Domain Name System

IPv4 addresses are necessary to forward packets on the Internet, but are impractical to handle for human beings. To visit a Web site, one uses for instance `www.domain.com`, instead of the IP address of the server that delivers the page, e.g. 130.104.230.67. `www.domain.com` is called a *host name*. Many hosts on the Internet are assigned such *names*, which must be mapped to their actual IPv4 address. The mappings between the *host name* and corresponding IP address were historically maintained in single files (`hosts.txt`), that were distributed to all hosts using the File Transfer Protocol [95, 96, 158]. For obvious scalability reasons, those mappings are now stored in a database globally distributed across the whole Internet. This database in called the *Domain Name System* (DNS) [132, 133]. To translate a host name to an IP address, a host queries a *DNS server*. The server looks in its database for an `A` DNS Resource Record (RR) containing the IP address that corresponds to the name. The DNS server can possibly forward the DNS query to other DNS servers if it cannot answer the query itself. Once the DNS response is received, the host can use the IP address to join the destination host.

Sometimes, a host may also need to map an IP address to a host name. A special domain is defined to look up a host name or another record, given an IPv4 address. The intent of this domain is to provide a way of mapping an IPv4 address to a host name. The domain is rooted at `IN-ADDR.ARPA`. An IPv4 address is represented as a name in the `IN-ADDR.ARPA` domain by a sequence of nibbles separated by dots with the suffix "`.IN-ADDR.ARPA`". For instance, to translate the `130.104.230.67` to the corresponding host name, one must issue a DNS query for the name `67.230.104.130.IN-ADDR-ARPA`. The DNS server will reply with a `PTR` resource record containing the actual name for `130.104.230.67`.

1.1.3 Private Address Space

Three blocks of the IP address space have been reserved for private uses : `10.0.0.0/8`, `172.16.0.0.0/12`, and `192.168.0.0/16` [171]. Any network may use these IP addresses, without any coordination with any authority, or any other sites. As a consequence, the addresses within this private address space are considered unique only within the network, and cannot be used to forward packets across the global Internet.

1.1.4 IPv4 Address Allocation Process

The responsibility for the management of the IPv4 address spaces is distributed globally in accordance with the hierarchical structure illustrated by Figure 1.1 [11].

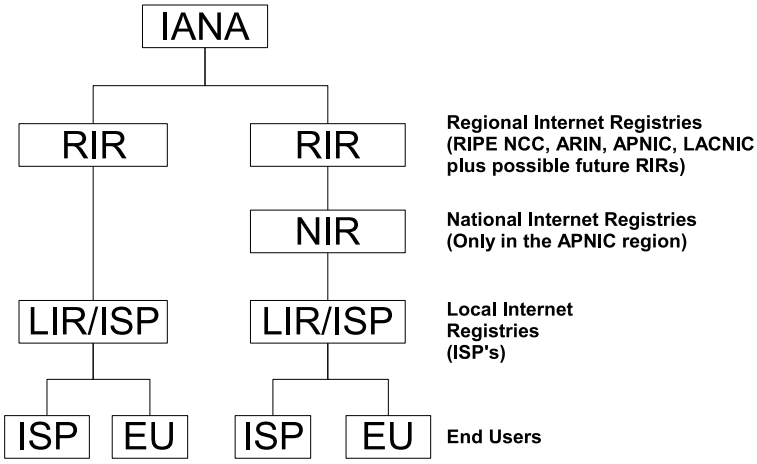


Figure 1.1. Hierarchical structure of the Internet Registries

The whole IPv4 address space was delegated by the Internet Architecture Board (IAB) to the Internet Assigned Numbers Authority (IANA). In practice, the IANA is at the head of the allocation chain. It allocates when needed large chunks of the address space to Regional Internet Registries (RIRs), such as RIPE NCC (Europe), ARIN (North America), APNIC (Asia-Pacific) or LACNIC (Latin America). In IPv4, a typical address allocation size made by IANA is one /8 at a time. The RIRs are recognised by the IANA to serve and represent large geographical regions. The primary role of RIRs is to manage and distribute public Internet address space within their respective regions.

The RIRs in turn allocate their address blocks to Local Internet Registries (LIR), or first to National Internet Registry (NIR) and then to LIRs for the Asia Pacific region. The H-Ratio [103] or the newer HD-Ratio [78] is used to determine

the utilisation thresholds that justify the allocation of additional addresses to LIRs. Typical sizes of IPv4 allocations made by RIRs to LIRs range from /21 to /16 [172], i.e. blocks of 2048 addresses to blocks of 65535 addresses.

A LIR assigns address space to its customers on request. These customers are typically enterprises and possibly other ISPs. Home users are usually not assigned addresses, they are only allowed to use the addresses assigned to their ISP.

Note that the words *assign* and *allocate* have specific meanings in Regional Internet Registry (RIR) address management policies. Address allocation to end-users should be called *assignment*, not allocation.

1.1.5 Allocation Types

A network can receive either Provider-Independent (PI) addresses, or Provider-Aggregatable (PA) addresses.

PI addresses are globally-unique addresses which are not assigned by a provider, but are provided by some other organisation, most usually an Internet Registry (RIR or LIR).

PA addresses are globally-unique addresses assigned by a provider to a customer. The addresses are considered *aggregatable* since the set of routes corresponding to the PA addresses are usually covered by an aggregate route set corresponding to the address space operated by the provider, from which the assignment was made.

1.2 Routing in the Internet

PI and PA addresses are used to identify and localise hosts connected on the Internet. When both the source and the destination hosts are connected to the same physical network, the packets can be delivered directly, through Ethernet or another local-area connection technology. Most of the time however, the hosts are located on different physical networks. In those cases, the packets must be relayed by one or more intermediate gateways, named *routers*, situated at interconnection nodes between the networks. To do the transfer, these routers have to know about the address space of both the source and the destination networks. They must also know how to forward the packet to move it closer to its ultimate destination. Depending on how distant two computers are, more than one router may handle the traffic. Each router, or *hop*, consults a map, called *routing table*, to move the traffic closer and closer to its target. A routing table contains, for each destination network prefix, the IP address of the next router to which the packet must be forwarded.

The routing tables can be configured manually for small networks. However, in larger networks, manually configuring the routing tables is simply unmanageable. Therefore, the routing tables are usually maintained automatically using

routing protocols, which allow routers to exchange and share subnet location or topology automatically. When using routing protocols, the routers use *route advertisements* to exchange routes and reachability informations about the networks they know about. These route advertisements are propagated from one router to another, until the whole network knows how to route a packet from A to B efficiently. When a failure occurs somewhere in the network, the router that detects the failure advertises new reachability informations, and a new path that does not make use of the failed link is selected, if it exists.

Routing in the Internet is separated into intradomain and interdomain routing. Intradomain routing, or routing inside a single domain, is handled by an interior gateway protocol (IGP), such as RIP [97, 127], EIGRP [47], OSPF [138], or IS-IS [211, 39]. They are used to dynamically advertise routes to route local traffic efficiently. An intradomain routing protocol typically distributes the entire network topology to all routers in the network, and selects the shortest path according to a metric chosen by the network administrator. The internal protocol chosen by a site depends on hardware support, staff experience, and internal policy. Typically, routers inside a single domain are under a single administration. They trust each other, and use according metrics to handle the traffic in a coherent way.

1.2.1 BGP Routing

Across the Internet at large, an exterior gateway protocol (EGP) is used to exchange network reachability informations between distinct domains, often competing with each other. In contrast with intradomain routing, routers participating in interdomain routing typically do not trust routers belonging to other domains. Another difference with intradomain routing is that, for scalability reasons, the interdomain routing protocol views each domain as a black box. It only knows the interconnections between domains, not their content.

The Border Gateway Protocol (BGP) is the only EGP currently used between routing domains on the Internet [169, 170, 186, 205]. The objective of BGP is to allow the transmission of IP packets to their destination along a *best* path, across different routing domains with many different routing policies, and without knowing the intradomain topologies of each particular domain. At the interdomain level, the best path is often the cheapest path that respects the business relationships between domains.

BGP is a path-vector protocol that works by sending *route advertisements*. A route advertisement made by a network can be viewed as a promise from this network to carry data to the IP address space represented in the route being advertised [116]. Each BGP route advertisement concerns a particular network prefix and includes the `next-hop`, i.e. the IP address of the router that must be used to reach this network, and a list of the intermediate transit networks in the

path, the `AS-Path`, along with other attributes. An example of route advertisements is illustrated by Figure 1.2. A BGP router stores its best and alternate routes for each network prefix in a BGP routing table, and uses this information to construct a forwarding table that controls the forwarding of each packet. An example of a BGP routing table is shown by Figure 1.5.

For instance, the *Level3* network owns the huge 4.0.0.0/8 prefix. When Level3 advertises its prefix, it promises that if someone sends data destined for any address in that address space, then it knows how to carry that data to the final destination and, more importantly, accepts to carry this data. Actually, the Level3 network is split in thousands of smaller subnets. However, other large ISPs do not need to know the internal topology of Level3. As far as the destination of a packet falls within the subnet 4.0.0.0/8, the packet can be sent to Level3, which will route the packet the rest of the way. In fact, routers in the Internet would be overwhelmed if they were required to keep track of every route to every subnet on the Internet. Instead, Level3 uses BGP to advertise a single 4.0.0.0/8 route to its BGP neighbours, called *peers*. This *route aggregation* ensures that the Internet routing tables are kept manageable. Route aggregation is further detailed in section 1.2.4.

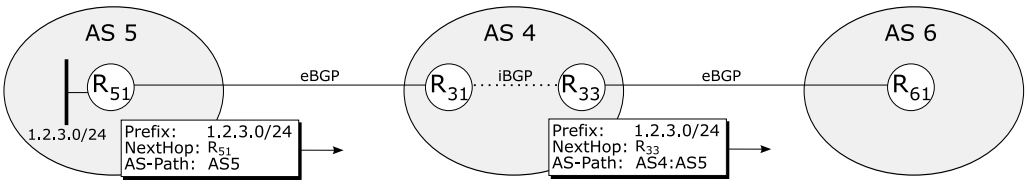


Figure 1.2. Propagation of a route advertisement from AS 5 to AS 6

In BGP, a domain is often referred to as an *Autonomous System*, or AS. The classic definition of an Autonomous System is a set of networks under a single technical administration. At the date of this writing, the Internet is composed of nearly 20,000 interconnected ASes [112]. The large majority of Autonomous Systems –82% according to [189]– do not allow external domains to use their infrastructure, except to reach them. These domains are named *stub* ASes. For instance, in Figure 1.3, AS 5 is a stub AS. The number of stub ASes is currently estimated to about 14,000 [112]. Autonomous systems that provide transit services to other ASes are called *transit* ASes. For instance, AS 1 in Figure 1.3 is typically a transit AS. Autonomous systems interconnect to each other by either using a dedicated point-to-point link, or by connecting to a public Internet Exchange (IX) point [108]. On Figure 1.3, ASes 1, 2 and 3 are interconnected using an IX, which provides a layer-2 connectivity between the three ASes.

When BGP is running inside an AS, it is referred to as iBGP. When BGP runs between autonomous systems, it is called eBGP. This is illustrated on Figure 1.4. Routers that sit on the boundary of an AS and that use eBGP to exchange

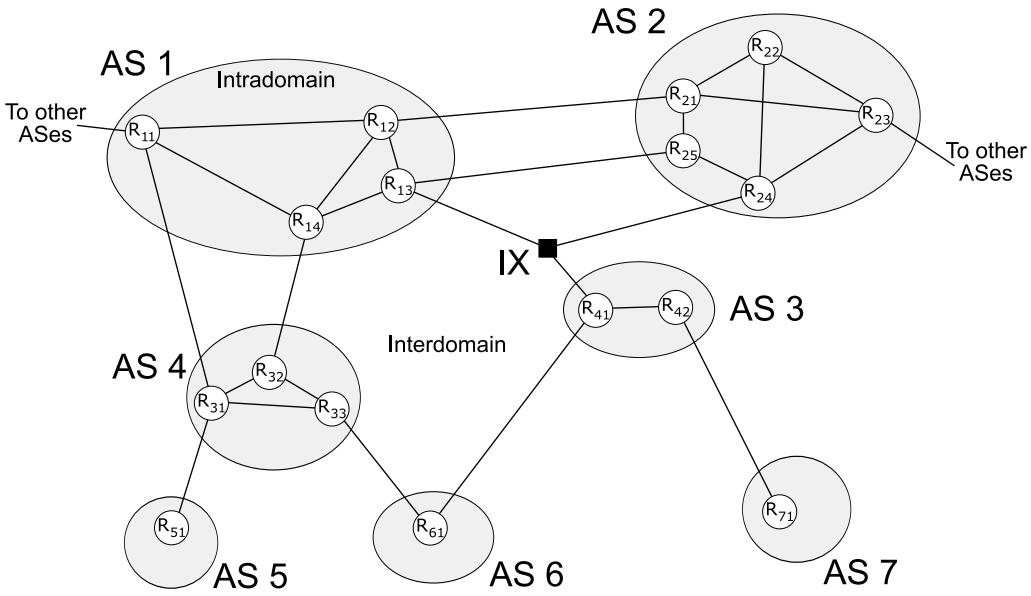


Figure 1.3. Simple Internet topology, showing physical link between ASes

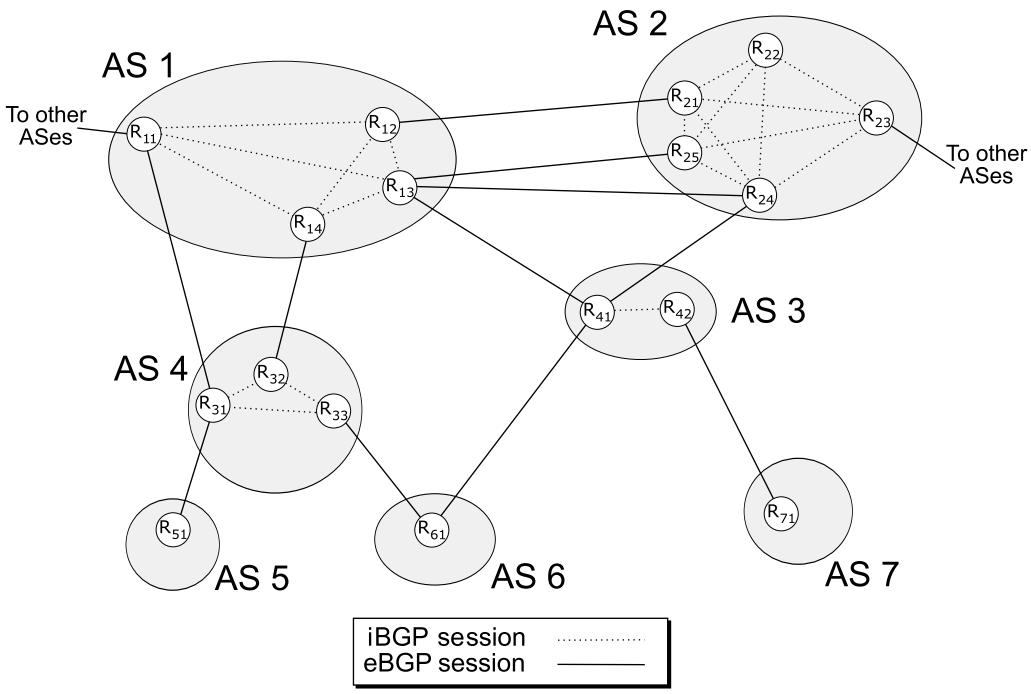


Figure 1.4. Corresponding eBGP and iBGP sessions

reachable prefixes with other BGP routers from distinct ASes are called *border routers*. For instance, R_{11}, R_{12}, R_{13} and R_{14} are border BGP routers of AS 1. The other routers inside the AS are called *transit router*. They usually do not run BGP.

A full-mesh of iBGP sessions is typically established between all BGP border routers in a single AS : each border router in an AS has an iBGP session configured with every other border router in the AS, as illustrated on Figure 1.4. As this can cause obvious scaling problems, solutions like route reflectors and confederations [28, 201] can be used to reduce the number of iBGP sessions.

A unique AS number, or ASN, is allocated to each AS participating in BGP routing. It is used to identify that AS to the world. AS numbers are currently 16-bit integers. There are public AS numbers, which may be used on the Internet and range from 1 to 64511, and private numbers from 64512 to 65535, which can be used within an organisation. For instance, Level3 is known as AS 3356. ASNs are fundamental to BGP. They are used in the BGP `AS-path` attribute, contained in each BGP route advertisement, together with the network prefix and the next-hop attributes. The `AS-Path` attribute is a sequence of AS numbers corresponding to the list of all ASes that must be traversed to reach the announced network. When a network wants to advertise itself via BGP, i.e. the network *originates* a route, the BGP router creates a new, empty AS-Path and advertises the network prefix to each of its external peers. By convention, whenever a BGP router advertises a route to an external BGP router, it prepends its own AS number to the `AS-Path`. For instance, Figure 1.2 illustrates how the route advertisement and its `AS-Path` attribute is propagated from AS 5 to AS 6.

Hence, an AS-Path reflects how the route was advertised. The `AS-Path` has two major functions in BGP. First, it is used to detect routing loops. A BGP router ignores any routing advertisement that contains its own AS number anywhere in the `AS-Path`. Second, the `AS-Path` can be used by the BGP routers as a kind of route metric. A route with a shorter `AS-Path` will usually be considered better than a route with a longer one. For each network prefix, the BGP router maintains a list of the available routes, together with their `AS-paths`. Figure 1.5 illustrates a BGP routing table. For prefix `3.0.0.0/8`, Figure 1.5 illustrates four possible routes. The one with the shortest `AS-path` is selected as the best route and inserted in the forwarding table.

1.2.2 BGP Route Filtering

Exterior gateway protocols like BGP are used between distinct domains, often competing with each other. In such environment, the goal of BGP is not to find the most efficient route, but instead to find the cheapest route that respects the business relationships between domains. For instance, an AS will rarely agree to carry the traffic of all its connected ASes towards all destinations. Therefore, BGP allows a router to be selective in the route advertisements that it sends

Network	Next Hop	AS Path	Status
3.0.0.0/8	209.123.12.4	8001 7018 80	
	204.29.239.1	6066 701 1239 80	
	144.228.241.81	1239 80	*Best*
	207.172.6.173	6079 3356 1239 80	
	⋮	⋮	
4.0.0.0/8	209.123.12.4	8001 7911 3356	
	204.29.239.1	6066 701 3356	
	207.172.6.173	6079 3356	
	4.0.4.90	3356	*Best*
	141.142.12.1	1224 38 7228 3356	
⋮	⋮	⋮	

Figure 1.5. Example of a BGP routing table

to its neighbour eBGP routers. *Policy routing* in the Internet is implemented by using to this route filtering process. Each domain is able to define its own routing policies with BGP, through the use of filters.

Input or *import filters* are used by an administrator to specify which routes can be accepted by a BGP router among all the routes received from a given peer. For example, a BGP router could only select the route advertisements with an `AS-Path` containing a set of trusted ASes. If a route advertisement is accepted by the input filter, the router places the new route in its BGP routing table, possibly after having updated some of its attributes. Input filters are essential to protect against erroneous advertisements. For instance, if a BGP peer announces the whole set of Internet routes, and if the `AS-paths` it announces is very short, all the traffic will flow through it, which will probably saturate the line. Worst, if the peer does not route correctly those prefixes, it creates a routing black hole.

Output or *export filters* are used to specify which routes can be advertised by the router to a given peer. At most one route will be advertised for each reachable destination. As said previously, a route advertisement made by AS A to AS B can be viewed as a promise from A to B to carry the traffic intended for any hosts inside the prefix advertised. Input and output filters used in combination with the BGP decision process allow ASes to implement many routing policies. In practice, two routing policies reflecting common commercial relationships are frequently used between ASes : the customer-to-provider relationship, and the peer-to-peer relationship, also named shared-cost peering [189, 86]. An example of commercial relationships is given by Figure 1.6, using the physical topology illustrated previously in Figure 1.3.

In a customer-to-provider relationship, a domain C buys Internet connectivity to another domain P, typically larger than C. P accepts to transmit packets coming from C towards any destination network. C will receive via P all packets that are destined for itself and for its own clients. For instance, in Figure 1.6, AS 5

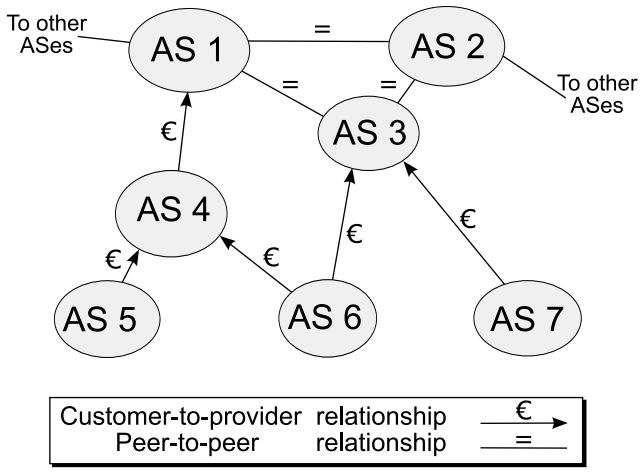


Figure 1.6. Business relationships between ASes

is a customer of AS 4, which means that AS 4 agrees to carry all the traffic coming from and going to AS 5. AS 6 has two providers, i.e. it is *multihomed*. In terms of BGP filters, the customer AS 5 will configure its BGP routers to announce to AS 4 only its own prefix, while the provider AS 4 will advertise to AS 5 all the Internet routes it knows. Note that the provider AS 4 is a customer of the even larger AS 1. In a shared-cost peering, domains X and Y are typically of comparable size, and have mutual advantage to exchange traffic between themselves and between their respective customers. In such relationships, X and Y will both announce their internal routes and the routes learned from their customers. For instance, ASes 1, 2 and 3 have a shared-cost peering between each other.

These commercial relationships can be used to assign hierarchy positions to autonomous systems [189]. Typically a customer AS would appear at a lower level in the hierarchy than its provider. According to these commercial relationships, the core of the Internet is made of about twenty large worldwide ASes, called tier-1's [189]. These ASes do not have any upstream provider and they provide connectivity to the global Internet. Their main purpose is to provide transit services. Almost every tier-1 AS peer with every other tier-1 to ensure reachability to all destinations. These tier-1's constitute the top level of the hierarchy. In Figure 1.6, ASes 1, 2 and 3 would be tier-1 providers. A second level in the hierarchy is made of regional or national ISP, called tier-2's. These ASes buy global Internet connectivity from one or several tier-1 ISPs, and often establish shared-cost peering relationships with other tier-2 ASes. These smaller transit ASes are sometimes further separated into tier-2, tier-3 and tier-4 ASes [189]. For instance, AS 4 in Figure 1.6 could be classified as a tier-2 provider. At the lowest level in the hierarchy, the customers are the stub networks that are origins and sinks of traffic and which do not carry any transit traffic. These are small ISP and corporate networks, for instance ASes 5, 6 and 7 in Figure 1.6.

1.2.3 BGP Decision Process

As explained previously, the `AS-Path` attribute of a route can be used to determine the best route towards a destination, among all the routes received from the BGP peers. However, the route with the *shortest* `AS-path` is not always the best one. Since BGP sees each AS as a black box, it would prefer a two-hop AS-path with 30 individual router hops over a four-hop AS-path with only four router hops. Thus, the network or system administrator often make use of *policy routing* in order to override the default behaviour of BGP and to leverage *longer* but better performing paths. To understand policy routing, one must understand the steps a BGP router uses for determining the *best* route for a given destination network, among the set of routes received from the peers. Those steps constitute what is called the *BGP decision process* [170, 33, 203].

The first BGP attribute checked in the BGP selection process is called the `local-preference`. This attribute is the preference a site administratively assigns to a given route. The route with the highest `local-pref` value is considered the preferred one. This attribute is local only to that AS. Across external BGP sessions, all `local-pref` attributes are reset to the default value.

When two different routes to the same destination prefix have the same `local-pref` value, the BGP decision process continues by considering the length of the `AS-path`. The second rule of the BGP decision process is to prefer, for a given prefix, a route with a short `AS-Path` over a route with a longer one. The length of the `AS-path` is seen as a measure of the quality of the route, and it is usually expected that the route with the shortest `AS-Path` is the best one. Some ASes prepend the `AS-path` in their BGP route advertisements with their local AS number multiple times. Those ASes hence artificially increase their AS paths, in the hope that BGP routers in distant ASes will not select this particular path. However, this method to influence the inbound path selection is easily overridden by adjusting the local-preference in the distant AS, in order to select a route with a longer `AS-Path`.

The third rule of the BGP decision process is to use the `Multiple-Exit-Discriminator` or `MED` to select among the remaining equal quality routes. This attribute allows to compare routes received from different routers of the same AS. The rule is to prefer the route with the lowest `med` value.

The fourth rule of the BGP decision process is to prefer routes learned over an eBGP session to routes learned over an iBGP session.

If the previous rules still do not suffice to select the best route, then the next rule is to prefer the route with the lowest path cost, which is provided by the interior gateway protocol, e.g. OSPF. This rules implements what is known as *hot-potato routing*. This means that an AS tries to get rid of the IP packet as soon as possible, in order to minimise the resource consumption implied by the forwarding of the IP packet.

If a given router learns a prefix from multiple ISPs and each route has the same `local-pref`, the same `AS-Path` length, and all other BGP path attributes are identical, then the BGP decision process relies on final tie-breaking rules which depend on the implementation. [170] recommends to select as the best route the one coming from the BGP router with the lowest *Router-ID* (now the BGP Identifier [46]), i.e. usually its IPv4 address. Additionally, if a route originates from the same BGP router but is received several times through multiple distinct paths, then the best route is the one coming from the lowest neighbour address, i.e. the address of the remote peer used in the TCP connection with the local router. While this may seem an odd candidate for a tie breaker, all BGP routers in the same AS will ultimately make the same routing decision, and thus avoid routing loops. Some implementations have chosen to keep the oldest route in order to avoid some oscillation problems [45].

1.2.4 Route Aggregation

Route aggregation refers to the practice of using a single prefix to announce the routes to multiple prefixes. An AS performs route aggregation by using the minimum number of prefixes to summarise all of its IP addresses. Two prefixes can be aggregated if the union of IP blocks represented by two prefixes can be summarised by a single prefix [37]. For example, prefixes 1.2.2.0/24 and 1.2.3.0/24 can be aggregated as prefix 1.2.2.0/23.

Autonomous systems typically aggregate their routing prefixes in two situations. First, an AS aggregates the routes it originates itself, if possible. For instance, if AS 123 originates prefixes 123.0.2.0/24 and 123.0.3.0/24, then it can announce prefix 123.0.2.0/23 only, instead of announcing both prefixes, see Figure 1.7. Second, the prefix announced by an AS can be aggregated by the provider of the AS, if the prefix is contained in the prefix of the provider. For example, suppose AS 20 received from its Internet Registry the 20.0.0.0/8 address block, and allocated the sub-prefix 20.123.0.0/16 to its customer AS 123. Instead of announcing both 20.0.0.0/8 and 20.123.0.0/16 prefixes, AS 20 can make use of an `as-set` to announce the 20.0.0.0/8 prefix only, as illustrated by Figure 1.8. This `as-set` argument summarises the `AS-Path` attributes of the two individual routes.

1.3 Multihoming

Multihoming means the practice of connecting to multiple distinct network service providers. For more and more companies, Internet connectivity takes a strategic importance. As a consequence, multihoming is becoming highly popular. Many ISPs and corporate networks wish to be connected via at least two providers to the Internet for economical and technical reasons. Technical reasons include the need for better redundancy, the need for load sharing and better performance.

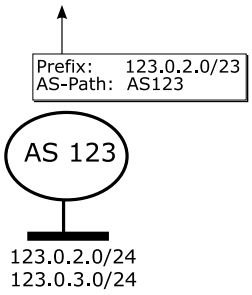


Figure 1.7. Aggregation of two routes originated within a single AS

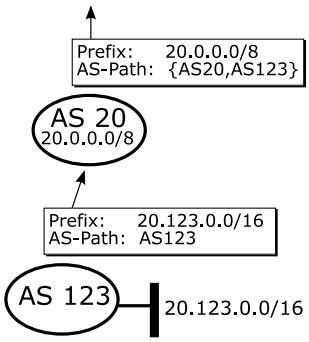


Figure 1.8. Aggregation of a customer prefix with the provider prefix

Economical reasons include policies beyond technical scope, e.g. cost or commercial reasons. These reasons are called here *motivations*. They will be further detailed in Chapter 3. Nowadays, at least 60% of stub domains are multihomed to two or more providers [3, 112], and this number is growing.

The term *multi-connecting* or *multi-attaching* is sometimes used when connecting several times to a single transit provider. In this thesis, multihoming refers only to the practice of connecting to different providers. We distinguish four types of multihoming : *Node Multihoming*, *Site Multihoming*, *Multihoming of Small ISPs*, and *Multihoming of Transit ISPs*.

1.3.1 Node Multihoming

Node multihoming means the practice of connecting a single node to multiple providers, such as a node that uses multiple interfaces with different IP addresses to connect to different network service providers. For instance, a laptop connected to a local area network through a Wireless Access, and connected to the Internet via an Ethernet link. Node Multihoming can be viewed as a degenerate case of Site Multihoming, explained below.

1.3.2 Site Multihoming

Site Multihoming refers to a stub network that connects to at least two different network service providers. These stub networks do not provide any transit service to other networks. Site multihoming potentially concerns a wide range of sites, from residential Internet users with two or three hosts, to very large enterprise networks composed of thousands of hosts.

1.3.3 Multihoming of Small ISPs

Multihoming of Small ISPs refers to ISPs that have a local presence, that provide transit services to stub networks, and that connect to multiple different upstream providers. Those small ISPs may be large enough to obtain their own provider-independent address space. They have their own AS numbers, and use BGP to advertise reachability informations to other domains.

1.3.4 Multihoming of Transit ISPs

Multihoming of Transit ISPs refers to medium and large ISPs, that provide transit services to other ISPs and/or stub networks, and that connect to multiple different upstream providers. They typically have more than a local presence, often a worldwide one. They always use their own provider-independent address space, and also use BGP to advertise reachability informations to other domains. Is is hoped that the number of those transit ISPs remains low, even in an IPv6 Internet. Currently, there are less than 2,000 medium and large transit ASes [112].

In this thesis, only *site multihoming* and the *multihoming of small ISPs* are considered. *Node multihoming* is a simpler and degenerate case of site multihoming. The *multihoming of Transit ISP* has a widely accepted solution, which is to announce a set of routes with BGP to the upstream providers. This provides the rest of the Internet with multiple paths back to the multihomed sites. This is the way how multihoming is achieved in the current IPv4 Internet. This solution is considered acceptable in IPv6 for the multihoming of medium and large transit ISPs, considering that the number of those ISPs will grow much slower than the number of stub networks and small ISPs. However, multihoming with BGP is not considered as a viable solution for site multihoming and for the multihoming of small ISPs, as it will be explained in Chapter 3.

1.4 Current IPv4 Multihoming Practices

This section describes how sites currently connect to multiple providers in the IPv4 Internet.

1.4.1 Site Multihoming with BGP

In the current IPv4 Internet, the traditional approach for multihoming is to announce, using BGP, a single prefix to each provider, as illustrated in Figure 1.9 or Figure 1.10.

When a failure occurs between the local site and a remote site, normal operation of the BGP routing protocol will ensure that the routing advertisement corresponding to this particular path will be withdrawn from the routing system.

The remote site who had selected this path as the best available will select another candidate path as the best path. Upon restoration of the path, the path is re-advertised in the interdomain routing system. The remote site will undertake a further selection of the best path based on this re-advertised reachability information.

Multihoming with BGP is essentially used by ISPs and very large enterprise networks. It obviously requires configuring and running the BGP protocol, in addition to obtaining an official AS number. When multihoming with BGP, the site can use provider-independent (PI) addresses or provider-aggregatable (PA) addresses.

Multihoming with PI addresses

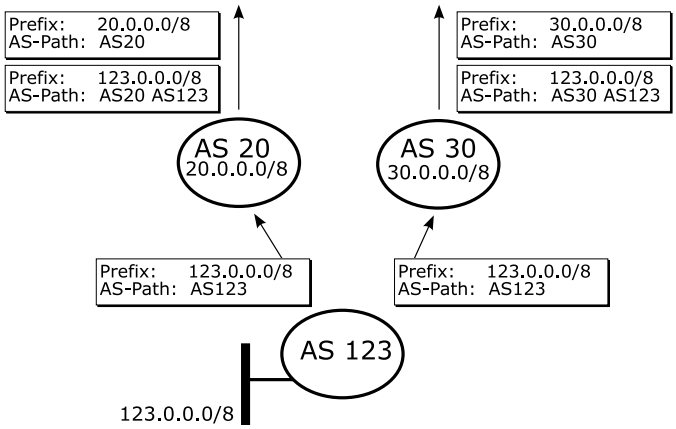


Figure 1.9. IPv4 Multihoming using provider-independent addresses

In Figure 1.9, AS 123 was large enough to obtain and use provider-independent (PI) addresses. It announces its PI prefix to both its providers AS 20 and AS 30. Neither AS 20 nor AS 30 is able to aggregate the announcement made by AS 123. Therefore, both AS 20 and AS 30 announce to the global Internet the prefix of AS 123, in addition to their own prefix. These ISPs will propagate the route received to the global Internet. This provides the rest of the Internet with multiple paths back to the multihomed sites. It is clear that the use of PI addresses introduces an additional routing entry in the global routing system. Widespread multihoming in this manner presents scaling concerns [110, 15].

The use of provider-independent addresses has long been the preferred way to multihome in IPv4 [111]. A reason for this preference is that a site does not have to renumber if it changes of provider. Until the mid 1990s, it was relatively easy for a site to obtain a fairly large provider-independent address space from an RIR. Little justification was needed to obtain a /24 PI assignment. Due to the

rapid depletion of the IPv4 address space, the RIRs no longer assign blocks as large as a /24 to small sites [172, 12, 10]. As a consequence, many sites are not able to obtain a PI assignment from their RIR.

Multihoming with PA addresses

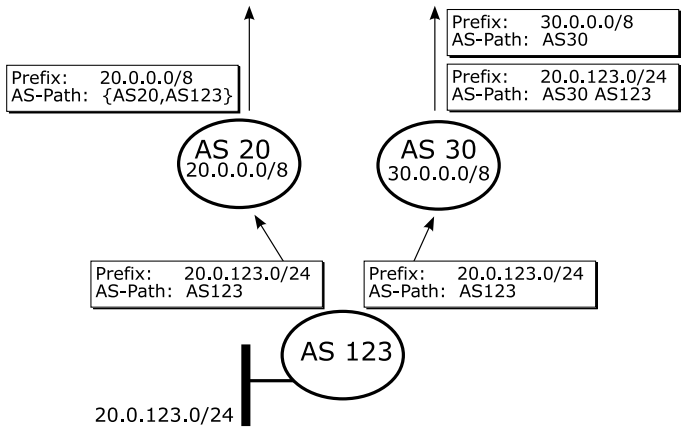


Figure 1.10. IPv4 Multihoming using provider-aggregatable addresses

In Figure 1.10, AS 123 uses instead a single provider-aggregatable address space. This address space is assigned by the primary transit provider AS 20. AS 123 announces prefix 20.0.123.0/24 to both its providers AS 20 and AS 30. Here, AS 20 is able to aggregate this prefix with its own 20.0.0.0/8 prefix. AS 20 only announces the aggregate to the Internet. However, AS 30 cannot aggregate this prefix with its own prefix. Thus AS 30 still has to announce the prefix of AS 123 in addition to its own prefix. An awkward side effect is that almost all packets will now enter AS 123 via AS 30, due to the BGP decision process which favours more specific prefix advertisements.

Sites use PA addresses when their addressing requirements are not sufficient to meet the requirements for a PI address block by RIRs. The drawback is that the site is usually required to renumber if it decides to change of primary transit provider.

Even when PA addresses are used, multihoming with BGP in this manner still introduces an additional routing entry in the global BGP routing tables, as AS 30 cannot aggregate the prefix announced by AS 123. In order to limit the size of the Internet routing tables, a common operational practice is to ignore routes with prefixes longer than 24 or even 22 bits [29]. Operators can also filter prefixes that have larger mask lengths than the address allocation guidelines published by the Regional Internet Registries [172, 12, 10]. In the example of Figure 1.10, AS 30 might refuse to propagate the route advertisement for prefix 20.0.123.0/24. A

consequence is that some routes are not distributed to the global Internet. Since the minimum /24 prefix required may be larger than the allocation available to small sites, these small sites are unable to get the multihoming benefits when using BGP.

1.4.2 Site Multihoming with NAT

Small sites that cannot multihome with BGP can still get multihomed with the help of Network Address Translation (NAT).

Network Address Translation

NAT is a method by which IP addresses are mapped from one realm to another, in an attempt to provide transparent routing to hosts [184, 80].

There are two variations to NAT, namely *Basic NAT* and *Network Address Port Translation* (NAPT). Basic NAT is a mechanism by which IP addresses are mapped from one address to another, in one-to-one fashion. NAPT extends the notion of translation one step further by also translating transport identifier (e.g., TCP and UDP port numbers, ICMP query identifiers). This allows the transport identifiers of a number of private hosts to be multiplexed into the transport identifiers of a single external address. For example, a NAPT device allows several hosts using private IP addresses to share a single external public IP address, a capability highly appreciated when the exhaustion of the IPv4 address space seems to be inevitable.

NAT devices do not examine or modify the transport payload. As a consequence, NAT devices presents limitations. For instance when an application payload includes an IP address, or when end-to-end security is needed. Application layer security techniques that do not make use of, or depend on, IP addresses will work correctly in the presence of NAT (e.g. SSL or SSH). In contrast, transport layer techniques such as IPsec transport mode [120], FTP [158], RTSP [179], or SIP [174] do not. For these applications that do not lend themselves easily to translation by NAT devices, *Application Level Gateways* (ALGs) can be used. ALGs are application specific translation agents that interact with NAT to set up state, use NAT state information, modify application specific payload and perform whatever else is necessary to get the application running through a NAT device.

Application to Multihoming

When multihoming with NAT, the site receives PA prefixes assigned from each transit providers. However, the hosts within the multihomed site do not use directly those addresses. They use instead private IP addresses [171]. Figure 1.11 illustrates such a scenario. In this figure, the site is using the `192.168.0.0/24`

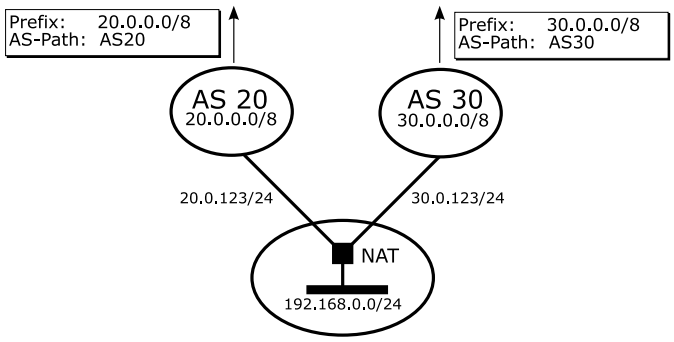


Figure 1.11. IPv4 Multihoming using Network Address Translation

private address space, and is connected to two providers : AS 20 and AS 30. The site received the `20.0.123.0/24` and `30.0.123.0/24` prefixes from AS 20 and AS 30 respectively. An IP packet leaving the network is processed by the NAT gateway, where the private IP address is translated into a public address belonging to the provider selected to carry the packet. For instance, the IP address of a packet sent to AS 30 will be translated from 192.168.0.1 to 30.0.123.1. An answer to this packet will be addressed to 30.0.123.1, received by the NAT gateway, and translated back to the original 192.168.0.1 private address.

The site is not really multihomed, but rather single-homed to several network service providers. No accommodation is required from the transit providers, beyond that which they would do for a non-multihomed customer. This approach allows a wide range of customer to multihome, from residential Internet users up to large enterprises. It does not require PI addresses nor an official AS number, and preserves the size of the Internet routing tables since no additional route specific to the multihomed site is announced.

However, this approach does not meet all motivations for multihoming, most notably transport-layer survivability. When a failure occurs that affect some connection, it is not possible to intercept and continue the connections since the outgoing public IP addresses cannot be modified without breaking the TCP session. Additionally, multihoming with NAT imposes the limitations introduced by the NAT, i.e. complex application protocols require explicit support for remapping the addresses and ports.

1.4.3 Site Multihoming with RSIP

We present here another solution for IPv4 multihoming, that does not make use of the NAT, and that still preserves the sizes of the Internet routing tables

Realm Specific IP

The Realm Specific IP (RSIP) [35, 34, 135] is a new protocol, designed as an alternative to NAT. It has the additional requirement to preserve end-to-end packet integrity, a feature not provided by NAT. RSIP is based on the concept of granting a host, called a *RSIP host*, from a network A a presence in another network B, by allowing it to use resources (e.g. addresses and other routing parameters) from the network B. The gateway, called *RSIP gateway*, between networks A and B owns a pool of such resources, that it can allocate to RSIP hosts. For instance, Figure 1.12 illustrates a private network that connects to two provider networks. The RSIP gateway on the border of the private network received from AS 20 and AS 30 the 20.0.123.0/24 and 30.0.123.0/24 address blocks. In the figure, Hosts A and B use private IP addresses. They use their private address to request a second, public, IP address from the gateway through a RSIP transaction. For instance, in Figure 1.12, Host A received the 20.0.123.1 IP address.

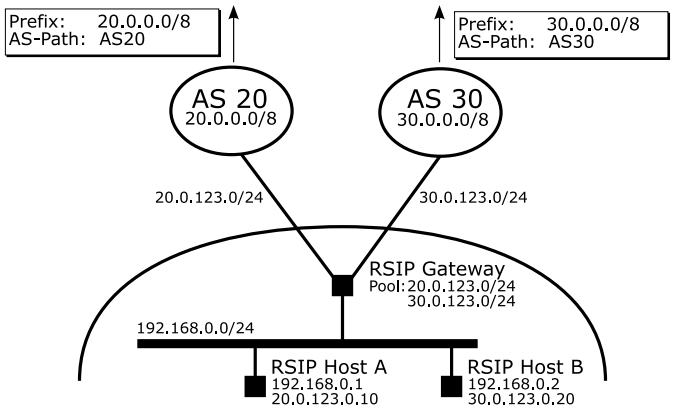


Figure 1.12. IPv4 Multihoming using the Realm Specific IP

RSIP has been defined in two basic flavours : RSA-IP and RSAP-IP. When using RSAP-IP, the RSIP gateway maintains a pool of IP addresses as well as pools of port numbers per address. The gateway allocates each IP address with one or more port numbers. A host may only use the tuples address/port that have been assigned to it. When RSAP-IP is used, a tunnel must be set up between the RSIP host and the RSIP gateway, since a same IP address (but with different TCP or UDP ports) could be used by two or more hosts on the same network. This is illustrated by Figure 1.13, where hosts A and B both use the same 20.0.123.10 address. Host A can use this address only in combination with TCP and UDP ports inside the 15000-17000 range. Host B can use the same address only with ports inside the 25000-26000 range If, for instance, Host A initiates a TCP flow using source port 15000, then the packets will flow through

an IP tunnel between Host A and the gateway. The reply packets sent by the remote destination host will be addressed to 20.0.123.10 port 15000. They will first reach the RSIP gateway, that will next tunnel these packets back to Host A.

Hosts A and B use their private IP address to communicate with each other. If a host needs to listen on a particular address or port, it must query the RSIP gateway. If the request is granted, the RSIP gateway will forward all packets destined for the address and port in question to the host.

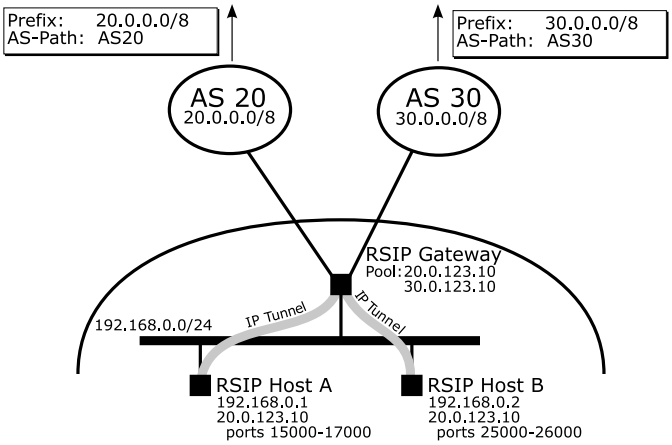


Figure 1.13. Establishment of tunnels when RSAP-IP is used

When using RSA-IP, a RSIP gateway only maintains a pool of IP addresses to be leased by RSIP hosts. Upon request, the gateway allocates an address to the host with an expiration time. The host may use this leased address with any TCP or UDP port. When the lease time is about to expire, the host asks for a lease extension. If granted, the host may continue to use the address, otherwise it must release it.

Application to Multihoming

The RSIP protocol can be used to provide some form of multihoming, using the configuration illustrated in Figure 1.12. In this example, hosts use RSA-IP to receive a temporary public IPv4 address. They use it to communicate with remote Internet hosts. Suppose the RSIP gateway detects a failure of the link with AS 20. The RSIP gateway can deny a client request for extending the lease time of an address within the 20.0.123.0/24 address block. A consequence is that a RSIP client using such address will request a new public IP address, which can be given from the 30.0.123.0/24 address block.

Compared to NAT, the advantage is that no Application Level Gateways (ALGs) are needed, since the payloads of the packets are not modified. Policies

can be implemented within the RSIP gateway. The extra cost of the RSIP solution is the necessity to add a software agent on each host in the multihomed network. To our knowledge, the RSIP implementation that we provide at [61] is the only complete implementation of the protocol, but limited to the Linux operating system.

Address sharing is still possible, through the use of the RSAP-IP flavour of the RSIP protocol. This is illustrated in Figure 1.13, where all hosts share a single address, but use it with different port numbers. Hence, this solution is applicable even if each provider only assigns one IP address to the site. Extensions to RSIP have been defined to improve the scalability of the architecture, and to offer extended possibilities to bind a RSIP host to a permanent host name. We refer the reader to our papers [66, 67, 68] for further details.

Chapter 2

Background on IPv6

This chapter briefly introduces IPv6. It focuses only on the differences with IPv4, and on the few IPv6 notions that are necessary to understand the rest of this work.

2.1 Addresses

When it was designed in the 1970's, a set of more than 4 billion IPv4 addresses seemed largely sufficient. However, the explosive growth of the Internet, inefficient addressing, and sub-optimal address allocation have caused a rapid exhaustion of the IPv4 address space. The complete exhaustion is expected between 2014 and 2018 [113]. It may even happen before, as suggested by Figure 2.1 [113]. This figure shows the evolution against time of the percentage of IPv4 addresses left for future allocations. In May 2005, less than 30% of the total IPv4 address space remains unallocated.

To support the sustained growth of the Internet, a new version of IP, IPv6, was designed. In IPv6, hosts and routers are identified with 128-bit addresses, instead of 32-bit ones. The address format is made of three parts [100, 101]. The first n bits represents the global routing prefix. n corresponds to the mask length. The m following bits are used to identify a sub-network. It is recommended that m is 16 bits, which leaves 48 bits for the global routing prefix n. The last 64 bits of the IPv6 address are reserved for the interface identifier of the end devices,

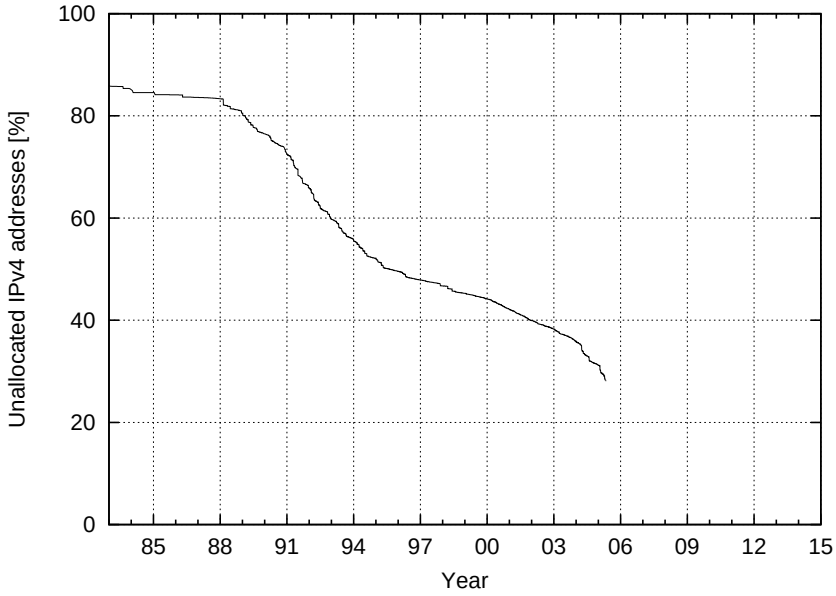


Figure 2.1. Evolution of the percentage of unallocated IPv4 address space

e.g. a 64-bit identifier derived from an Ethernet's MAC address. The resulting address format is illustrated on Figure 2.2, and further detailed in [100, 101]. This IPv6 address space is large enough to contain billions of billions of networks, each network containing billions times the whole today's Internet.

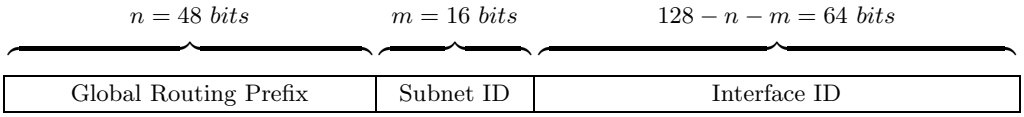


Figure 2.2. Typical IPv6 Address Format

The textual representation of an IPv6 address is `x:x:x:x:x:x:x:x`, where the "`x`"s are the hexadecimal values of the eight 16-bit pieces of the address [99]. The use of "`::`" indicates multiple groups of 16-bits of zeros. The "`::`" can only appear once in an address, and it can also be used to compress the leading and/or trailing zeros in an address. The representation of an IPv6 address prefix is similar to the way IPv4 addresses prefixes are written in CIDR notation, i.e. `ipv6-address/prefix-length`. The `prefix-length` is a decimal value that specifies how many of the leftmost contiguous bits of the address comprise the prefix [99].

2.2 Address Types

The IPv6 architecture defines several types of addresses [100]. These types are indicated on Table 2.1.

Address type	Binary prefix	IPv6 notation
Unspecified	`00...0 (128 bits)`	`::/128`
Loopback	`00...1 (128 bits)`	`::1/128`
Multicast	`11111111`	`FF00::/8`
Link-Local unicast	`1111111010`	`FE80::/10`
Site-Local unicast	`1111111011`	`FEC0::/10`
Global unicast	(everything else)	

Table 2.1. IPv6 Address Types

The unspecified address `0:0:0:0:0:0:0:0` must never be assigned to any node, as it indicates the absence of an address. The loopback address `0:0:0:0:0:0:0:1` may be used by a node to send an IPv6 packet to itself. It plays the same role as the IPv4 127.0.0.1 address. An IPv6 anycast address is an address that is assigned to more than one interface, typically belonging to different nodes. Anycast addresses are taken from the unicast address spaces (of any scope) and are not syntactically distinguishable from unicast addresses. A packet sent to an anycast address is routed to the *nearest* interface having that address. An IPv6 multicast address is an identifier for a group of interfaces, typically belonging to different nodes.

Every IPv6 address other than the unspecified address has a specific *scope*. A scope is a topological span within which the address may be used as a unique identifier for an interface or set of interfaces [71]. A Link-Local unicast address is for use on a single link, i.e. its scope is the link. A Link-Local address starts with the well-known prefix `FE80::/10`. Such address is automatically assigned to an interface by concatenating the `FE80::/10` prefix with the interface identifier (64 bits). A Site-Local[1] unicast address is for use in a single site, i.e. its scope is the site. Finally, a Global unicast address is for use in the global Internet. An example of a global-scope unicast IPv6 address is `2001:DB8:3080:5::ABCD/48`, where the global routing prefix is `2001:DB8:3080::/48`, the subnet ID is `5`, and the interface ID is `::ABCD`. The mask length is 48 bits, out of 128 bits.

[1] Site-Local addresses are considered deprecated since September 2004 [105], essentially because of implementation difficulties and the fuzzy nature of the *site* concept.

2.3 Stateless Address Autoconfiguration

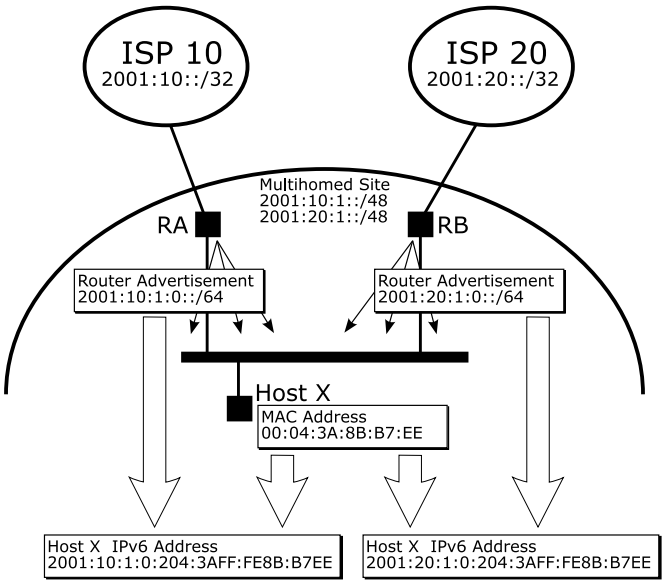


Figure 2.3. Stateless address autoconfiguration

In IPv6, a stateless mechanism has been defined to allow a host to generate its own addresses, using a combination of locally available information, and information advertised by routers [198].

In typical IPv6 sites, routers periodically advertise the global prefixes that identify the subnet associated with a link. Hosts generate a complete IPv6 address for each of their interfaces by combining the advertised prefix with each interface identifier. The interface identifier for an Ethernet interface is itself based on the EUI-64 identifier [14] derived from the built-in 48-bit IEEE 802 address of the interface. This process is described in [53], and illustrated by Figure 2.3.

In this figure, RA (resp. RB) is the site exit router connected with ISP 10 (resp. ISP 20). RA advertises the subnet prefix `2001:10:1:0::/64`, and RB advertises prefix `2001:20:1:0::/64`. Suppose Host X has a single interface. Host X will generate two global-scope IPv6 addresses for this interface, by concatenating the two advertised prefixes with its interface identifier. This interface identifier is itself generated by adding the fixed hexadecimal value 0xFFFE between the third and fourth octets of the IEEE 802 address, and by setting to 1 the "Universal/Local" (U/L) bit, i.e. the lowest order bit of the first octet of the identifier. For instance, if the IEEE 802 address of Host X is `00:04:3A:8B:B7:EE`, the generated interface identifier is `0204:3AFF:FE8B:B7EE`, and the two IPv6 addresses generated for this interface are `2001:10:1:0:204:3AFF:FE8B:B7EE` and `2001:20:1:0:204:3AFF:FE8B:B7EE`. This process is illustrated by Figure 2.4.

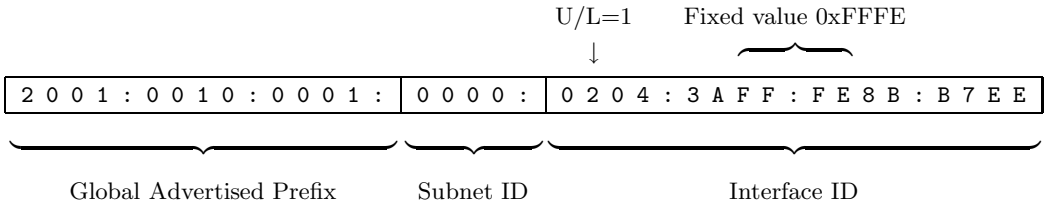


Figure 2.4. Generation of an IPv6 Address

In the absence of routers, a host can only generate link-local addresses. Nodes can use these link-local addresses for communicating with other nodes attached to the same link.

IPv6 also defines a stateful address autoconfiguration model [76, 75], where hosts obtain addresses, other configuration informations, or both from a server. Stateless and stateful autoconfiguration complement each other, as a host can use stateless autoconfiguration to configure its own addresses, and use stateful autoconfiguration to obtain other informations, such as the addresses of DNS recursive name servers.

2.4 Address Selection

As presented in section 2.2, the IPv6 architecture defines several types of IPv6 addresses, with different scopes (global, site, or link). A consequence is that an IPv6 host typically ends up with several IPv6 addresses per interface. Additionally, If the host is connected to different networks, it may also have several global-scope IPv6 addresses. This is the case for Host X in Figure 2.3. The end result is that IPv6 implementations will very often be faced with multiple possible source and destination addresses when initiating a communication. A document [73] defines a default algorithm for selecting source and destination addresses, so that developers and administrators can reason about and predict the behaviour of their systems. The algorithm is specified as a set of rules that define a partial ordering on the set of addresses that are available for use.

2.4.1 Default Policy Table

The specification optionally allows the administrative configuration of policy that can override the default behaviour of the algorithm. The "policy override" takes the form of a configurable table that specifies precedence values, and preferred source prefixes for destination prefixes. This table is called the *policy table*, and is a longest-matching-prefix lookup table, much like a routing table. It tells which destination addresses must be preferred over others, and which source addresses must be used with a given destination address. Given an address A, a lookup in the policy table produces a precedence value $Precedence(A)$ and a

label $Label(A)$. The precedence value is used for sorting destination addresses. If $Precedence(A) > Precedence(B)$, then the algorithm will prefer destination address A over destination address B. The label value allows for policies that prefer a particular source address prefix for use with a destination address prefix. The algorithm prefers to use a given source address S with a destination address D if $Label(S) = Label(D)$. The recommended default policy table is illustrated on Table 2.2 [73].

Prefix	Precedence	Label
::1/128	50	0
::/0	40	1
2002::/16	30	2
::/96	20	3
::ffff:0:0/96	10	4

Table 2.2. The default policy table

The two following subsections detail how the address selection algorithm works [73]. The algorithm is given a list of possible destination addresses (e.g. from the DNS), and has a list of possible source addresses (all addresses assigned to the interfaces).

2.4.2 Source Address Selection

First, for each destination address, the algorithm selects the best possible source address. Eight rules exist to make this selection. We can highlight three interesting rules. The first is to prefer an appropriate scope, e.g. use a global-scope source address with a global-scope destination address, and use a link-local source address with a link-local destination address. To make a choice among the source addresses that have an appropriate scope, the algorithm uses a second rule, which is to prefer matching labels. If the label of one source address matches the label of the destination address, then this source address is preferred over another that does not match. The third and last interesting rule is to prefer the source address with the longest matching prefix for the destination address. We refer the reader to [73] for the description of the other rules that we do not use in this work.

2.4.3 Destination Address Selection

After the source address selection, the algorithm knows which is the best source address for each destination address given by the DNS. The algorithm must now determine the best (source, destination) address pair. Like for the source address selection, there are many rules on how to perform this selection. We highlight six interesting rules, over a total of ten. The first rule is to prefer a couple of

(source, destination) addresses that have the same scope, over a couple where the source and destination addresses have different scopes. The second is to prefer a couple of (source, destination) addresses that have matching labels over a couple of addresses that do not. Note that this rule is different from the matching label rule of the source address selection. The third rule is to prefer the destination addresses with the highest precedence, given by the policy table. The fourth rule is to prefer destination addresses with the smaller scopes. This means that a couple of link-local source and destination addresses will be preferred over a couple of global-scope source and destination addresses. The fifth rule is to prefer a couple of source and destination addresses that have the longest matching prefix. Finally, the last rule is to leave the order unchanged and select the first destination address on the list. Again, we refer the reader to [73] for the description of the four other rules. We do not use them in this work.

One effect of this default policy table is to prefer using native source addresses with native destination addresses, 6to4 [44] source addresses with 6to4 destination addresses, and v4-compatible [99, 88] source addresses with v4-compatible destination addresses. Another effect of the default policy table is to prefer communication using IPv6 addresses to communication using IPv4 addresses, if matching source addresses are available.

2.4.4 Example

Assume that the destination host has two IPv6 addresses `2001:bbbb:bbbb::b` and `2007:0:bbbb::b`. The initiating host has three IPv6 addresses `2001:aaaa:aaaa::a`, `2007:0:aaaa::a`, and `fe80::a`. The algorithm will first select the best source address for each destination address. With the default policy table, this will result in the addresses shown in Table 2.3.

Destination Address	Best Source Address	Rule Used
`2001:bbbb:bbbb::b`	`2001:aaaa:aaaa::a`	Longest Matching Prefix
`2007:0:bbbb::b`	`2007:0:aaaa::a`	Longest Matching Prefix

Table 2.3. Example of source address selection

Next, the initiating host selects the best destination address. In this example, all couples of source and destination addresses have the same scope, the same matching labels, and the same precedences. The couple (`2007:0:bbbb::b`, `2007:0:aaaa::a`) has a longer matching prefix (`2007:0::/32`) than the matching prefix (`2001::/16`) of the couple (`2001:bbbb:bbbb::b`, `2001:aaaa:aaaa::a`). Therefore, the first couple is preferred. As result of the source and destination address selection algorithm, Host X will join Host Y with the destination address `2007:0:bbbb::b` and the source address `2007:0:aaaa::a`.

2.5 DNS Extensions

Like in IPv4, hosts usually have an associated name stored in the Domain Name System. The name is stored using a new DNS resource record, with mnemonic `AAAA` [197]. The `IP6.ARPA` Domain is defined for providing a way of mapping an IPv6 address to a host name. An IPv6 address is represented as a name in the IP6.ARPA domain by a sequence of nibbles separated by dots with the suffix "`.IP6.ARPA`". For instance, the reverse lookup domain name corresponding to the address `2001:0:1:2:3:4:567:89ab` would be `b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.1.0.0.2.IP6.ARPA`.

2.6 Address Allocation Process

The responsibility for the management of the IPv6 address spaces is identical to the one of the IPv4 address space. It was illustrated by Figure 1.1 in Chapter 1. In IPv6, the block 2000::/3 was designated to be the global unicast address space that the Internet Assigned Numbers Authority (IANA) may allocate to the Regional Internet Registries [101]. In IPv6, the allocation unit from IANA to RIRs is usually one /23 at a time. As in IPv4, the RIRs in turn allocate their address blocks to Local Internet Registries (LIR). In IPv6, LIRs that meet the initial allocation criteria receive a minimum allocation of /32 out of 128 bits, i.e. enough for more than 4 billion of networks, each containing 2^{64} addresses. Those LIRs next assign address spaces to their customers on request. In IPv6, the customers will generally be given /48 assignments [114, 11, 156].

Chapter 3

IPv6 Multihoming Problem Statement

This chapter focuses on the reasons why a new multihoming mechanism is needed in an IPv6 Internet. It presents the challenges faced by the IPv6 multihoming architectures to respect motivations and constraints related to multihoming.

3.1 The BGP Routing Tables Growth

The size of a BGP routing table is measured in the number of prefixes it contains. A BGP routing table entry contains reachability information for a single prefix. Over the last ten years, the size of these routing tables has risen from 20,000 IPv4 prefixes in 1995 to more than 200,000 in 2005 [112, 109, 131]. Figure 3.1 shows the number of active IPv4 prefixes in the BGP routing tables of the routers in the Telstra AS (AS 1221). This figure illustrates the sustained growth of the BGP routing tables. Since IPv6 is currently not largely deployed, the number of IPv6 prefixes contained in the BGP routing tables is still low, with a few hundreds of prefixes, as illustrated by Figure 3.2. However, their number is also increasing rapidly. Without better route aggregation, it is expected that the number of IPv6 prefixes in the BGP routing tables will be similar or even larger than the number of IPv4 prefixes.

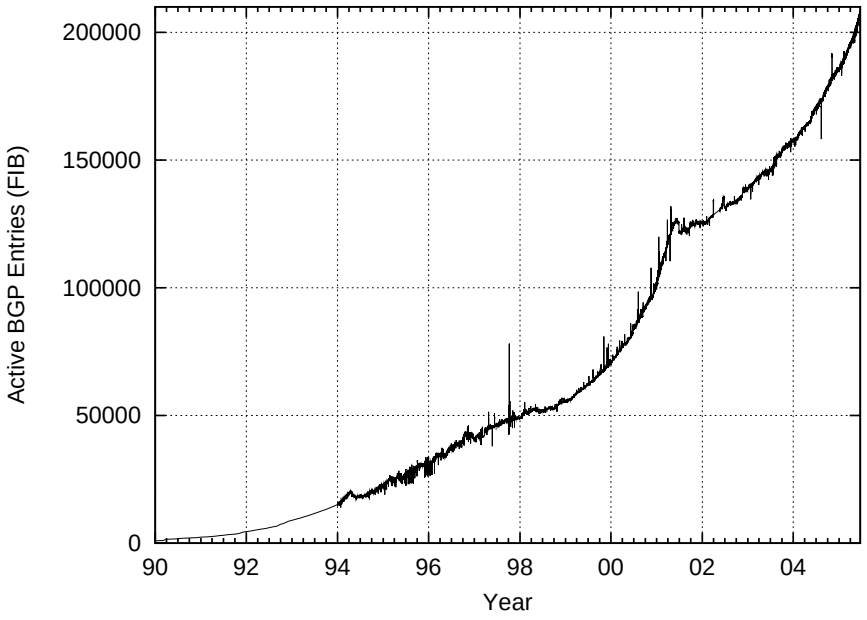


Figure 3.1. Number of active BGP IPv4 entries in the Telstra routers

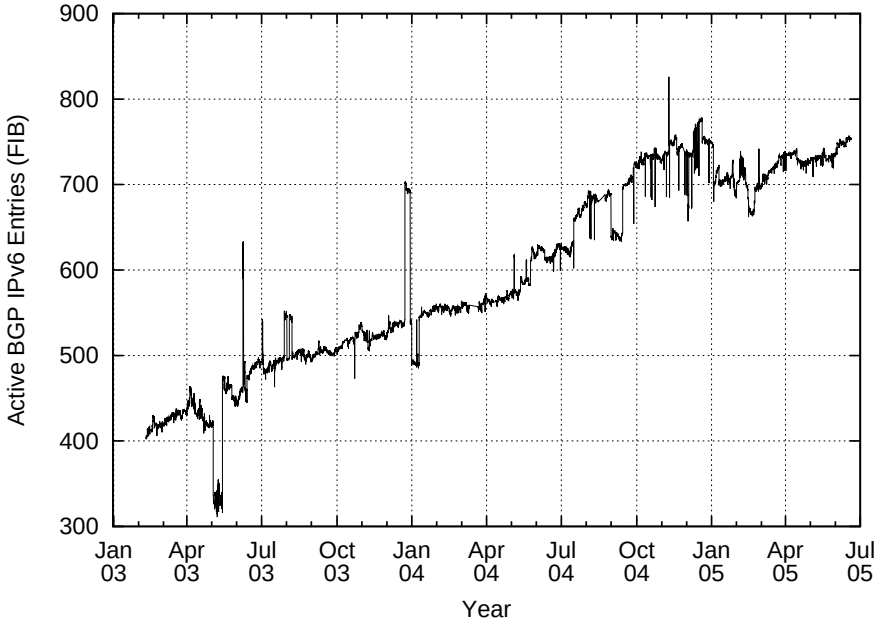


Figure 3.2. Number of active BGP IPv6 entries in the Telstra routers

The current size of those tables already causes operational issues for some Internet Service Providers, as it has major implications on storage requirements, computational load, forwarding performance, and protocol overhead for Internet routers [110, 29]. Moreover, several experts are concerned about the increasing risk of instability of BGP [15, 123].

The scalability of the Internet routing infrastructure depends on the number of prefixes that appear in the BGP routing tables. This in turn depends on how often autonomous systems use the minimum number of prefixes to summarise all of its IP addresses, process known as *route aggregation*, see section 1.2.4. However, route aggregation cannot be performed in all situations. One reason is multihoming. As described in 1.4, IPv4 multihoming is achieved by announcing with BGP a single prefix to each provider. A multihomed AS receives either provider-aggregatable addresses from some or all of its providers, or it receives a provider-independent address block directly from an Internet Registry. In any case, at least one provider is unable to aggregate the prefix from the multihomed AS. Figure 1.9 and Figure 1.10 in section 1.4.1 illustrated this problem.

According to a work about the growth of the BGP routing tables [37], multihoming is responsible for approximately 20 to 30% more prefixes. Moreover, the number of multihomed sites is growing, since many ISPs and corporate networks wish to enhance their reliability and improve their network performance such as network latency. Nowadays, at least 60% of stub domains are connected to two or more providers [3]. Many sites are expected to also require to be multihomed in IPv6, even end users with multiple interfaces to GSM, UMTS, or 802.11 networks. In order to preserve the size of the BGP routing tables in the Internet, the IPv4 multihoming mechanism is prevented in IPv6 by operational procedures [173]. A new multihoming solution that ensures route aggregation is required for IPv6 [15].

The overall issue of IPv6 site multihoming is to provide some functionalities to meet various motivations, under several technical and non-technical constraints. For instance, redundancy is a major motivation that pushes sites to become multihomed, while *scalability* is a technical *constraint* that any IPv6 multihoming solution must respect. RFC 3582 [1] mixes functionalities, motivations and constraints all together, and calls them *goals*. As this term is sometimes ambiguous, e.g. redundancy and scalability can both be referred to as *goals*, we chose to distinguish these three notions for clarity purposes.

The motivations for multihoming are described in section 3.2, while the functionalities needed to meet these motivations are detailed in section 3.3. The constraints are presented in section 3.4.

3.2 Multihoming Motivations

This section identifies major reasons why sites wish to be multihomed in the current IPv4 Internet. These motivations reflect the benefits that sites expect to obtain by connecting to multiple providers. These motivations are expected to be the same in an IPv6 Internet. This section is mostly based on [2, 1, 177].

3.2.1 Redundancy

A major motivation to connect to multiple providers is to protect the multihomed site against various failure modes, within one or more transit providers, as well as link failures between the multihomed site and the providers. Multihoming is expected to accommodate –in the general case, issues of shared fate notwithstanding– continuity of connectivity during the following failures :

- physical failure (e.g. a fibre cut during construction works or a router hardware failure)
- logical link failure (e.g. a misbehaving router interface)
- routing protocol failure (e.g. a BGP peer reset or misbehaving)
- human errors (e.g. configuration errors)
- transit provider failure (any of the above error occurring in an upstream provider, e.g. a backbone-wide IGP failure)

3.2.2 Load Sharing

By being multihomed, a site is able to distribute its traffic between multiple transit providers. The objective is to use the multiple providers concurrently, not just the usage of one provider over one interval of time and another provider over a different interval. A reason behind this might be that any one transit provider might not fulfil alone all the requirements of the site. In particular, reasons behind sharing the load between several providers may include performance and policy, as explained below. For example, a site might use one provider for its global connectivity, and another provider for the traffic it exchanges locally.

3.2.3 Performance

Many sites use multiple providers to improve their network performance, e.g. delay, loss, jitter or raw bandwidth. For instance, a site might use a high-bandwidth, low delay but expensive link for its real-time traffic, and use another, cheaper link for the rest of its traffic. Sometimes a site can experience performance difficulties to reach some destinations. In this case, it may be useful to redistribute the traffic on the links with other providers.

3.2.4 Policy

The traffic is sometimes distributed based on some policy beyond technical scope, e.g. cost or commercial reasons. An academic institution might for instance direct the commercial traffic to the provider offering global Internet connectivity, while sending its research traffic through a national or international research network. Another example is a site that might prefer to use several cheap low-bandwidth connections instead of a single expensive high-bandwidth connection, only for cost reasons.

3.2.5 Independence

Technical reasons for independence are mainly covered under redundancy. Independence here focuses on economic, political and administrative perspectives. Multihoming with your own address space brings some degree of provider independence. This in turn may help the multihomed site in obtaining better service level agreements, or being able to get lower prices.

3.3 Multihoming Functionalities

As explained in section 3.2, sites wish to be multihomed for various technical (e.g. redundancy, load-sharing, network performance) and non-technical reasons (e.g. cost, commercial relationships, independence). These reasons can be referred to as *motivations*. In order to meet these motivations, any IPv6 multihoming solution must provide sufficient *functionalities*. Two main functionalities can be distinguished : full fault tolerance and traffic engineering capabilities.

3.3.1 Fault-Tolerance

The most common motivation for multihoming is probably redundancy. Hence, any multihoming solution must provide mechanisms to protect the multihomed site against the failure of a link with one of its providers, as well as distant failures in the Internet, see section 3.2.1. This is illustrated on Figure 3.3, where the multihomed site protects itself against the distant failure of Link C, not only against the failure of Link A or Link B.

A site is said to *re-home* when it is changing of provider for carrying its packets, for instance because the link with the previous provider failed. This change should be as quick as possible. A required mechanism is to provide transparency for transport-layer sessions across such re-homing events. This means that the exchange of data between devices on the multihomed site and devices elsewhere on the Internet should proceed with no greater interruption than that associated with the transient packet loss during the re-homing event. Transport-layer sessions include protocols such as TCP, UDP and SCTP over IP. Applications which

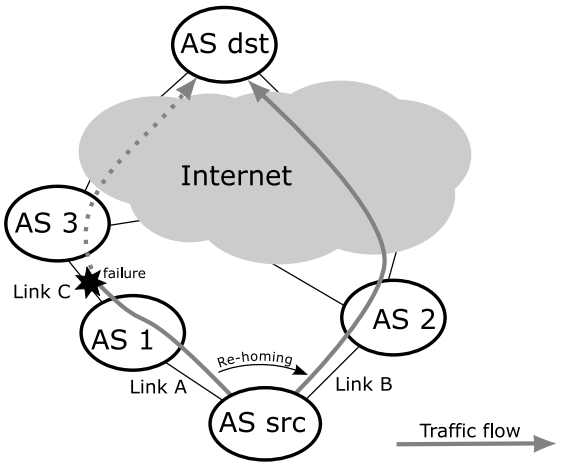


Figure 3.3. Re-homing after the detection of a distant failure

communicate over raw IP and other network-layer protocols may also benefit from re-homing transparency.

3.3.2 Traffic Engineering Capabilities

An adequate set of traffic engineering functionalities is required to fulfil the load-sharing, performance, and policy motivations for multihoming, described previously in section 3.2. Required traffic engineering functionalities include :

- the ability to distribute both inbound and outbound traffic between multiple transit providers,
- the ability to switch the traffic from one provider to another in case of performance difficulties,
- the ability to cope with policy reasons beyond technical scope (e.g. cost, or acceptable use conditions).

3.4 Multihoming Constraints

In addition to providing the previous functionalities, any multihoming solutions for IPv6 must also respect technical and non-technical *constraints* [1].

3.4.1 Scalability

As explained in section 3.1, current IPv4 multihoming practices contribute significantly to the rapid growth of the BGP routing tables in the Internet [110, 37]. This is a concern, both because of the hardware requirements it imposes, and

also because it has an impact on the stability of the routing system. A major constraint required for an IPv6 multihoming solution is to preserve the scalability of the routing system, by allowing route aggregation at the level of the providers of the multihomed site. A new IPv6 multihoming architecture should scale to accommodate orders of magnitude more multihomed sites than the current IPv4 Internet, without imposing unreasonable requirements on the routing infrastructure.

3.4.2 Compatibility with Packet Filtering

For security reasons, the IETF recommends that ISPs drop the customers' traffic entering their networks that is coming from a source address not legitimately in use by the customer network [82, 25]. This practice, called *ingress filtering*, aims at limiting the impact of Distributed Denial of Service Attacks (DDOS), by denying traffic with spoofed addresses access to the network, and to help ensure that traffic is traceable to its correct source network. The reasoning behind the ingress filtering procedure is that distributed denial of service attacks frequently spoof the source addresses of other systems, making it difficult to trace the source of the attack. If the traffic leaving a stub network and entering an ISP can be limited to traffic with legitimate source addresses, attacks can be somewhat mitigated : traffic with random or improper source addresses can be suppressed before it does significant damage, and attacks can be readily traced back to at least their source networks [82].

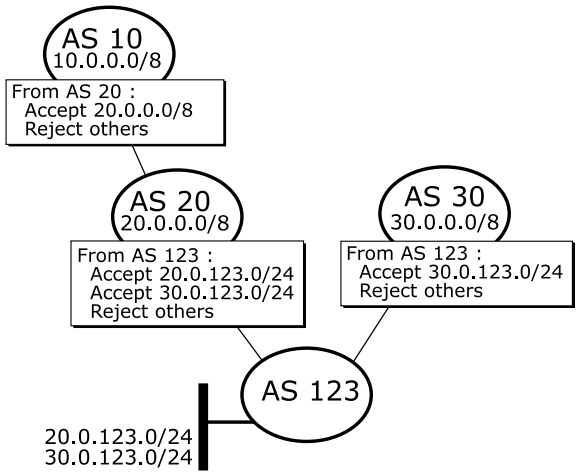


Figure 3.4. Examples of ingress filters applied by the providers

A multihomed site can negotiate with a provider so that another prefix is allowed by the provider. For instance, in Figure 3.4, AS 20 normally only accepts packets with source addresses belonging to the prefix it assigned to AS 123, i.e.

the 20.0.123.0/24 prefix. In this example, AS 20 agreed to relax its ingress filter in order to also accept packets with a source address belonging to the 30.0.123.0/24 prefix. However, it is not always possible for a site to require that a provider relaxes its filters. For instance, in Figure 3.4, AS 30 refuses to accept packets with source addresses belonging to the 20.0.123.0/24 prefix. Moreover, even if a provider agrees to relax its filters, like AS 20 in the example, it is not guaranteed that the packet will not be stopped by an upstream provider, like AS 10.

Ingress filtering can be implemented in at least five ways, with varying impact : Ingress Access Lists, Strict Reverse Path Forwarding, Feasible Path Reverse Path Forwarding, Loose Reverse Path Forwarding, and Loose Reverse Path Forwarding ignoring default routes. These mechanisms are described in details in [25].

For security reasons, a multihoming solution should not preclude this ingress filtering procedure.

3.4.3 Multihoming Independence

Multihoming independence means here the absence of cooperation related specifically to the multihoming technique. One can distinguish the independence between the transit providers, and the independence between a multihomed site and its providers.

Independence between a multihomed site and its providers means that no particular configuration or service is required from the providers to allow the site to be multihomed. In other words, no accommodation is required beyond that a provider would do for a single-homed customer. A provider might even not be aware that its customer is multihomed. In particular, a provider should not be required to modify its ingress filtering policy. Independence between a multihomed site and its providers is desirable since it would allow home office networks to easily become multihomed. This independence is however not strictly required for an IPv6 multihoming solution.

Although a multihoming architecture may require cooperation between a site and its transit providers, it should not require any cooperation directly between the transit providers.

A multihoming solution provides *Complete Multihoming Independence* when independence is assured both between a site and its providers, and between the transit providers of the site. Otherwise, when the solution requires some cooperation between the multihomed site and its providers, then the solution is said to provide only *Partial Multihoming Independence*. It can be noted that complete multihoming independence implies the previous constraint, i.e. compatibility with packet filtering. Complete multihoming independence is typically achieved through the use of multiple prefixes per site, or through the use of Network Address Translation (NAT). For instance, IPv4 multihoming with BGP provides partial multihoming independence, while IPv4 multihoming with NAT provides complete multihoming independence.

3.4.4 Impact on Legacy System

A new multihoming strategy should limit its impact on the hosts, the routers and the Domain Name System (DNS), in order to be gradually deployed and to not prevent normal operation of legacy hosts and routers.

The multihoming solution may require changes to hosts or applications, in particular if the changes would enhance the transport-layer survivability. However, a legacy host that implements basic IPv6 specifications [100, 98, 89] should still be able to work in a multihomed site, even if it cannot benefit from site-multihoming. For instance, a legacy host may lose connectivity if the connection with one transit provider is lost, even if another provider is still operational. The changes required by the solution to the host stack, socket API or transport layer should be distinct functions added to existing functions, so that it is possible to retain the original socket API or transport protocols in parallel, even if they cannot benefit from multihoming.

The solutions may also require changes to IPv6 router implementations, but these changes should not prevent normal single-homed operation. Routers that implement these changes should be able to interoperate fully with hosts and routers not implementing them.

Finally, the solution may also require modifications to the Domain Name System. These solutions either should be compatible with the observed dynamics of the current DNS system, or the solutions should demonstrate that the modified name resolution system required to support them is readily deployable.

3.4.5 Simplicity

Since any proposed multihoming solution must be deployed in real networks with real customers, simplicity is highly desired. The current multihoming solution is quite straightforward to deploy and maintain. A new IPv6 multihoming solution should not be substantially more complex to deploy and operate, for multihomed sites or for the rest of the Internet, than the current IPv4 multihoming practices.

Chapter 4

IPv6 Multihoming State of the Art

Over 40 solutions for IPv6 multihoming have been proposed during the last few years [22]. This chapter reviews only on the most relevant ones. It classifies the solutions for IPv6 multihoming according to their *architectural* approaches, rather than their implementation techniques and common issues as in [111]. This state of the art also methodically compares the solutions according to their mechanisms, benefits and drawbacks, outlining in this way the major steps that have led to a new multihoming architecture for IPv6.

4.1 Introduction

Briefly stated, the IPv6 multihoming problem consists in finding a solution that provides complete fault-tolerance and traffic engineering functionalities, while respecting two main constraints : scalability and multihoming independence. The main architectural approaches for multihoming can be classified according to where are located the fundamental mechanisms that provide fault-tolerance, traffic engineering functionalities, route aggregation and multihoming independence. Three approaches are distinguished : *Routing*, *Middle-Box*, and *Host-Centric* approaches. These approaches are basically illustrated on Figure 4.1. In Routing approaches, the mechanisms for supporting IPv6 multihoming are located in

Internet routers in general. In Middle-Box approaches, these mechanisms are provided by intermediate agents, located between the multihomed site and the global Internet. Finally, in Host-Centric approaches, multihoming functionalities are provided mainly by the multihomed hosts themselves.

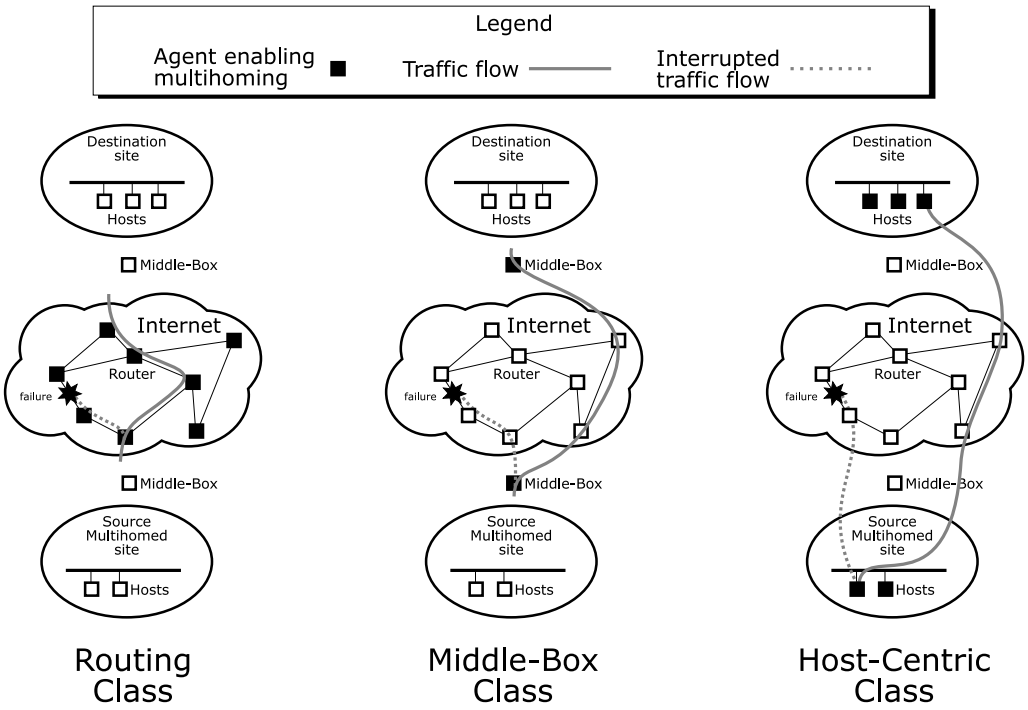


Figure 4.1. Classes of IPv6 multihoming solutions

This chapter will follow a path in the decision tree that yields to the solution selected by the IETF. The first three sections focus on the Routing, Middle-Box and Host-Centric approaches. As it will appear that Host-Centric is the most promising approach, Section 4.4 will detail several proposed solutions that belong to this class. These solutions are classified into *Transport Layer* (Section 4.4.2), *Pure Network Layer* (Section 4.4.3), and *Intermediate Network Layer* (Section 4.4.4) approaches, the latter approach being the one chosen by the IETF. Finally, at the end of the decision tree, several Intermediate Network Layer solutions are presented, differing from the kind of identifier namespace. Among the solutions proposed, the SHIM approach is the one fostered by the IETF. It is detailed in Section 4.6.

Mechanisms, advantages and drawbacks of all solutions presented in this chapter will be summarised in Table 4.2 and Table 4.3.

4.2 Routing Approaches for IPv6 Multihoming

The class of Routing solutions groups multihoming architectures that rely on the routing system in general to roughly provide fault-tolerance and traffic engineering functionalities, while allowing route aggregation. Routing mechanisms include the use of BGP, the filtering of BGP route advertisements, or the use of tunnels. IPv6 multihoming solutions that belong to this class are *IPv6 Multihoming with BGP* [125], *IPv6 Multihoming using Cooperation between Providers* [117] and *IPv6 Multihoming Support at Site Exit Router* [92]. A few other Routing approaches are also shortly presented.

4.2.1 IPv6 Multihoming with BGP

IPv6 multihoming with BGP adapts to IPv6 the traditional IPv4 multihoming method. The procedure is detailed in [125]. Like in IPv4, a site uses BGP to announce its own prefix to each provider. As explained previously and illustrated in Figure 4.2, this solution causes scalability problems, as each multihomed site introduces a new prefix in the BGP routing tables of all routers in the Internet, even if the site is using provider-aggregatable prefixes.

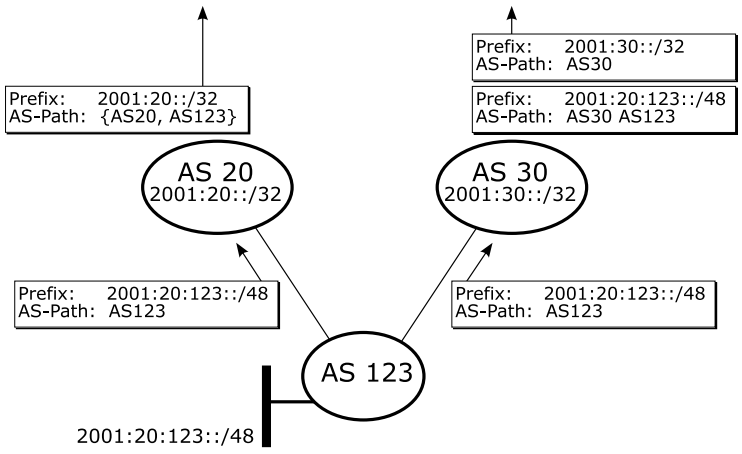


Figure 4.2. IPv6 Multihoming with BGP using provider-aggregatable prefixes

This multihoming solution is considered acceptable only for large ISPs and transit providers, which use provider-independent prefixes. It is not a solution for small ISP, home and enterprise networks, for scalability reasons but also because it requires the use of BGP.

Fault-tolerance and traffic engineering are typically provided by adequate configuration of BGP and IGP. For instance, a multihomed site can engineer its

outbound traffic by assigning appropriate IGP weights to its intradomain links, or it can use more complex techniques [168, 203]. Inbound traffic engineering is as difficult to control as with IPv4. AS-Path prepending, MED or community attributes can be used to roughly control the amount of traffic received from the providers [33, 165, 203].

4.2.2 IPv6 Multihoming using Cooperation between Providers

This solution relies on providers that cooperate to filter BGP routes, in order to enable route aggregation while still providing some fault-tolerance. It uses the existing routing protocols and implementations. The solution is named *IPv6 Multi-Homing with Route Aggregation*. [117, 23], but it could be used also with IPv4. Figure 4.3 illustrates a multihomed site that uses this mechanism. It is connected to two providers, AS 10 and AS 20. The multihomed site received a single provider-aggregatable prefix 2001:10:1::/48 from one of its providers, AS 10 in this example. This particular ISP is named the *primary ISP*.

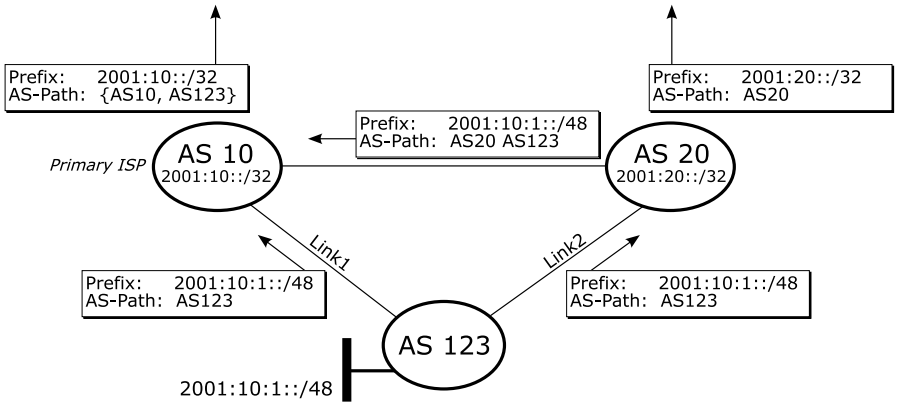


Figure 4.3. IPv6 Multihoming through cooperation between providers

The multihomed site advertises its prefix to both AS 10 and AS 20. In order to preserve the interdomain routing, AS 20 propagates prefix 2001:10:1::/48 to AS 10 and to AS 10 only. AS 10 is able to aggregate this prefix with its own prefix 2001:10::/32, and it announces only the aggregated prefix 2001:10::/32 to the global Internet. AS 20 does not propagate prefix 200:10:1::/48 to the global Internet. This can be done for example by using BGP redistribution communities [166, 167, 168]. As a result of this routing advertisement, the traffic coming from the Internet and destined for the site is always routed through AS 10, since only AS 10 announced the prefix of the site to the Internet. AS 10 will forward the traffic destined for its customer either directly through Link1 or via AS 20, according to some routing policy. The multihomed site can send its outbound traffic indifferently through AS 10 or AS 20.

If Link2 fails, both inbound and outbound traffic will flow through link1. Similarly, if Link1 fails, the inbound traffic will reach the multihomed site by taking the path AS 10 → AS 20 → multihomed site. The outbound traffic will take the reverse forwarding path.

Route aggregation is achieved because only the provider that assigned the PA prefix to the multihomed site aggregates and announces it to the Internet. This solution does not provide fault tolerance neither in case of a failure within the primary ISP (AS 10), nor for primary ISP Internet link failure (Link1). Additionally, if there is no direct link between AS 10 and AS 20, then the prefix 2001:10:1::/48 announced to AS 20 must be propagated to AS 10 through an intermediary transit provider. In such cases, a tunnel must be used, otherwise less aggregation is achieved and/or cooperation is needed between the transit providers, for instance through the use of the BGP Communities Attribute [200, 32]. This cooperation may conflict with their commercial interests, and may become unmanageable if the number of multihomed sites using this mechanism increases. This solution also forces the client to depend on its primary provider.

A similar solution is described in [199], where a group of providers administrates cooperatively one prefix and one ASN for their multihomed customers. Each customer is assigned a subprefix, e.g. a /48, based on the current assignment rules. However, the group of providers advertises to the global Internet only the aggregated prefix, e.g. a /32.

4.2.3 IPv6 Multihoming Support at Site Exit Router

IPv6 Multihoming Support at Site Exit Router is a Routing solution that relies on the use of tunnels and multiple prefixes. It is described in RFC 3178 [92, 23]. In this approach, a multihomed site is assigned multiple prefixes, one per provider.

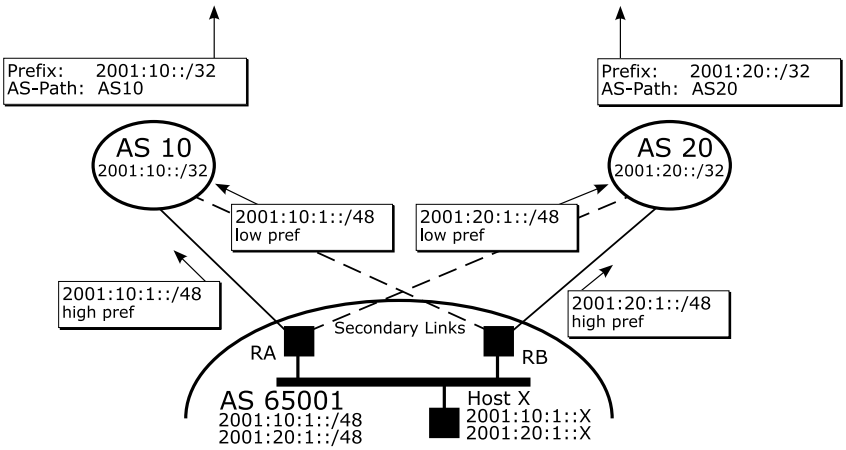


Figure 4.4. IPv6 Multihoming support at site exit router

In the example illustrated in Figure 4.4, AS 65001 obtained prefix 2001:10:1::/48 from AS 10 and prefix 2001:20:1::/48 from AS 20. The two prefixes are advertised by the site exit routers RA and RB to every host inside AS 65001. These prefixes are used to derive one IPv6 address per provider for each host interface. Route aggregation is achieved by announcing to a given provider only the prefix allocated by this provider, so that each provider is able to perform route aggregation. For instance, AS 65001 advertises prefix 2001:10:1::/48 only to AS 10, and AS 10 announces only its own IPv6 aggregate 2001:10::/32 to the global Internet. Besides route aggregation, an advantage provided by this solution is that no particular service related to multihoming is required from the transit providers, as the site is not really multihomed, but rather single-homed to each provider. How hosts select the source address of their packets is not defined in this solution.

Redundancy is provided by using secondary links, established between RA and AS 20, and between RB and AS 10. In Figure 4.4, RA advertises prefix 2001:20:1::/48 towards AS 20 over the secondary link, which is typically an IP-over-IP tunnel that goes physically through AS 10. Similarly, RB advertises prefix 2001:10:1::/48 towards AS 10 over the secondary link. In normal conditions, secondary links are advertised by the routing protocol with a low preference, so that the primary links are used. When a failure occurs on a primary link between the site and its ISP, normal operation of the routing protocol ensures that the routing advertisement corresponding to this particular path is withdrawn. In this case, the path using the secondary link becomes a valid option for the routers.

This architecture provides route aggregation and is able to preserve the established connections across link failures. The main concerns are that it does not cope with the failure of any of the upstream ISPs, and that it forces each ISP to configure tunnels. Moreover, in the case of a long-term failure, the traffic that flows through the secondary link should be switched to the primary link of the valid provider. This requires some mechanism to prevent the use of addresses belonging to the failed ISP.

4.2.4 Miscellaneous Routing Solutions

Miscellaneous solutions related to Routing solutions are briefly described here.

Multihoming Using IPv6 Addressing Derived from AS Numbers

IPv4 Multihoming with BGP is prevented in IPv6 by operational procedures [173]. A document [178] suggests a few possible approaches which could be used to derive the IPv6 address space automatically from an AS number. This solution would implicitly give a provider-independent prefix to everyone who has an AS number. However, routability is not guaranteed, and the solution is not applicable to sites that do not have an AS number.

Geographic Address Allocation

Several proposals like [161, 94] have been made to allocate IP addresses in some geographic fashion, in order to address the route aggregation problem. To get around the scalability problem, a geographical aggregation scheme splits the global routing domain into a number of smaller regional ones, where flat routing happens in each region. Ideally, only aggregates are visible outside the regions. For instance, the core routers in the Internet would contain one route per country, or even one route per part of continent. A major issue with geographic address allocation is global routing. No transit provider is willing to advertise the aggregate for a region if not all sites in the region are customers and pay for the transit service. Moreover, it is not uncommon that sites move from one region to another, e.g. when two ISPs merge together. Thus, economic and operational realities would seem to indicate that it is not a realistic approach.

A similar proposal [206] suggests to aggregate the routing information for multihomed destinations inside service provider networks based on geography to accomplish scalable multihoming in IPv6. This would mitigate the route aggregation issue, but it does not solve it.

Routing support for IPv6 Multi-homing

A proposal [36] suggests that each provider advertises its own prefix downwards to all its customers, and also propagates down to the customers all prefixes received from higher level providers. This is clearly redundant for upward routing purposes since each provider still advertises the default route ::/0 to the customer. The utility is that if a customer receives a BGP advertisement for the prefix of an upper-level provider, then he can deduce that the Internet can reach him through this provider. Conversely, the withdrawal of such route indicates to the receiving AS that the addresses are no longer reachable from the Internet. This reachability information can be used by hosts in the customer network to avoid the use of those addresses as source addresses.

4.3 Middle-Box Approaches for IPv6 Multihoming

Middle-Box approaches provide multihoming functionalities through services offered by intermediary boxes between multihomed hosts and the Internet, for instance a NAT box. Multihoming architectures that belong to this Middle-Box class include *Multihoming with NAT* [6], *Multihoming Aliasing Protocol* [159], and *Multihoming Translation Protocol* [160].

4.3.1 IPv6 Multihoming with NAT

IPv6 Multihoming with NAT relies on the use of Network Address Translation to direct packets towards a working provider. Typically, a NAT router is installed at the edge of the network, and knows which provider is up and which is not. Based on this knowledge, the NAT router substitutes the source IP address of an outgoing packets with an IP address belonging to the prefix of a operational provider. Figure 4.5 illustrates a site that is multihomed by using NAT. The site received two IPv6 prefixes, one from each of its providers. Hosts within the site may use addresses from either the first or the second provider. When a failure occurs, the source addresses contained in outgoing packets can be rewritten by the NAT box, in order to pass the filters applied by the newly selected provider.

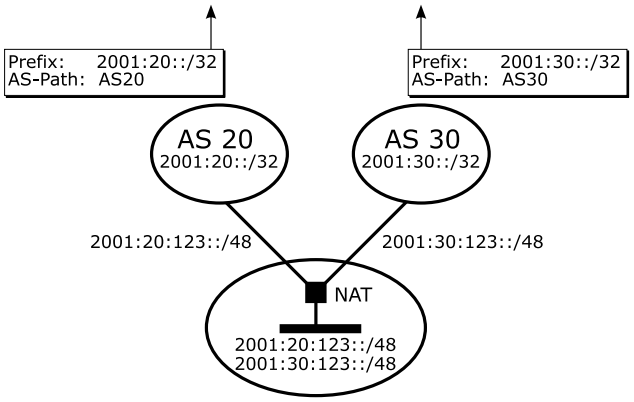


Figure 4.5. IPv6 Multihoming with NAT

Multihoming with NAT allows route aggregation and provides complete independence with respect to the providers. The site does not need to run BGP. The NAT box can also be used to somewhat control the amount of traffic sent and received through each provider, as it is often done today in IPv4 [7].

However, NAT is not considered as a good engineering practice in an IPv6 Internet, because it has many architectural implications [93]. In particular, NAT alters packets and is not sufficient to achieve transport-layer survivability. Indeed, as explained in Section 1.4.2, if a failure occurs that affect some connection, it is not possible to intercept and continue the connection, since the outgoing public IP addresses cannot be modified without breaking the TCP session. A solution would be to establish some cooperation with a NAT box installed in the remote site, so that the new IP address can be rewritten back to its original form when a packet reaches the destination site. This idea is used by the MHTP and MHAP solutions, described in the next section.

The use of middles boxes causes other difficulties : it is not well suited for geographically distributed configurations. Additionally, the use of middle boxes

breaks the leading design principle for Internet protocols, called the *end-to-end principle* [38, 43, 176]. According to this principle, it is beneficial to avoid polluting the network with state, and to limit new state creation to the involved end nodes.

For all these reasons, this approach is not considered suitable for IPv6 site multihoming, although it may be interesting for use as a transition mechanism, or for small residential networks.

4.3.2 Multihoming Aliasing and Translation Protocols

Multihoming Translation Protocol (MHTP) [159] and its variant *Multihoming Aliasing Protocol* (MHAP) [160] propose to set up middle-boxes at the edges of the multihomed sites. The middle-boxes, called *endpoints* use a protocol to convert provider-independent addresses to provider-aggregatable addresses when leaving the multihomed site, and convert the addresses back to the original form when reaching the edge of the destination site. Hence a fully routable and aggregatable space is used in the core of the Internet, while a provider-independent space is used at the edge of the Internet, e.g. in multihomed sites. The drawback is that an additional routing table (the MHTP routing table) must be maintained for multihomed networks. This adds an additional layer of indirection, shifting the scalability issue of site multihoming to a separate protocol. Moreover, it is difficult to ensure in an Internet-wide environment that the addresses will be written back, and that no intermediate router will need to access the original addresses, e.g. to send an ICMP message. In addition, it is not evident that this approach can be made reasonably secure, though no security analysis was ever performed on this approach.

4.4 Host-Centric Approaches for IPv6 Multihoming

Host-Centric Multihoming groups all solutions that rely on the use of multiple prefixes and host capabilities to provide fault-tolerance, traffic engineering and route aggregation. Many proposed solutions belong to this class : *SIM* [147], *NOID* [148], *CB64* [146], *SHIM* [150], *WIMP* [212], *HIP* [137], *LIN6* [195] and and *E2E* [153]. Several transport protocols (*SCTP* [188], *DCCP* [122] and *TCP-MH* [129]) have been adapted to operate in such an environment.

The architecture common to all Host-Centric multihoming approaches is first described in section 4.4.1. Next, Host-Centric approaches are further divided into *Transport Layer*, *Pure Network Layer*, and *Intermediate Network Layer* approaches, described respectively in sections 4.4.2, 4.4.3, and 4.4.4. Finally, mechanisms, advantages and drawbacks of all solutions will be summarised in Table 4.2 and Table 4.3.

4.4.1 Architecture

Figure 4.6 illustrates a standard IPv6 multihomed site. Two Internet Service Providers, AS 10 and AS 20, provide connectivity to the multihomed site AS 65001. AS 65001 received one prefix per provider, e.g. 2001:10:1::/48 from AS 10 and 2001:20:1::/48 from AS 20. The two prefixes are advertised by the site exit routers RA and RB to every host within AS 65001. These prefixes are used to derive one IPv6 address per provider for each host interface. Route aggregation is achieved, because AS 65001 advertises prefix 2001:10:1::/48 only to AS 10, and AS 10 only announces its own IPv6 aggregate 2001:10::/32 to the global Internet.

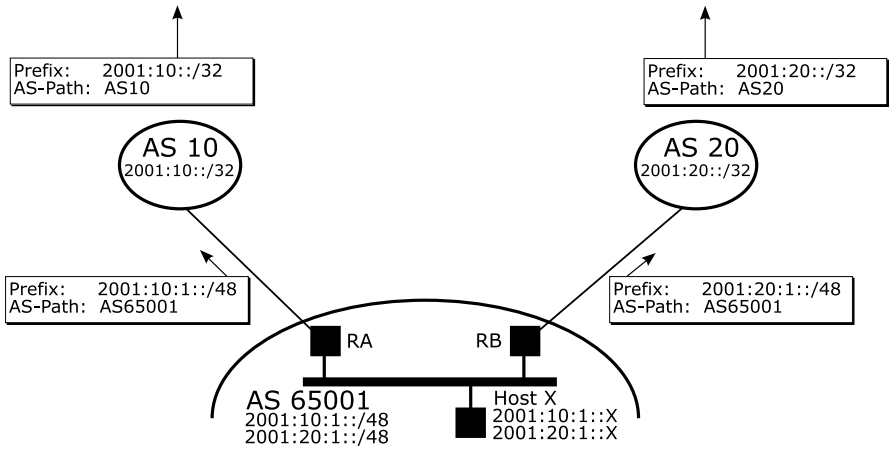


Figure 4.6. Host-Centric IPv6 Multihoming

Source Address Dependent Routing

To respect ingress filtering policies applied by the providers [25], the site must ensure that all outgoing packets with a source address within prefix 2001:10:1::/48 are sent through AS 10. Similarly, outgoing packets with a source address within prefix 2001:20:1::/48 must be sent through AS 20, otherwise they would be dropped when entering in AS 10. As a consequence, the source address selected by a host determines the upstream provider used.

The simplest solution to respect this constraint is to have only one site exit router connected to all providers. This router selects the exit link on the basis of the source address contained in the packet [107]. Another solution is to implement source address dependent routing, either in the whole multihomed sites, or simply in a connected domain that includes all the site exit routers [107], as illustrated by Figure 4.7. Tunnels can be used if the site exit routers are not directly interconnected.

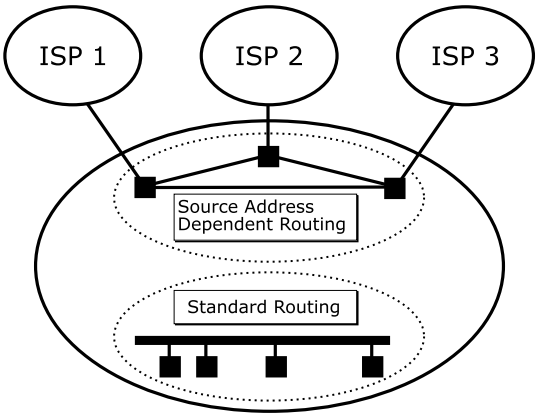


Figure 4.7. Source address dependent routing between the site exit routers

Preservation of Established Connections

The difference with IPv6 Multihoming Support at Site Exit Router is that fault-tolerance is not provided by the use of backup links. Instead, it is based on enhanced host capacity to detect the failure of a path, and to switch from one provider to another. In Figure 4.6, suppose for instance that some failure occurs within AS 10. The host detects this failure, for example by examining the number of packet losses. To switch from the failed provider to the other provider, the host simply selects another source address for its outgoing packets. This would break the transport connections, unless some new protocol or mechanism is used by the host to preserve its established connections. This can be done in practice by using an enhanced TCP like TCP-MH [129] or a dedicated transport-layer protocol like SCTP [188, 50, 154, 49] or DCCP [121, 122, 83], or by designing something new below the transport layer like HIP [137], SHIM [150], SIM [147], NOID [148], CB64 [146], or WIMP [212].

As applications have many different requirements for the quality of their network connection, an advantage brought by this solution is that each application can decide by itself if the connection is good enough. If needed, an application may switch to another provider by simply using another source address. The major drawback is that new mechanisms are required to preserve the established connections, in both source and destination hosts. This solution also heavily change how traffic engineering is achieved as hosts can decide which provider they use to send their outgoing traffic. This will be largely discussed in this thesis. Further issues related to this class of solutions are described in [106]. However, a quick look to Table 4.2 shows that Host-Centric approaches provide all major functionalities required, at the price of some modifications to end-hosts. This trade-off is considered acceptable, especially if the required modifications also produce benefits outside the framework of IPv6 multihoming.

4.4.2 Transport Layer Approaches

As previously stated, the mechanisms for the provision of multihoming support can be located in the transport layer or in the network layer of the stack. Current transport protocols, e.g. TCP or UDP, identify the endpoints involved in a communication through their IP addresses. If one address changes, the transport-level flow breaks. With the Host-Centric solutions, hosts have several addresses. They need to be able to use them interchangeably during the lifetime of a flow in order to survive outages affecting any of those addresses. Transport Layer approaches suggest the support of multiple addresses per endpoint in the transport layer, so that an address can be substituted with another without breaking the communication. A few current transport protocols already support this feature, like SCTP [188] and DCCP [122]. For those protocols, only minor modifications [51, 121] are required to adapt them to the multihoming scenario. Older transport protocols, like TCP or UDP, do not currently support the use of multiple addresses, and require substantial modifications, like TCP-MH [129].

It is usually believed that the transport layer has a better understanding than the network layer of which address is working and which is not. The transport layer naturally obtains information on the quality of different paths. However, working at the transport level requires a different mechanism for every transport protocol. With respect to security, cookie based protections may be enough to ensure that no new security threats are introduced. The reason is that attacks to transport layer solutions are performed on a per connection basis, in contrast to network layer solutions.

TCP-MH

The existing TCP is not designed to manipulate multiple addresses in one TCP session. If a network outage occurs and the access-line associated with the local and remote IP addresses is down, the TCP session will finally time out and terminate.

An extension to TCP has been proposed in [129], where SYN segments contain all the IP addresses available to reach the source node. TCP-MH also defines MH-Add and MH-Delete options in order to convey local address information from the sender to the receiver over an established TCP connection. These options are used one at a time during a connection to add or remove usable IP addresses. When an outage is detected, the endpoints switch to another available pair of IP addresses. A serial number is added in the MH-Add and MH-Delete options so that these options are more difficult to fake for an attacker trying to hijack an existing TCP connection, but many other security issues still exist [17].

TCP-MH is not the first proposition to enhance TCP. *Multi-homed TCP* [104] already suggested in 1995 to identify packets belonging the same connection by using a context identifier that is sent in a TCP option, rather than using the source

and destination addresses and ports. Outgoing packets are sent to one or more addresses from which data has recently been received for the same connection.

Yet another proposal [192] defines additional options to TCP, to be used when initiating the connection. During this phase, both endpoints agree on the alternative addresses that can be used to deliver packets belonging to the TCP session.

SCTP

The Stream Control Transmission Protocol (SCTP) is a new, reliable, connection-oriented transport protocol [154, 188, 50]. It can be used as an alternative to TCP and UDP. A relationship is created and maintained between two endpoints of an SCTP association until all packets have been successfully transmitted. SCTP allows data to be partitioned into multiple streams that have the property of independently sequenced delivery, so that a message lost in any one stream will only initially affect delivery within that stream, and not delivery in other streams.

A core feature of SCTP is the ability for a SCTP endpoint to support multiple IP addresses. SCTP endpoints exchange their lists of addresses during the initiation of an association. No IP address can be added or deleted once the association has been established, although an extension to SCTP can provide this feature [187]. Each endpoint is able to receive messages from any of the addresses associated with the remote endpoint. However, a single address is chosen as the *primary* address and is used as the destination for normal transmission. Each endpoint monitors the reachability of the secondary addresses of its peer so that it always knows which addresses are available for the failover. The monitoring is done by sending a heartbeat packet to an idle destination address, which the peer acknowledges. A secondary address is used when continued failure to send to the primary address is noticed, until heartbeat packets determine that the primary address is reachable again.

Issues and discussion on the applicability of SCTP to the multihoming problem are presented in [49, 51]. From the security viewpoint, SCTP uses a random verification tag as a weak security mechanism to avoid packet injection. SCTP protects itself against TCP SYN flooding attacks by remaining stateless during the handshake in order to prevent state-exhaustion attacks. Security issues are discussed in details in [17].

DCCP

The Datagram Congestion Control Protocol (DCCP) [122, 83] did not originally support multihoming. An extension to DCCP [121] provides primitive support for multihoming and mobility via a mechanism for transferring a connection endpoint from one address to another. The moving endpoint must negotiate this support

beforehand. When the moving endpoint gets a new address, it sends a DCCP-Move packet from that address to the stationary endpoint. Next, the stationary endpoint changes its connection state to use the new address.

DCCP support for mobility is intended to solve only the simplest multihoming and mobility problems; for instance, there is no support for simultaneous moves.

4.4.3 Pure Network Layer Approaches

In the early stages of development of the IPv6 multihoming solution, the Mobile IPv6 (MIPv6) [119] protocol was often proposed as a useful tool to deal with the multihoming problem, since the preservation of established communications through movement is similar to the preservation of established communications through outages in multihomed sites. Both approaches need to dynamically change the addresses used for their transport-level flows without breaking them. Since the MIPv6 protocol already provides support for preserving established communications through movement, it seems worthwhile to explore whether it could also be used to provide session survivability in multihomed environments. However, MIPv6 has strong limitations, as described in [24]. The use of MIPv6 for IPv6 multihoming and its limitations are well summarised in [111], and rephrased below.

MIPv6 is the only solution that can be classified as a pure network layer approach. In this approach, each mobile node (MN) uses a preferred IP address, the Home Address (HoA), as a stable identifier, regardless of its current point of attachment to the Internet. When the MN is at the Home Network, the HoA is used both as locator and as identifier. When the MN is not at the Home Network, the HoA is used as an identifier, and the CoA is used as locator. A relaying agent (Home Agent) placed in the Home Network is used to forward packets addressed to the HoA to the current location, specified by the CoA. After each movement, the MN must inform its Home Agent of the new CoA, and optionally inform its correspondent nodes (CNs). The mapping information between the HoA and the current CoA is conveyed using Binding Update (BU) messages. When a BU message is exchanged between the MN and the Home Agent, it is possible to assume the existence of a pre-established Security Association that can be used to protect the binding information. However, when the BU message is exchanged between the MN and the CN, it is not possible to assume the existence of such Security Association. In this case, it is necessary to adopt an alternative mechanism to protect the binding information contained in the message. The selected mechanism is called Return Routability procedure and the background for its design is detailed in [144]. This procedure is illustrated on Figure 4.8.

After a movement (❶), the MN sends to the CN a Care-of Test Init (CoTI) message using its new Care-of Address, and a Home Test Init (HoTI) message using its Home Address (❷). Upon reception of those messages, the CN starts the Return Routability procedure, in order to verify that the HoA is actually assigned

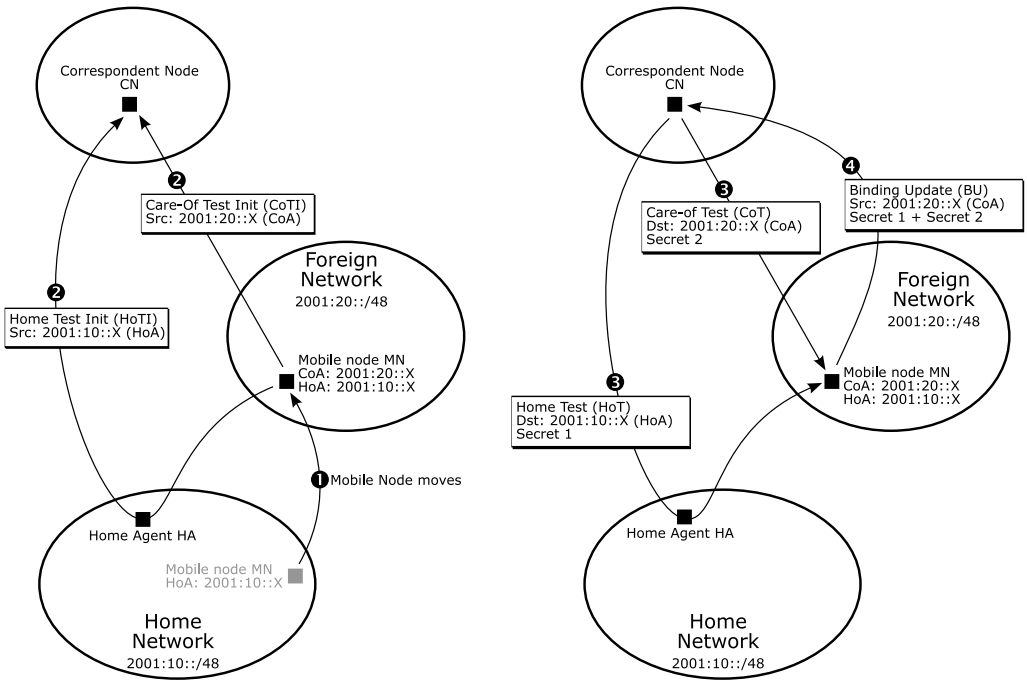


Figure 4.8. Return Routability procedure when a mobile node moves to a new Care-of Address.

to the MN, and that the MN is currently located at the CoA. The CN sends two different secrets, one to the claimed HoA using a Home Test (HoT) message, and another one to the claimed CoA, using a Care-of Test (CoT) message (❸). If the MN receives both secrets, this means that the Home Agent located at the Home Network has a trust relationship with the MN, and that it has forwarded the secret sent to the HoA, and that the MN is receiving packets sent to the CoA. In order to prove the validity of the binding between the HoA and the CoA, the MN sends to the CN a BU message that include authorisation information derived from both secrets (❹).

The lifetime of the binding that is created in the CN using authorisation information obtained through the Return Routability procedure is limited to 7 minutes [144]. After that period a new Return Routability procedure is needed to extend the binding lifetime.

The possible application of the MIPv6 protocol to the multihoming problem would be to send BU messages in order to inform the CN that an alternative address is to be used, when the node detects that the current path becomes unavailable.

In this scenario, the multihomed host adopts the MN role and the host outside the multihomed site adopts the CN role. Home Agent capabilities are not required.

When a communication is established between the multihomed host and the external host, the address used for initiating the communication is the HoA. For instance, in Figure 4.9, MN is using `2001:10:123::X` as HoA. The communication continues using this address as long as no outage occurs. If an outage occurs and the HoA becomes unreachable (❶), an alternative address of the multihomed node is used as a CoA, for instance `2001:20:123::X`. In this case, the multihomed node sends a BU message to the external host (❷), informing about the new CoA to be used for the HoA, so that the established communication can be preserved using the alternative address. However, such BU message has to be validated by the Return Routability procedure, which implies that the binding lifetime will be limited to a fixed period of no more than 7 minutes. The result is that the binding between the HoA and the new CoA will expire after this interval has elapsed, and then the HoA will be used for the communication. Since the HoA is unreachable because of the outage (❸), the communication will be interrupted. It should be noted that it is not possible to acquire new authorisation information by performing a new Return Routability procedure, because it requires communication through the HoA, which is no longer reachable (❸). Consequently, a mechanism based on the MIPv6 BU messages to convey information about alternative addresses will preserve communications only for 7 minutes.

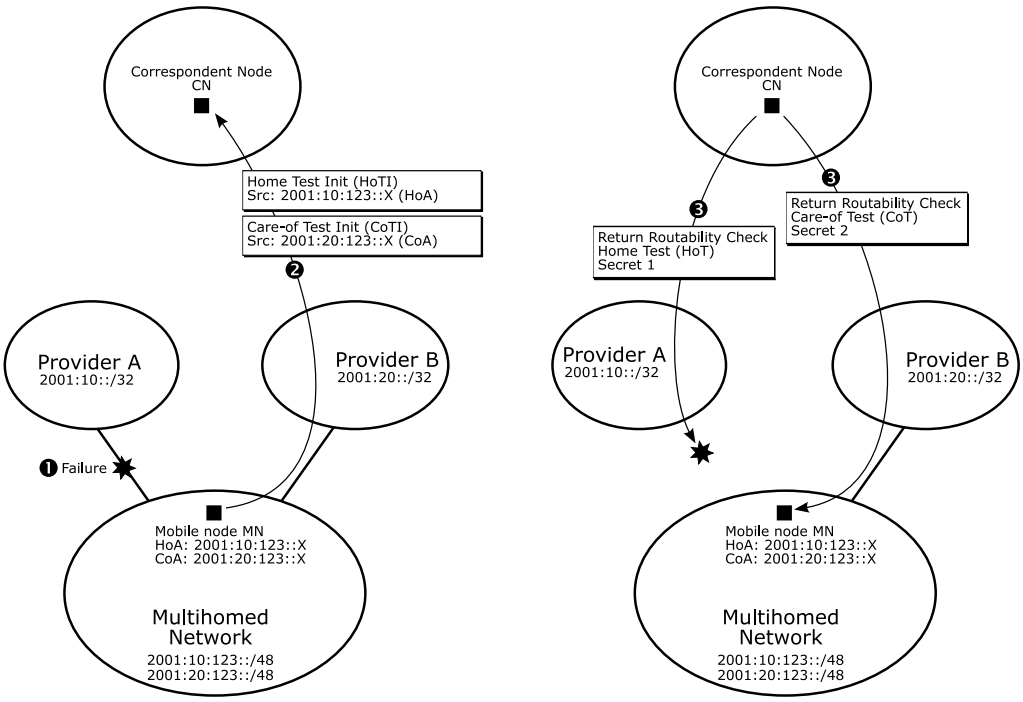


Figure 4.9. Application of MIPv6 to the multihoming problem. The Return Routability procedure does not allow to preserve communications.

Another detected limitation [24] is that once a path has failed and another one is being used, it is not possible to switch the communication to a third path. The reason is that no valid authorisation information is available, since the return routability procedure requires exchange of information using the HoA, which is in this case unavailable. The conclusion is that the use of MIPv6 is not a suitable mechanism for IPv6 multihoming.

4.4.4 Intermediate Network Layer Approaches

Intermediate Network Layer approaches suggest to support multiple addresses in an intermediate layer between the transport and the network layers. More precisely, this intermediate layer is located above the IP routing sub-layer (that performs network related functions like forwarding), but below the IP endpoint sub-layer (that performs end-to-end functions like fragmenting and IPsec). The whole protocol stack, including the new layer, is depicted in Figure 4.10 [21]. This new layer 3.5 aims at separating two entirely separate functions that are included in an IP address : the location of a node, and the identity of the node. The *locator* of a node specifies how to reach the node. It specifies a network attachment point, in term of the network topology. A locator is mostly used to forward packets in routers. The *identifier* of a node is a label at the IP layer, which is presented to upper layers. This is illustrated on Figure 4.10. An identifier is used for distinguishing one node from another and is independent from the node's attachment to the network. A node can have multiple identifiers, but each identifier is supposedly globally unique.

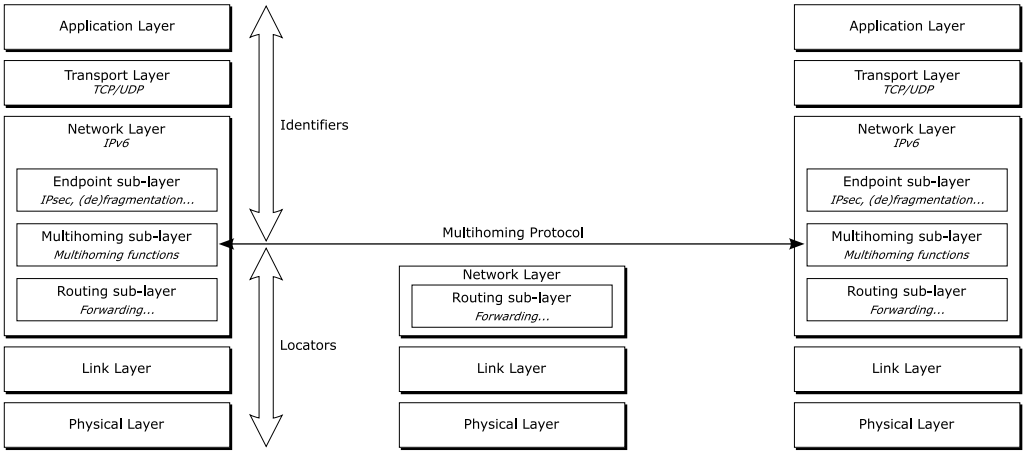


Figure 4.10. Multihoming layer in the protocol stack

Current IP addresses are both locators and identifiers, because they contain topological significance and act as a unique identifier for an interface. Identifiers form a new intermediate namespace between the two currently global namespaces

that the Internet has created : Internet Protocol (IP) addresses and Domain Name Service (DNS) names.

The separation between locators and identifiers allows applications to only use identifiers, which are mapped to locators by the intermediate layer. When a locator is no longer valid, the identifier is mapped to another locator. This change of locator is transparent for the upper layers, since applications and transport protocols bind only to the identifier, which never changes. Consequently, this approach is available for any transport protocol, including the installed base of TCP. However, endpoints using this approach require additional mechanisms to coherently map the identifiers presented to the upper layers and the IP addresses actually contained in the packets. This mapping between identifiers and locators may be vulnerable to redirection attacks if no proper protection is provided [151, 21]. Such a vulnerability is introduced when an attacker can benefit from the mapping mechanism to induce a victim to believe that she is communicating with the owner of a given identifier, while she is actually exchanging packets with a locator that does not belong to the owner of the perceived identifier. In other words, a redirection attack consists in creating a false mapping between an identifier and a locator.

Several Intermediate Network Layer approaches have been proposed, differing from the kind of identifier namespace. Some proposals suggest the creation of a new identifier namespace, of cryptographical nature or not. Others promote the use of regular IP addresses as identifiers. In addition, hybrid proposals suggest the use of cryptographical addresses as identifiers. We will now present each one of them.

New Non-Cryptographic Identifier Namespace

Location Independent Network Architecture for IPv6 (LIN6) [195] proposes the creation of a new identifier namespace, completely separated from the locator namespace. Here, an identifier and a locator are called respectively a *LIN6 generalised ID* and a *LIN6 address*. The *LIN6 generalised ID* is a 128-bit identifier used to identify the hosts only in the transport layer and the upper layers. This identifier is not used in the IP layer. The low order 64 bits of the LIN6 generalised ID consist of a globally unique *LIN6 ID*, typically derived from the MAC address of an interface. The high order 64 bits of the LIN6 generalised ID are a static predefined value, called *LIN6 prefix*. Figure 4.1 illustrates those concepts.

The *LIN6 address* is formed by substituting the constant LIN6 prefix with the actual 64-bit subnet to which the node is connected.

The mapping between the LIN6 ID and the network prefix of a LIN6 node is maintained by a *Mapping Agent* (MA). A LIN6 node registers its network prefixes with its Mapping Agent when it boots or when its network prefixes change.

Assume that a node A initiates a communication with a node B. First A and B securely register their network prefixes with MA_A and MA_B, respectively. A

	64 bits	64 bits
LIN6 generalised ID	LIN6 prefix (constant)	LIN6 ID
LIN6 address	Network prefix	LIN6 ID

Table 4.1. The LIN6 generalised ID and the LIN6 address

obtains the AAAA record of B from the name server by indicating the domain name of B. The obtained AAAA record consists of the network prefix of MA_B and the LIN6 ID of B. A generates the LIN6 generalised ID of B by concatenating the LIN6 prefix and the lower 64-bits of the AAAA record. A also generates the IPv6 address of MA_B by concatenating the upper 64-bits of the AAAA record and a well-known interface identifier (the $MAIFID$) common to all mapping agents. Finally, A obtains the network prefix of B from MA_B by indicating the LIN6 ID of B. A can now send a packet to B. To avoid impersonation, B must also convert the received LIN6 ID of A to the actual LIN6 address of A, by following the same steps.

LIN6 thus heavily relies on the DNS and on its mapping agents. It is not evident that this kind of mapping can be made reasonably secure or scalable. For instance, [139] identifies a major security threat that exists even if the binding updates are secure and that a redirection is not possible at the time of locator updates. The initial DNS request by a correspondent node returns the address of the Mapping Agent of the initiator. If this message is modified, the correspondent node can be forced to learn the wrong address of the Mapping Agent. A malicious node could easily take advantage of this to perform a redirection attack.

The *8+8 End to End Multihoming* (E2E) [153] approach is a variant of the LIN6 approach. It also suggests to use the eight most significant bytes of an IPv6 address as the locator, and the eight less significant bytes as the identifier. The DNS is used to map the identifier of a host to its hostname, using a reverse DNS look up. Then the DNS is used again to map the hostname to the corresponding 8+8 address, which is a combination of a locator of location agents of the host and ID of the host. Like for LIN6, hosts register their locations to their respective location agents, which maintain the list of current locators of the hosts. This approach suffers from the same issues as LIN6.

New Cryptographic Identifier Namespace

Several proposals suggest the creation of a new cryptographic identifier namespace. Among those, we can find the *Host Identity Protocol* (HIP) approach to multihoming [137, 136, 143, 145], and the *Strong Identify Multihoming* [147] (SIM). Both HIP and SIM implement the separation between identifiers and locators by defining a separate name space and a new layer between the network and

transport layers. This new structure insulates the transport layer protocols from the networking layer, thereby allowing transport sessions to remain unaffected even if the underlying IP addresses change. They both propose the creation of a new 128-bit identifier as the cryptographic hash of a public key associated to the endpoint. The public key is typically stored in the DNS, using a new DNS Resource Record.

The result is a secure binding between the identifier and the associated key pair. This allows the node to use the corresponding private key to sign the control packets that convey alternative address information. The trust chain is the following : the identifier used for the communication is securely bound to the key pair because it contains the hash of the public key, and the alternative address is bound to the public key through the signature. This approach effectively protects against redirection attacks. Additional protection against flooding attacks is obtained through a reachability test before actually sending packets to the alternative locators.

A main difficulty identified for this approach is the high cost of public key operations. HIP is a four-way handshake, requiring public key cryptographic operations. While such a heavy exchange makes sense for applications where hosts have a fairly long term relationship, it may be too heavy for short term transactions, such as web browsing. SIM limits the use of public key signatures only to the time of locator prefix changes for a host or when two hosts claim to use the same identifier. Another difficulty is the support for referrals and call-backs when embedding the identifiers into applications protocols [149]. This last problem is related to the extreme difficulty of building a directory service that maps identifiers to locators when the identifier namespace is flat.

New Ephemeral Identifier Namespace

The protection of the identifier is important because it represents the identity of the owner. A possible approach to avoid the security issues is to simply remove this functionality from the identifier. The *Weak Identifier Multihoming Protocol* (WIMP) [212] proposes to use ephemeral 128-bit identifiers, which are only valid as long as the flow is active. Once the flow is over, the identifier is meaningless, hence worthless. So there is no need to protect it. However, this is only possible for the identifier of the source endpoint, but not for the identifier of the destination endpoint. Indeed, in order to be able to establish a communication with a given target, a stable identifier of the target is required. The stable identifier proposed by WIMP is a hash of the Fully Qualified Domain Name (FQDN). The result is that source identifiers are worthless, so there is no need to protect them, while the destination identifiers are intrinsically bound to the FQDN of the target, providing the required protection.

During WIMP session establishment, WIMP introduces a separate header into the data packets, between the IP and TCP/UDP headers that contains information about the WIMP session. WIMP does not introduce a separate header into all IPv6 packets. Instead, once a WIMP session is established, the IPv6 FlowID is used to hold an identifier for the WIMP host-pair context associated with a given packet. The FlowIDs serve as a convenient *compression tag* without increasing the packet size.

In order to prevent redirection attacks WIMP relies on the ability to verify that the entity requesting redirection indeed holds the successor values of a hash chain and is able to combine a divided secret that is sent via parallel paths. WIMP can be divided into two exchanges : context establishment and re-addressing exchange. The former exchange establishes a state for both communication endpoints. The re-addressing exchange is used to update the locators belonging to the communicating parties.

The main difficulty with the WIMP approach is that referrals and call-backs are not supported, due to the ephemeral nature of the identifier. Another issue with this approach is that it depends on the DNS system.

Plain IPv6 Addresses as Identifiers

HIP, SIM and WIMP propose to create a new namespace, completely separated from the locators namespace. Instead, *Multihoming without IP Identifiers* (NOID) [148] suggests that the identifier of a host may be chosen from its set of regular IPv6 addresses. Since the address used as an identifier by the upper layers is not intrinsically bound to a public key, an external trusted entity is required to secure the binding between the identifier and the locators. NOID relies on the DNS infrastructure to verify the relationship between a given locator, the corresponding FQDN and the set of locators for the host. More precisely, the initiating node uses a DNS query to obtain all the available addresses of the target node. Next, the initiator selects an identifier among the obtained addresses. The remaining addresses are used as alternative locators to establish the flow. The target node obtains the set of IPv6 addresses available for the initiator by querying the reverse DNS tree. NOID makes use of flow IDs so that mapping to the correct identifier at the receiving end can be accomplished, without relying on the locators in the packet.

The main difficulty of this approach is the dependence on the DNS, especially on the proper population of the reverse DNS tree, which may be difficult to achieve for unmanaged networks.

Hybrid Approaches: Addresses with Cryptographical Features as Identifiers

Hybrid approaches attempt to achieve the benefits of the previous approaches without their limitations. They propose the use of addresses as identifiers, as in NOID, in order to properly support referrals and call-backs. However, the addresses used as identifiers are not regular addresses. Instead, they contain cryptographical information in the interface identifier part, providing a secure binding between the identifier and the alternative locators. One approach is *Multihoming using 64-bit Crypto-Based IDs* (CB64) [146], where the address of a multihomed node is a Cryptographic Generated Address (CGA) [16, 155], that contains a 64-bit hash of a public key in its interface identifier. In this case, the set of alternative locators can be authenticated through a signature with the corresponding private key.

In order to prevent redirection attacks, this protocol relies on the ability to verify, using public key cryptography as in SIM, that the entity requesting redirection indeed holds the private key where the hash of the corresponding public key hashes to the ID itself. Hence, CB64 does not use DNS for verification as in NOID. However, the cost of those public key operations involved is a limitation for CB64.

An alternative approach is based on the use of Hash Based Addresses [20]. *Multihoming L3 Shim Approach* (SHIM) [21, 150] suggests that information about the multiple prefixes is included within the addresses themselves. This is achieved by generating the interface identifiers of the addresses of a host as hashes of the available prefixes and a random number. The multiple addresses are next generated by appending the different network prefixes to the generated interface identifiers. The result is a set of addresses, called Hash Based Addresses (HBAs), that are inherently bound. A cost efficient mechanism is available to determine if two addresses belong to the same set : given the prefix set and the additional parameters used to generate the HBA, a single hash operation is enough to verify if an HBA belongs to a given HBA set. No public key operations are involved in the verification process, as long as the prefix set is stable. Protection against flooding is obtained through a reachability test that verifies the willingness of the target to receive traffic through the alternative locators. An incremental approach to IPv6 multihoming, that uses the SHIM approach, is described in [22]. We will futher detail the SHIM approach in Section 4.6, as this is the solution chosen by the IETF.

4.5 Looking Towards the Future of IPv6 Multihoming

During the last years, the IETF has made several explicit or implicit architectural decisions regarding IPv6 multihoming. The main decision is to go down the path

of developing the Host-Centric approaches. The IETF made this choice in 2003. It can be explained by looking at Table 4.2.

This table summarises the mechanisms, advantages and drawbacks of the three approaches to IPv6 Multihoming : Routing, Middle-Box, and Host-Centric approaches. By taking a look at the features provided by each approach, we can observe that Routing approaches either do not allow route aggregation, or cannot provide complete fault-tolerance. Moreover, Routing approaches typically require running BGP, and do not provide Site-ISP independence. Middle-Box approaches provide many required features, but basically fail to preserve the End-to-End Principle [38, 43, 176]. Moreover, IPv6 multihoming with NAT does not preserve the transport-layer flows in case of failure, and MHAP or MHTP shows security concerns. Hence, the Middle-Box approach is not considered as a suitable solution for IPv6 site multihoming, although it may still be interesting as a transition mechanism, or for small residential networks.

In Table 4.2, Host-Centric approaches appear to be the most promising IPv6 multihoming architectures, provided that functionalities are added to end hosts. This is considered acceptable as many hosts already require to be updated for the new IPv6 Internet, and because Routing approaches for IPv6 multihoming can provide transition mechanisms. As a consequence of this decision, it can be expected that the use of several IPv6 addresses on each end-host will become widely prevalent on the IPv6 Internet. This is a drastic architectural change compared to today's IPv4 Internet, where hosts are typically identified by a single IPv4 address.

Table 4.3 summarises the mechanisms, advantages and drawbacks of the main Host-Centric solutions. The Host-Centric architecture requires that end hosts be able to switch between IPv6 addresses without breaking transport-level flows. Many Host-Centric solutions were proposed. Among all proposed mechanisms, the IETF promotes Intermediate Network Layer approaches, as they can provide multihoming support to any transport level protocol such as TCP and UDP. Moreover, Transport Layer approaches appear to have some difficulties for protecting against *Man-In-The-Middle* (MITM) attacks. Among the Intermediate Network Layer approaches, SIM, HIP, WIMP, LIN6 and E2E use a new identifier namespace, which creates concerns for supporting referrals and call-backs when embedding the identifiers into applications protocols [149]. A look at this table shows that SHIM and CB64 are the most promising solutions, due to their security features, and their low requirements on the infrastructure.

SHIM can be seen as a superset of CB64, since SHIM supports multiple security mechanisms. In particular, SHIM can provide a more efficient support when the set of prefixes allocated to a multihomed site is stable. For this reason, the IETF has decided by the end of 2004 to foster the SHIM approach.

		IPv6 Multihoming Solution					
		IPv6 with BGP	Provider Cooperation	RFC 3178	IPv6 with NAT	MHAP, MHTP	Host-Centric*
		Routing approaches			*Middle-Box approaches*		*Host-Centric approaches*
Mechanism	IP version support	IPv4+IPv6	IPv4+IPv6	IPv4+IPv6	IPv4	IPv6	IPv6
	Fault-tolerance	R	R	R+TUN	MB	MB	H
	Traffic engineering capability	R	R	R	MB	MB	H
	Route aggregation	N/A	SA	MP	MP	MP	MP
	Complete Independence	N/A	N/A	N/A	MP	MP	MP
Feature	Route aggregation		✓	✓	✓	✓	✓
	Traffic engineering capability	✓	✓	✓	✓	✓	✓
	Link fault tolerance	✓	✓	✓	✓	✓	✓
	ISP fault tolerance	✓			✓	✓	✓
	Transport-layer survivability	✓	✓	✓		✓	✓
	Stable configuration in case of long term failure	✓			✓	✓	✓
	Site - ISP Independence				✓	✓	✓
	ISP - ISP Independence	✓		✓	✓	✓	✓
	End-to-End Principle preserved	✓	✓	✓			✓
	No modification to hosts	✓	✓	✓	✓	✓	

R = Based on routing system, H = Based on host capability, MP = Based on the use of multiple prefixes, MB = Based on the use of a middle-box, SA = Based on selective route announcements, TUN = Based on the use of tunnels, N/A = Not available, ✓= provides this feature. * Groups many Host-Centric solutions.

Table 4.2. Overview of IPv6 multihoming approaches

		Transport Layer Approaches			*Intermediate Network Layer Approaches*							
		SCTP	TCP-MH	DCCP	NOID	CB64	SHIM	SIM	HIP	WIMP	LIN6	E2E
Mechanism	IP version support	4+6	4+6	4+6	4+6	6	6	6	4+6	6	6	6
	Layer	4	4	4	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5
	ULID namespace	IP	IP	IP	IP	IP+	IP+	ID	ID	ID	ID	ID
	Layer-4 applicability	TCP+UDP	TCP	UDP	ANY	ANY	ANY	ANY	ANY	ANY	ANY	ANY
	Secure change of locators	PK/key	Cookie	PK/key	DNS	PK	HBA/PK	PK	PK	DNS Ephemeral	MA	MA
Feature	DNS independence	✓	✓	✓		✓	✓	✓	✓			
	Allows referrals	✓	✓	✓	✓	✓	✓					
	No new identifier namespace	✓	✓	✓	✓	✓	✓					
	Protection against MITM					✓	✓	✓	✓			
	Protection against redirection	✓	✓	✓	✓	✓	✓	✓	✓	✓		

IP = plain IP address space, ID = new IP address space, IP+ = plain IP with modified interface identifier PK = uses Public Key cryptography, key = uses symmetric key operations, DNS = uses DNS, MA = uses a Mapping Agent, Ephemeral = security provided through the use of ephemeral identifiers, HBA = uses Hash Based Addresses. ✓= provides this feature.

Table 4.3. Host-Centric approaches: overview of mechanisms for the provision of IPv6 multihoming

4.6 The SHIM Approach

As explained in the previous sections, the *Multihoming L3 Shim Approach* (SHIM) [21, 150] proposes the use of an intermediate layer located above the IP routing sub-layer, but below the IP endpoint sub-layer. This approach uses, at least initially, routable IP locators as the identifiers visible above the SHIM layer. This ensures that all upper layer protocols can operate unmodified in a multihoming environment, as they always see a stable IPv6 address. The locator used in the address fields of the packets can change over time in response to failures affecting the original locator. This is illustrated in Figure 4.11. In this figure, Host X has addresses IP1(X) and IP2(X), and Host Y has addresses IP1(Y), IP2(Y) and IP3(Y). The stable source and destination addresses seen by the transport and upper layers are IP1(X) and IP2(Y), while the actual addresses used in the packets are IP2(X) and IP3(Y). The mapping between the stable and actual addresses is done by the new SHIM layer.

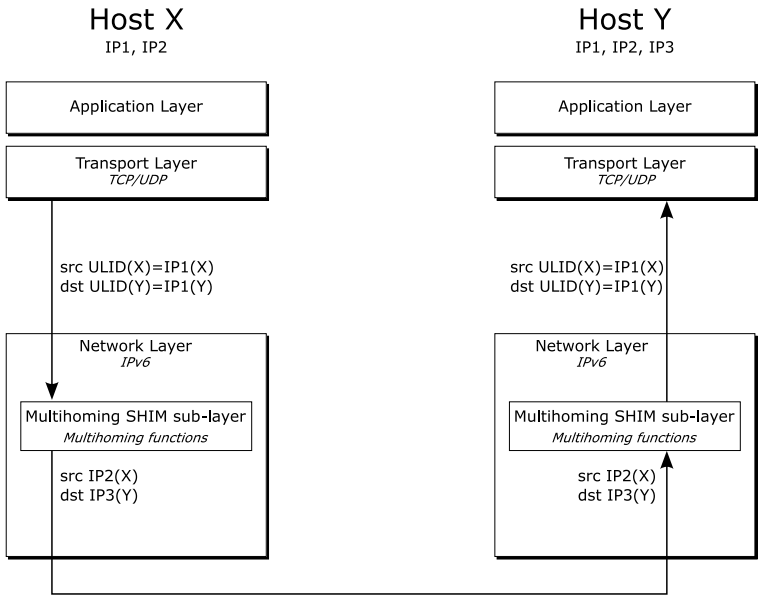


Figure 4.11. Mapping with changed locators

The SHIM approach is best explained by describing the sequence of events that occur when a multihomed capable Host X starts talking to another multihomed capable Host Y.

When Host X wants to initiate a communication with Host Y, it first typically issues a DNS request for a name of Host Y. It receives in the DNS response some or all the addresses assigned to Host Y. Host X uses the default address selection algorithm [73] to select both the source and destination addresses that will be used for its outgoing packets. These initial source and destination addresses will

also be used as endpoint identifier for all transport and application layers on Host X and Host Y. So far, no multihoming protocol exchange is needed.

At some point in time, one of the host, e.g. Host X, may request to take advantage of multihoming, e.g. in order to obtain a higher reliability. Hence, it initiates the SHIM protocol exchange. This exchange will fail if Host Y does not support the SHIM protocol. If it succeeds, Hosts X and Y will exchange their respective set of available addresses. In order to prevent redirection attacks, Host X uses the HBA mechanism [20] described in Section 4.4.4 to ensure that the additional addresses given by Host Y are compatible with the initial address of Host Y, i.e. that all addresses of Y belong to the same HBA set. At this point in time it is possible for both hosts to change to a different address in the set.

Suppose that a failure of a provider prevents Host Y from receiving packets from Host X. A timeout is raised on Host Y, and a reachability test packet is sent to Host X to check if the path is still available. If no answer is received, Host Y initiates an address pair exploration procedure by sending several test packets to Host X with different source and destination addresses, until a reply packet is received. When Host X receives packets from Host Y with new addresses, it also checks the currently used addresses, and switches to a new address pair if needed. It is not required that both Host X and Host Y use the same address pair for communicating. This address exploration procedure is explained in [13, 207], and is still being discussed at the IETF.

4.7 Conclusion

Constraints for the scalability of the Interdomain routing system have led the IETF and the research community to propose drastic changes in how site multihoming shall be achieved in an IPv6 Internet. This chapter has reviewed and compared all major architectures that were proposed for IPv6 site multihoming, their mechanisms, advantages and concerns. Host-Centric multihoming, the IPv6 multihoming approach promoted by the IETF, affects the fundamental multihoming mechanism, as well as intradomain routing, traffic engineering and hosts within the multihomed site. In particular, the use of multiple IPv6 addresses per end hosts introduces a major architectural change compared to today's IPv4 Internet, where hosts are typically identified by a single IPv4 address. Fortunately, we will show in the next part of this thesis that these changes also bring wide opportunities to develop multihoming for small and residential networks, as well as the quality of service in the future Internet.

Part II

Traffic Engineering and Host-Centric IPv6 Multihoming

Chapter 5

Leveraging Internet Path Diversity and Network Performance with IPv6 Multihoming

5.1 Introduction

Chapter 4 reviewed all proposed IPv6 multihoming solutions. Three representative approaches were described : *Routing*, *Middle-Box*, and *Host-Centric*. Middlebox approaches have many concerns, and do not appear to be the best solutions for IPv6 multihoming. Routing and Host-Centric approaches have a fundamental difference. In Routing approaches, hosts use a single IPv6 address, and transport-layer survivability is provided by routing mechanisms. In Host-Centric approaches, hosts use several IPv6 addresses. They use them interchangeably during the lifetime of a flow in order to survive outages affecting any of those addresses.

Both Routing and Host-Centric approaches allow route aggregation. Routing approaches are easier to deploy, as they do not require the hosts be updated. Unfortunately, they provide fault-tolerance only for the links with the providers.

On the other hand, Host-Centric approaches provide complete fault-tolerance, but require to modify the hosts.

Without further study, one would perhaps prefer Routing approaches because they are easier to deploy. However, these two approaches have never been really compared in terms of fault-tolerance and network performance. Therefore, this chapter aims at answering the following questions :

> *Do Host-Centric approaches bring significant advantages over Routing approaches, in terms of fault-tolerance and network performance ?*
>
> *Does the use of multiple provider-dependent aggregatable (PA) IPv6 prefixes per site bring any other advantage than route aggregation ?*

This chapter evaluates the benefits yielded by *Host-Centric* approaches for IPv6 multihoming, over *Routing* approaches like traditional multihoming with BGP. The two approaches are compared in terms of resilience and traffic engineering possibilities, through the notion of *AS path diversity*.

The Internet AS path diversity is the number of distinct AS paths that exist between an AS and the rest of the Internet. Section 5.2 explains how the path diversity directly impacts the resilience to failure of a site, and also its ability to support quality of service and traffic engineering. For example, a site for which all paths merge in a single AS in the Internet is dependent on the performance of this particular AS. Having a wide variety of paths to join and to be joined by other ASes ensures a better resilience to failures. It could also allow to obtain better network latency by bypassing congested networks in the Internet.

Section 5.3 compares the diversities when using multiple PA prefixes and when using a single PI prefix. It shows how stub ASes that use multiple PA prefixes can exploit paths that are otherwise unavailable. In other words, it explains how the use of such prefixes increases the number of concurrent paths, i.e. the AS-level path diversity. For instance, it shows that a dual-homed stub AS that uses several prefixes has a better Internet path diversity than any multihomed stub AS that uses a single PI prefix, whatever its number of providers. In addition, section 5.3 shows that the gain in path diversity does not depend much on the topology, and that this result should also hold for the actual Internet.

Section 5.4 focuses on the delays of interdomain paths between pairs of multihomed stub ASes. It shows that lower delays can often be found among the new paths introduced by the use of multiple PA prefixes. Simulations suggest that a delay improvement is observed for approximately 60% of the stub-stub AS pairs, and that the delay improvement could be higher in the actual Internet.

Finally, the related work is presented in section 5.5.

The use of multiple addresses is natural in an IPv6 multihoming environment. However, it is not impossible to use the same multihoming technique in IPv4. For example, this can be done by delegating two IPv4 prefixes to a site, and by using

network address translation (NAT) or tunnels if necessary. Unfortunately, due to the current lack of IPv4 addresses, the need to delegate several IPv4 prefixes to a single multihomed site makes this solution less attractive.

To our knowledge, this is the first study on the impact of Host-Centric IPv6 multihoming on interdomain path diversity. It has been published in [69].

5.2 Simulation Setup

This section focuses on the method used to measure the AS-level path diversity. The path diversity metric that we use is first presented in section 5.2.1. Next, section 5.2.2 details how a BGP simulator is used to extract the AS-level paths from a given topology. Finally, section 5.2.3 explains how the use of several prefixes per sites can improve the AS-level path diversity.

5.2.1 Measuring the Path Diversity

In this study, we are interested in measuring the path diversity between any two stub ASes, at the level of autonomous systems. A path diversity metric is usually computed as the number of link-disjoint or node-disjoint paths between a couple of nodes [193, 194]. Obviously, the focus is on nodes that are at least multihomed, since there is only one BGP path between two single-homed stub ASes. The AS-level path diversity depends on the network infrastructure, but also on routing policies used by BGP along the path. For example, the traditional metric computation, based on the number of disjoint paths, gives a number between 0 and 2 for any two dual-homed ASes. Indeed, it exists at most two disjoint paths announced by BGP towards a dual-homed AS. In this study, we would like to differentiate two paths that join early from two paths that join only at the destination AS. The traditional diversity metrics are not appropriate in this case since they cannot distinguish these paths.

In this section, we propose the use of another path diversity metric, derived from the notion of *Availability* of a network system [185]. The availability can be expressed as the percentage of time that a network system, component, or application is available for a user. It depends on the availability A_i of each individual component plus the system organisation [185]. Two components A_1 and A_2 can be connected in series or in parallel. In the first case, the availability of the combination is $A = A_1 \times A_2$. In the second case, the availability is given by $A = 1 - (1 - A_1) \times (1 - A_2) = A_1 + A_2 - A_1 \times A_2$.

We propose to reuse this notion to measure the path diversity that exists between two stub ASes. We explain how to apply this notion, and why this metric can take into account the length of the paths and how much they overlap.

The path diversity metric proposed for this study is defined, from a source S to a destination D, as follows. Let $P_1, P_2, ..., P_n$ be the n providers of S. Consider the tree of all paths starting from providers P_i of S to destination D, for $i = 1, ..., n$. This tree represents all the BGP paths towards D that are advertised by the providers P_i to S. The path diversity metric should have the following properties. First, it should favour path trees that have many branches, because each branch is an additional opportunity to bypass problematic links or nodes. Next, it should prefer trees where paths join lately near the destination node over trees where paths merge near the source node. Finally, the metric should prefer trees with short branches over trees with long branches.

The path diversity for this tree is proportional to the availability of a path between the source and destination nodes, i.e. the probability that such a communication path exists between the nodes. Since the goal is only to compare the diversity of different path trees, we do not attribute a specific meaning to the absolute path diversity measure. Consequently, we can assign an A_i, i.e. an initial diversity, of 0.5 to each link in the tree. This number is chosen in order to best distribute the values of the path diversity metric in the range $[0, 1]$.

An algorithm for computing the path diversity metric is given by Alg. 1. The diversity is computed recursively link by link, from the leaves to the root. It always returns a number between 0 and 1. At each computation step, two cases are considered, to which all other cases can be reduced. Either two links are in sequence, or the links join in parallel at the same node.

In the first case, two links with diversity d_1 and d_2 in sequence can be merged into a single link with a combined diversity $d_{1,2} = d_1 \cdot d_2$, as illustrated on Figure 5.1 (left).

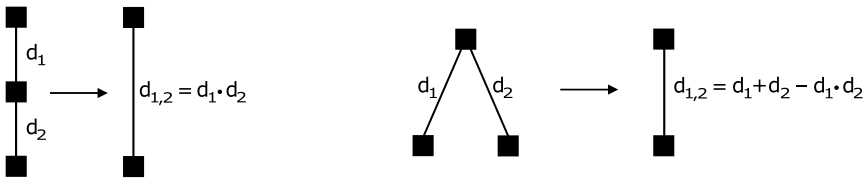


Figure 5.1. Merging two sequential (left) or parallel (right) paths into a single path

The combined diversity $d_{1,2}$ is a number in $[0, 1]$ lower than both d_1 and d_2, so that the metric favours short paths over longer ones. This step also implicitly gives a higher importance to the path diversity that exists near the root of the tree. This behaviour ensures that the metric prefers trees where paths join near the destination node over trees where paths merge near the source node.

In the second case, when a link with a diversity d_1 and another link with a diversity d_2 join, the two links can be merged into a single link with a combined

diversity $d_{1,2}$, computed as $d_{1,2} = d_1 + d_2 - d_1 \cdot d_2$, as illustrated on Figure 5.1 (right).

The diversity $d_{1,2}$ is a number greater than both d_1 and d_2, which corresponds adequately to an improvement in terms of diversity. Moreover, because both d_1 and d_2 are always between 0 and 1, the combined diversity is also a number between 0 and 1. A recursive algorithm to compute d for a whole tree is presented in Alg. 1.

Alg. 1. Computing Diversity Metric

```
Diversity(root)
{
    d = 0 ;
    if ( Children(root) == ∅ )
        return 1 ;

    foreach child ∈ Children(root) {
        d_child = 0.5 × Diversity(child) ;
        d = d + d_child − d × d_child ;
    }

    return d ;
}
```

A diversity of 1 is impossible to reach in practice, since it would mean that an infinity of disjoint paths exist between the source and the destination nodes. A null diversity means that there is no path between the two nodes, or that the path has an infinite length. These two extreme cases are illustrated on Figure 5.2.

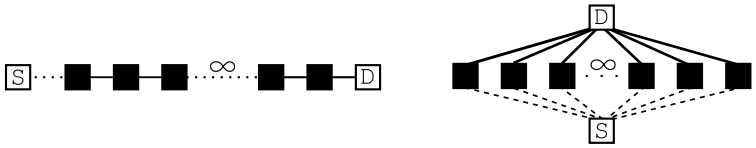


Figure 5.2. A null diversity (left) and a diversity of 1 (right)

The path diversity metric d gives higher preference to shorter AS paths. Sometimes, this bias in favour of short paths is undesirable. For instance, it is not obvious that a long AS path comprising 1 or 2 routers in each AS is not as good as a single hop AS path where the AS has dozens of intermediate routers. In this case, the tree can be processed to reduce long branches as illustrated on Figure 5.3. Next, the unbiased diversity is given by applying Alg. 1 with $root$ = D. This alternate metric is noted d'. Two particular cases for the computation of d' must

be handled separately for this algorithm. The first is when the tree has a single node. The second is when the tree is made of a single branch. In both cases, the diversity d' is set to null.

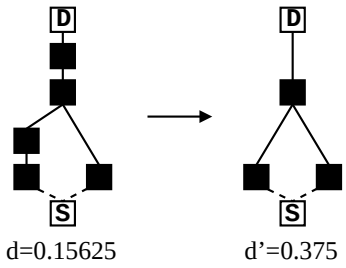


Figure 5.3. Removing the bias in favour of short paths

Examples of values for d and d' are shown in Figure 5.4. In Figures 5.4(a) and (b), the source S is dual-homed and the destination D is single-homed. d in Figure 5.4(b) is better than d in Figure 5.4(a) because the tree (b) contains a path with 3 hops and a path with 2 hops, while the tree (a) contains 2 paths of 3 hops each. However, d' is identical for trees (a) and (b) when considering only the number of branches. Both d and d' for trees (c) and (e) show a better path diversity than trees (a) and (b) because they contain 3 disjoint paths instead of 2. However, d in tree (d) has a slightly better diversity than d in tree (c), even if (c) has 3 disjoint paths while (d) has only 2. The reason is that the 2 disjoint paths of (d) have 2 sub-branches each, while the diversity of the 3 disjoint paths of (c) is mitigated due to their lengths. However, when considering d', this bias in favour of short paths is removed, and both trees (c) and (e) are better than tree (d).

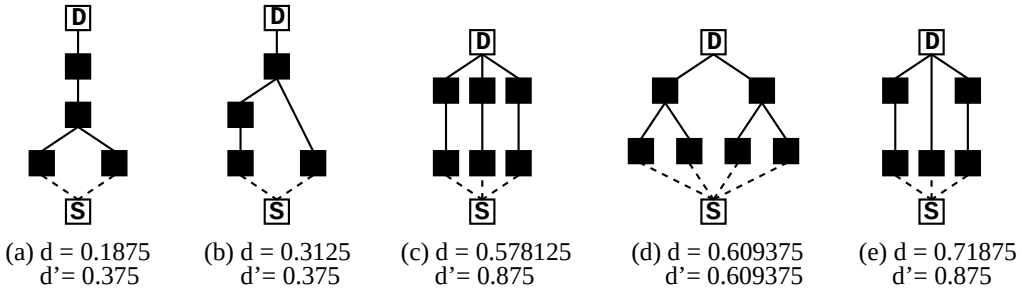


Figure 5.4. Path diversity metric examples

Other metrics exist to compute the path diversity [4, 193, 194]. In Table 5.1, the first and second metrics are those presented in this work. The third is a metric

used in [4] to quantify the diversity of network paths that multihoming provides. This metric considers the tree of paths from the source to the destination node. The expected fraction of links that are shared by two or more paths in the tree is given by $\frac{P-E}{E}$ where P denotes the sum of the hop-counts of the individual paths from the source to the destination, and E is the total number of edges in the tree. Thus $1 - \frac{P-E}{E}$ could be used to estimate the fraction of paths that are non overlapping, i.e. to estimate path diversity. The last four metrics are used in [193, 194] to characterise the path diversity of complete ISP topologies. The fourth and fifth metrics calculate respectively the number of node-disjoint and link-disjoint paths. A partially node- or link-disjoint path is defined as one for which there are respectively some nodes or links that appear in more than one path. These last four metrics were adapted to compute the interdomain AS-level path diversity.

Values of these metrics for the examples of Figure 5.4 are shown in Table 5.1. For all these metrics, a higher value suggests a better diversity.

	Metric	*(a)*	*(b)*	*(c)*	*(d)*	*(e)*
1.	Our metric d	0.19	0.31	0.58	0.61	0.72
2.	Our second metric d'	0.375	0.375	0.875	0.61	0.875
3.	$(1 - \frac{P-E}{E})$ [4]	0.5	0.75	1	0.67	1
4.	Node-disjoint paths	1	1	3	2	3
5.	Link-disjoint paths	1	1	3	2	3
6.	Partially node-disjoint paths	2	2	3	4	3
7.	Partially link-disjoint paths	2	2	3	4	3

Table 5.1. Path diversity values computed by different metrics

For our study, the third metric has an undesirable bias in favour of longer paths. Moreover, it cannot differentiate a few cases such as those illustrated in Figure 5.4(c) and Figure 5.4(e). Finally, this third metric is unable to correctly compare some cases. For example, when comparing trees in Figure 5.4(b) and Figure 5.4(d), the metric evaluates that tree 5.4(b) has a better diversity than tree 5.4(d). This is obviously wrong. The 4th, 5th, 6th and 7th metrics are not fine-grain enough for our analysis. For example, none of them is able to distinguish cases 5.4(a) and 5.4(b), or cases 5.4(c) and 5.4(e). Only the two first metrics d and d' are able to provide precise and fine-grained measures of path diversity between two nodes. We will use d instead of d' throughout this document, unless stated otherwise.

5.2.2 Simulation of BGP

IPv6 multihoming with multiple provider-dependent aggregatable prefixes is currently not deployed. As a consequence, we set up simulations made on various

Internet-like topologies instead of conducting experiments on the current IPv4 Internet. We detail in this section how the AS-level paths can be extracted from a given topology.

In this study, we focus on the paths announced by BGP between each pair of stub ASes in a given topology. These paths depend on the topology, and also on the commercial relationships between ASes. As explained in Chapter 1, the commercial agreements between two ASes are usually classified into customer-to-provider and peer-to-peer relationships [86, 189]. These relationships are either inferred [86, 189] from given BGP routing tables, or directly provided by the topology description. We compute, for each AS, the BGP configuration that corresponds to its commercial relationships with the other ASes. The BGP export policies basically define that an AS announces all the routes to its customers, but announces to its peers and providers only the internal routes and the routes of its customers. Additionally, the configuration defines that an AS prefers routes received from a customer, then routes received from a peer, and finally routes received from a provider [86, 189]. These filters ensure that an AS path will never contain a customer-to-provider or peer-to-peer edge after traversing a provider-to-customer or peer-to-peer edge. This property is known as the the *valley-free* property [86]. One prefix per AS is announced in our simulations.

The paths for a given topology are obtained by simulating the BGP route distribution over the whole topology. For this purpose, a dedicated BGP simulator is used, named C-BGP [165, 162, 164, 204]. C-BGP supports import and export filters, and uses the full BGP decision process. The last tie-breaking rule used by C-BGP to choose between two equivalent routes is to prefer the route learned from the router with the lowest router IP address, as recommended by [170].

As soon as all the routes have been distributed and BGP has converged, we perform traceroute measurements on the simulated topology and deduce the AS paths.

5.2.3 Impact of PI and PA Prefixes on Available AS Paths

This section introduces how the use of several prefixes per sites can improve the AS-level path diversity. Figure 5.5 shows an AS-level interdomain topology with shared-cost peerings and customer-provider relationships. An arrow labelled with "€" from AS x to AS y means that x is a customer of y. A link labelled with "=" means that the ASes have a shared-cost peering relationship [86]. In this figure, both S and D are multihomed. S has two providers, E and F. D has also two providers, A and B. B and C have a peer-peer relationship. E and F both have B and C as providers.

In IPv4, a stub AS typically becomes multihomed by announcing its provider-independent prefix to all its providers. This PI prefix is next propagated by BGP routers all over the Internet. This propagation is illustrated in Figure 5.6. In this figure, the stub AS D announces its prefix $P1$ to its two providers A and

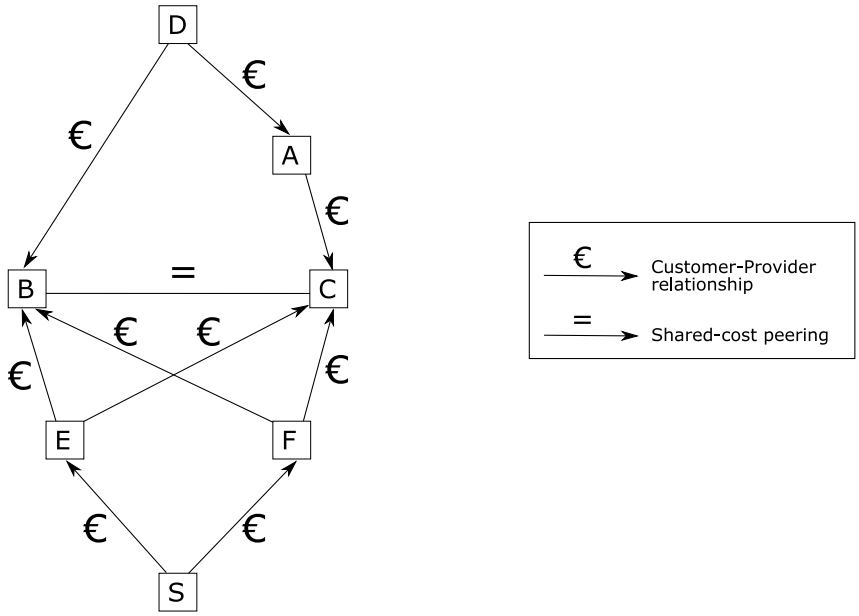


Figure 5.5. AS Topology with business relationships between ASes

B. A advertises prefix $P1$ to C, which in turn advertises it to E and F. In the same way, B advertises prefix $P1$ to E and F. At this point, both E and F have received two routes towards destination D : a route through B and another route through C and A. Both E and F will select the route through B, because it is one AS hop shorter than the route through C and A. Hence, E will announce to its client S that the best route towards prefix $P1$ is the $E : B : D$ route, while F will announce that the best route to prefix $P1$ is the $F : B : D$ route. Thus S will receive two routes towards D, i.e. the $E : B : D$ route from its provider E, and the $F : B : D$ from its provider F. More generally, a stub AS that is multihomed by using a single prefix will receive one route per provider, for each destination prefix. In particular, if AS S is single-homed, it will only receive a single route from its provider to reach the dual-homed AS D. The two BGP routes towards D ($S : E : B : D$ and $S : F : B : D$) that are learned by S in the IPv4 scenario are illustrated on Figure 5.8 (left). In this example, we see that the two routes join early at AS B.

When stub ASes use IPv6 multihoming with multiple PA prefixes, additional routes exist.

Suppose again that a host inside AS S wants to reach a host inside D in Figure 5.5. With IPv6, each provider assigns one prefix to the multihomed site. In Figure 5.7, A assigns prefix $P2$ to D, while B assigns prefix $P1$ to D. To ensure the scalability of the BGP routing tables in the Internet, AS D announces prefix $P1$ only to AS B, and prefix $P2$ only to AS A. This prefix is then propagated in the Internet as illustrated by Figure 5.7. In particular, AS E will get one BGP

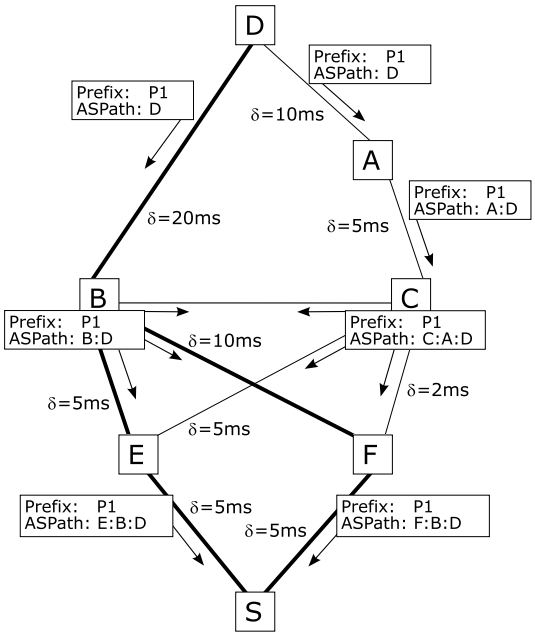


Figure 5.6. BGP prefix advertisement with IPv4 multihoming

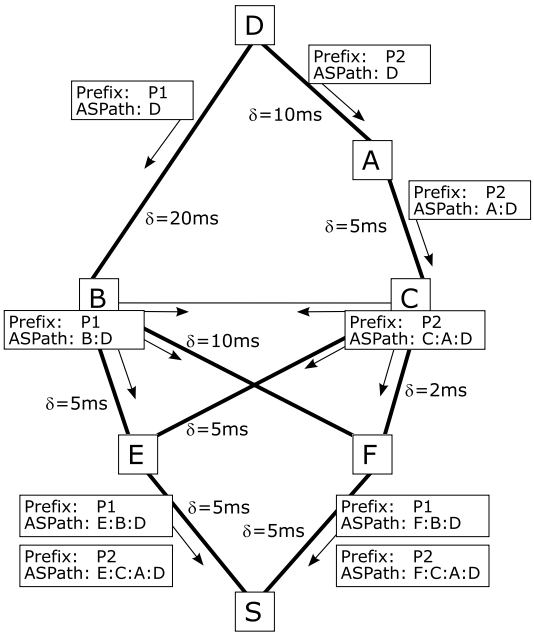


Figure 5.7. BGP prefix advertisement with IPv6 multihoming

route towards prefix $P1$, i.e. route $E : B : D$, and one BGP route towards prefix $P2$, i.e. route $E : C : A : D$. In the same manner, AS F will get two routes, the $F : B : D$ route for prefix $P1$, and the $F : C : A : D$ route for prefix $P2$. Neither E nor F have to select a best route since the routes received are for different prefixes. Hence, S will receive both $E : B : D$ and $E : C : A : D$ routes from provider E, and both $F : B : D$ and $F : C : A : D$ routes from provider F. Thus, a total of four routes are received by S, two routes for destination prefix $P1$ and two routes for destination prefix $P2$. These four BGP routes towards D, that are learned by S in the IPv6 scenario are illustrated on Figure 5.8. The path actually used to reach D depends on which destination address of D is used. For instance, if a host uses a destination address that belongs to prefix $P1$ (resp. $P2$), then its traffic will flow through B (resp. A). Moreover, the path used to reach D also depends on which source address is used. As said in Chapter 3 section 3.4.2, most providers refuse to convey packets with source addresses outside their address range, for security reasons. For example, E refuses to forward a packet with a source address belonging to F. Therefore, depending on the source address selected, the upstream provider used to convey the packet is either E or F.

In summary, two routes are available when a stub AS is multihomed by using a single prefix, while four routes are received by a stub AS that uses multiple PA prefixes. The corresponding path trees are illustrated by Figure 5.8. When multihoming by announcing a single prefix, the resulting path diversity computed

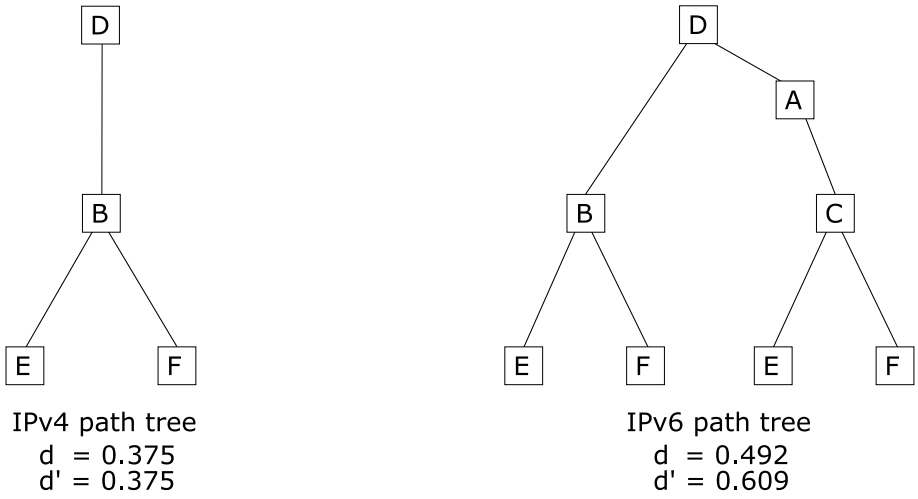


Figure 5.8. Values of d and d' for the resulting IPv4 and IPv6 path trees

by the metric d is 0.375. When multihoming by announcing several PA prefixes, the diversity d is about 0.492. Values for the alternate path diversity metric d' are also indicated.

Note that, if the delays are those indicated by Figure 5.6 and Figure 5.7, and if S uses a PI prefix to be multihomed, then S is unable to use the path with the lowest delay, i.e. 22ms for the $S : F : C : A : D$ route. This is because BGP will only advertise the $S : E : B : D$ and $S : F : B : D$ routes to S, with delays of 30ms and 35ms respectively. This will be discussed in section 5.4.

5.3 Leveraging Internet Path Diversity with Multiple prefixes

Section 5.2.3 has shown that stub ASes that use multiple PA prefixes can exploit paths that are otherwise unavailable. In other words, the use of multiple PA prefixes increases the number of paths available, i.e. the Internet path diversity. In this section, we use simulations and our diversity metric d to quantify the Internet path diversity that exists when a multihomed stub AS uses either multiple PA prefixes or a single PI prefix.

We first detail an inferred AS-level Internet topology in section 5.3.1. We present and discuss the results of simulations made on this topology in sections 5.3.2 and 5.3.3. Next, we explain in section 5.3.4 how we generate various Internet-like topologies, and use them in section 5.3.5 in order to evaluate the impact of the topology on the AS-level path diversity. Finally, in section 5.3.6, we use a router-level generated topology to investigate the impact of hot-potato routing on the path diversity.

5.3.1 Inferred Internet Topology

IPv6 multihoming with multiple PA prefixes is currently not deployed. As a consequence, our evaluations are performed on synthetic Internet topologies, instead of conducting measurement experiments on the actual IPv4 Internet. No accurate model of the global Internet currently exists. Modelling the Internet, even only at the AS level, remains an active research topic [213].

The first topology we use is an inferred AS-level Internet topology. This topology is inferred from several BGP routing tables using the method developed by Subramanian et al., and described in [189]. The topology dates from January 2003, and contains 14695 ASes and about 30000 links. It is publicly available at [190].

5.3.2 Simulation Results

Figure 5.9 presents the AS-level path diversity d when stub ASes are multihomed by announcing a single PI prefix, in the inferred AS-level Internet topology. Figure 5.10 shows the path diversity d when all stub ASes use IPv6 multihoming with multiple PA prefixes, in the same inferred topology.

The figures show $p(x)$: the percentage of couples *(source AS, destination AS)* having an AS-level path diversity greater than x. The results are classified according to the number of providers of the destination stub AS. This number is indicated beside each curve. In Figure 5.9, we see for example that only 12% of single-homed stub ASes using a single PI prefix have a diversity higher than 0.2. This percentage raises to a bit more than 20% for dual-homed stub ASes. It reaches about 55% when the stub has 25 providers. To give an idea, an example of a tree with a diversity of about 0.2 is given in Figure 5.4(a). In Figure 5.10, about 50% dual-homed stub ASes that uses multiple PA prefixes have a quality better than 0.2. In the studied topology, 90% of the stub ASes with more than 4 providers have a diversity higher than 0.2, when they use multiple PA prefixes.

Figure 5.11 and Figure 5.12 present the AS-level path diversity computed by the alternate path diversity metric d', which removes the bias in favour of short paths. Figure 5.11 shows that only 60% of couples *(source AS, destination AS)* have a a non null diversity d'. This means that 60% of pairs of ASes have at least two paths between them, while 40% have only a single path. This is easily explained by the fact 40% of stub ASes are single-homed in the considered topology. Indeed, a single-homed stub AS receives only a single BGP route towards a given destination prefix, whatever the number of providers of the destination AS. It can further be observed that, for the vast majority of stub ASes, the path diversity d' is below or equal to 0.75. This means that stub ASes have rarely more than two completely disjoint paths towards a destination stub AS, even if the destination AS has 25 providers. This is no longer the case when multiple PA prefixes are used, as illustrated by Figure 5.12.

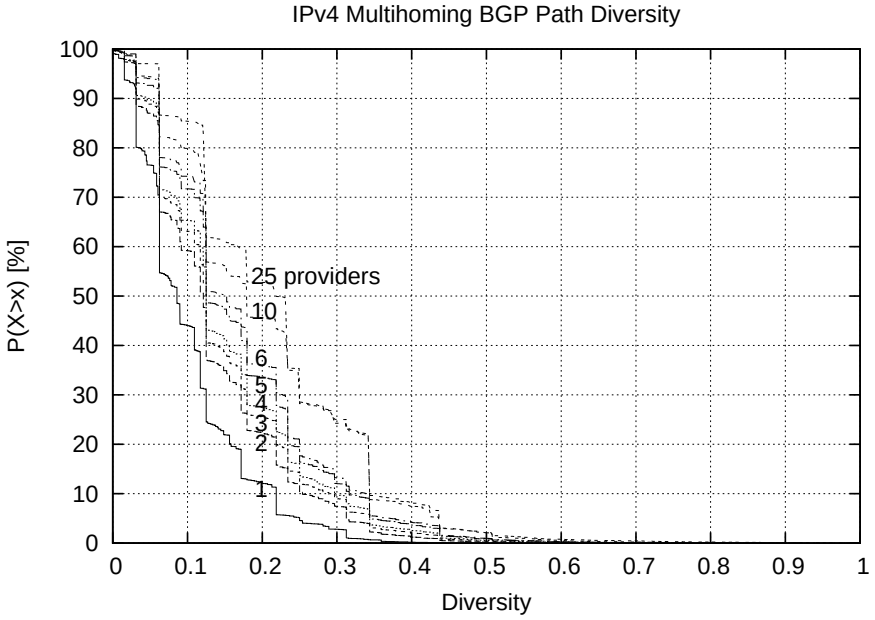


Figure 5.9. AS-level path diversity d for the inferred Internet topology, using multihoming with a single PI prefix

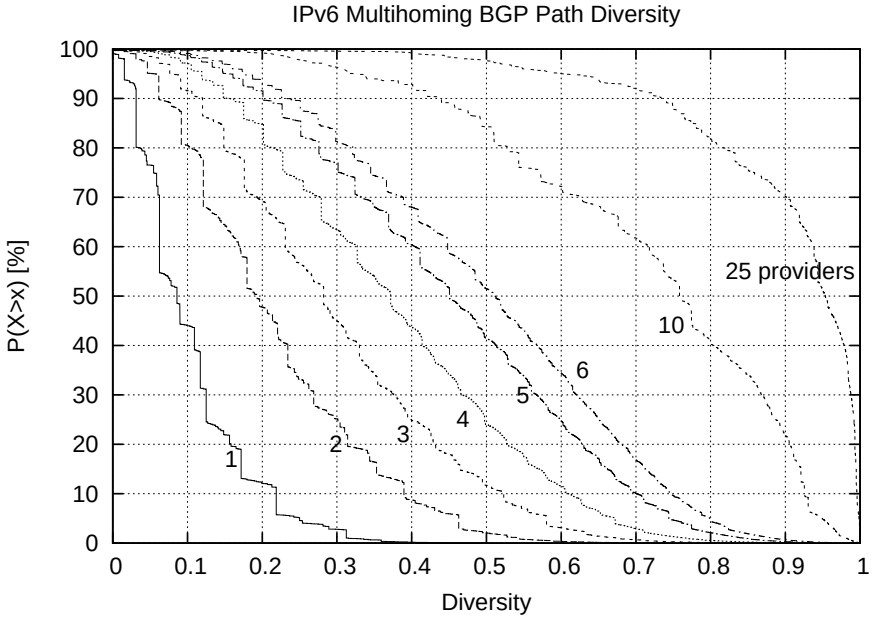


Figure 5.10. AS-level path diversity d for the inferred Internet topology, using multihoming with multiple PA prefixes

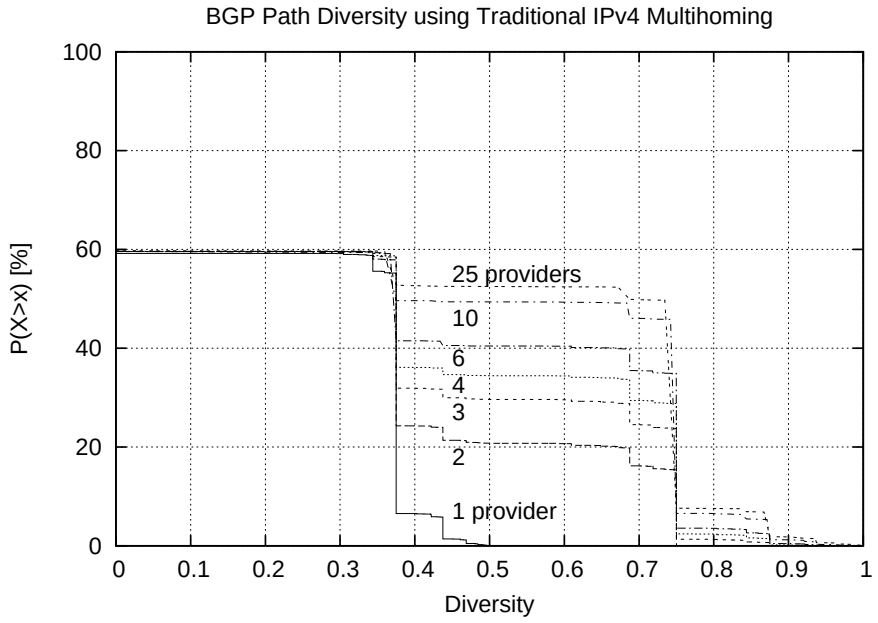


Figure 5.11. AS-level path diversity d' for the inferred Internet topology, using multihoming with a single PI prefix

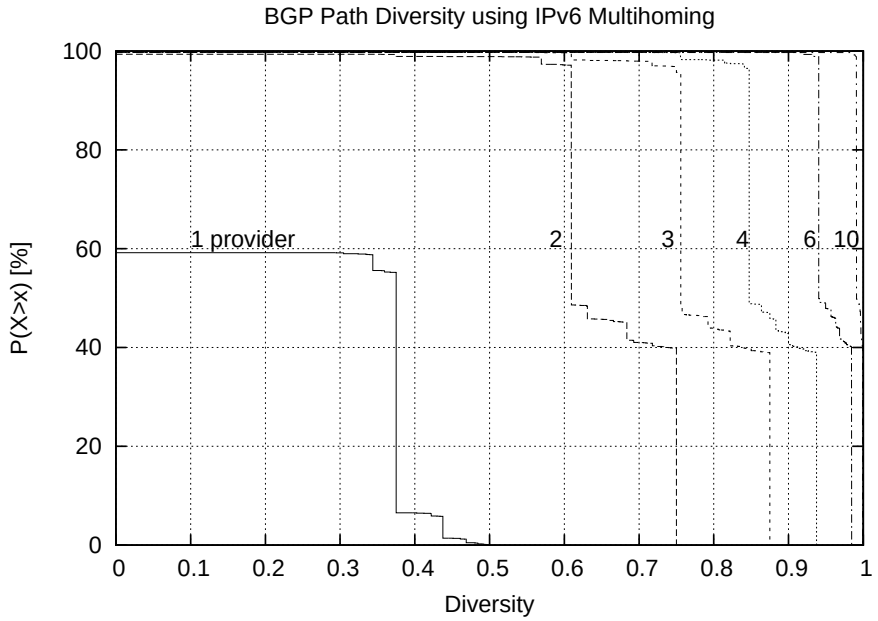


Figure 5.12. AS-level path diversity d' for the inferred Internet topology, using multihoming with multiple PA prefixes

When comparing Figures 5.9 and 5.11 with Figures 5.10 and 5.12, it appears clearly that the AS-level path diversity is much higher when stub ASes use multiple PA prefixes than when they use a single PI prefix. This is observed for both metrics d and d', i.e. when the bias in favour of short AS paths is taken into account or not. For example, Figure 5.10 shows that the path diversity observed for a dual-homed stub AS that uses multiple prefixes is already as good as the path diversity of a 25-homed stub AS that uses a single prefix. The path diversity obtained by a 3-homed stub AS using multiple PA prefixes completely surpasses the diversity of even a 25-homed stub AS that uses a single PI prefix.

We can notice that the diversity d or d' remains the same when considering only single-homed destinations. Indeed, only one prefix is announced by a single-homed stub AS. In this case, the PA or PI nature of the announced prefix has no impact on the AS-level path diversity.

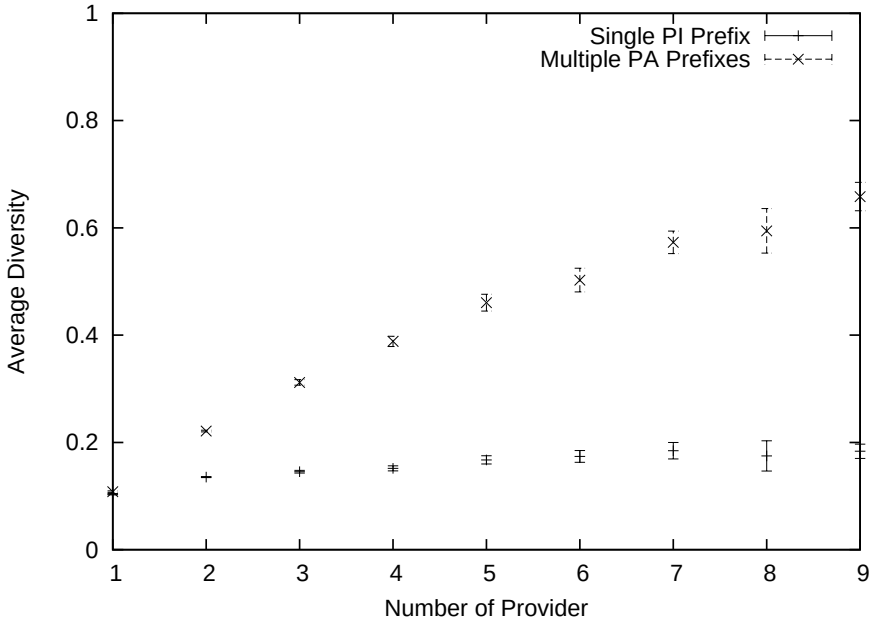


Figure 5.13. Average path diversity d for the inferred Internet topology

We can also evaluate the average path diversity d in function of the number of providers. For each stub AS x, we compute the mean of the path diversities for all paths towards x. We then group the stub ASes according to their number of providers, and compute their mean. Figure 5.13 shows the results for stub ASes using multiple PA prefixes, and for stub ASes using a single PI prefix, together with the 95% confidence interval. It shows that the average path diversity d obtained using multiple PA prefixes rises substantially with the number of providers. It rises much faster than the path diversity obtained when a single PI prefix is used, where the average slowly rises from about 0.10 for single-homed

sites to about 0.18 for sites having 9 providers. To give an idea to the reader, this improvement is similar to the one observed between the two path trees illustrated in Figure 5.14.

(a) d=0.109375 (b) d=0.1875

Figure 5.14. Example of improvement of 0.08

Figure 5.9 and 5.13 suggest that it is nearly impossible for a stub AS to achieve a good path diversity when it is multihomed by using a single PI prefix, whatever its number of providers. This result is also obtained when the alternate diversity metric d' is considered, which removes the preference for shorter paths.

5.3.3 Impact of BGP on the Path Diversity

This section examines how the Internet path diversity is affected by the BGP protocol.

Multihoming is assumed to increase the number of alternative paths. However, the AS-level path diversity offered by multihoming depends on how much the interdomain routes, as distributed by BGP, overlap. As explained in Chapter 1, the selection of the best route is first based on the highest local preference and shortest interdomain AS path. If the best route is not yet identified, the route with the smallest *Multiple-Exit-Discriminator* or *MED* is used. Then, the decision process prefers routes learned over an eBGP session to routes learned over an iBGP session. Among the remaining routes, a router prefers those with the closest IGP next hop. When these rules are not sufficient to select the best route, further tie-breaking rules are applied, such as to keep the oldest route or to prefer the route learned from the router with the lowest ID.

The results presented in the previous sections suggest that BGP heavily reduces the path diversity, at the level of autonomous systems. Two factors can explain why the diversity is so much reduced.

The first and primary factor is that, for each destination prefix, each BGP router in the Internet receives one route from a subset of its neighbours. Based on this set of received routes, BGP selects a single best route towards the destination prefix, and next advertises this single best route to its neighbours. Therefore, each BGP router reduces the diversity of available paths. As a consequence, a single

homed stub AS will receive from its provider only a single route towards each destination prefix, even if the destination site is connected to the Internet through multiple providers. Unfortunately, BGP is designed as a single path routing protocol. It is thus difficult to do better with BGP. Two extensions to BGP [186] have been discussed, but not accepted, within IETF to allow BGP to advertise several paths. The first solution [209] was proposed to solve the MED oscillation problem [90] by advertising several paths to a destination inside a BGP UPDATE. However, in this case only one of the advertised paths is used to forward IP packets. The second solution [30] was proposed to allow a router to learn several next-hops to reach an external destination in a transit AS. To our knowledge, those extensions have not been implemented and are not used in the Internet.

A second factor exists that further reduces the path diversity. The tie-breaking rule used by BGP to decide between two equivalent routes often prefers the same next-hops. Let us consider a BGP router that receives two routes from its providers towards a destination D. According to the BGP decision process, the shortest AS path is selected. However the diameter of the current Internet is small, more or less 4 hops [112]. As a consequence, AS paths are often of the same length, and do not suffice to select the best path. It has been shown that between 40% and 50% of routes in core and large transit ASes are selected using tie-breaking rules of the BGP decision process [165]. In our model with one router per AS, the tie-breaking rule used in this case is to prefer routes learned from the router with the lowest router address. This is the standard rule used by BGP-4 [170]. Unfortunately it yields to always prefer the same next-hop, a practice that further degrades the path diversity.

The first factor suppresses paths, while the second factor increases the probability that paths overlap. The use of multiple prefixes removes the first factor. Figure 5.10 and Figure 5.13 have shown the resulting benefit. However, the use of multiple prefixes has no impact on the second factor, since it does not modify BGP and its decision process in particular.

The overlapping of AS-level paths is illustrated in Figure 5.15. Each path between two stub ASes consists of multiple interdomain links. We compute how many times an interdomain link appear in the set of all paths between stub ASes. Next, we order the links by their probability of occurrence. Figure 5.15 shows $P(x)$, the probability that x% of the most used links appear in a path between any two dual-homed stub ASes. For instance, the figure shows that the 20% top used links appear in more than 80% of all paths between any two dual-homed stub ASes. A closer look to the results shows that the most used links are those that belong to Tier-1 providers.

Figure 5.15 shows that the concentration of paths is the same using either multiple PA prefixes, or a single PI prefix. It confirms that multihoming with multiple PA prefixes only impacts the number of paths available towards a stub AS, not how they overlap. Therefore, multihoming with multiple PA prefixes can

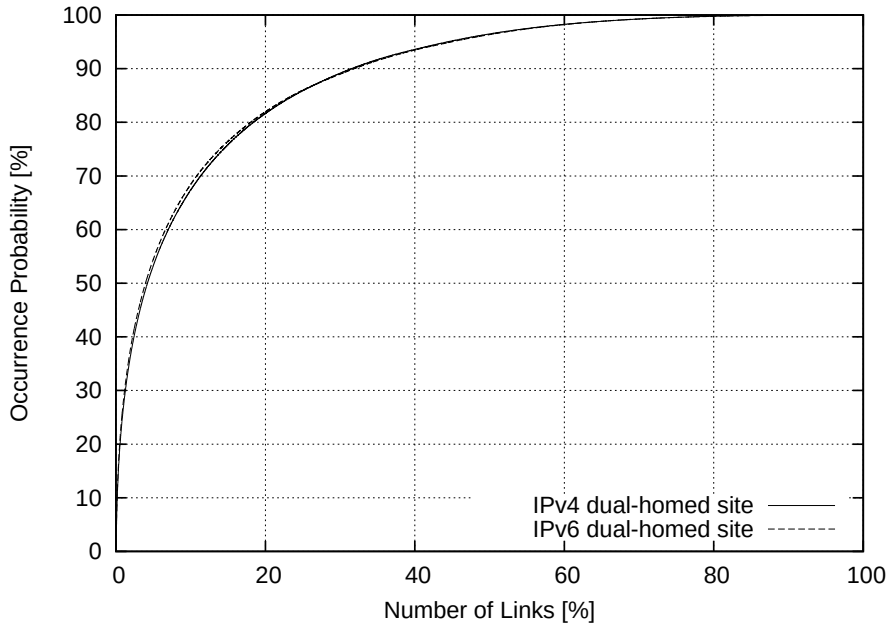


Figure 5.15. Probability $P(x)$ that the top x% of the links appear in a path. The top 20% links appear in more than 80% of the paths. The use of multiple prefixes instead of a single PI prefix makes no difference.

increase the AS-level path diversity without changing how interdomain links are used.

5.3.4 Generation of Hierarchical Internet Topologies

So far, all results were obtained using the same inferred topology. In order to draw some conclusions about the real Internet, the simulations are performed on several Internet-like topologies, with different properties. The simulations on these various topologies will allow us to determine the impact of the topology on the results, but also to explore possible evolution scenarios for the Internet.

In order to generate several AS-level Internet-like topologies, we use the GHITLE topology generator [58]. A topology is generated level by level, from the dense core to the customer level. Four levels are typically created : a fully-meshed dense core, a level of large transit ASes, a level of local transit ASes and a level of stub ASes. Additional levels can be created if needed. An AS inside a given level may have peer-to-peer relationships with other ASes from the same level, or customer-provider relationships with ASes from any upper level in the hierarchy. Customer-provider relationships are created using a traditional Barabási-Albert preferential connectivity model [26], i.e. new nodes tend to connect to existing nodes that are highly connected or popular. As a consequence, the degree distribution of the generated topologies follows a power law. Peer-to-peer relationships

are created randomly between nodes of the same level. The number of customer-provider links per node reflects the distribution seen in the Internet. GHITLE let us specify the number of levels, and the number of nodes in each level. This generator can produce small- or large-diameter Internet topologies while keeping the same overall number of stub ASes and transit ASes. This feature is used in the next section to explore different scenarios of the Internet evolution.

GHITLE was developed for this analysis because no existing generator can produce a topology that provides details about customer-provider and peer-to-peer relationships, and where the number of levels and nodes in each level can be specified. In particular, Inet [118] does not provide commercial relationships between ASes. Brite [130] and GT-ITM [40] do not produce a hierarchical topology with more levels than just transit and stub AS levels. Those transit-stubs networks do not correspond to the current Internet [189]. A second version of GT-ITM, presented in [41] is supposed to support policies, but unfortunately this version is not currently available.

5.3.5 Influence of Topology on Path Diversity

The way Internet will evolve in the future remains essentially unknown. In order to delimit the range of variation for our results, we perform simulations with three different topologies. The first topology tries to resemble the current Internet [189]. The second is a small-diameter Internet topology, consisting of stub ASes directly connected to a fully meshed dense core, as illustrated on Figure 5.16. The third is a large-diameter topology, generated using eight levels of ASes. An example of a topology with 8 levels of ASes is given by Figure 5.17. All these topologies have the same number of stub and transit ASes. They are generated using the GHITLE topology generator, described in the previous section. These topologies are publicly available at [59].

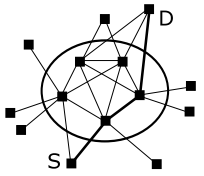


Figure 5.16. Example of a small-diameter generated Internet topology

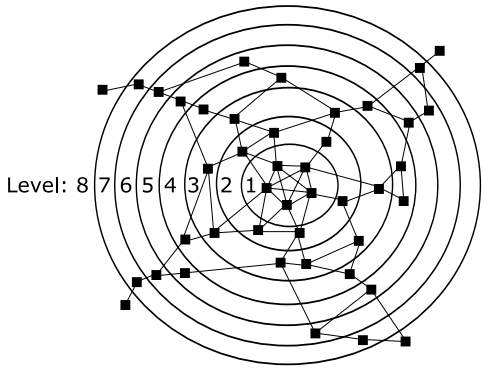


Figure 5.17. Example of a large-diameter generated Internet topology

Figure 5.18 and Figure 5.19 show the AS-level path diversity for the first topology, aimed at reproducing the current Internet. We used four levels of ASes : a dense core, a level of large transit ASes, a level of local transit ASes, and a level of stub ASes. The generated dense core is a full mesh of 22 ASes. The two generated transit levels contain respectively 200 and 2600 ASes. Finally, the stub level contains about 12000 stub ASes. This generated topology is similar to the inferred topology in terms of AS number and connectivity between ASes. As expected, the path diversity results for this generated topology are almost identical to the results obtained for the inferred topology.

We now explore different scenarios for the evolution of the Internet. A first scenario is a concentration, for commercial reasons, of tier-1 and tier-2 providers, i.e. ASes in the core and large transit ASes. At the extreme, the Internet could consist of a small core of large transit providers, together with a large number of stub ASes directly connected to the transit core. This would lead to an Internet topology with a small diameter, i.e. with short AS paths.

Another scenario is that the Internet continues to grow, with more and more core, continental, national and metropolitan transit providers. In this case, the Internet might evolve towards a network with a large diameter, i.e. an Internet with long AS paths.

We evaluate the path diversity for these two extreme cases : a small- and a large-diameter topology. The generated small-diameter topology consists of a full mesh core of 50 ASes and a single level of customers containing about 12000 stub ASes. The number of customer-provider links per stub AS still reflects the distribution seen in the Internet. In this small-diameter topology, the majority of the AS paths between stub ASes contain two intermediate hops : the first intermediate hop is the provider of the source AS, and the second and last intermediate hop is the provider of the destination AS, as illustrated on Figure 5.16. The AS-level path diversity for such a topology is illustrated on Figure 5.20 and Figure 5.21. As expected, the diversity in the small-diameter topology is better, since the paths are shorter than in the current Internet. It can be observed that 20% of single-homed stub ASes that use a single PI prefix have a diversity better than 0.22. This is better than with the inferred topology in Figure 5.9. Again, when a single prefix is used, the gain in path diversity does not rise much with the number of providers. In contrast, when multiple prefixes are used, the gain in path diversity rises fast with the number of providers.

The large-diameter topology is generated using eight levels of ASes. The same simulations are performed. The path diversity results are presented by Figure 5.22 and Figure 5.23. These figures show a poor path diversity in comparison with the path diversity of the previous topologies. This is due to the paths being longer. Again, Figure 5.22 shows that the path diversity remains low when stub ASes use a single PI prefix, whatever their number of providers. When multiple PA prefixes are used, the path diversity rises much faster with the number of providers, as shown by Figure 5.23.

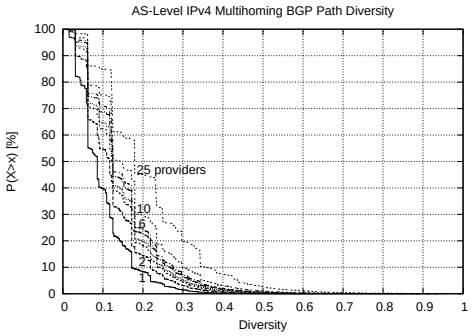


Figure 5.18. AS-level path diversity d for a generated Internet-like topology, using a single PI prefix

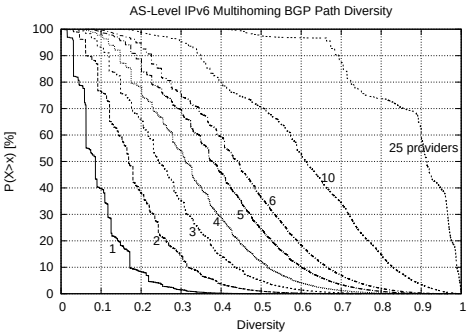


Figure 5.19. AS-level path diversity d for a generated Internet-like topology, using multiple PA prefixes

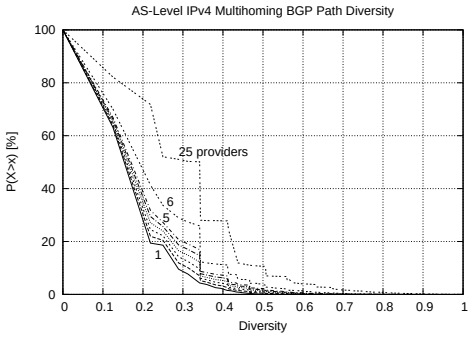


Figure 5.20. AS-level path diversity d for a small-diameter generated topology, using a single PI prefix

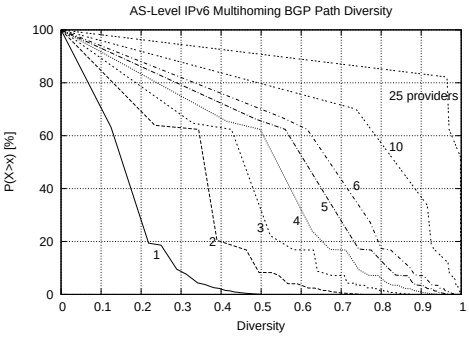


Figure 5.21. AS-level path diversity d for a small-diameter generated topology, using multiple PA prefixes

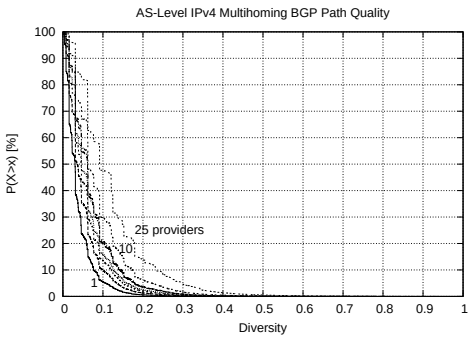


Figure 5.22. AS-level path diversity d for a large-diameter generated topology, using a single PI prefix

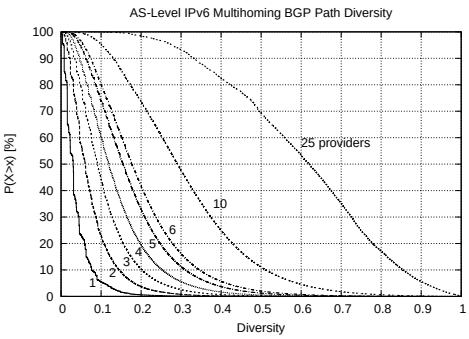


Figure 5.23. AS-level path diversity d for a large-diameter generated topology, using multiple PA prefixes

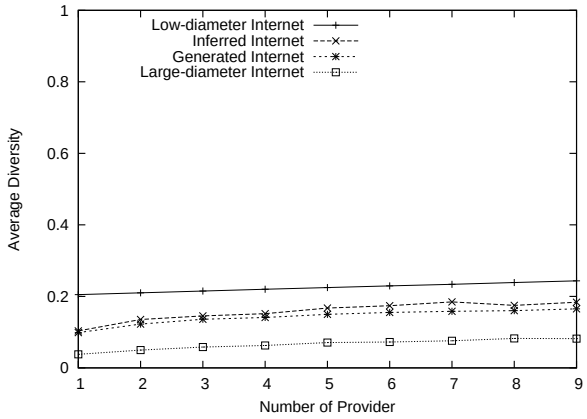


Figure 5.24. Average path diversity using a single PI prefix

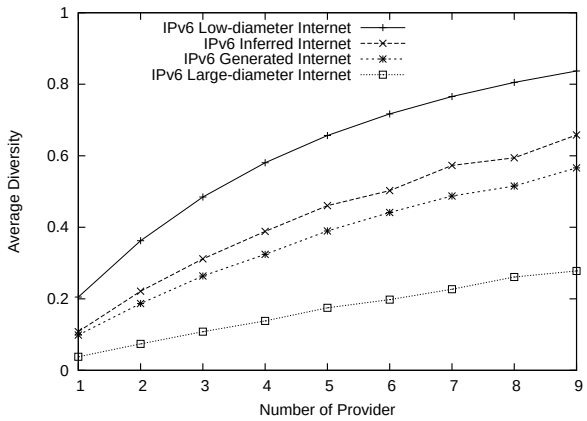


Figure 5.25. Average path diversity using multiple PA prefixes

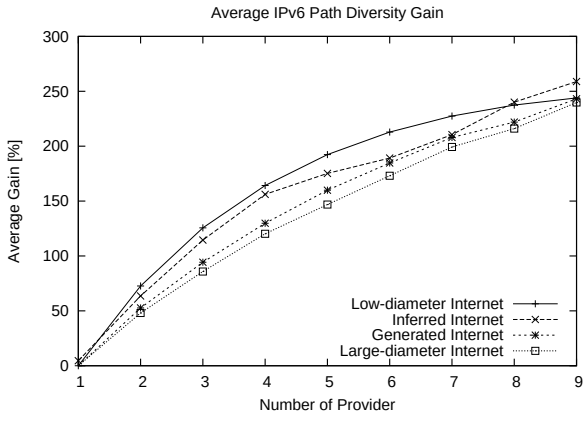


Figure 5.26. Summary of path diversity gains when using multiple PA prefixes instead of a single PI prefix

Figure 5.24 and Figure 5.25 show the average path diversity in function of the number of providers for all previously considered topologies. Figure 5.24 show that the average path diversity obtained when using a single PI prefix does not rise much in function of the number of providers, whatever the topology. In contrast, the average path diversity rises fast with the number of providers, when multiple PA prefixes are used. A comparison between Figure 5.24 and Figure 5.25 indicates that a dual-homed stub AS using multiple PA prefixes already gets a better path diversity than any multihomed stub AS that uses a single PI prefix, whatever its number of providers.

Figure 5.26 summarises the results for all topologies studied. It shows the path diversity benefit in percent that a stub AS obtains when it uses multiple PA prefixes instead of a single PI prefix. We can first notice that the gain is obviously null for single-homed stub ASes, as the use of one PA prefix instead of one PI prefix has no impact on the path diversity. Next, the figure shows that the gain is high when multiple PA prefixes are used, as soon as the stub AS has more than a single provider. For instance, the gain rises up to about 50% for dual-homed stub ASes, in all topologies considered. The path diversity of a 3-homed stub AS that uses multiple prefixes is about twice the diversity that it gets when it uses a single prefix. In a small-diameter Internet, the gain in path diversity rises fast in function of the number of providers, but also shows a marginal gain that diminishes quickly. In a large-diameter Internet, the gain in path diversity rises more slowly. But it keeps a substantial marginal gain, at least if the number of providers is below 10.

Additionally, Figure 5.26 shows that the gain for the current Internet is almost everywhere included between the gains of the two extreme cases. The deviation observed for stub ASes with 8 and 9 providers can be explained by the small number of these stub ASes, less than 10 in the inferred topology.

We can observe that the gain does not vary much with the topology considered. Figure 5.26 strongly suggests that the results observed for our synthetic topologies will also hold for the real Internet. In particular, the gain curve for the real Internet should most likely lies somewhere between the two extreme cases.

5.3.6 Impact of Hot-Potato Routing on Path Diversity

In the previous sections, the AS-level path diversity was analysed by considering one router per AS. However, a factor that can impact the path from a source to a destination is the intradomain routing policy used inside transit ASes. In this section, we explain how we model the intradomain policy using a generated router-level topology.

Usually, ISP routing policies in the Internet conform to hot-potato routing [196]. In hot-potato routing, an ISP hands off traffic to a downstream ISP as quickly as possible. In order to model this behaviour, we need a router-level

Internet topology with costs associated to intradomain links. It is currently not possible to generate and handle a router-level topology of a size similar to the Internet, i.e. probably composed of a few millions of routers. However, hot-potato routing can be modelled without using so many routers. The topology presented here is a trade-off between the realism of the topology and the simulation cost. No adequate router-level model currently exists for the interdomain. In particular, the transit-stubs networks generated by GT-ITM are not satisfactory, as they do not correspond to the current Internet [189].

We model hot-potato routing by generating a router-level topology with the GHITLE generator as follows. The topology is structured in four levels of ASes : 22 tier-1 ASes, 200 tier-2 ASes, 2600 tier-3 ASes and about 12000 stub ASes. The intradomain topology is generated differently for each AS-level. Figure 5.27 illustrates a generated router-level topology, where small filled squares represent routers and bold lines interdomain links. The cost associated to an intradomain link is indicated next to the link.

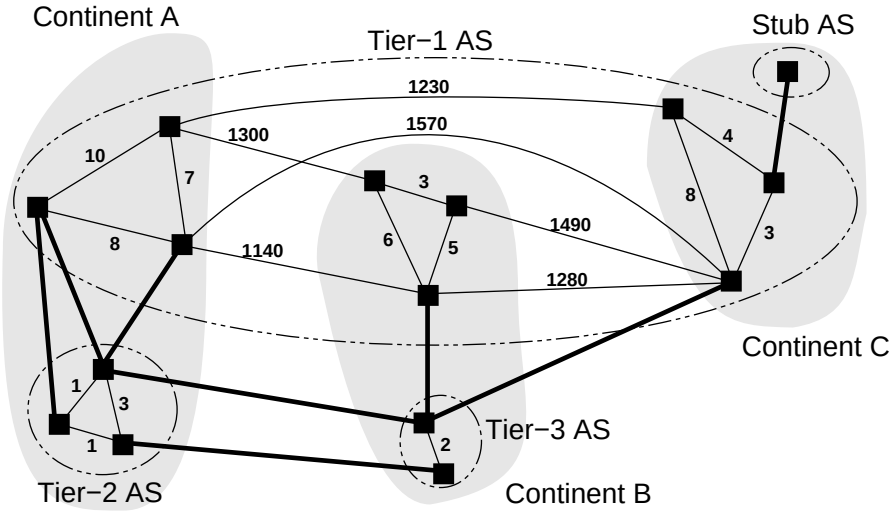


Figure 5.27. A router-level topology modelling hot-potato routing

We first divide the world into three major continents, in grey in Figure 5.27. We consider that tier-1 providers have a global presence on the three major continents. Each continent has three major geographical regions. We model the global presence of tier-1 providers by generating a triangle of routers in each continent. For redundancy reasons, all pairs of triangles are interconnected using two intercontinental links. We assign costs between 1 and 10 at random to intradomain links between routers in the same continent. Intercontinental links are assigned costs between 1000 and 2000. These costs are assigned in such a way that they model hot-potato routing, although they do do not reproduce exactly the costs

observed in real transit ASes. In particular, the costs are chosen so that a router-level path between an entry and an exit point from the same continent does not include an intercontinental link. For each continent, an interdomain link is generated between two routers belonging to two different tier-1 providers. By this way we generate three peer-to-peer interdomain links between every tier-1 providers.

We consider that tier-2 providers cover a single continent, composed of 3 regions. The intradomain topology of tier-2 ASes is modelled as a triangle of routers, where each router represents one geographical region. If a customer-provider link exists between a tier-2 AS and a tier-1 AS, then three interdomain router-level links are generated between routers of the two ASes. Between 1 and 2 peer-to-peer links are generated between tier-2 providers.

Similarly, the intradomain topology of tier-3 ASes is modelled as being composed of two routers, where each router represents a geographical region. Two router-level interdomain links are generated for each customer-provider AS-level link between a tier-3 and a provider. Between 1 and 2 peer-to-peer links are generated between two tier-3 ASes.

We still model a stub AS by a single router, assuming that a stub AS is a very localised network.

Once the router-level topology has been generated, we build the corresponding BGP configuration, and use C-BGP to simulate the distribution of the BGP routes. We then perform traceroute measurements at the AS level, and compute the path diversity using our metric.

The results for the path diversity are presented in Figure 5.28 and Figure 5.29. We see that the benefit of using multiple PA prefixes instead of a single PI prefix remains large, even when hot-potato routing is taken into account. When comparing Figures 5.28 and 5.29 with Figures 5.18 and 5.19, we see that hot-potato routing has no significant impact on the AS-level path diversity. This suggests that hot-potato routing will not have a major impact on the results presented in the previous sections.

5.3.7 Further Analysis of Internet Path Diversity

The previous sections have shown that the path diversity is much higher when a stub AS uses multiple PA prefixes instead of a single PI prefix. This section examines and discusses the reasons of this effect, and corroborates this claim by presenting other simulation results, using more traditional path diversity metrics.

First, we compute the probability that a stub AS has at least two disjoint paths towards any other stub AS, when it uses a single PI prefix. The inferred Internet topology is used for this simulation. Figure 5.30 shows the mean, 5^e percentile, median and 95^e percentile of this probability. For instance, it shows that a dual-homed AS has disjoint paths towards 20% of the destination ASes in average. It also shows that having more providers increases the probability of

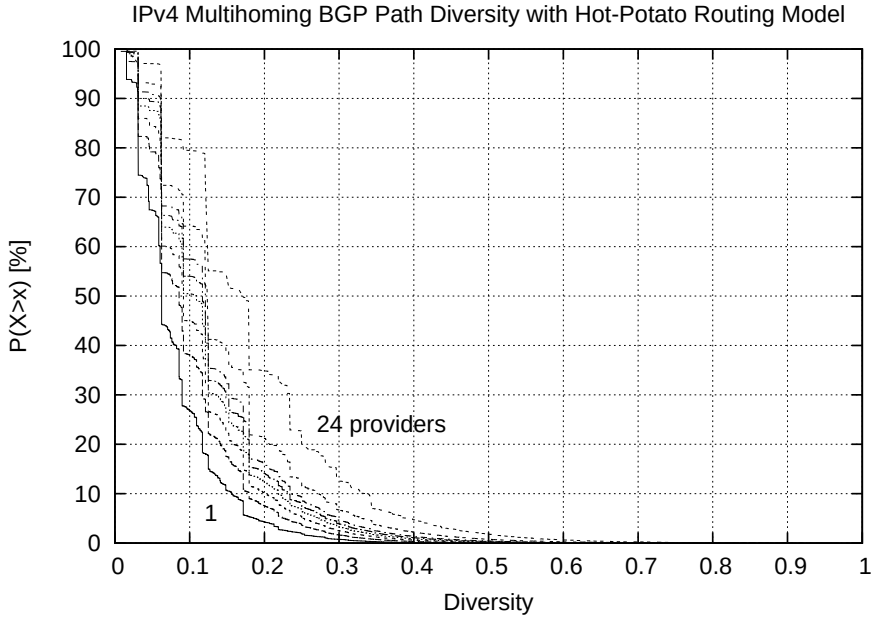


Figure 5.28. Path diversity for a generated Internet-like topology, with hot-potato routing model, using a single PI prefix

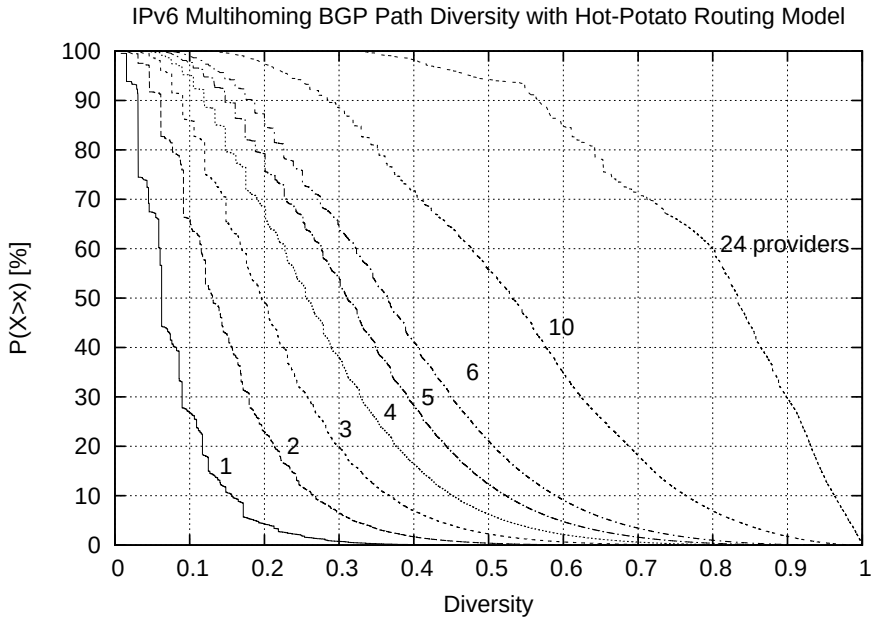


Figure 5.29. Path diversity for a generated Internet-like topology, with hot-potato routing model, using multiple PA prefixes

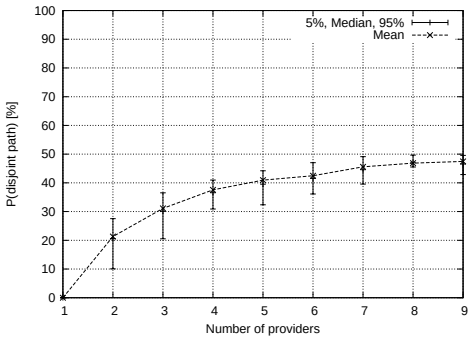


Figure 5.30. Probability that a stub AS has at least two disjoint paths towards any other stub AS, when it uses a single PI prefix

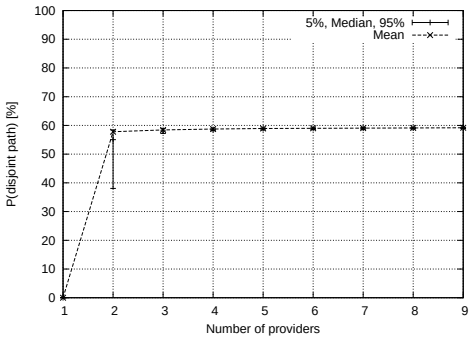


Figure 5.31. Probability that a stub AS has at least two disjoint paths towards any other stub AS, when it uses multiple PA prefixes

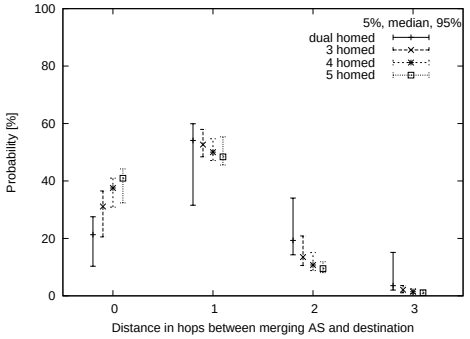


Figure 5.32. Distance in hops between the merging AS and the final destination stub AS, when a single PI prefix is used

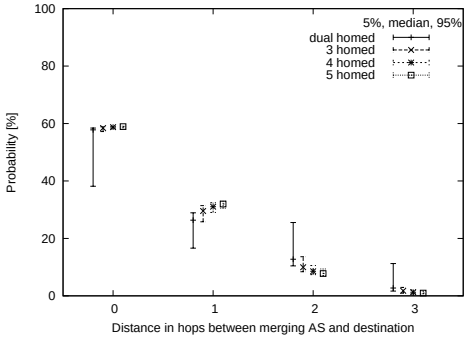


Figure 5.33. Distance in hops between the merging AS and the final destination stub AS, when multiple PA prefixes are used

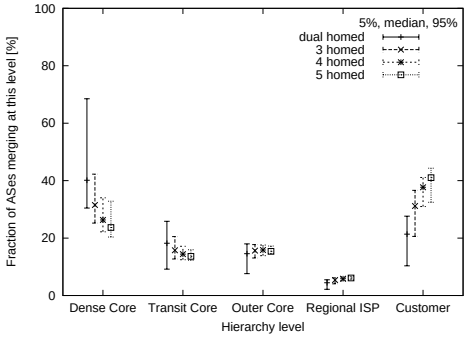


Figure 5.34. Hierarchy level of the merging AS, using a single PI prefix

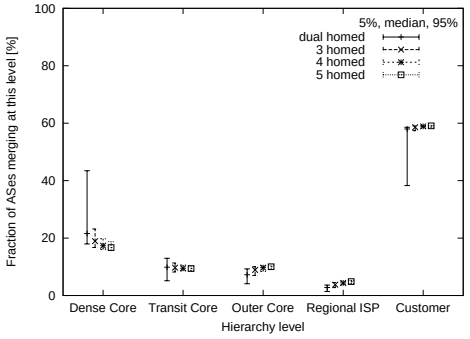


Figure 5.35. Hierarchy level of the merging AS, using multiple PA prefixes

having completely disjoint paths towards a destination AS. Obviously, a single-homed stub AS cannot have two disjoint paths. We can see that the probability increases with the number of providers but with a diminishing marginal gain. Note that the percentage of single-homed stub ASes in this topology is 40%, and thus the probability for having disjoint paths is at most 60%, whatever the number of providers.

We next perform the same evaluation, considering the use of multiple PA prefixes. Figure 5.31 shows that only two providers suffice to reach the maximum probability of having at least two disjoint paths up to the destination AS. Note that the maximum of about 60% is reached here, while it is still not with 9 providers when a single PI prefix is used.

Hence, the use of multiple PA prefixes increases the probability of having at least two completely disjoint paths up to the destination. It is obvious that completely disjoint paths cannot exist towards single-homed destination ASes. However, even when paths do not or cannot merge at the destination AS, it is still interesting to know the benefits from the use of multiple prefixes, compared to the use of a single PI prefix. We call *merging AS* the AS from which all paths from a multihomed stub AS to another stub AS merge into a single path. We now look at the distance, in hops, that exists between the merging AS and the destination AS. If paths merge at the destination, then this distance is 0 hop. The stub-stub AS paths are classified according to the number of providers of the source stub AS.

Figure 5.32 shows the distance in hops between the merging AS and the final destination stub AS, when a single PI prefix is used. It shows that most paths merge at one hop from the destination AS, essentially because 40% of stub ASes are single-homed, for which paths merge at their providers. The figure also shows that, depending on the destination stub AS, a dual-homed stub AS has a probability comprised between 30% and 60% that its paths merge at the provider of the destination stub AS. As expected, the probability that paths merge at the destination AS increases with the number of providers of the source AS. This probability is still limited to about 60%, i.e. the percentage of multihomed stub ASes in this topology.

The same evaluation is performed when considering the use of multiple PA prefixes. Figure 5.33 shows that having two or three providers nearly ensures that paths will merge at the nearest possible AS from the destination AS. Indeed the maximum probability that can be reached is 60% of paths merging at the destination stub AS (i.e. the percentage of multihomed stub ASes), and 40% of paths merging at the provider of the destination stub AS. Figure 5.33 also shows that the percentage of paths that merge at one hop from the destination stub AS increases with the number of providers of the source stub AS. This rise is entirely made at the expense of paths that merge at two hops or more.

In short, the use of IPv6 multihoming with multiple PA prefixes not only increases the number of completely disjoint paths towards a destination AS, but also increases the path diversity by making paths merge nearer the destination AS.

A consequence of this finding is that AS paths merge less often in the dense core when using multiple PA prefixes. Figure 5.34 and Figure 5.35 illustrate this phenomenon. Figure 5.34 shows where paths merge in the Internet hierarchy, using a single prefix. It shows for instance that, depending on the destination stub AS, between 30 and 70% of the paths from a dual-homed site merge at the dense core.

On the other hand, when multiple prefixes are used, Figure 5.33 has shown that 60% of paths merge at the destination AS. Consequently, in Figure 5.35, 60% of paths merge at the customer hierarchy level, while only about 20% paths still merge at the dense core.

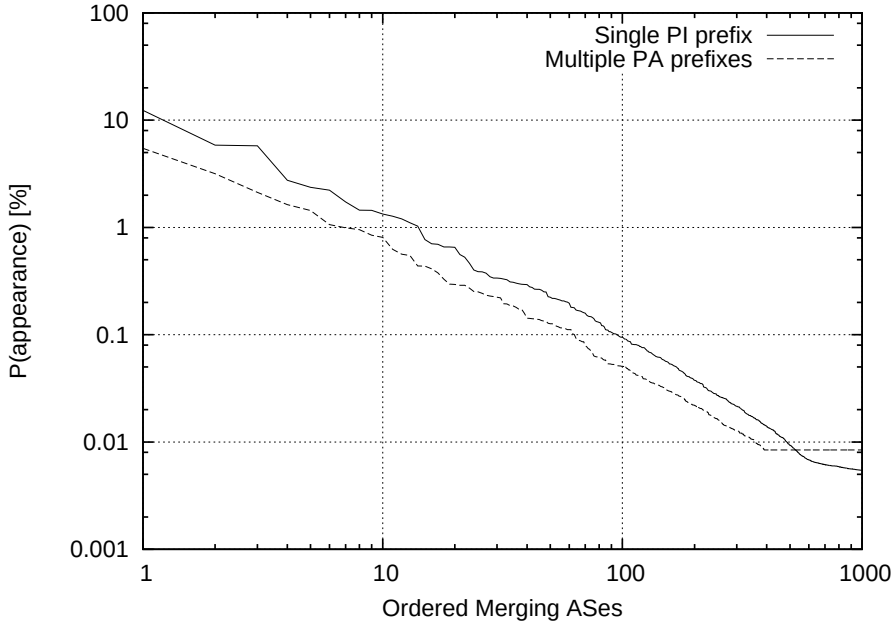


Figure 5.36. Distribution of ASes appearing as merging points, using a single PI prefix, and using multiple PA prefixes

We can also analyse how often an AS appears as a merging point between a multihomed stub AS and any other stub AS. Figure 5.36 shows that, in the inferred Internet topology, a single Tier-1 AS is the merging AS for about 12% of paths, when a single PI prefix is used. In contrast, when multiple PA prefixes are used, Figure 5.36 shows that a single AS is the merging AS for about 5% of

paths. This result confirms that, when multiple prefixes are use, the interdomain AS paths are more evenly distributed over transit ASes.

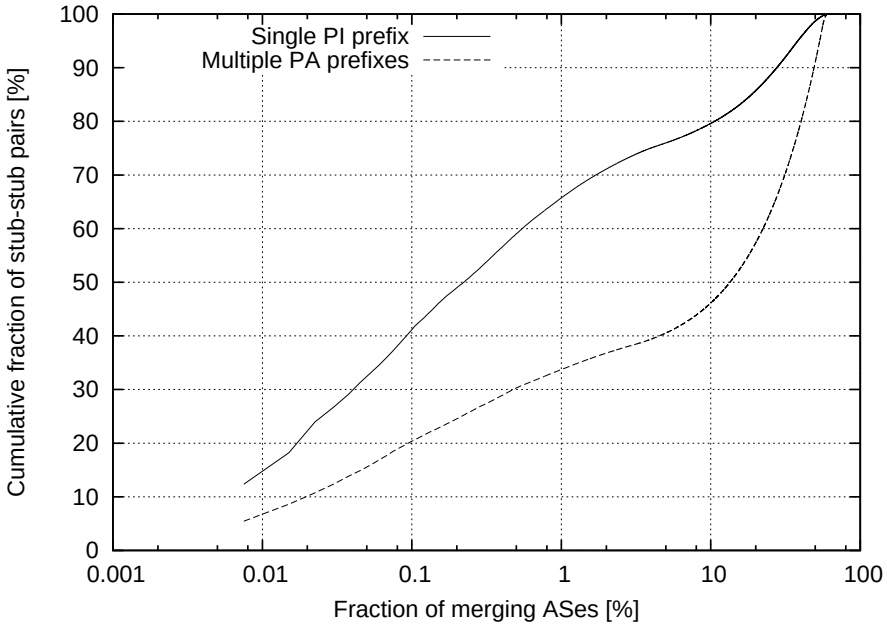


Figure 5.37. Cumulative distribution of ASes appearing as merging points, when a single PI prefix is used, and when multiple PA prefixes are used.

Similarly, Figure 5.37 shows that 1% ASes are the merging points of more than 65% stub-stub AS paths when a single prefix is used. When multiple prefixes are used, 1% ASes are the merging points of about 35% stub-stub AS paths. This is 30% less than using a single prefix. This result agrees with the results presented on Figure 5.31, Figure 5.33 and Figure 5.35, that showed that paths most often merge at the destination AS when multiple PA prefixes are used.

5.3.8 Summary

In the previous section, we have shown that a dual-homed stub AS that uses multiple PA prefixes has already a better Internet path diversity than any multihomed stub AS that uses a single provider-independent prefix, whatever its number of providers. We have also shown that this result does not depend much on the topology, and that hot-potato routing has no major impact on the diversity. Next, we have shown that the rise in path diversity using multiple PA prefixes is a consequence of paths merging nearer the destination AS. We have claimed that a dual-homed stub AS that uses multiple PA prefixes is almost certain to have at least two paths that do not merge before the destination stub

AS itself, provided that the destination AS is not single-homed. Finally we have shown that a consequence of the use of multiple prefixes is that the AS paths available to stub ASes are more evenly distributed on the Internet.

We now examine the consequence of this gain in diversity on network latency. We will evaluate in the next section how often the delay can be improved by leveraging this Internet path diversity.

5.4 Improving Delays with Multiple Prefixes per Site

We have shown that stub ASes that use multiple PA prefixes can exploit paths that are otherwise unavailable. Thus, the use of multiple PA prefixes increases the number of paths available, i.e. the Internet path diversity. Among the new paths, some of them may provide lower delays. We evaluate in this section how often this improvement in network latency occurs. First, we detail in section 5.4.1 the topology used to perform this evaluation. Next, we present the simulation results in section 5.4.2. Finally, we present the related work in section 5.5 and conclude this chapter in section 5.6.

5.4.1 A Two-Level Topology with Delays

In order to simulate delays along paths, we cannot rely on topologies provided by Brite [130], Inet [118], or GT-ITM [40] since they either do not model business relationships or do not provide delays along links. A topology that contains both delays and commercial relationships is available at [163]. In this topology, the interdomain links and the business relationships are given by a topology inferred from multiple BGP routing tables [189]. For each peering relationship found between two domains in this topology, interdomain links are added.

The different points of presence of each domain are geographically determined by relying on a database that maps blocks of IP addresses and locations worldwide. The intradomain topology is generated by first grouping routers that are close to each other in clusters, and next by interconnecting these clusters with backbone links.

The delays along the links are the propagation delays computed from the distances between the routers. The IGP weights used are the propagation delays for links shorter than 1000 km, twice the delay for links longer than 1000 km but shorter than 5000 km and 5 times the delay for links longer than 5000 km.

The resulting topology is described in more details at [163]. It contains about 40,000 routers, 100,000 links and requires about 400,000 BGP sessions. C-BGP [162] is used to simulate the BGP protocol on this topology, in order to obtain the router-router paths, and thus the delays between multihomed stub ASes. To

reduce the simulation time, we conduct the simulation for 2086 multihomed stub ASes randomly chosen among the 8026 multihomed stub ASes.

In this topology, 55% of the delays along the BGP routes are comprised between 10 and 50ms. About 20% of the delays are below 10ms and 25% sit between 50 and 100ms. These delays are considered as minimal bounds for the real delays, since only the propagation delay is taken into account. Factors that increase delays like limited bandwidth or congestion delays are not considered here. Although the simulated delays are inferior bounds to the delays observed in the global Internet, their order of magnitude is preserved.

5.4.2 Simulation Results

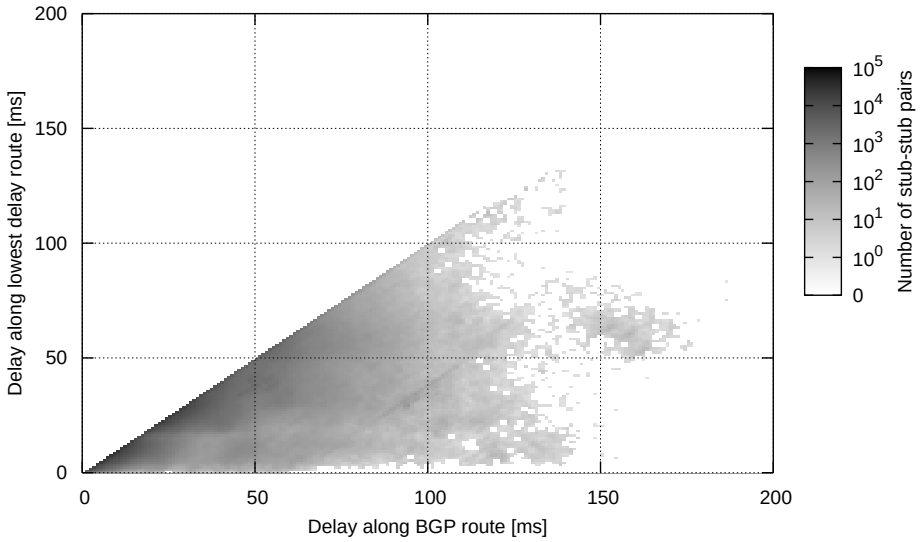


Figure 5.38. Delay along the BGP route versus delay along the lowest delay route

Figure 5.38 and Figure 5.39 plot the best delay obtained when multihomed stub ASes use a single PI prefix (x-axis), against the best delay obtained when stub ASes use multiple PA prefixes (y-axis). The gray-scale indicates the number of stub-stub AS pairs, on a logarithmic scale. In Figure 5.38, the diagonal line that appears represents stub-stub AS pairs for which both multihoming mechanisms yield to the same delay.

As explained in section 5.2.3, the use of multiple PA prefixes provides additional paths, beside traditional paths that are still available. As a consequence, delays can only improve, and no dot can appear above the diagonal line. A dot under this diagonal line indicates that the use of multiple PA prefixes introduces a new path with a delay lower than the delay along the best BGP path obtained

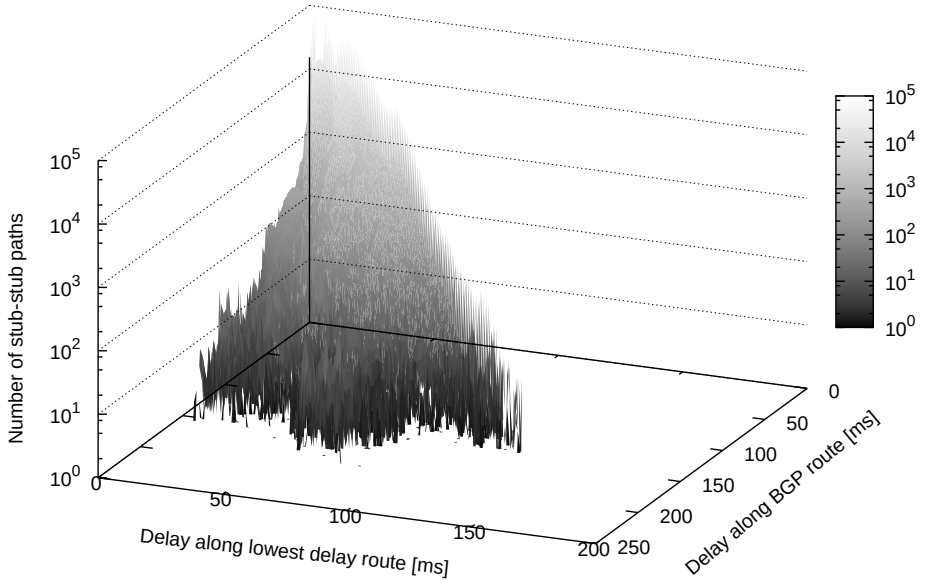


Figure 5.39. Delay along the BGP route versus delay along the lowest delay route

when a single PI prefix is used. We can see that many dots are located under this line. Sometimes, the improvement can even reach 150ms in this topology.

Figure 5.40 shows the cumulative distribution of the relative delay improvement. It shows that no improvement is observed for approximately 40% of the stub-stub AS pairs. However, the relative improvement is more than 20% for 30% of the stub-stub AS pairs. Delays are cut by half for about 8% of the stub-stub AS pairs.

As said in section 5.4.1, the delays observed in this topology are expected to be minimal bounds to those seen in the real Internet. Thus, we can reasonably assume that the absolute delay improvements presented in Figure 5.38 will not be lower in the actual Internet.

These simulation results show that improving delays is a benefit of IPv6 multihoming with multiple PA prefixes, without increasing the BGP routing tables.

5.5 Related Work

A work about IPv4 multihoming path diversity appeared in [4], where the authors define two path diversity metrics to quantify the reliability benefits of multihoming for high-volume Internet servers and receivers. They notice however that their metrics have an undesirable bias in favour of long paths. Their study draws empirical observations from measurement data sets collected at servers and monitoring

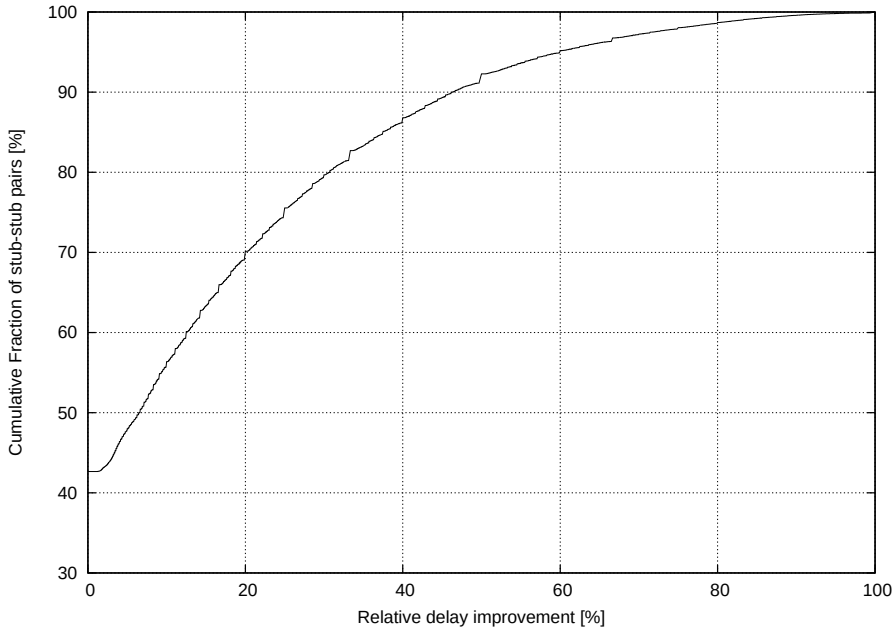


Figure 5.40. Distribution of the relative delay improvement

nodes, whereas our work is based on inferred and generated global-scale AS-level topologies.

A comparison of Overlay Routing and Multihoming Route Control appeared in [126]. In this study, the authors demonstrate that an intelligent control of BGP routes, coupled with ISP multihoming, can provide competitive end-to-end performance and reliability compared to overlay routing. Our results agree with this finding. In addition, our work explicits the impact of the path diversity on performance, and shows that IPv6 multihoming with multiple PA prefixes is able to actually provide these benefits.

In another study [194], Teixeira et al. characterise the path diversity of complete ISP topologies obtained using the Rocketfuel tool [183]. They next compare the inferred topology with the real one. They find that the Rocketfuel topologies both contain false links and miss actual links. They also find that these topologies have a significantly higher apparent path diversity, due to their large number of false links. In this work, we use inferred and generated AS-level Internet topologies to estimate the AS-level path diversity that is available to stub ASes that use either a single PI prefix, or multiple PA prefixes.

In [124], the authors analyse how paths from a dual-homed stub AS to a destination stub AS merge in the Internet. They used traceroute queries from their campus network multihomed to two commercial ISPs. They observed that the paths via their two ISPs merge at the dense core of the Internet for more than 76% of destination nodes. In this work, we also performed this experiment but

using an inferred Internet topology. We observed where the paths merge between any pair of stub ASes. We show in contrast that usually paths do not so often merge in the dense core, in particular for stub ASes having three providers or more.

It is well known that the use of provider-dependent aggregatable prefixes preserves the scalability of the interdomain routing system [85]. To our knowledge, this is the first study that shows that the use of multiple PA prefixes also increases network performance by leveraging the Internet path diversity, compared to the use of traditional multihoming with a single PI prefix.

5.6 Conclusion

In this chapter, we first proposed to reuse the notion of availability to compute the path diversity between two stub ASes. This metric measures how paths from a source to a destination AS overlap. It can take into account or not the lengths of the paths. We used this metric to study the AS-level path diversity that exists between pairs of multihomed stub ASes, first by considering the use of a single PI prefix, next by considering the use of multiple PA prefixes. We used an inferred Internet topology, and we generated several hierarchical Internet topologies. We computed the AS-level paths with a BGP simulator, and used our metric to measure the Internet path diversity between stub ASes.

We have shown that, in all considered topologies, the path diversity remains low when sites use a single PI prefix, whatever their number of providers. The main result found in this study is that the use of multiple PA prefixes as in the Host-Centric IPv6 multihoming approach largely improves the AS-level path diversity. This benefit adds up to the advantage of reducing the size of the Internet routing tables. We have also shown that the gain from using multiple PA prefixes rises fast with the number of providers, with a diminishing marginal gain. We have observed that this gain does not vary much with the topology considered, which suggests that the results obtained will most likely also hold for the real Internet. Next, based on further analysis, we have claimed that a dual-domed stub AS that uses multiple PA prefixes is almost certain to have at least two paths that do not merge before the destination stub AS itself, provided that the destination AS is not single-homed. Finally, we have shown that delays between multihomed stub ASes can be improved by leveraging this Internet path diversity.

All these observations show that, from a performance point of view, IPv6 multihomed stub ASes get benefits from the use of multiple provider-dependent aggregatable prefixes instead of a single provider-independent prefix. This study thus naturally encourages the IETF to pursue the development of Host-Centric IPv6 multihoming. The use of multiple PA prefixes enables route aggregation, and provides lower delays and more diverse Internet paths. This in turn yields to better possibilities to balance the traffic load and to support quality of service.

Chapter 6

The NAROS Approach for IPv6 Multihoming with Traffic Engineering

6.1 Introduction

Once multihomed, stub Autonomous Systems usually need to engineer their interdomain traffic. There is a growing demand for highly efficient and cost-effective mechanisms to improve the end-to-end performance of those stub ASes [175, 152, 7, 5]. They typically wish to use the paths with the best quality for sending and receiving packets. Various quality metrics can be used depending on the applications, e.g. low delay, high bandwidth, low jitter, or low loss rate. Unfortunately, the available interdomain traffic engineering techniques are currently based on the manipulation of BGP attributes [168], which contributes to the growth and instability of the BGP routing tables [15, 123]. In addition, BGP does not currently carry QoS metrics and BGP routers do not always select the paths with the best "quality" [102]. The reason is that BGP was designed to provide reachability and to allow domains to locally apply route selection policies. BGP does not take any QoS metric into account. As a consequence, multihoming does not always lead to improved performance.

Chapter 5 has shown that the use of multiple PA prefixes not only preserves the scalability of the interdomain routing, but also provides more diverse interdomain paths and lower delays, hence increasing the possibilities to balance the traffic load and to support the quality of service.

However, as described in Chapter 4, this approach has implications on traffic engineering methods, as the source address selected by a host determines the upstream provider used. As a consequence, outbound and inbound traffic engineering of flows initiated from hosts inside the multihomed site is entirely determined by how these hosts select their source addresses. No traffic engineering mechanism has ever been developed for such Host-Centric environment, despite the growing demand for such mechanism in IPv4, and despite the large opportunity for leveraging the interdomain path diversity.

This chapter aims at filling this gap by proposing and evaluating a technique to engineer the traffic of Host-Centric IPv6 multihomed sites. The new mechanism should also allow to effectively exploit the multiple interdomain paths that exist between multihomed sites. The technique must support a wide range of policies, possibly varying in time, and of technical or non-technical nature. This chapter tries to answer the following questions :

How to share the load in Host-Centric IPv6 multihomed sites, without manipulating BGP attributes ?

How to take into account complex load sharing policies ?

Can we handle very dynamic routing policies ?

Is the mechanism efficient and scalable ?

The solution proposed and evaluated in this chapter is based on a new service, inside the multihomed site, called *Name, Address and ROute System* (NAROS). NAROS is a mechanism to perform traffic engineering in a Host-Centric IPv6 multihoming environment. The basic principle of NAROS is that, when initiating a new flow, hosts inquire the NAROS service to determine the best source address to use to contact the destination node. By selecting the source address, the NAROS service can control how the locally initiated flows are distributed over the links with the providers. It thereby provides a traffic engineering mechanism, without transmitting any BGP advertisement and without affecting the BGP routing tables.

This chapter is organised as follows. In section 6.2, we explain how policy can be implemented in IPv6 multihomed sites by using a policy table. In section 6.3, we detail how NAROS allows a multihomed site to engineer its traffic. Finally, in section 6.4, the performance of the NAROS server is evaluated by using trace-driven simulations. We will show that the load on the NAROS server is reasonable and that we can obtain good load-balancing performance.

The technique proposed in this chapter has never been developed, although briefly suggested in [106]. To the best of our knowledge, this is the first approach that explicitly allows load-balancing and traffic engineering in IPv6 Host-Centric multihomed sites. It has been published in [65] and [64].

6.2 The Source Address Selection Policy in IPv6 Multihomed Sites

Figure 6.1 illustrates a multihomed site that uses the Host-Centric approach. Three Internet Service Providers (ISP 1, ISP 2 and ISP 3) provide connectivity to the multihomed site. The site exit router connecting with ISP 1 (resp. ISP 2 and ISP 3) is RA (resp. RB and RC). Each ISP assigns a site prefix (PA, PB and PC) to the multihomed site. As detailed in sections 2.3 and 4.4, these prefixes, together with a subnet ID (SA, SB and SC) are advertised by the site exit routers, and used to derive one IPv6 address per provider for each host interface.

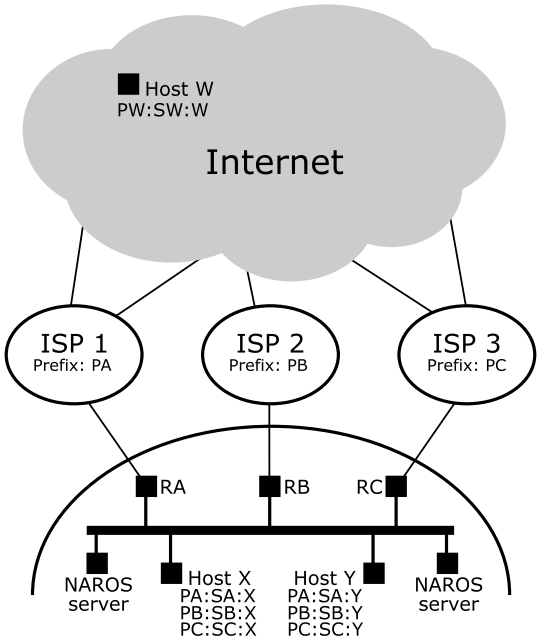


Figure 6.1. A multihomed site connected with three providers

The use of multiple addresses introduces a new architectural approach to engineer the traffic. As explained in Chapter 3 section 3.4.2, the site must ensure that all outgoing packets with a source address within the prefix assigned by a given ISP are sent through the same ISP.

In order to control how its flows are distributed over the links with the providers, the multihomed site must instruct its hosts how they should select their source addresses. This traffic engineering approach is said to be based on host capability. In practice, by default, hosts use the source address selection algorithm to select an appropriate address [73]. The selection algorithm relies on the RFC 3484 *policy table* [73], and was briefly described in Chapter 2, section 2.4.

In this section, we will show through an example that the default policy table may not lead to an adequate address selection in Host-Centric multihomed sites. We will also present how we can somehow engineer the traffic by tweaking the address selection through the addition of rules in the policy table.

First, let us observe that the default source address selection is arbitrary when a host has several global-scope IPv6 addresses, such as in Host-Centric multihomed sites. Consider the two Host-Centric multihomed sites illustrated on Figure 6.2.

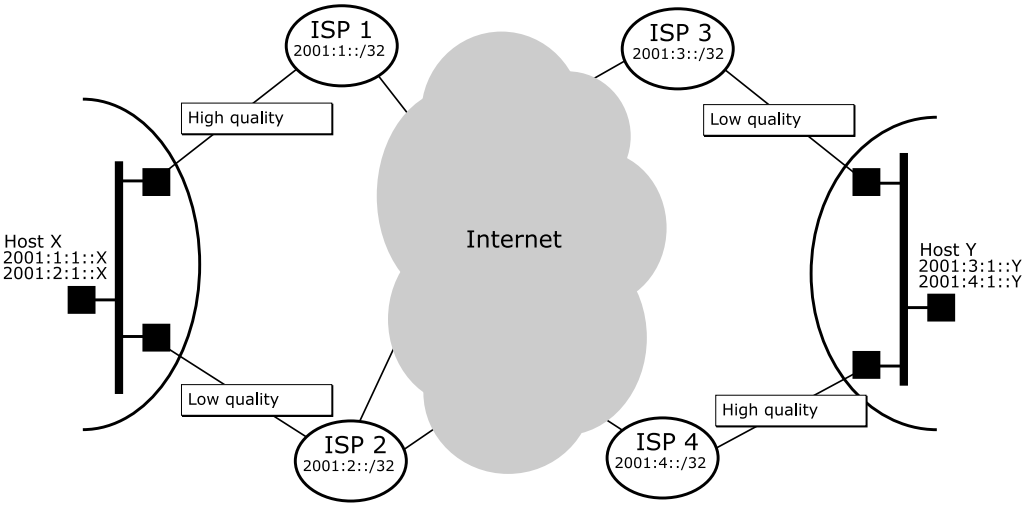


Figure 6.2. Arbitrary selection of source and destination addresses

In this figure, Host X has two global-scope IPv6 addresses, `2001:1:1::X` and `2001:2:1::X`. Similarly, the destination Host Y has two IPv6 addresses, `2001:3:1::Y` and `2001:4:1::Y`. Host X prefers the link through ISP 1, and Y prefers the one through ISP 4. The default address selection algorithm first selects the best source address of Host X for each destination address of Host Y. According to the rules explained in Chapter 2, the output is the one illustrated on Table 6.1.

For the destination address `2001:3:1::Y`, `2001:2:1::X` is the best source address because its matching prefix (`2001:0002::/31`) is longer than the matching prefix (`2001:0000::/30`) of the source address `2001:1:1::X`. For destination

Destination Address	Best Source Address	Rule
2001:3:1::Y	2001:2:1::X	Longest Matching Prefix
2001:4:1::Y	Unspecified	No rule

Table 6.1. Output of the source address selection

address 2001:4:1::Y, both source addresses of Host X have the same matching prefix, i.e. 2001:0000::/29. In such a case, the algorithm is unable to select the best source address, and the best source address is unspecified. Once the best source address is selected for each destination address, the algorithm has to select the best couple of source and destination addresses, which is the first one in Table 6.1. Thus, in Figure 6.2, Host X will join Host Y by using destination address 2001:3:1::Y and source address 2001:2:1::X. Unfortunately, this choice leads to the use of the low quality links.

To avoid this undesirable behaviour, the address selection algorithm can be tuned by inserting additional rules in the policy table. For instance, in Figure 6.2, in order to favour the high quality link with ISP 1, Host X may alter its policy table in the way illustrated by Table 6.2. Consider that Host X wants to contact Host Y. The available destination addresses are 2001:3:1::Y or 2001:4:1::Y. According to Table 6.2, these two destination addresses are assigned label 1, as they match only entry 2001::/16 with label 1. Hence, 2001:1:1::X will be the best source address for both destinations 2001:3:1::Y and 2001:4:1::Y, because address 2001:1:1::X is the only source address that has the same label (i.e. label 1) as the destination addresses 2001:3:1::Y or 2001:4:1::Y. As a consequence, all inbound and outbound packets exchanged with Host Y will flow through ISP 1. This will be true for any flow, provided that the flow is initiated by Host X.

Prefix	Precedence	Label
2001:1::/32	30	1
2001:2::/32	30	2
2001::/16	30	1

Table 6.2. Policy table of Host X

This example shows that traffic engineering can be achieved for locally initiated flows, by adding appropriate rules in the policy tables. This configuration can be done either by hand, or by using an extension to DHCP [128, 202], or in a fully dynamic way as we propose here with NAROS.

6.3 The NAROS Service

We propose to let the NAROS service manage the selection of the source addresses, in a fully dynamic way. The address selection will influence how the traffic flows through the upstream providers, and a good selection method will allow the site to engineer its interdomain traffic. The NAROS service is provided by one or more NAROS servers maintained inside the multihomed site, as illustrated in Figure 6.1.

We now consider in details how NAROS handles four main aspects of Host-Centric multihomed sites : source address selection, inbound and outbound traffic engineering, and fault-tolerance.

6.3.1 Source Address Selection

When a host inside the multihomed site initiates a flow with a remote node, it must determine the best source address to use among its available addresses. The proposed mechanism is that the host queries a NAROS server for the source address to be used. It complements in this way the default IPv6 source address selection algorithm [73, 74].

The rationale behind having a service for the source address selection is that the selection process could possibly be influenced by many factors. Examples of such factors are the current loads and states of the links, or administrative policies. A NAROS server could also rely on informations contained in BGP tables, e.g. the path length towards the destination. When engineering the traffic of a site, these policies may have to change very often. Storing statically the full set of rules in every host is not efficient, as a host is typically concerned by only a small subset of these rules. It is even almost useless when the routing policies change frequently. In this case, manual configuration of hosts is unmanageable, and the use of DHCPv6 is not sufficient. In order to not overflow the network with policies that concern only a small subset of the hosts, NAROS is designed as a service that must be queried by hosts, instead of a server that periodically advertises policies. The benefit of this approach is that a host receives routing policies only for destinations that it actually uses, in a dynamic way. The use of a request-response NAROS service may hence drastically reduce the number of policy informations that hosts have to handle. This can in turn reduce the resources needed to perform the source and destination address selection. It can also be particularly useful for *dumb* devices, that have no or very few memory. In this case, the device is even not required to remember NAROS informations it receives, so that it is able to respect complex and very dynamic routing policies with no memory requirement.

In its simplest form, the basic NAROS service is independent from any other service. A NAROS server does not maintain state about the internal hosts. It is thus possible to deploy several NAROS servers in anycast mode inside a site for

redundancy or load-balancing reasons. A NAROS server can also be installed on routers such as the site exit routers.

The NAROS protocol runs over UDP and contains two main messages : NAROS request and NAROS response. The NAROS request message is used by a client to request its flow parameters. The parameters included in a NAROS request are at least the destination address of the remote node and the source addresses currently allocated to the client. The NAROS server should only be contacted when the default source address selection procedure [73] cannot select the source address otherwise than by using the longest matching prefix rule.

The NAROS response message is sent by a NAROS server and contains the connection parameters to be used by the client. The parameters include at least the selected best source address, a prefix, and a lifetime. It tells that the client can use the selected source address to contact any destination address matching the given prefix. These parameters remain valid and can be cached by the client during the announced lifetime. The NAROS server gives only a hint about the best source address to use. It is recommended but not mandatory that the host actually uses this address, especially if it has other ways to determine the best source address. A situation where a host must not respect the NAROS selection is when the host knows that a failure affects the proposed best address, but the NAROS server did not detect it. This situation is discussed in section 6.3.4.

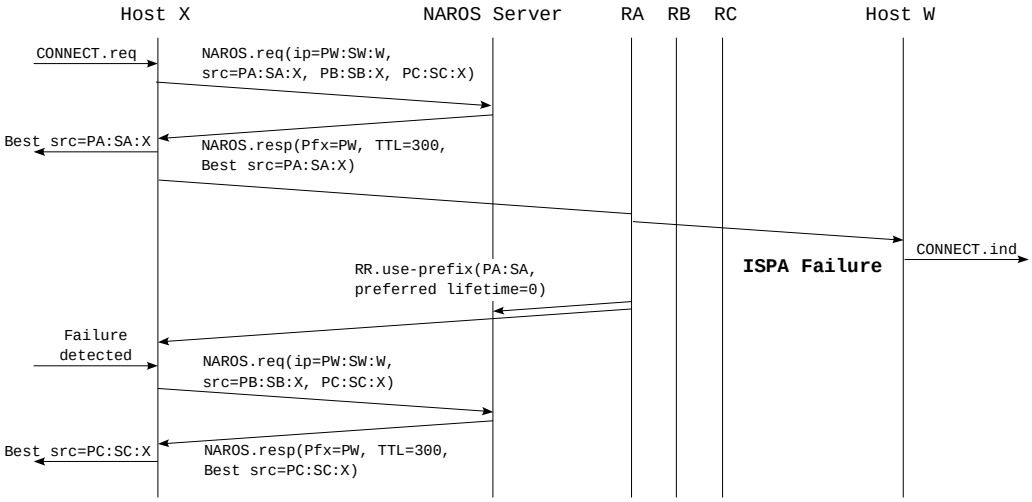


Figure 6.3. Basic NAROS scenario

The upper part of Figure 6.3 illustrates how the NAROS messages and parameters are used. The detailed formats of the NAROS messages are described in Appendix A. Before Host X sends its first packet to remote Host W (PW:SW:W), it issues a NAROS request in order to obtain the source address to reach Host

W. Upon receipt of the request, the NAROS server identifies the prefix PW associated with Host W, and selects for example PA:SA:X as the best source address to reach any destination hosts inside prefix PW. The prefix PW can be determined arbitrarily, e.g. using the /8 prefix covering the prefix of the destination address. Another solution is to extract from a BGP table the most specific BGP prefix that matches the destination address. The server assigns a lifetime (e.g. 300 seconds) to these parameters in the NAROS response message.

Once the reply is received, Host X knows that it should prefer address PA:SA:X to reach any destination inside prefix PW, including Host W. This preference is typically converted into a rule that is inserted in the policy table of Host X. This rule is removed once the lifetime has expired. The selected source address should be used for the whole duration of the flow, unless some failure affecting the flow occurs. If new TCP or UDP connections to the same destination are initiated before the announced lifetime expires, Host X can use the cached parameters. Otherwise it must issue a new NAROS request, and the best source address for the same destination may be different. By using appropriate values for the lifetime and the prefix in the NAROS response, it is possible to balance the load of the locally initiated traffic flows, and to limit the number of NAROS requests sent by hosts. This is demonstrated in section 6.4.

An optimisation is to couple the NAROS server with a DNS server. In this case, we can merge the DNS and NAROS request and response messages, hence minimising the overhead induced by the NAROS service. Figure 6.4 illustrates the messages exchanged when the NAROS and the DNS servers operate together. In this scenario, Host X wants to initiate a connection towards Host W, using the FQDN of Host W. The name resolver on Host X is modified so that the DNS request is put inside a NAROS request message addressed to the NAROS server. Upon receipt of the NAROS+DNS request, the NAROS server forwards the DNS request to the DNS server and waits for its resolution. Once the IP addresses of Host W have been received, the NAROS server selects the best source address among the addresses of Host X. Next, it replies with a NAROS+DNS response message to Host X. The modified name resolver library on Host X finally separates the NAROS and DNS responses, and Host X can initiate a connection with Host W. The merging of the NAROS and DNS servers appears here only as an optimisation. However, we will show in Section 6.3.3 that this coupling enables extended traffic engineering capabilities, and may thus be the preferred architecture for many sites.

6.3.2 Traffic Engineering for Locally Initiated Flows

By using the NAROS service, the source address selection is driven by the NAROS server. This can naturally be used to perform traffic engineering. For example, in order to balance the traffic among the three providers in Figure 6.1, a NAROS server can use a round-robin approach. For each new NAROS request, the server

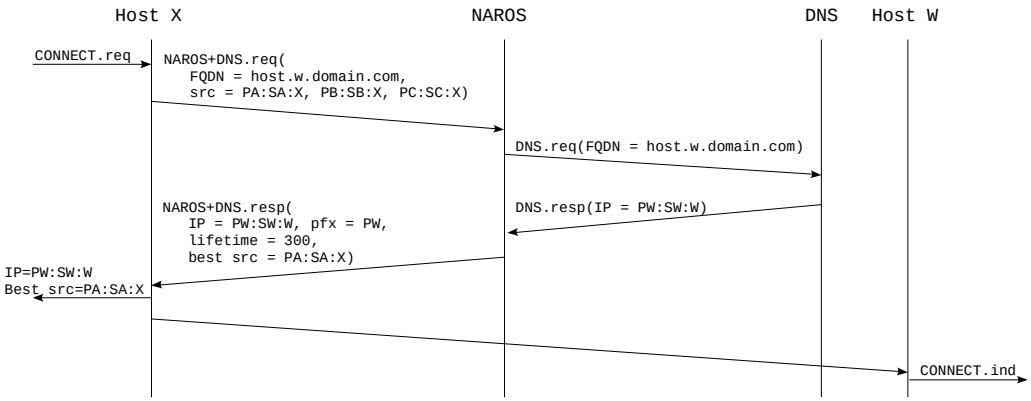


Figure 6.4. Basic scenario with a NAROS server coupled with a DNS server

selects another provider and replies with the corresponding source address. Except when a provider fails, this source address, and thus the upstream provider, remains the same for the whole duration of the flow.

The NAROS service allows traffic engineering without injecting any information in the Internet routing infrastructure. For instance, we will show in section 6.4.2 that a site can obtain similar performance as classical CRC-16 based load-balancing mechanisms. NAROS can also easily support unequal load distribution, without any additional complexity.

The NAROS mechanism allows to engineer all flows, inbound and outbound, provided that these flows are initiated by the hosts of the multihomed site. In particular, if hosts are sending lots of data to remote hosts in the Internet, NAROS allows the site to distribute this outgoing traffic over the ISP links by selecting the source addresses of the flows. Similarly, if hosts in the multihomed site receive lots of traffic from the Internet but send back very few traffic, then NAROS can also be used to control the amount of ingoing traffic received from each provider, provided that the flows are initiated from the multihomed site. This inbound traffic engineering mechanism is specific to Host-Centric IPv6 multihomed sites, where the ISPs of the multihomed site announce only their own BGP prefix, and do not announce the other prefixes of the multihomed site. As a result, a packet intended for an address belonging to the prefix of one ISP will necessarily enter the multihomed site through this particular ISP, making it possible to engineer the inbound traffic. This is not the case for traditional IPv4 multihoming with BGP.

6.3.3 Traffic Engineering for Remotely Initiated Flows

Sometimes, most flows of a multihomed site are initiated by remote hosts. A typical example is a content provider network. In such cases, a server that replies to a query cannot freely choose the source address of its packets. Indeed, as in

IPv4, the source address of a response packets must be the destination address of the query packet. Hence, the source address selection cannot help here to engineer this traffic.

However, when flows are initiated by remote hosts, limited traffic engineering can still be achieved by again coupling the NAROS and DNS servers of a site. Assume that the multihomed site has three providers, as depicted in Figure 6.1. Host X inside the multihomed site has three IPv6 addresses, PA:SA:X, PB:SB:X and PC:SC:X. If a remote host W in the Internet uses destination address PA:SA:X to query Host X, then the traffic sent by Host W towards Host X will necessarily enter in the multihomed site through ISP 1, as explained in the previous section. Thus, the IPv6 address selected by remote hosts determines the inbound provider used.

The address of Host X used as destination address by Host W is often provided through a query to a DNS server. As Host X has three addresses, the DNS server will reply to Host W with a list of three addresses. When receiving this list, Host W typically tries the first address. Limited traffic engineering can be achieved by ordering the list of addresses in the DNS response message. This method assumes that the list is not reordered when the DNS response message is sent back to the host, and that Host W does effectively try the first address of the list. It also assumes that remote Host W relies on the DNS to contact Host X. This last assumption may not always hold, for instance in the case of many peer-to-peer applications.

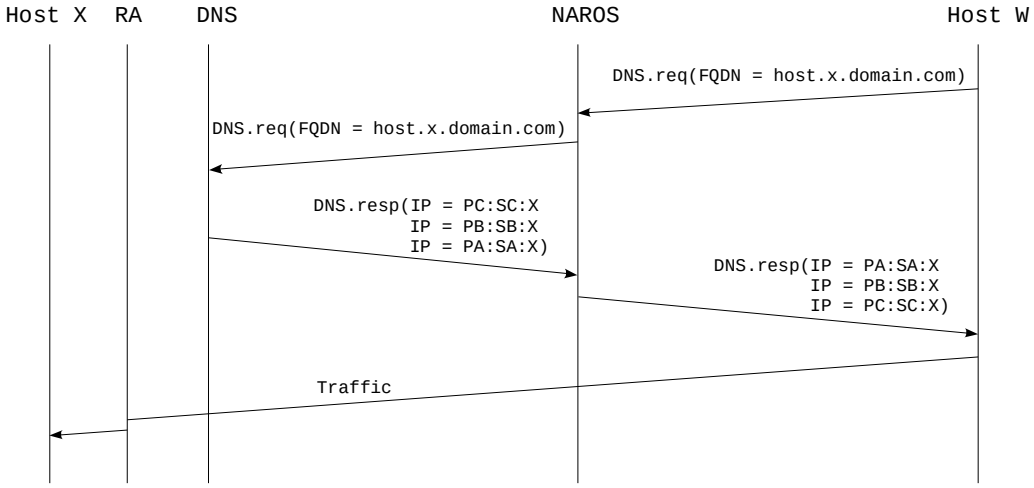


Figure 6.5. Inbound traffic engineering scenario, using a NAROS server coupled with a DNS server

When Host W in the Internet wants to contact Host X in the multihomed site, the basic scenario is illustrated on Figure 6.5. First, the NAROS server receives

from Host W the DNS request message for Host X. The server forwards the DNS request to the DNS server, and waits for the DNS response message that contains all the addresses assigned to Host X. Upon receipt of the DNS response message, the NAROS server reorders the list of addresses corresponding to Host X. For instance, it can place the PA:SA:X address at the top of the list to favour the use of ISP 1. The NAROS server finally transmits the altered DNS response message to Host W. As Host W usually tries the first address, the PA:SA:X address for Host X is used, and the traffic enters the multihomed site through router RA connected with ISP 1. Although the NAROS and DNS servers are separated on Figure 6.5, in practice they can be implemented as a single server.

6.3.4 Fault Tolerance

The NAROS server in itself does not provide fault-tolerance to Hosts inside the multihomed site. In Host-Centric environments, fault-tolerance can only be ensured by the hosts themselves. Hosts are the only entities that can fully decide if their flows suffer from a problem that requires some action.

However, the NAROS server must not ignore failures that can be clearly identified. For instance, if the link with one of the upstream providers fails, the site exit router connected to this provider can detect the event, and can advertise a null preferred lifetime for that prefix [140, 54], as in the solution described in [77] or [106]. This event must be taken into account by the NAROS server, so that further NAROS response messages will not include deprecated addresses, and will not lead to the use of the failed provider. Otherwise the NAROS service becomes useless. A host can also take this failure event into account, and it may immediately ask new parameters to the NAROS server. However, the host always makes the final decision about which address to use. More generally, a host can ask updated parameters each time it detects a failure which affects one of its flows. Once the new source address is known, mechanisms like SHIM or SCTP can be used to switch to the new address while preserving the established connections, as explained in Chapter 4, section 4.4.

In the lower part of Figure 6.3, assume for example that ISPA becomes unavailable. The site exit router connected to ISPA detects the failure and advertises a null preferred lifetime for prefix PA. The NAROS server immediately takes this advertisement into account, and future NAROS replies will not contain this prefix. Host X will also receive this advertisement. The standard effect is that it should no longer use this source address for new TCP or UDP flows. If Host X is currently using a deprecated address, it can issue a new NAROS request to choose among its other available source addresses. Next, the host can use e.g. SHIM to switch to the new source address while preserving its established connections.

6.4 Performance Evaluation

The performance of the NAROS protocol depends on two base parameters : the size of the prefix associated with the destination and its lifetime. We evaluate in this section the impact of these parameters on the cache size of the hosts, the number of NAROS requests and consequently the server load, and finally the quality of the load-balancing. The evaluation results are presented in section 6.4.2.

The evaluation of the NAROS service is based on a real traffic trace [57]. This trace is a flow-trace collected during 24 hours on November 18, 2002 and contains all the interdomain traffic of a university site. The trace is described in section 6.4.1. The NAROS performance is evaluated using an IPv4 traffic trace because we could not get a significant traffic trace of an IPv6 network.

6.4.1 Traffic Traces

The trace-driven simulation is based on real IPv4 flow traces collected on the interconnection point of the Université catholique de Louvain with the BELNET network during 24 hours on November 18, 2002. The trace contains only the outgoing traffic, i.e. the traffic that flows from UCL to BELNET, as depicted on Figure 6.6. 7687 hosts were active in the campus network. No firewall is used to filter the traffic received from the Internet.

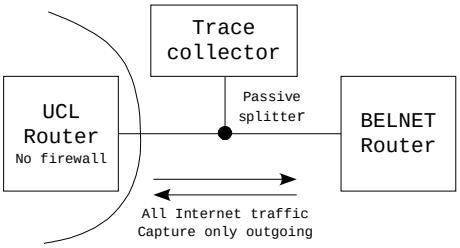


Figure 6.6. The trace was captured between the UCL and BELNET networks

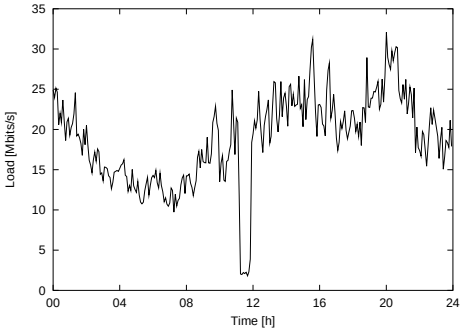


Figure 6.7. Traffic load

The trace is a flow-trace collected with `nprobe` [72]. The expired flows were saved every 30 seconds. The maximum flow idle lifetime was set to 30 seconds. This means that a flow is over when the last packet received is older that 30 seconds. However, due to memory limitations, we defined that a flow that has been active for more than 30 seconds was also considered expired, regardless of the flow duration. Further packets belonging to the same flow were accounted on a new flow. This has a non-negligible impact on our evaluation and it is discussed in section 6.4.3. The `nprobe` command used was :

Total Active Hosts	7687
Total Flows	17,511,778
Total Octets	203,067,716,744
Total Packets	322,377,278
Duration of trace (secs)	86384
Average flow duration (secs)	12.06
Average packet size (bytes)	629.00
Average flow size (bytes)	11596.00
Average packets per flow	18.00
Average flows / second	202.72
Average Kbits / second	18806.05

Table 6.3. Trace characteristics

Application	*Port(s)*	*%Volume*
edonkey	4661,4662,4665,4672	9.71
kazaa	1214	7.47
http	80	2.33
ftp	20,21	1.04
smtp	25	0.85
opennap	6699	0.51
ssh	22	0.24

Table 6.4. Traffic volumes per application

Application	*Port(s)*	*%Flow*
http	80,8080	25.17
edonkey	4661,4662,4665,4672	18.08
kazaa	1214	3.18
dns	53	2.95
irc	6665-6669	1.45
gnutella	6346	0.53
smtp	25	0.50
ftp	20,21	0.14
ssh	22	0.01

Table 6.5. Traffic flows per application

```
nprobe -s 30 -d 30 -t 30 -w 262144
```

The traffic trace contains origin, destination, port, protocol, size, and timing informations of about 322 million packets forming more than 17.5 million flows. The total amount of volume captured is about 200 GB (18.8 Mb/s in average). The average flow lifetime is 12 seconds. A summary of the traffic characteristics is shown in Table 6.3. Figure 6.7 shows the evolution of the traffic load during the day. Due to technical reasons, the traffic was disturbed between 11am and 12am. The trace contains a large amount of HTTP and peer-to-peer traffic. A summary is shown in Table 6.4 and Table 6.5.

6.4.2 Evaluation Results

The first performance parameter to consider is the size of the NAROS cache maintained by the hosts. We evaluate the impact of the prefix length used in the NAROS responses on the cache size of the hosts. For example, if a host requests a NAROS hint for destination 1.2.3.4, the NAROS server may reply with a /24 prefix, meaning that the parameters are valid for all addresses inside prefix 1.2.3.0/24. It may also extract the corresponding prefix from a BGP table. In this case, the prefix length is variable because it depends on the most specific matching prefix found in the BGP table for this destination address.

Figure 6.8 shows on a log-log scale $p_1(x)$: the percentage of hosts having a maximum cache size greater than x. For instance, it shows that if we use /24 prefixes as in the example, the cache size remains below 100 entries during the whole day for more than 90% of the hosts. We used a lifetime of 300s. The hosts which present the largest cache size were found to be either compromised machines sending lots of probes or very active peer-to-peer clients.

In Figure 6.9, the impact of the lifetime on the cache size is evaluated. Unsurprisingly, it shows that the use of lower lifetimes yields to smaller cache sizes. A consequence of Figure 6.8 and Figure 6.9 is that small prefix lengths and low lifetime contribute to small cache sizes. A value of 300s seems appropriate for the site studied.

We also evaluate the impact of the lifetime on the cache performance. A good cache performance is necessary to limit the number of NAROS requests that a host issues. Figure 6.10 evaluates the percentage of cache hits versus the lifetime in seconds. As expected, it shows that the cache hit ratio is higher when longer lifetime or less specific prefixes are used. However, for the trace studied, we get no significant improvement by using lifetimes longer than about 300 seconds. We can also observe that the lifetime has little impact on the cache hit ratio when /8, /16 or BGP prefixes are used.

A second element to consider is the server load. Figure 6.11 shows on a log-log scale $p_2(x)$: the percentage of hosts issuing a number of NAROS requests greater than x, during the whole day. We use here a lifetime of 300s and simulate various

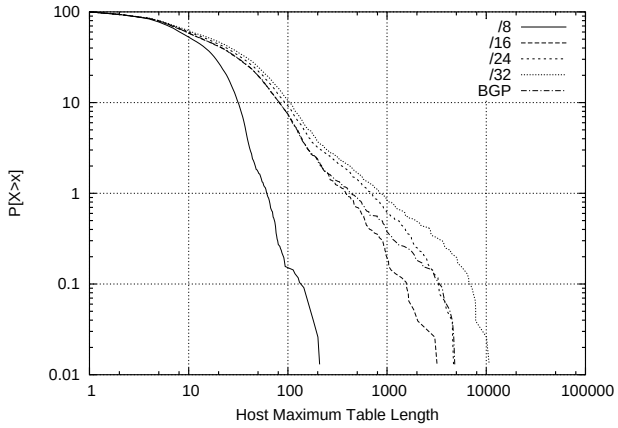


Figure 6.8. NAROS Cache size for a lifetime of 300s and various prefix lengths

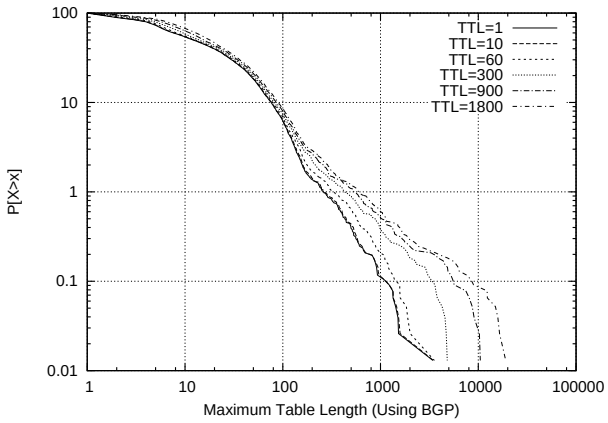


Figure 6.9. Impact of the response lifetime on the cache size

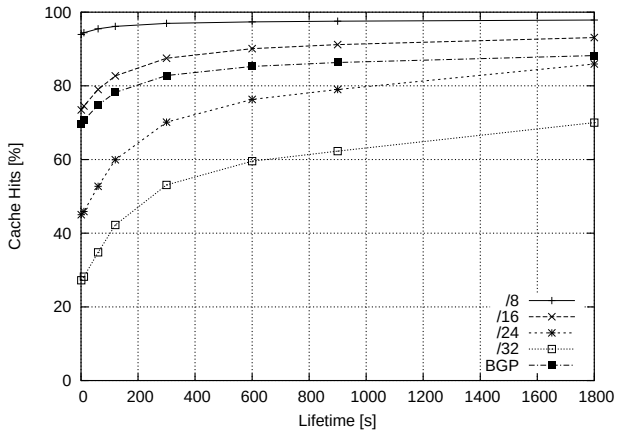


Figure 6.10. Impact of the response lifetime on the cache performance

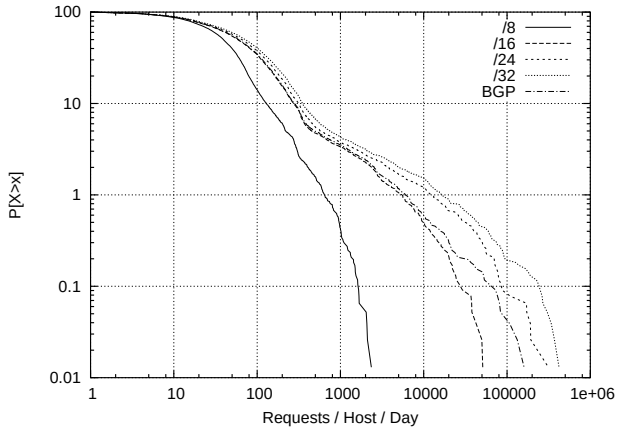


Figure 6.11. Number of requests per host during one day, for various prefixes and a lifetime of 300s

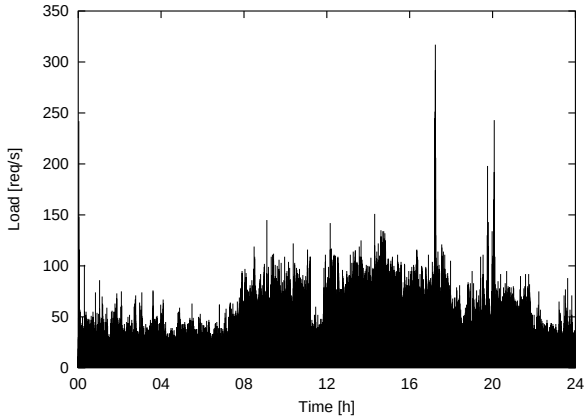


Figure 6.12. Server load using BGP prefixes and a lifetime of 300s

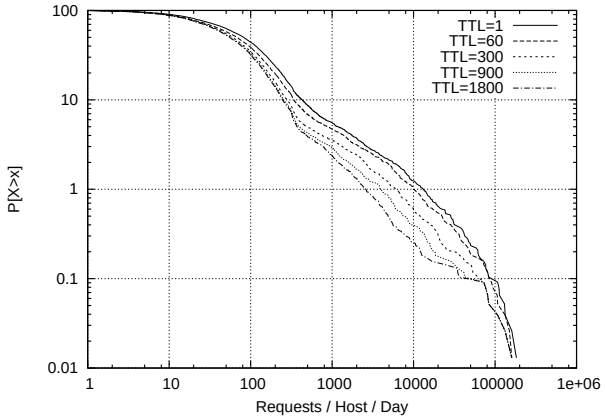


Figure 6.13. Impact of the response lifetime on the number of requests per host during one day

prefix lengths. Figure 6.11 shows that when BGP prefixes are used, 90% of the hosts issue less than about 300 requests during the whole day. The resulting server load is illustrated in Figure 6.12. This load is proportional to the number of hosts. It essentially follows the traffic load. The load average is about 35 requests per second, which is still reasonable. In comparison, this is no more than the number of DNS requests coming from the Internet and addressed to the site studied. The bandwidth overhead of the NAROS approach is estimated to about 0.35%.

In Figure 6.13, we evaluate the number of NAROS requests per host during the day, for various lifetime values. As expected, the figure shows that the number of requests per host diminishes when the lifetime increases. This is most noticeable for hosts that issue a large number of requests. Figure 6.14 shows the impact of the lifetime value on the average NAROS server load. The figure shows that the load diminishes quickly as the lifetime increases. It also shows that there is no significant improvement from using lifetimes longer than about 300 seconds.

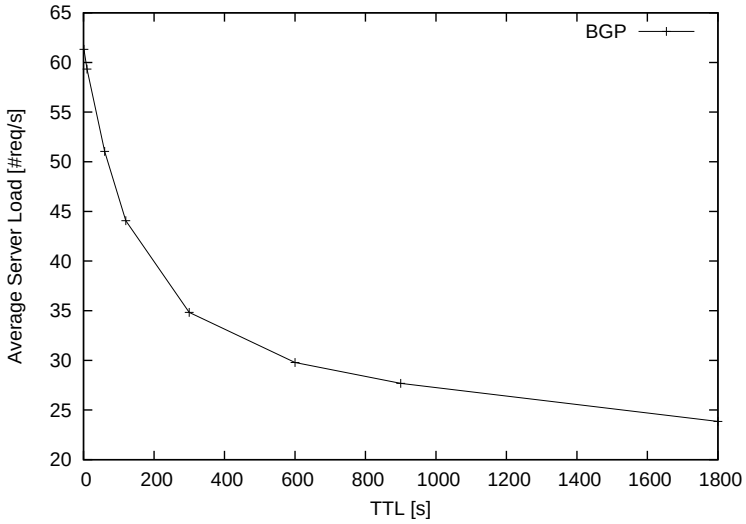


Figure 6.14. Impact of the response lifetime on the average server load, using BGP prefixes

We now compare the performance of the NAROS load-balancing mechanism with the best widely used load-balancing mechanism that preserves packet ordering, i.e. CRC16 [42]. We focus on the common case of balancing the load between two links of the same capacity. For the NAROS load-balancing, we use a simple round-robin approach, i.e. a new flow is alternatively assigned to the first and the second links. CRC16 is a direct hashing-based scheme for load-balancing where the traffic splitter uses the 16-bit *Cyclic Redundant Checksum* (CRC) algorithm as a hash function to determine the outgoing link for every packet. The index of the outgoing link is given by the 16-bit CRC checksum of the tuple (source

IP, destination IP, source port, destination port, protocol number), modulo the number of links. CRC16 is often used on parallel links from the same router.

We measure the performance of the load-balancing by looking at the deviation from an even traffic load between the two links. Let $load_1$ and $load_2$ be respectively the traffic load of the first and the second link. We define the deviation as a number in $[-1, 1]$ computed as $(load_1 - load_2)/(load_1 + load_2)$. A null deviation means that the traffic is balanced, while a deviation of 1 or -1 means that all the traffic flows respectively through the first or the second link.

Figure 6.15 compares the deviation in percent of the NAROS and CRC16 load-balancing. For NAROS, we use BGP prefixes and a lifetime of 300s. The sum of all deviations obtained for the NAROS curve equals 23.38, while this number equals 23.40 for the CRC16 curve. This shows that, for this traffic trace, the NAROS solution is able to provide a load-balancing mechanism with similar performance as a widely used static load-balancing mechanism. Figure 6.16 compares the NAROS load-balancing quality for a lifetime of 1s and a lifetime of 1800s. When comparing the sums of the deviations of the the two curves, it appears that the quality of the load-balancing is only slightly better when a lifetime of 1s is used, at the expense of a higher server load. From Figure 6.16 and Figure 6.15, it appears that the use of a lifetime of 1s does not yield to better load-balancing performance than the use of a lifetime of 300s. Hence, for this traffic trace, the lifetime parameter has few impact on the quality of the load-balancing. Simulations should be performed on other traffic traces to confirm these results.

6.4.3 Impact of the Trace Collection Method on the NAROS Evaluation

The method used to collect the trace has a non-negligible impact on the performance evaluation. This is analysed in this section.

Due to memory limitations, flows active for more than 30 seconds are considered expired, regardless of the flow duration. Hence, the collected flows are never longer than 30 seconds, and are shorter and more frequent than in reality. Thus, the number of NAROS requests and the NAROS server load are greater than they should be. This factor, together with the fact that no filter is applied on the traffic, suggests that our evaluation of the server load and the number of request is a worst case evaluation.

The number of cache entries is not affected since the cache entry of a flow is updated at every NAROS request for this flow. So, the entry remains in the cache until the real flow expires.

The NAROS cache hit ratio is variously affected by this factor. When the NAROS lifetime is shorter than 30 seconds, a cache miss happens every 30 seconds since the cache entry expired. In this case, the evaluation of the cache hit ratio is pessimistic. If the lifetime is longer than 30 seconds, a cache hit happens each

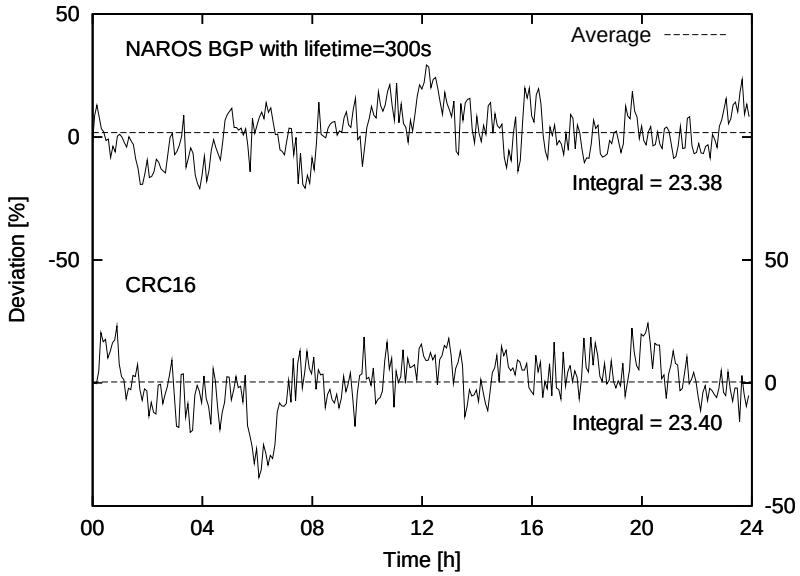


Figure 6.15. NAROS and CRC16 load balancing comparison

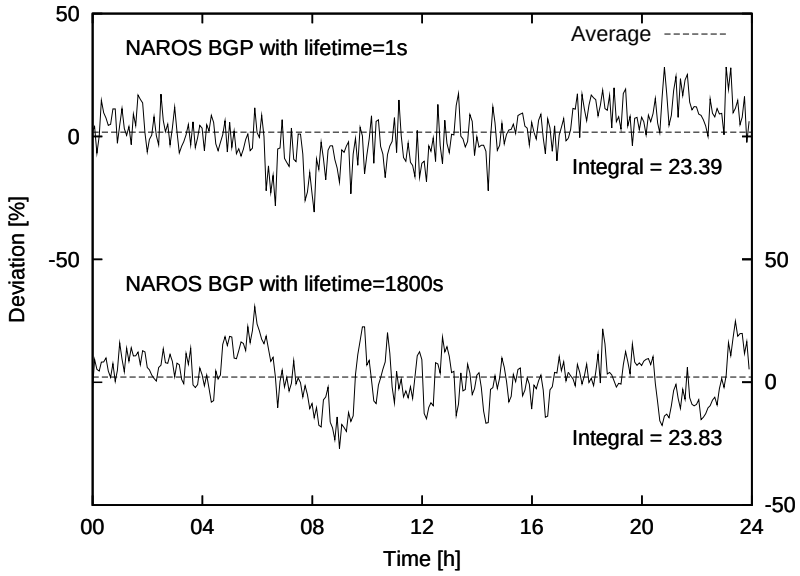


Figure 6.16. NAROS load balancing with lifetime of 1s and 1800s

30 seconds since the cache entry is still valid. Here, the evaluation of the cache hit ratio is optimistic.

However, all these impacts are reduced by the small number of long flows : only 12% of the flows are longer than 30 seconds.

6.5 Summary and Conclusion

In this chapter, we have shown that a solution exists to perform complex traffic engineering in Host-Centric IPv6 multihomed sites. The proposed solution is named NAROS. When a host in the multihomed site initiates a communication with a remote host, it contacts its NAROS server to determine the best IPv6 source address to use. By selecting the source address, the NAROS service can control how the locally initiated flows are distributed over the links with the providers. It thereby provides an inbound and outbound traffic engineering mechanism, without transmitting any BGP advertisement, and without affecting the interdomain BGP routing tables. The NAROS server does not maintain any per-host state, and can easily be deployed as an anycast service inside each site, which allows this service to scale. It can be set up independently from the providers, and the overhead induced by the NAROS service is very low, especially if the NAROS service and the DNS are coupled. This coupling of NAROS and DNS could also enable extended traffic engineering capabilities, and may be the preferred architecture for many sites. Changes are limited to hosts inside the multihomed site. Legacy hosts are still able to work, but their flows cannot be driven by the NAROS service.

We have evaluated the performance of the NAROS server on a real traffic, and have shown that the load on the NAROS server is reasonable and that, when used to load-balance the locally initiated traffic between two providers, the NAROS server obtains similar performance as classical CRC-16 based load-balancing mechanisms. Further simulations, using other traffic traces, should be performed to confirm these performance results.

Chapter 7

Scalable Route Selection for IPv6 Multihomed Sites

7.1 Introduction

Chapter 5 has shown that the use of multiple PA prefixes increases the number of interdomain paths available between IPv6 multihomed sites. NAROS was presented in Chapter 6 as a mechanism to enable Host-Centric IPv6 multihomed sites to control the selection of the interdomain path. The remaining issue studied in this chapter is which interdomain path to select, knowing that some interdomain paths provide much lower delays than others.

Let us consider two stub ASes, that have respectively m and n providers, and that use multiple prefixes to get multihomed in IPv6. The total number of interdomain paths between these two ASes is typically $m \times n$, i.e. many more than when traditional multihoming is used. Chapter 5 has shown that, among the newly available paths, some offer lower delays. Since the number of providers today ranges from 2 to more than 20, the number of paths available between two stub ASes is often higher than or equal to 4. AS a consequence, the selection of the *best* path may become more difficult.

This chapter focuses on how to identify low delay paths, in order to better exploit the path diversity leveraged by the use of Host-Centric IPv6 multihoming. The end-to-end delay is a major component of the performance of a path since

it reflects its length and bandwidth. Choosing paths that offer low delays is an important traffic engineering goal [7, 27, 48], in particular for many interactive and real-time applications.

One approach to identify the path with the lowest delay is to probe all available paths. However, measuring adequately the delay of a path typically requires many probes, distributed over time [8]. For example, *Resilient Overlay Network* (RON) uses at least 10 probes to estimate the latency of a path, with 14 seconds in average between two consecutive probes [9]. Besides the high measurement overhead imposed, the cost of probing all available paths to identify the one with the lowest delay will probably outweigh the benefit of an intelligent choice. This is particularly evident when sites use Host-Centric IPv6 multihoming. In this case, many interdomain paths exists between two multihomed sites, much more than when IPv4 multihoming is used. For instance, suppose that a multihomed site has about 2.5 providers on average. If each multihomed site has a significant amount of traffic towards 1000 other multihomed sites, then the average number of paths to probe is $1000 \times (2.5)^2 = 6250$ for each site.

The questions that we try to answer in this chapter are :

How to identify the interdomain paths with the "best quality" ?

Is it possible to identify the low delay paths for a given destination, without prior communication with this destination ?

Can we do that for any destination node in a scalable way, i.e. without actively probing all paths ?

Can we provide informations about the quality of paths without adding QoS attributes to BGP ?

We propose to use a network coordinate system as a scalable an efficient way to help hosts in selecting, for each pair of source and destination hosts, the IPv6 addresses that will lead to a low delay path. Our goal is to avoid all paths with really bad delays, and to use the path with the lowest delay as much as possible. Relying on RTT measurements from the RIPE NCC data set, our experiments show that, using synthetic coordinates, all paths with really bad delays can be avoided. Moreover, we are able to select paths with a delay at most 20% worse than the lowest delay for more than 85% of the pairs of multihomed sites. A second contribution is SVivaldi, an improved version of the Vivaldi distributed algorithm for computing synthetic coordinates. We show that SVivaldi produces more accurate coordinates, and is able to stabilise Vivaldi's coordinates. This work has been published in [70].

This chapter is organised as follows. In section 7.2, we explain how synthetic coordinates can help hosts in IPv6 multihomed sites to select their best paths towards a destination host. We next present and improve a distributed algorithm

that computes these coordinates in a scalable way. Next, in section 7.4, we evaluate the quality of the route selection. Finally, the related work is presented in section 7.6.

7.2 Identifying Low Delay Paths by Using Network Coordinate Systems

Synthetic coordinate systems have been originally developed to allow an Internet host to predict the round-trip delays to other hosts, without having to contact them first [141, 142, 157, 55]. Each host computes its synthetic coordinates in some coordinate space such that the distance between the synthetic coordinates of two hosts predicts the RTT between them. For example, if two hosts have coordinates x and y respectively in the coordinate space, the distance $\|x - y\|$ is a predictor of the RTT between them. One unit of distance in the coordinate space corresponds to a RTT of one millisecond.

As explained in Chapter 6, section 6.2, IPv6 hosts currently arbitrarily choose between two or more global-scope IPv6 source and destination addresses. The address selection algorithm specifies that the pair of *(source, destination)* addresses with the longest matching prefix must be preferred [73]. This pair of addresses entirely determines the interdomain path used. The selection of the interdomain path is thus arbitrary, unless hosts have some clue on how to select the pair of source and destination addresses.

We propose that hosts in IPv6 multihomed sites base their source and destination IPv6 address selection on the predicted delays provided by synthetic coordinates. The selection of those addresses is made once at the start of a flow (e.g. TCP connection) between two hosts. When the addresses are selected, they do not change during the connection to avoid packet re-ordering.

Each host can compute its own coordinates, for every assigned IPv6 address. We propose that the hosts dynamically update their DNS server when their coordinates change. DNS extensions can be used to allow hosts to securely update their DNS records [210, 79, 208]. In most cases however, hosts within a single IPv6 prefix should have approximately the same coordinates. The coordinates of a prefix should thus be a good estimate of the coordinates of any host within this prefix. In such cases, we propose that the DNS server of the site computes the coordinates of its few assigned prefixes, and publishes them in the Domain Name System. If a site specifies the coordinates of a network or subnet, it is not required to specify the coordinates for each individual host. For example, a computer lab with 24 workstations, all of which are on the same subnet and in basically the same location, would only need a single coordinate for the subnet. However, if the location of the Web server must be more precisely estimated, a separate coordinate for it can be placed in the DNS.

The synthetic coordinates can be advertised using a new DNS Resource Record (RR), with mnemonic SLOC. This resource record is detailed in Appendix B, and we provide an implementation of the SLOC RR at [60], for the ISC BIND 9.3.0 DNS server [115]. The creation of a new resource record instead of using the TXT resource record is the recommended way to store new data in the DNS [81]. In practice, the SLOC resource record can be associated directly with the domain name of a host, so that the coordinates and the domain name of a host can be provided together in a single DNS response message.

Figure 7.1 illustrates two dual-homed IPv6 sites. The DNS server in AS 65001 has computed coordinates $(15, 25)$ and $(21, 30)$ for its prefixes `2001:10:1::/48` and `2001:20:1::/48` respectively, while the DNS server in AS 65002 has computed coordinates $(5, 12)$ and $(2, 16)$ for its prefixes `2001:30:1::/48` and `2001:40:1::/48` respectively. Note that we could use other coordinate spaces than the 2-dimensional Euclidean one, for instance 3-dimensional Euclidean or Spherical coordinates.

In order to predict the lowest delay path towards a host Y in AS 65002, a host X in AS 65001 first issues a DNS request for the addresses and coordinates of Y. For example in Figure 7.1, Host X will receive coordinates $(5, 12)$ and $(2, 16)$ associated with addresses `2001:30:1::y` and `2001:40:1::y` respectively. Next, Host X uses those coordinates together with the coordinates of its own addresses to compute the predicted RTT for all possible couples of (source, destination) addresses. In the example, Host X will compute the predicted RTT for four AS paths. These paths and their corresponding RTT predictions are shown in Table 7.1 [1]. In this example, the path with the lowest delay is obtained by using addresses `2001:10:1::x` and `2001:40:1::y`, with an estimated RTT of $\|(15, 25) - (2, 16)\| = 15.8ms$.

The best path can thus be predicted using a single DNS request, instead of performing four series of delay measurements, each involving several probes. Moreover, the coordinates should not change frequently. They can be cached in the DNS resolvers, further reducing the cost associated to the prediction of the best path. In the previous example, each host determines by itself the destination address to use. Another option is that the DNS server makes this choice on behalf of the host. For instance by removing from the DNS response message the destination addresses that lead to low quality paths.

The accuracy of the predicted delays depends on the synthetic coordinates computed for each node. Any algorithm like GNP[141], NPS [142], BBS [181] [182], Vivaldi [55], Virtual Landmarks [191], or Lighthouse [157] can be used to assign these synthetic coordinates to hosts. In this chapter, we evaluate the

[1] Note that the distance between prefixes `2001:10:1::/48` and `2001:20:1::/48` is not null because it corresponds to the delay of the path going through AS 10 and AS 20

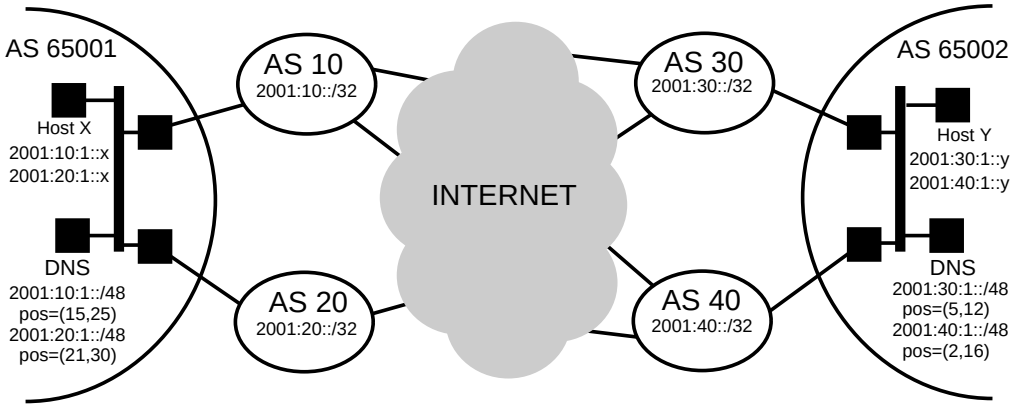


Figure 7.1. Both dual-homed IPv6 sites have computed the coordinates associated to their prefixes. Those coordinates are published using the DNS.

Source Prefix	Dest. Prefix	AS Path	Estimated Distance
2001:10:1::/48	2001:30:1::/48	$10 \rightarrow 30$	$\|\|(15,25)-(5,12)\|\| = 16.4ms$
2001:20:1::/48	2001:30:1::/48	$20 \rightarrow 30$	$\|\|(21,30)-(5,12)\|\| = 24.1ms$
2001:10:1::/48	2001:40:1::/48	$10 \rightarrow 40$	$\|\|(15,25)-(2,16)\|\| = \mathbf{15.8}ms$
2001:20:1::/48	2001:40:1::/48	$20 \rightarrow 40$	$\|\|(21,30)-(2,16)\|\| = 23.6ms$
2001:10:1::/48	2001:20:1::/48	$10 \rightarrow 20$	$\|\|(15,25)-(21,30)\|\| = 7.8ms$
2001:30:1::/48	2001:40:1::/48	$30 \rightarrow 40$	$\|\|(5,12)-(2,16)\|\| = 5ms$

Table 7.1. RTTs estimated by coordinates

use of the Vivaldi decentralised network coordinate system, because it has the advantages of being simpler and fully decentralised.

7.3 Computing Stable Synthetic Coordinates

Vivaldi is a light-weight, adaptative and fully distributed algorithm for computing synthetic coordinates for Internet nodes. It does not require any fixed infrastructure and can compute coordinates for itself after collecting latency information from only a few other nodes. This number of nodes is constant and does not depend on the total number of nodes. For instance, a node sends probes to a few tens of distant nodes. Based on these delay measurements, the node can compute its own coordinates. The main interest of the coordinate system is that the Euclidean distance between the coordinates of two nodes is a good prediction of the round-trip-time between the two nodes in the Internet.

Furthermore, the use of coordinates to estimate delays between nodes has a much lower overhead than active probing. Assume a network with n nodes, where each node wants to know its end-to-end delay with any other node in the network. When active probing is used, each node must probe $n-1$ other paths. In this case, a total of $n \times (n-1)$ paths are probed, which is in $O(n^2)$. Instead, when using synthetic coordinates, each node needs only to probe a constant number c of neighbours to compute its coordinates. In this case, a total of $c \times n$ paths are probed, which is in $O(n)$.

7.3.1 The Simple and Adaptative Vivaldi Algorithms

Conceptually, Vivaldi places a spring between each pair of nodes (i, j) with a rest length set to the known RTT [55]. The current length of the spring is considered to be the distance between the nodes in the coordinate space. Vivaldi minimises the spring energy in the system by simulating the movements of nodes under the spring forces.

Alg. 1 shows the pseudo-code for the simple Vivaldi algorithm, as originally described [55]. Initially, all nodes starts from the same location. To separate them, Vivaldi defines u(0) as a unit-length vector in a randomly chosen direction. When a node i with coordinates $\vec{x}_i$ measures a RTT of rtt_{ij} towards node j with coordinates $\vec{x}_j$, it updates its coordinates using the following rule :

$$\vec{x}_i = \vec{x}_i + \delta \times (rtt_{ij} - \|\vec{x}_i - \vec{x}_j\|) \times \vec{u}(x_i - x_j)$$

where the scalar quantity $(rtt_{ij} - \|\vec{x}_i - \vec{x}_j\|)$ is viewed as the displacement of the spring from the rest, and the unit vector $\vec{u}(x_i - x_j)$ gives the direction of the force on node i. In other words, node i will move towards or away from node j based on the magnitude of the error, and on a constant time step δ.

Algorithm 1. The Simple Vivaldi algorithm

// Node j has been measured to be rtt ms away,
// and Node j says it has coordinates $\vec{x}_j$.
// Constant δ is a tuning parameter.
simple_vivaldi(rtt, $\vec{x}_j$)
{
    // Compute error of this sample. (1)
    $e = rtt - \|\vec{x}_i - \vec{x}_j\|$

    // Find the direction of the force the error is causing. (2)
    $dir = \vec{u}(x_i - x_j) = \frac{(\vec{x}_i - \vec{x}_j)}{\|\vec{x}_i - \vec{x}_j\|}$

    // The force vector is proportional to the error. (3)
    $\vec{f} = \vec{dir} \times e$

    // Move a small step in the direction of the force. (4)
    $\vec{x}_i = \vec{x}_i + \delta \times \vec{f}$
}

The rate of convergence of the simple Vivaldi algorithm is governed by this δ time step. Large δ values allow the nodes to adapt quickly their coordinates, but cause oscillations and failures to converge to useful coordinates. In order to both obtain fast convergence and to avoid oscillations, it is suggested to vary δ depending on how certain the node is about its coordinates [55]. The proposed adaptative time step δ is computed as :

$$\delta = c_c \times \frac{local\ error}{local\ error + remote\ error} \tag{7.1}$$

where the constant c_c is a tuning parameter.

Using this δ, an accurate node that samples an inaccurate node will not move much, an inaccurate node that samples an accurate node will move a lot, and two nodes of similar accuracy will move towards each other by the same distance.

Algorithm 2 shows the pseudo-code for the adaptative Vivaldi algorithm, as described in [55]. The Vivaldi procedure uses each RTT sample to update the coordinates of a node. The weight w of a sample is based on the ratio between the local and the remote error estimates (line 1). The algorithm tracks the local relative error by using a weighted moving average (lines 2 and 3). The coordinates are updated by moving a small step towards the position that best reflects the RTT measured. The amplitude of the move depends on the weight of the sample, and on the difference between the measured RTT and the predicted RTT (lines 4 and 5).

Algorithm 2. The Adaptative Vivaldi algorithm

// Node j has been measured to be rtt ms away, has coordinate $\vec{x}_j$,
// and an error estimate of e_j.
//
// Our own coordinates and error estimate are $\vec{x}_i$ and e_i.
//
// The constants c_e and c_c are tuning parameters.
adaptative_vivaldi(rtt, $\vec{x}_j$, e_j)
{
    // Sample weight balances local and remote error. (1)
    $w = e_i/(e_i + e_j)$

    // Compute relative error of this sample. (2)
    $e_s = |\|\vec{x}_i - \vec{x}_j\| - rtt| \, / rtt$

    // Update weighted moving average of local error. (3)
    $e_i = e_s \times c_e \times w + e_i \times (1 - c_e \times w)$

    // Update local coordinates, with the adaptative δ. (4) (5)
    $\delta = c_c \times w$
    $\vec{x}_i = \vec{x}_i + \delta \times (rtt - \|\vec{x}_i - \vec{x}_j\|) \times \frac{(\vec{x}_i - \vec{x}_j)}{\|\vec{x}_i - \vec{x}_j\|}$
}

7.3.2 Vivaldi's Unstable Coordinates

The adaptative Vivaldi algorithm quickly converges towards a solution when latencies satisfy the triangle inequality, which states that the distance directly between two nodes a and c must be less than or equal to the distance along any path going through an intermediate node b. Unfortunately, in the Internet, the latencies sometimes violate this inequality, due for example to policy-based routing with BGP. In such cases, we will show that the nodes may never converge towards stable coordinates, if the original Vivaldi algorithm is used.

In order to illustrate this phenomenon, we simulate the Vivaldi algorithm using RTT measurements collected from 58 active test boxes from the RIPE NCC Test Traffic Measurements Service. The RIPE measurement configuration is described in [87]. The test boxes are scattered over Europe and a few are located in the US, Australia, New Zealand and Japan. Every test box is equipped with a GPS clock so that the one-way delays between each pair of boxes can be measured accurately (within 10μs). Every day, more than 2000 probes are performed for each test box pair. The interval between two consecutive probes is randomised according to a Poisson distribution, as recommended in [8]. The measurements also include packet losses, path information, bandwidth and delay variations. They are regularly used by ISPs and in the literature. In this work,

we use only the RTTs computed as the sum of the two one-way delays, between every pair of test boxes. In this simulation, 2-dimensional Euclidean coordinates are used for the sake of simplicity. As in [55], the Vivaldi simulations are driven by a matrix of inter-host Internet RTTs. Vivaldi computes coordinates using a subset of the RTTs, and the full matrix is needed to evaluate the quality of the predictions made by using those coordinates.

Figure 7.2 shows the evolution of the coordinates chosen by Vivaldi for the nodes of the RIPE data set. The figure shows that all the nodes constantly update their coordinates, making the whole system to shift slowly. While the relative distances between individual nodes remain approximately stable, the set of nodes as a whole is moving and rotating slowly.

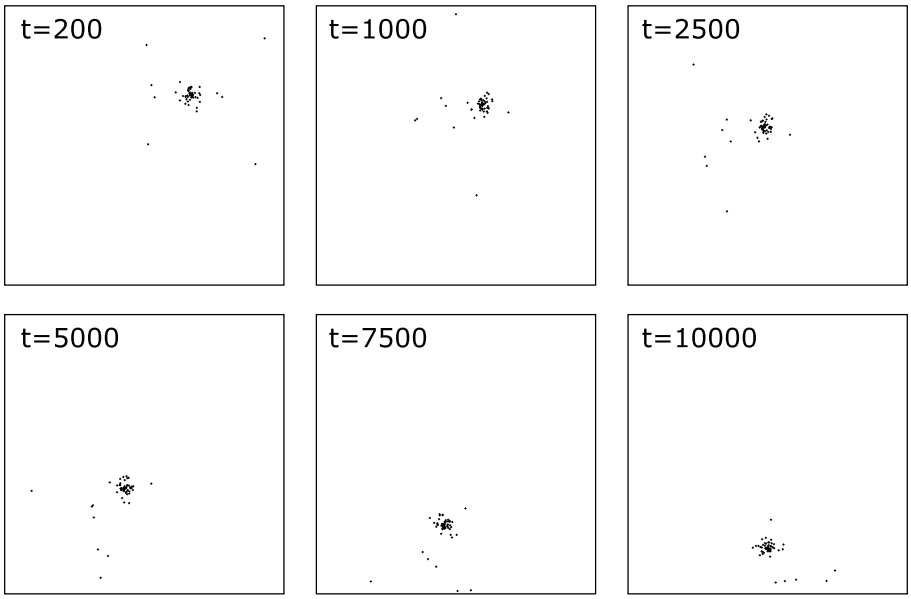


Figure 7.2. Evolution of the Vivaldi coordinates of the RIPE nodes

This phenomenon occurs even with a small system composed of only three nodes that violate the triangle inequality. Such a system is illustrated on top of Figure 7.3. The algorithm that computes synthetic coordinates should minimise the sum of prediction errors for all nodes, i.e. either

$$L1 = \sum_i \sum_j |rtt_{ij} - \|\vec{x}_i - \vec{x}_j\||$$

or

$$L2 = \sum_i \sum_j (rtt_{ij} - \|\vec{x}_i - \vec{x}_j\|)^2$$

An optimal analytical solution for the illustrated example can be computed easily if we use the analogy with a spring system. With such system in mind, when all nodes have reached a stable position, node b will be attracted by both nodes a and c, while nodes a and c will repulse each other. Hence, we can assume that the stable position of node b will certainly stand on the line segment between nodes a and c. In this case, the problem is reduced to the computation of positions x_b and x_c in the system given on Figure 7.3, where the position of node a is fixed to 0.

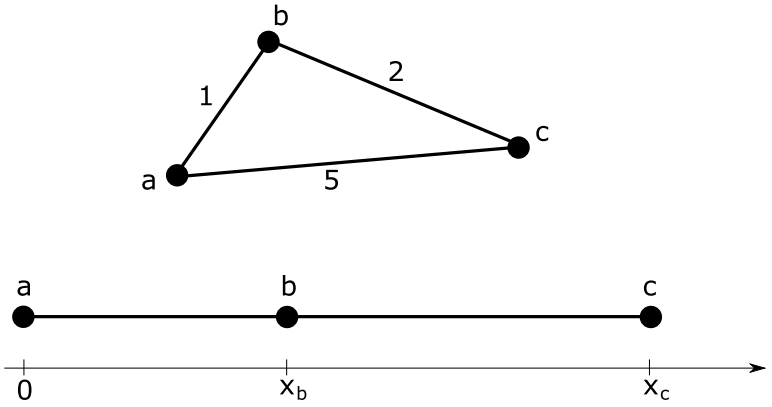


Figure 7.3. The RTTs between nodes a, b and c violate the triangle inequality (top). The optimal solution for the system can be computed easily if we observe that node b will stand on the line segment between nodes a and c (bottom).

The x_b and x_c coordinates must minimise L2, the sum of prediction errors, i.e. x_b and x_c must minimise :

$$f(x_b, x_c) = (x_b - 1)^2 + ((x_c - x_b) - 2)^2 + (x_c - 5)^2$$

The solutions for x_b and x_c are given by computing the roots of the partial derivative functions in x_b and x_c :

$$\begin{cases} \frac{\partial f}{\partial x_b} = 2(x_b - 1) - 2((x_c - x_b) - 2) = 0 \\ \frac{\partial f}{\partial x_c} = 2((x_c - x_b) - 2) + 2(x_c - 5) = 0 \end{cases}$$

$$\Leftrightarrow \quad \begin{cases} x_c = 2x_b + 1 \\ x_b = 2x_c - 7 \end{cases} \quad \Leftrightarrow \begin{cases} x_c = 13/3 \\ x_b = 5/3 \end{cases}$$

Hence the optimal solution is when nodes a, b and c are aligned, and when the distances (a, b), (b, c) and (a, c) are respectively 5/3, 8/3 and 13/3, as depicted in Figure 7.4. In this case, L1 equals 4 and L2 equals 8/3, i.e. 2.66.

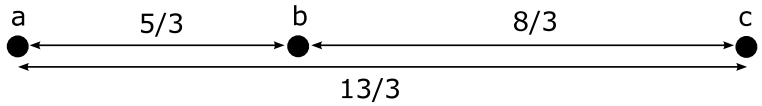


Figure 7.4. Optimal distances between the three nodes

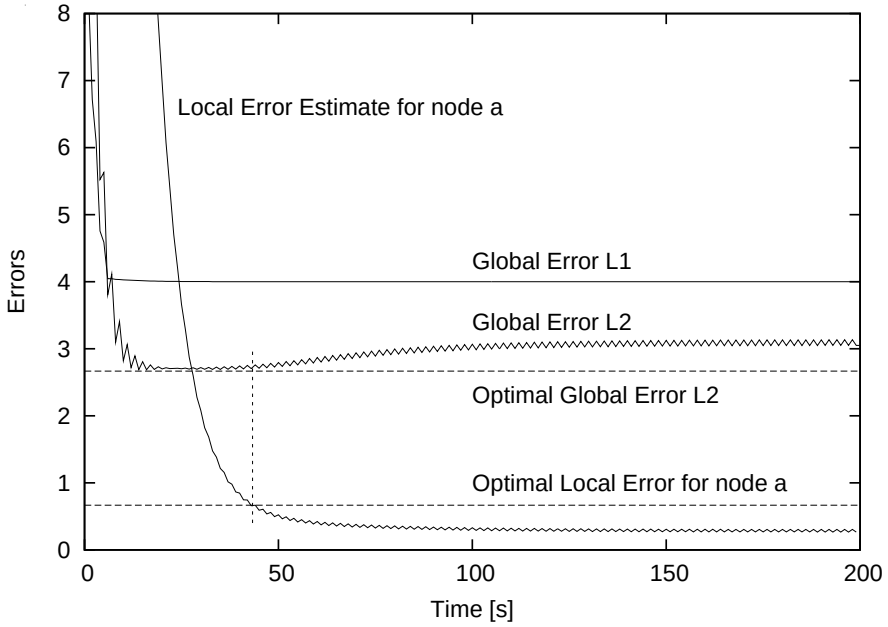


Figure 7.5. Evolution against time of the global errors L1 and L2, together with the local error estimation of node a, for the Vivaldi algorithm

Figure 7.5 shows the evolution against time of the global errors L1 and L2, together with the evolution of the local error e_s (Algorithm 2, line 2) estimated by node a. We can observe that the Vivaldi algorithm quickly finds coordinates that minimise error L1, but fails to produce coordinates that minimise the global error L2. The figure shows that the global error L2 first reaches the optimum, but next rises and stabilises around a value slightly above 3, while the optimum is 2.66. We can observe that the global error L2 starts moving away from the optimum when the local error estimated by node a passes under its optimum value (0.66), as illustrated by the vertical dotted line in Figure 7.5.

Worse, the global error L2 oscillates between two values, and Vivaldi fails to provide stable coordinates for the nodes, as illustrated by Figure 7.6. This figure shows that the tree nodes move along a line, never reaching stable coordinates. An animation of the system is available at [62].

This behaviour can be explained by looking at how the nodes update their coordinates. Figure 7.7 illustrates again the system composed of the three nodes a, b and c. Assume that the system has reached the optimum solution described in

Figure 7.4. At step 1, node a measures its RTT with node b. Since the measured RTT is 1ms and the current distance between nodes a and b is 1.67, node a will move one step towards node b, in order to minimise its local prediction error. At step 2, node b probes node c. It observes that the measured RTT is 2ms while the current distance is 2.67. Hence, node b will move one step towards node c, in order to reduce its distance from c. Finally, at step 3, node c measures a 5ms RTT with node a. Since the distance between their coordinates is only 4.33, node c will move a step away from node a, in order to increase its distance with node a. At this point, the system is back to the initial situation, except that the three nodes have all moved one step Δ in the same direction. The evolution of the system on a longer time scale is illustrated on Figure 7.6.

This example shows that the coordinates provided by the Vivaldi algorithm can easily be unstable when latencies violate the triangle inequality, even if the system is composed of very few nodes, and even if it has reached an optimum. It shows that the violation of the triangle inequality is the cause of the coordinates stability problem with the RIPE data set. Such behaviour is unacceptable in our case since it would require that hosts constantly update the DNS with their new coordinates.

7.3.3 SVivaldi : a Stable Vivaldi Algorithm

In this chapter, we propose two modifications to the original Vivaldi algorithm. The first is to improve the local error estimate, such that each node can compute a better estimation of the accuracy of its coordinates. The second is to introduce a new factor to prevent the system from oscillating indefinitely. The new algorithm is presented in Algorithm 3.

A More Accurate Local Error Estimate

Our first modification is to use a more accurate local error estimate, in order to improve the convergence of coordinates. In the adaptative Vivaldi, each node maintains an estimate of the accuracy of its coordinates. The error estimates are used by nodes to compute their adaptative time step δ, computed by Equation 7.1. This adaptative time step is used by a node to quickly converge when it is still learning its rough place in the network, and to move slowly when it refines its position. In the adaptative Vivaldi algorithm, a node estimates its local error by maintaining a moving average of recent relative errors. At each step, the local error estimate is modified so that it better matches the error for the node being sampled (Algorithm 2, line 3). Consequently, the Vivaldi algorithm gives much importance to the RTT measured with the last node. The prediction error depends on the neighbour node considered, i.e. the distances of some nodes are better predicted that the distances of other nodes. Thus, the local error estimate may vary each time the node samples another neighbour. Those variations prevent the node from having a stable and accurate local error estimate, which in turn can

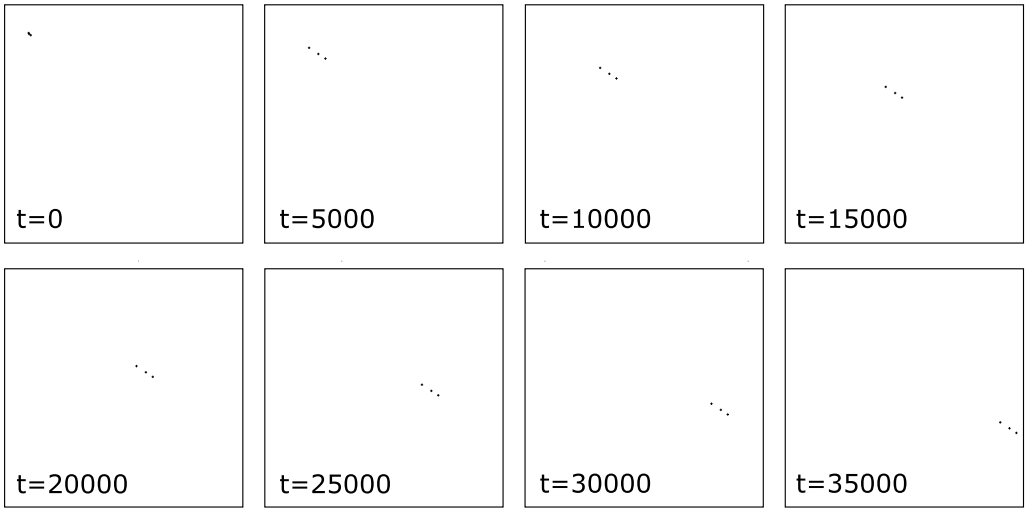


Figure 7.6. Evolution of the Vivaldi coordinates of the three nodes

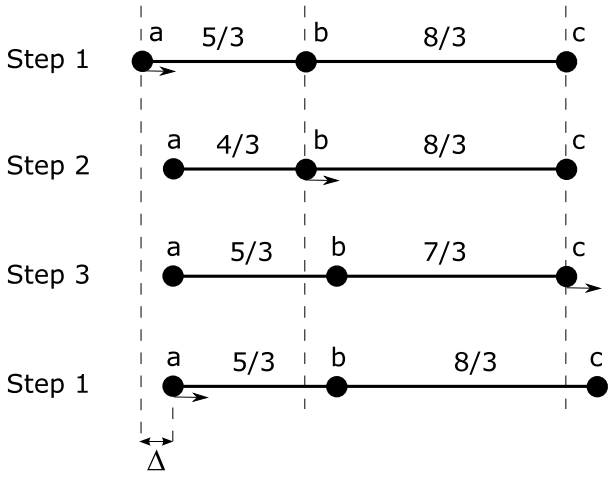


Figure 7.7. Steps of the Vivaldi algorithm making the coordinates of the three node diverge

Algorithm 3. The SVivaldi algorithm

// Node j has been measured to be rtt_{ij} ms away, has coordinate $\vec{x}_j$, and an
// error estimate of e_j. Our own coordinates and error estimate are $\vec{x}_i$ and e_i.
//
// The set of our neighbours is $Neigh_i$. The set of the last rtt measured to
// our neighbours is RTT_i.
//
// The constants c_e, c_c and c_f are tuning parameters.
// The $loss_i$ factor is initialised to null for all nodes i.

svivaldi(rtt_{ij}, $\vec{x}_j$, e_j)
{

// Sample weight balances local and remote error. (1)

$w = e_i/(e_i + e_j)$

// Cache the last rtt_{ij} and update the set of neighbours. (2) (3)

$RTT_i[j] = rtt_{ij}$

if $(j \notin Neigh_i)$ **then** $Neigh_i = Neigh_i \cup \{j\}$

// Compute local error estimate. It is computed as the mean of the
// differences between the predicted and the measured RTTs, over
// all our neighbours. (4)

$e_i = \frac{\sum_{j \in Neigh_i} |\|\vec{x}_i - \vec{x}_j\| - RTT_i[j]|}{\#Neigh_i} \times c_e + e_i \times (1 - c_e)$

// Lose energy, i.e. update the loss factor. (5)

$loss_i = c_f + (1 - c_f) * loss_i$

// Update local coordinates. (6) (7)

$\delta = c_c \times w \times (1 - loss_i)$

$\vec{x}_i = \vec{x}_i + \delta \times (rtt - \|\vec{x}_i - \vec{x}_j\|) \times \frac{(\vec{x}_i - \vec{x}_j)}{\|\vec{x}_i - \vec{x}_j\|}$

}

sometimes prevent it to find an optimum solution, as illustrated by Figure 7.8 with the system composed of three nodes.

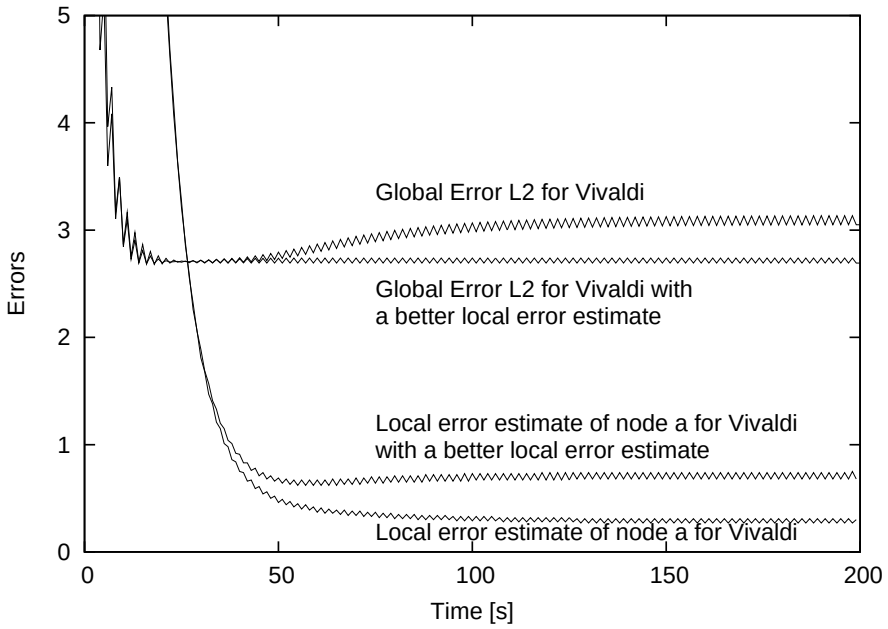


Figure 7.8. Influence of the improved local error estimate on the global error L2 for the system with three nodes

As a more accurate local error estimate we propose the mean of the absolute differences between the predicted RTT for neighbour i and the actual RTT measured for this neighbour, for any neighbour i (Algorithm 3, line 4). In other words, a node retains the last prediction error computed for each of its neighbour, and computes the local error estimate as the average of those prediction errors. This estimate only requires that a node retains the last RTT measured for each of its neighbours (Algorithm 3, lines 2 and 3). This local error estimate gives the same weight to each neighbour, in contrast with Vivaldi's local error estimate where the probe with the last node has a heavier weight.

Figure 7.8 compares the Vivaldi algorithm with and without the new local error estimate. It illustrates the evolution against time of the global error L2, for the system composed of three nodes. The figure shows that the newly proposed local error estimate allows the Vivaldi algorithm to find better coordinates for the nodes. In particular, it allows here to find coordinates that are very close to the optimum solution. We will show in section 7.3.4 that this improved local error estimate also allows us to find better coordinates for the nodes of the RIPE data set.

A Loss Factor

While a more accurate local error estimator improves the solution, we can notice in Figure 7.8 that it still does not prevent the system from oscillating.

As said earlier, the Vivaldi algorithm simulates a physical spring between each pair of nodes (i, j), with a rest length set to the measured rtt_{ij}. When the triangle inequality is not always staisfied, some of those springs can oscillate indefinitely around their rest length, and make the whole system move. This behaviour was previously explained in section 7.3.2.

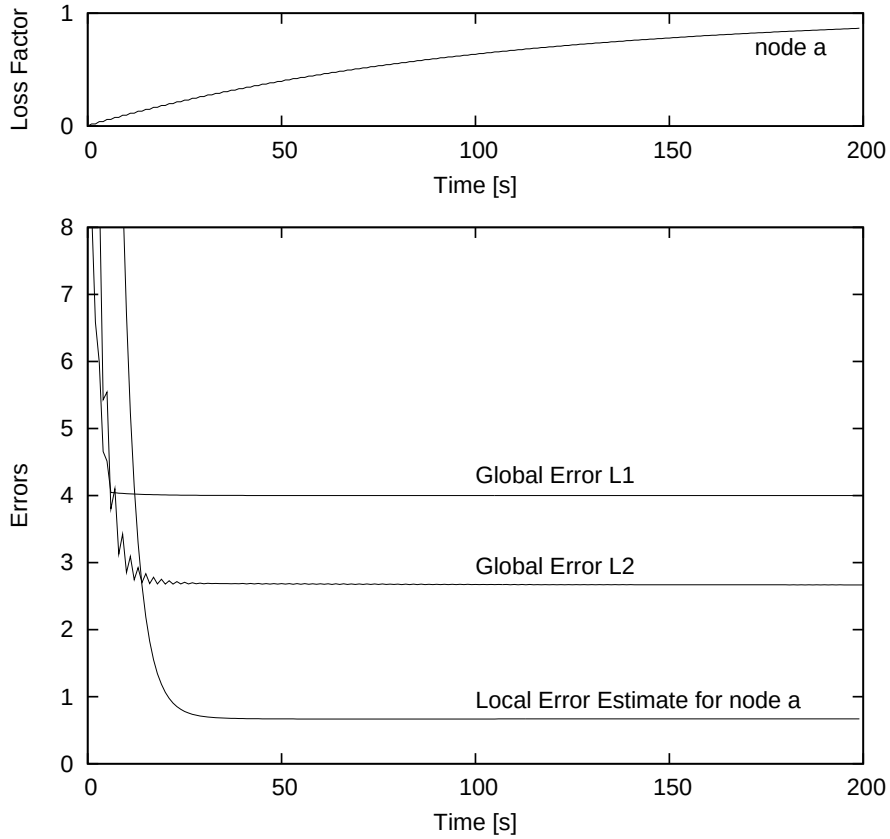


Figure 7.9. Evolution against time of the global errors L1 and L2, together with the local error estimation of node a, for the SVivaldi algorithm (bottom). Evolution against time of the loss factor for node a (top).

Our second modification to Vivaldi is to introduce a *loss* factor to allow these springs to progressively rest in a local minimum (Algorithm 3, line 5). The *loss* factor comes from the analogy with the springs. As the springs oscillate, some energy is lost due to friction. The friction force slows the nodes down. It depends on the normal force, i.e. the difference between the measured and predicted RTTs,

and on a constant friction coefficient $c_f < 1$. c_f controls the quantity of energy that is lost at each oscillation. The value c_f is a tradeoff between accurate coordinates and short convergence times. A high value for c_f quickly stabilises the coordinates but prevents the system from converging to accurate coordinates. A low value for c_f allows the system to find accurate coordinates at the price of a longer convergence period. Our experiments show that a value of $c_f = 0.02$ is a good compromise, so we use it in our simulations. The *loss* factor (Algorithm 3, line 5) is set to zero when the node starts to converge, and gradually grows to 1, as illustrated by Figure 7.9 (top). Whenever a significant change is observed for the measured RTT with a neighbour, the *loss* factor can be reset to zero in order to let the node move again.

We now evaluate the accuracy of the coordinates provided by our improved algorithm. Figure 7.9 (bottom) shows the evolution of the global errors L1 and L2 of the system composed of three nodes. It also gives the local error estimate of node a. The figure shows that the introduction of a loss factor allows the nodes to converge to stable coordinates, and in this case, to reach the optimum solution. We will show in the next section that this conclusion is also true for a larger number of nodes.

A friction coefficient has already been used in another coordinate algorithm, called Big-Bang simulation (BBS) [181]. Their authors have shown that the introduction of friction allows their algorithm to find better coordinates. However, their algorithm seems to require global knowledge of the system, and it is not clear how to decentralise it.

Our new version of the Vivaldi algorithm, detailed in Algorithm 3, is called *SVivaldi* throughout this chapter. This algorithm includes the improved local error estimate, described in the previous section, and the loss factor.

7.3.4 Evaluation of SVivaldi

We evaluate in this section the stability and the accuracy of the coordinates provided by SVivaldi, and compare them with Vivaldi. The evaluation is performed by simulating the two algorithms with the same data set. The simulations rely on the full matrix of Internet RTTs between 59 nodes of the RIPE data set. Vivaldi and SVivaldi use a subset of these RTTs, but the full matrix is used to evaluate the accuracy of predictions made by using those coordinates. For the sake of simplicity, 2-dimensional Euclidean coordinates are used in order to compare the behaviours of the SVivaldi and Vivaldi algorithms. Section 7.4.1 will present other coordinate spaces that can be used with those algorithms. Each node takes measurements every second from another node, chosen uniformly at random in a set of 20 neighbour nodes.

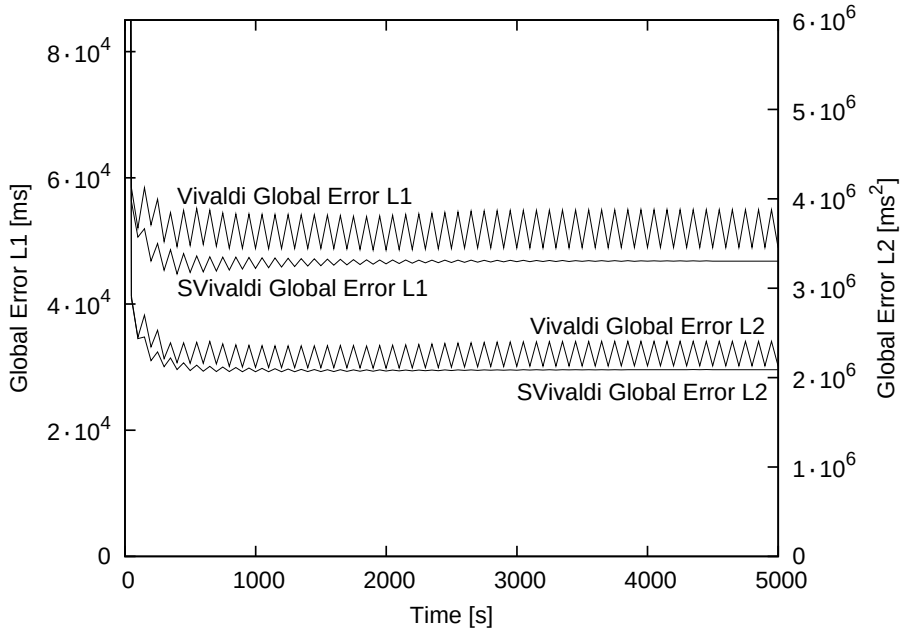


Figure 7.10. Evolution against time of the global errors L1 and L2 for 2D coordinates chosen by Vivaldi and SVivaldi

Figure 7.10 shows the evolution against time of the global errors L1 and L2 for the coordinates chosen by SVivaldi and Vivaldi. The figure shows that, when the Vivaldi algorithm is used, nodes constantly update their coordinates, making both global errors L1 and L2 oscillate around a constant value. Instead, the *loss* factor of our improved SVivaldi algorithm allows the nodes to converge to stable coordinates. In Figure 7.10, both errors L1 and L2 for SVivaldi progressively stop oscillating. After about 30 minutes, the coordinates are practically stable for all nodes. Additionally, Figure 7.10 suggests that the improved SVivaldi produces slightly more accurate coordinates than the Vivaldi algorithm.

Figure 7.11 compares the cumulative distribution of relative prediction errors for the 2-dimensional Euclidean coordinates chosen by Vivaldi and SVivaldi for the nodes of the RIPE data set. As the coordinates provided by the algorithms depend on the neighbours each node chooses, the figure shows the median distribution among 100 simulations with different sets of neighbours for each node. The 5^{th} and 95^{th} percentiles are also indicated for SVivaldi. The percentiles for the Vivaldi algorithm are similar but are not shown for graphical reasons.

Figure 7.11 shows that the relative error of the delays predicted by the use of 2-dimensional coordinates is less than 50% for more than 80% of pairs of nodes. The figure shows that the SVivaldi algorithm produces more accurate coordinates than the Vivaldi algorithm, at least for the RIPE data set. The improvement is due to its improved local error estimator. In Figure 7.11, a non negligible fraction

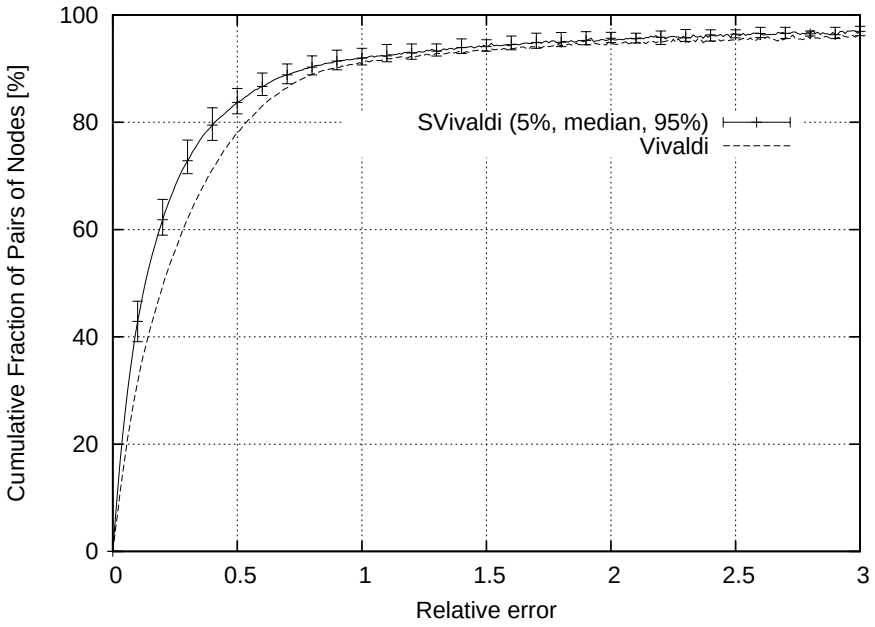


Figure 7.11. Cumulative distribution of prediction errors for 2D coordinates chosen by Vivaldi and SVivaldi

of relative prediction errors are higher than 300%. These large relative errors appears mostly for the prediction of small absolute delays, and the use of other coordinate spaces with more dimensions can reduce those errors.

We have also verified that SVivaldi preserves Vivaldi's ability to cope well with large numbers of newly-joined nodes with inconsistent coordinates. Figure 7.12 illustrates SVivaldi's robustness against high-error nodes. The simulation is identical to the one used in [55]. It starts with 200 nodes that already know coordinates that predict latency well. At $t = 0$, 200 new nodes are added to the system. Figure 7.12 shows the evolution against time of the coordinates of the nodes. It shows that the structure of the original nodes is preserved while new nodes join, i.e. new high-error nodes do not significantly disturb old low-error nodes.

7.4 Evaluation of the Quality of the Route Selection

We evaluate in this section how the use of the SVivaldi coordinate system can help in the selection of the lowest delay path between two multihomed IPv6 sites.

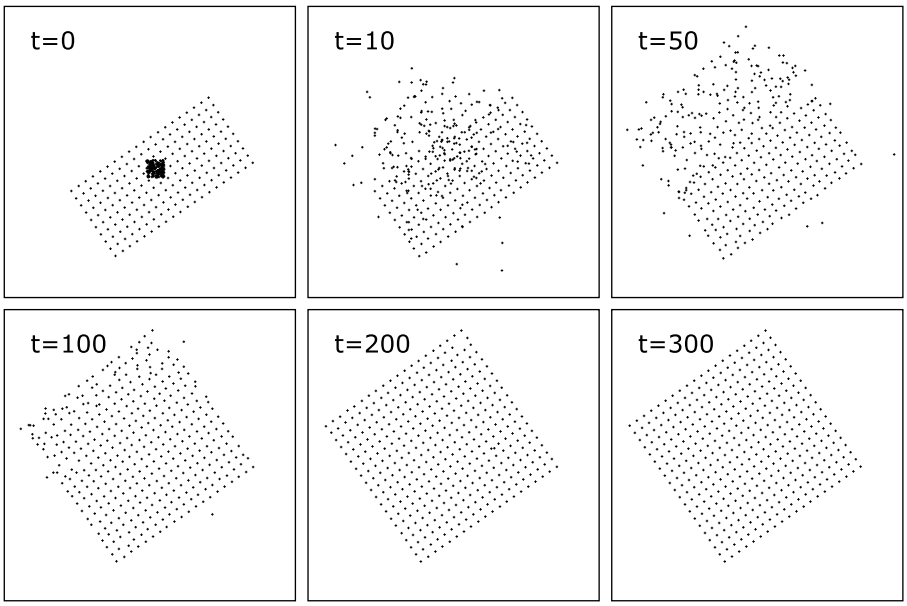


Figure 7.12. Evolution of a stable 200-node network after 200 new nodes join, using the new SVivaldi algorithm

7.4.1 Coordinate Space

Vivaldi and SVivaldi can operate with any coordinate space by redefining the coordinate substraction, vector norm, and scalar multiplication operations. Coordinate spaces that were considered by Vivaldi's authors are Euclidean coordinates, spherical coordinates and an Euclidean coordinate augmented with a height.

In this evaluation, we used this last coordinate space, i.e. Euclidean coordinates augmented with a height, as suggested by Vivaldi's authors [55]. In the height vector model, the usual vector operations are redefined as follows :

$$\begin{aligned} [x_1, \ldots, x_n,\ x_h] - [y_1, \ldots, y_n,\ y_h] &= [x_1 - y_1, \ldots, x_n - y_n,\ x_h + y_h] \\ \|[x, \ldots, x_n,\ x_h]\| &= \sqrt{x_1^2 + \ldots + x_n^2}\ + x_h \\ \alpha \times [x_1, \ldots, x_n,\ x_h] &= [\alpha x_1, \ldots, \alpha x_n,\ \alpha x_h] \end{aligned}$$

A packet sent from one node to another must first travel the source node's height, next travel in the Euclidean space, next travel the destination node's height. [55] has shown that so called height vectors perform better than both 2D and 3D Euclidean coordinates. Even more dimensions can be used to gain precision, but with diminishing marginal gain. Liying and Crovella [191] have shown that 6 or 7 dimensions are sufficient for accurate embedding of network distances of several datasets in an Euclidean space.

Another promising coordinate space is an hyperbolic space with an optimal curvature. This space is used in the BBS algorithm [182], and appears to give a better accuracy for the coordinates. Its usage by the SVivaldi algorithm is reserved for further work.

7.4.2 Multihoming Simulation

IPv6 multihoming with multiple prefixes is not currently deployed in the Internet. In order to simulate IPv6 multihoming, we follow a similar methodology to the one used in [4, 126]. We emulate a multihoming site by selecting a few RIPE nodes in the same metropolitan area, and use them collectively as a stand-in for a multihomed network. This is illustrated by Figure 7.13, where two RIPE nodes situated in Brussels but belonging to two different providers are used together to emulate a single IPv6 dual-homed site situated in Brussels.

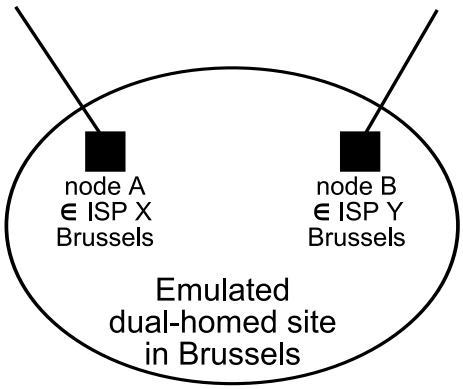


Figure 7.13. Emulation of an IPv6 multihomed site by grouping two nodes in the same metropolitan area, but belonging to two different providers

This method actually models IPv6 multihoming where the provider-dependent prefixes advertised by the virtual site are aggregated by its providers. In our data set, a total of 13 multihomed sites are emulated by this method, a number similar to the study of Akella et al. on multihoming [126]. In our study, 10 sites are dual-homed, 1 is 3-homed, 1 is 4-homed and a last one has 8 providers. One multihomed site is located in the US, one in Japan, and all the others in Europe. Unfortunately, since the BGP routing tables of the RIPE test boxes are not available, we cannot compare the SVivaldi path selection directly with the BGP path selection.

The conclusions discussed in the next sections are drawn only from simulations with the RIPE dataset. These conclusions should be confirmed by simulations using other datasets. There is however difficulties in obtaining datasets such as the RIPE one, where precise measures of delays are available for a large number of nodes and for several consecutive days.

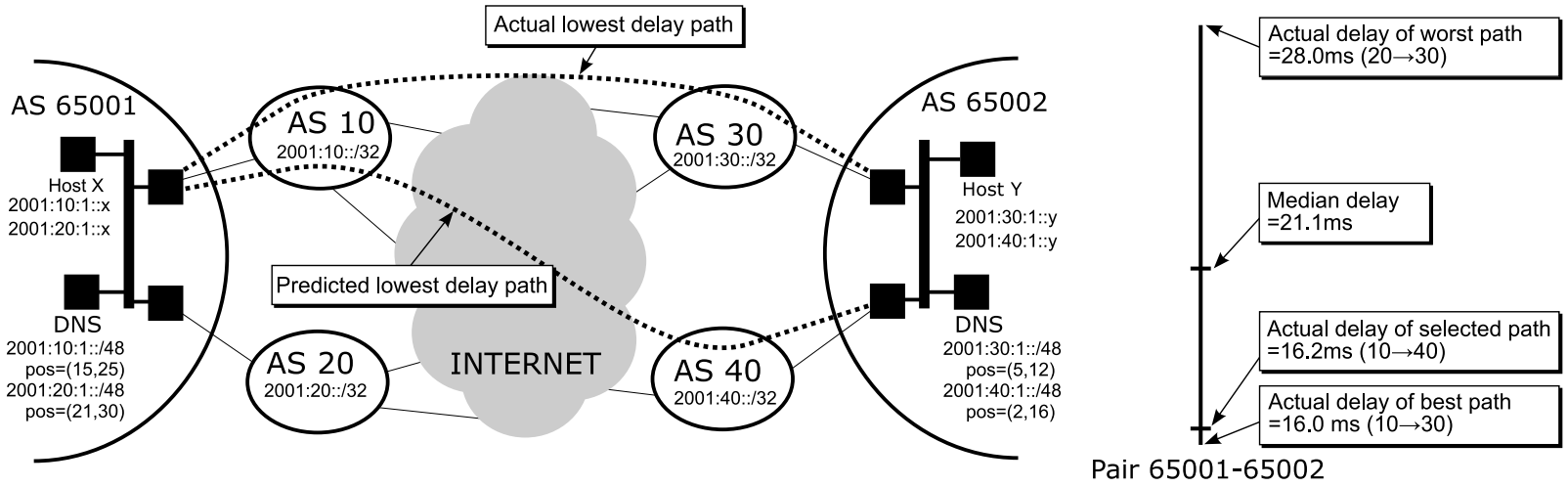
7.4.3 Simulation Results

In order to evaluate the quality of the paths selected by SVivaldi, we consider every possible pair of multihomed sites. Since our data set consists in 13 emulated multihomed sites, a total of 78 pairs are considered. Figure 7.14 illustrates a pair of multihomed sites. For each pair, we first extract from the data set the actual RTTs of every paths between them. Next, we compute the delays of those paths, predicted by the SVivaldi coordinates. In Figure 7.14, the path (AS 10 → AS 40) is selected as the best path because the predicted delay is the lowest one. This path is called the *selected path*. However, due to the delay approximations inherent to the use of coordinates, the selected path may not be the path with the real lowest delay. For instance, in Figure 7.14, the actual best path is (AS 10 → AS 30), instead of (AS 10 → AS 40). We evaluate here how often this phenomenon happens, and how large is the difference between the delay of the selected path and the lowest delay.

For each pair of multihomed sites, we draw a vertical bar. The upper end of the bar indicates the actual delay of the worst path, while the lower end indicates the actual delay of the best path, i.e. the lowest possible delay. Two graduations are added on a bar. They indicate the actual delay (i.e. not the estimated delay) of the path selected by SVivaldi, and the median delay among the paths. For instance, in Figure 7.14, the actual delays of the best, worst and selected path are respectively $16.0ms$, $28ms$ and $16.2ms$. The median delay is given by $(16.2ms + 26.0ms)/2 = 21.1ms$. The corresponding bar is illustrated on the right part of Figure 7.14. Those bars are used in Figure 7.15 to estimate the quality of the path selection for the emulated multihomed sites of the RIPE data set.

Figure 7.15. shows the delay of the paths chosen by SVivaldi for each pair of multihomed sites, sorted by increasing delay. The curve connects the graduations of the actual delays of the selected paths. In this figure, we can first observe that the delay of the worst paths can sometimes be several times larger than than the delay of the best path. For instance, the 11th pair indicates a lowest delay of about $20ms$ and a highest delay of more than $130ms$. Fortunately, SVivaldi selected the lowest delay path for this pair. Actually, in this data set, SVivaldi never selects those really bad paths. For the large majority of multihomed pairs, SVivaldi even manages to select almost the best path. When SVivaldi does not select the best path, the difference between the delay of the selected path and the lowest delay is not that large.

Figure 7.16 shows the relative difference between the delay of the best path and the delay of the path selected by various path selection algorithms. Two figures are presented for readability. They show $f(x)$, the fraction of pairs of multihomed sites where a relative difference lower than x is observed.



Path	Predicted delay	Actual delay	
(AS 10 → AS 30)	$\|\|(15,25)-(5,12)\|\| = 16.4ms$	$\mathbf{16.0}ms$	*Best Path*
(AS 20 → AS 30)	$\|\|(21,30)-(5,12)\|\| = 24.1ms$	$28.0ms$	*Worst Path*
(AS 10 → AS 40)	$\|\|(15,25)-(2,16)\|\| = \mathbf{15.8}ms$	$16.2ms$	*Selected Path*
(AS 20 → AS 40)	$\|\|(21,30)-(2,16)\|\| = 23.6ms$	$26.0ms$	

Figure 7.14. Comparing the RTTs of the predicted and actual lowest delay paths between AS 65001 and AS 65002

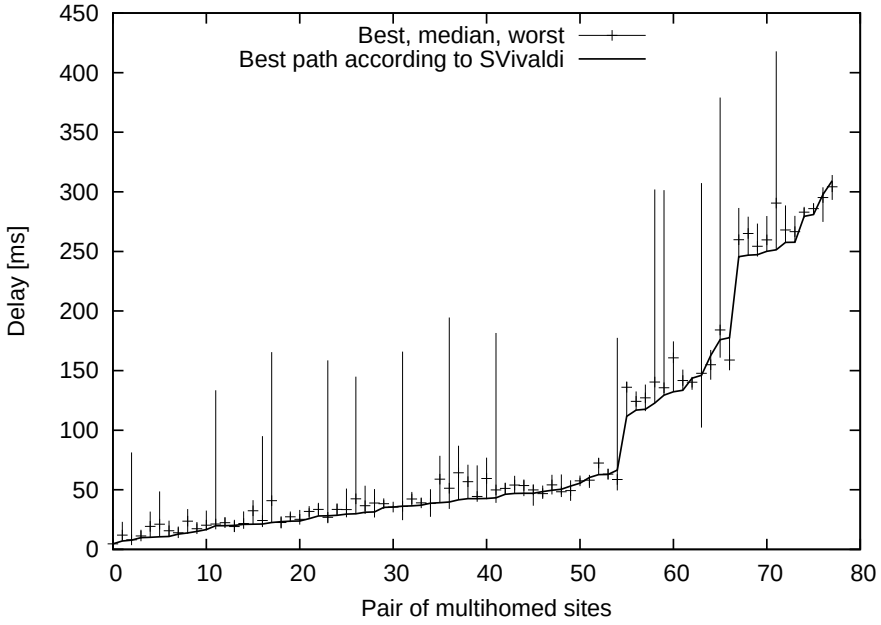


Figure 7.15. Delay of the path chosen by SVivaldi for each pair of multihomed sites, in the RIPE data set

In the upper figure, the SVivaldi path selection is compared with the mean, median and worst delays. The figure shows that SVivaldi allows us to find the absolute best path for about 40% of pairs of multihomed sites, and selects a path with a delay at most 20% worse than the best delay for more than 85% of those pairs. It should be noted that in IDMaps [84], a path selection is considered correct as long as the delay of the selected path is within a factor of 2 times the delay of the best path. Following this criterion, SVivaldi practically never selects a wrong path, at least for this dataset. The upper figure also confirms that large differences are not uncommon between the delays of the best and worst paths. The figure shows that the delay of the worst path is more than twice the delay of the best path for about 25% of pairs of IPv6 multihomed sites.

In the lower figure, the SVivaldi path selection is compared with a random path selection, and with the Vivaldi path selection. Since the BGP path lengths are not correlated with their delays [102], we may consider that a random path selection may approximate the BGP path selection. The figure shows that, in contrast with SVivaldi, a random selection of path does not avoid the selection of very high delay paths. For 8% of pairs, the random path selection provides a path with a delay more than 100% larger than the delay of the best path, while this percentage is nearly null when SVivaldi is used. The lower figure also compares the SVivaldi and Vivaldi path selections. It shows that SVivaldi better selects the lowest delay paths, although the main advantage in our case is that SVivaldi's

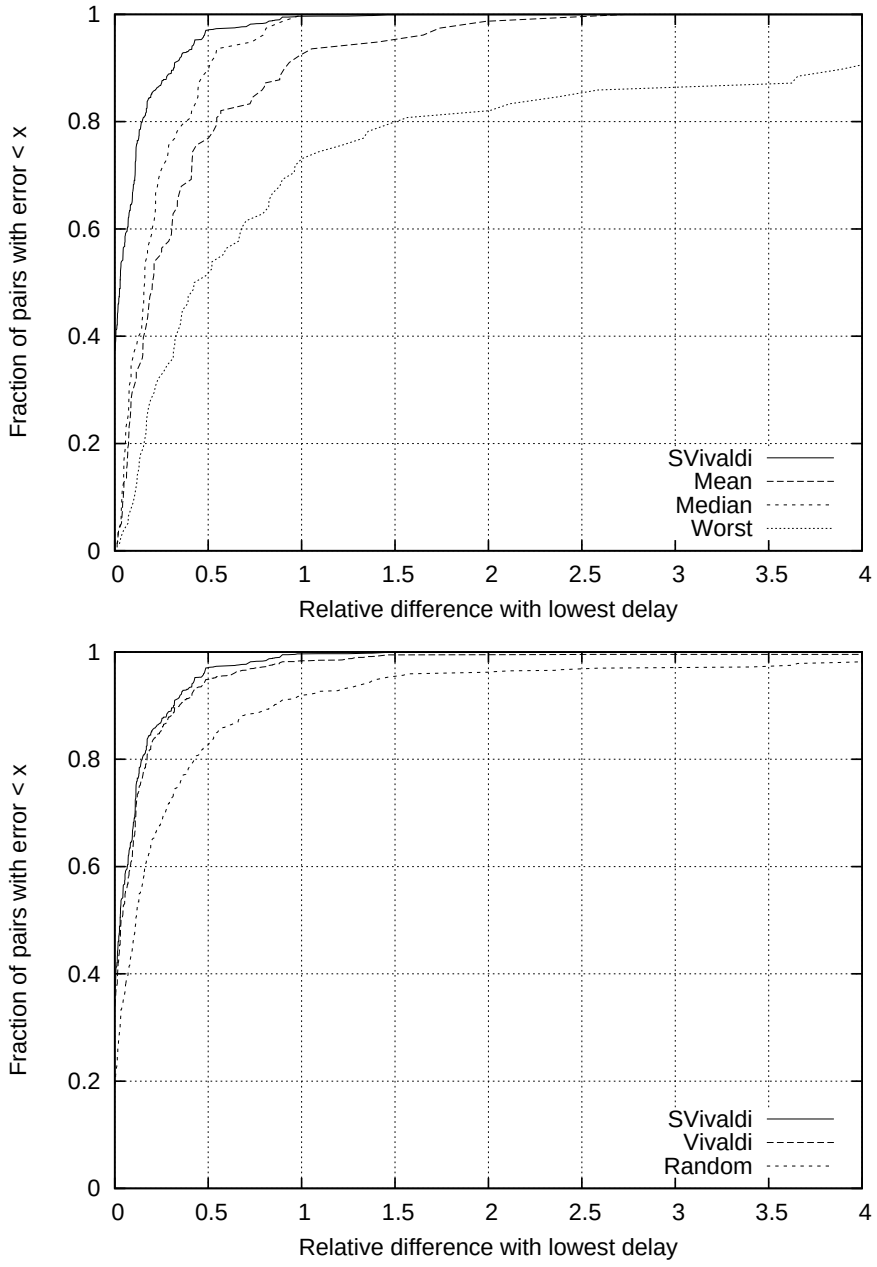


Figure 7.16. Cumulative distribution of the relative difference between the delay of the best path and the delay of the path selected by SVivaldi

coordinates are stable.

In summary, Figure 7.16 confirms that, for the RIPE dataset studied, SVivaldi successfully manages to avoid really bad paths, i.e. paths where the delay is more than twice the lowest delay. According those figures, these bad paths are not unusual. Finally, we can observe again that, beside producing stable coordinates, the use of SVivaldi also produces slightly lower relative errors than the use of Vivaldi.

7.5 Coupling NAROS and the Use of Network Coordinates

The NAROS solution presented in Chapter 6 and the use of coordinates to select the lowest delay path are two solutions that can be used independently from each other. However, it could be useful to combine them to constitute a more comprehensive service. This is illustrated in Figure 7.17.

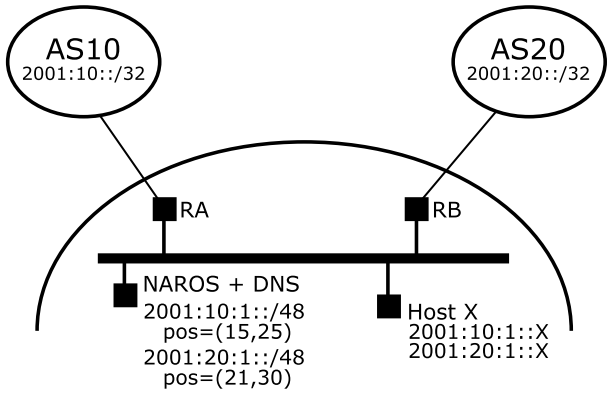


Figure 7.17. Coupling the NAROS service with the use of coordinates

In this figure, the NAROS server is coupled with the DNS. Coordinates are assigned to the two prefixes `2001:10:1::/48` and `2001:20:1::/48` of the multihomed sites. These coordinates are computed by the NAROS+DNS server, and stored in the DNS. All hosts inside the multihomed site share these coordinates, and do not have to compute them. When a host X wants to initiate a connection with another host Y in the Internet, it issues a NAROS+DNS request for the remote host Y. Host X sends this request to its local NAROS+DNS server. When receiving the request, the NAROS+DNS server retrieves the addresses of Host Y, and uses the synthetic coordinates to select the pair of source and destination addresses that yield the lowest delay path. This information is finally transmitted to Host X using a NAROS+DNS response message.

7.6 Related Work

Other solutions have been proposed to perform intelligent route selection. A Resilient Overlay Network (RON) [9] is an application-layer overlay on top of the existing Internet. It aims at detecting and recovering from path outages and periods of degraded performance. RON nodes regularly monitor the quality of paths to each other, and use this information to dynamically select between the direct path and an indirect path via other RON nodes. However, it has been shown that the use of overlays is not necessary to achieve good end-to-end resilience and performance [126]. Moreover, traffic in overlays can circumvent BGP routing policies, which can be problematic. Our utilisation of coordinate systems to select the routes relies on the existing BGP routes, and does not circumvent the BGP's policy-driven routing. Moreover, the use of coordinates is more scalable and does not impose the costs associated to overlays.

Several vendors also enable route selection [7, 27, 48]. However, these solutions rely on actively probing popular nodes. In an IPv6 multihoming environment, these solutions will have to cope with the multiplication of paths available, and their pressure imposed on the network infrastructure will increase. Moreover, those solutions cannot help for prefixes for which they do not have active measurements.

The King method [91] is a tool that predicts Internet RTTs between arbitrary end-hosts by using recursive DNS queries. Unlike the use of coordinates, King would need to probe all available paths in order to identify the one with the lowest delay.

7.7 Conclusion

With IPv6, the use of multiple prefixes increases the number of paths available to a multihomed site. Selecting the path with the lowest delay is important for many interactive and real-time applications.

The first contribution in this chapter is to propose the use of a network coordinate system as an efficient and scalable way to help IPv6 hosts in selecting the best source and destination IPv6 addresses that will lead to the use of the lowest delay path. Our observations have shown that the use of synthetic coordinates is a scalable way to select good paths and to avoid all really bad paths, i.e. paths where the delay is more than twice the lowest delay. Our experiments with the RIPE data set have shown that we are able to select paths with a delay at most 20% worse than the lowest delay for more than 85% of the pairs of multihomed sites. Further simulations, with other datasets, should be performed to confirm this finding.

The second contribution is SVivaldi, an improved version of the Vivaldi distributed algorithm for computing synthetic coordinates. Vivaldi suffers from coordinate stability problems. We have shown that the addition of a loss factor to the Vivaldi algorithm stabilises the coordinates. Moreover, our observations have shown that our proposed local error estimate allows the nodes to compute more accurate coordinates. SVivaldi is completely distributed and requires no infrastructure. It only requires that a multihomed site maintains its coordinates by measuring its RTTs with a small number of other multihomed sites.

Chapter 8

Conclusion

In this concluding chapter, we first summarise in section 8.1 the main contributions of this thesis. In section 8.2 we envision the future of traffic engineering mechanisms in IPv6 multihomed sites. In particular, we discuss some questions and challenges that remain. Finally, in section 8.3, we conlude this thesis by presenting possible further works.

8.1 Contributions

For reliability and performance reasons, more and more Internet service providers and corporate networks are multihomed. However, the current multihoming mechanism contributes to the explosive growth of the Internet routing tables, which has major implications on storage requirements, protocol overhead and stability, and forwarding performance. As a consequence, the traditional way to be multihomed in IPv4 can simply not be used for IPv6 multihoming.

Many approaches for IPv6 multihoming were proposed, with little consideration for traffic engineering aspects. The overall contribution of this thesis is to unleash traffic engineering for IPv6 multihomed sites.

First, the thesis reviewed in Chapter 4 the many proposed architectural approaches for IPv6 multihoming. The most relevant approaches were methodically compared, according to their mechanisms, benefits and drawbacks.

Next, the thesis revealed in Chapter 5 the most promising architectural approach for IPv6 multihoming, in terms of traffic engineering and fault-tolerance capabilities. This approach, named Host-Centric IPv6 multihoming, relies on the use of multiple provider dependent aggregatable (PA) prefixes per site. We demonstrated that the use of multiple PA prefixes largely increases the number of interdomain paths available to Host-Centric multihomed sites. This can thereby reduce the delays between multihomed sites, compared to the use of traditional IPv4 Multihoming.

Another contribution is a generic tool, named NAROS, to perform traffic engineering inside Host-Centric IPv6 multihomed sites. By driving the source address selection of multihomed hosts, the NAROS service can control how the locally initiated flows are distributed over the links with the providers. It thereby provides an inbound and outbound traffic engineering mechanism, without transmitting any BGP advertisement, and without affecting the interdomain BGP routing tables. This mechanism can handle complex and very dynamic routing policies. It was presented in Chapter 6.

Many interdomain paths are available to Host-Centric multihomed sites, and NAROS allows a site to effectively select one of those paths. The last contribution of this work is a a scalable an efficient way to select a *good* path, and to avoid all really *bad* paths. The metric considered in Chapter 7 was the end-to-end delay, as it is a major component of the performance of a path since it reflects its length and bandwidth.

In summary, the thesis shed some light on three different faces of traffic engineering for IPv6 multihoming.

The first is *why* should IPv6 multihomed sites use the Host-Centric approach, from a traffic engineering point of view ? We have shown that those sites can get better resilience and lower delays, thanks to a higher AS-level path diversity.

The second is *what* should they do in order to effectively exploit this diversity ? Host-Centric multihomed sites must be able to control which interdomain path is used for their flows. In this work, we proposed the NAROS mechanism to perform such control.

The third is *how* can we identify the "best" interdomain path ? We proposed the use of synthetic coordinates to identify the low delay paths, and to avoid the really high delay paths.

8.2 The Future of IPv6 Multihoming

By the end of 2004, the IETF fortunately decided to go down the path of developing Host-Centric approaches, more precisely the SHIM approach described in Chapter 4, section 4.4.4. This decision was not driven by traffic engineering considerations. Rather, it was made because Host-Centric approaches provide

complete fault-tolerance, while preserving the scalability of the interdomain routing. As a consequence of this decision, it can be expected that the use of several IPv6 addresses on each end-host will become widely prevalent on the IPv6 Internet. These hosts will use their addresses interchangeably during the lifetime of a flow in order to survive outages affecting any of those addresses. An intermediate layer, between the network and the transport layers, is used to preserve the established connections. This layer separates two functions that are included in an IP address : the location of a node, and the identity of the node. This is a drastic architectural change compared to today's IPv4 Internet, where hosts are typically identified and located by a single IPv4 address.

In addition, another drastic change will impact the routing inside Host-Centric IPv6 multihomed sites. For security reasons, the IETF recommends that each ISP drops the customers' traffic entering its network that is coming from a source address not allocated by the ISP to the customer network. For Host-Centric IPv6 multihomed sites, this ingress filtering procedure means that the source address of a packet determines the upstream provider used. This has two major consequences.

The first consequence is that, inside the multihomed site, packets must be forwarded according to their source addresses. This source-address dependent routing can be limited to the border routers of the multihomed sites. In this case, routers may need to be upgraded to support this form of routing, which has cost implications. An alternative is to establish tunnels, but this practice complicates the management of the network.

The second consequence is that new traffic engineering techniques, such as the NAROS approach presented in Chapter 6, can be developed and deployed, so that Host-Centric multihomed sites are able to control how their flows are distributed on the links with their providers.

These two major architectural changes require to add a new intermediate layer in the network stacks of end-hosts, and to implement source-address dependent routing. Hence, it seems wise to ask ourself if sites will agree to implement these techniques. We should not neglect the reluctance of most sites for adopting drastic architectural changes. Fortunately, the switch to an IPv6 Internet represents a considerable opportunity to bring those architectural changes. However, the time window is not infinite, and the IPv6 deployment is already a reality in some regions. An unanswered question is thus :

Isn't it too late to introduce such drastic changes ?

We believe it is still time to introduce these changes. The reason is that the deployment of IPv6 is still slow. This was indicated by Figure 3.2, which showed a slow current growth of the number of active IPv6 BGP entries. This should leave enough time to develop Host-Centric approaches, provided that IPv6 multihomed

sites see enough incentives for using this approach. Indeed, routing approach for IPv6 multihoming can also preserve the scalability of the interdomain routing, at the price of fault-tolerance and traffic engineering capability. Without sufficient incentives, it may happen that current multihomed sites would prefer Routing approaches. The reason is that these approaches are easier to deploy, but also because the traffic engineering methods currently used in IPv4 multihomed sites can be readily adapted to Routing approaches for IPv6 multihoming.

This thesis introduced new incentives for Host-Centric IPv6 multihoming. It explained that Host-Centric multihoming approach brings many advantages, both in terms of fault-tolerance and traffic engineering capabilities.

However, we might still ask ourself :

Are the incentives for Host-Centric multihoming strong enough ?

We believe that Host-Centric multihoming will finally be largely adopted, because small and home networks will not use the routing approaches since they require to run BGP. For those networks, Host-Centric multihoming is the only viable solution, and the one that brings the best resilience to failures, and the best possibilities to increase their network performance. The actual question for those small and home networks is rather :

Will these small networks be multihomed ?

The answer is probably *yes* for small networks, as many of them become more and more dependent on their Internet connectivity. The answer to this question remains unclear however for home networks.

For the small and home networks that will choose to be multihomed by using the Host-Centric approach, it still remains unclear if they will engineer their traffic, and which method they will use. We presented NAROS as a first mechanism to help Host-Centric multihomed sites in engineering their traffic. We also proposed the use of synthetic coordinates to predict delays between any two multihomed sites in the Internet. The proposed mechanism is highly scalable and efficient, especially compared to actively probing all available paths. However, while highly efficient for the global Internet, this approach may yield to delay predictions of lower quality than the delay predictions obtained using active probing. Considering the selfish behaviour of most sites, it is not certain that they will always agree to trade a better delay prediction for a lower overall network overhead.

8.3 Future Work

Let us conclude this thesis by presenting possible further works. Working on traffic engineering tools for Host-Centric multihomed sites is crucial for the deployment of IPv6 multihoming, which in turn is crucial for a large adoption of IPv6. Most multihomed sites that today use traffic engineering techniques in IPv4 will not switch to IPv6 before they know they will be able to also engineer their traffic in IPv6. It is surprising that little consideration has been given until now to this aspect of IPv6 multihoming. Traffic engineering methods for IPv6 Host-Centric multihomed sites must be developed in order to at least provide basic features like load-sharing and some performance optimisation mechanisms. Chapter 7 provides a way to predict delays of interdomain paths. A future work consists in engineering the traffic according to other metrics like bandwidth and cost. NAROS can be used to define a combined metric that groups delay, bandwidth, policy and costs criteria.

Another future work is how to improve the delay predictions made by synthetic coordinates between multihomed sites. The path selection procedure depends on the quality of those delay predictions. If the predictions are not good enough, sites will tend to prefer active probing. A possible way to improve the accuracy of the coordinates is to run the SVivaldi algorithm over hyperbolic spaces. Shavitt and Tankel have shown that Internet delays can be embedded in hyperbolic spaces to achieve an accuracy better than achieved before for the Euclidean space [182]. Future simulations are needed to determine whether this can improve the path selection procedure.

Regarding the use of synthetic coordinates, other questions remain unanswered. A first question is how to protect the coordinates from malicious attacks. What happens if a site deliberately or unintentionally maintains completely wrong coordinates ? A second question is how to handle the case where the network becomes completely partitioned ? How to react to a link failure, and how fast ?

If a group of nodes is disconnected from another group, one option is to make the distances between the coordinates of the groups grow to infinity. When the groups are connected again, all the coordinates must change again to reflect the new actual distance.

Another option is to simply ignore probes towards nodes that are unreachable, and only take into account successful probes. In the first option, the coordinates are used to predict delays *and* to detect unreachability. In the second option, coordinates are used only to predict delays, and reachability detection is provided by other techniques. The advantage of the second option is a better stability of the coordinates.

Another future work is to investigate the possibility and cost of implementing the support of the coordinates directly inside the SHIM layer, hence removing the dependency against the Domain Name System. This implies that every host will have to compute its own coordinates, hence increasing the network overhead. However, this option diminishes the deployment cost, as a DNS server is no longer required.

Finally, a last future work is related to the AS-level path diversity metric detailed in Chapter 5. A possible usage of this metric is to help sites in choosing the provider AS that most increases their AS-level path diversity. This usage remains to be evaluated.

Part III

Appendix

Appendix A

The NAROS Protocol

This appendix gives details about the NAROS protocol presented in Chapter 6. It describes the formats of the various NAROS messages.

A.1 Message Types

The general format of all NAROS messages is illustrated by Figure A.1. The NAROS messages consist of two mandatory fields : a Version field, and a Message Type field, followed by one or more required parameters that depend on the message type. NAROS runs over UDP. A NAROS message must not be longer than 4096 bytes and there may be only one NAROS message per UDP datagram.

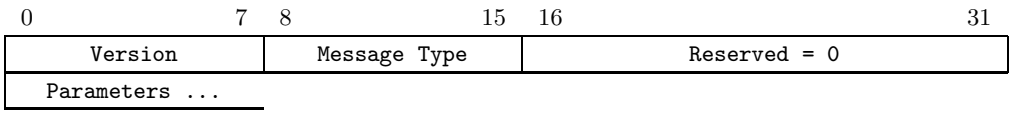


Figure A.1. The NAROS message format

The Version field is a single byte that specifies the NAROS version number that is being used. The current NAROS version number is 1.

The Message Type field is a single byte that specifies the type of the message. Message types currently defined are shown in Table A.1. The first column indicates the values for the message types. The second column gives the names of the messages, and the third indicates who sends and who receives a given message.

Value	Message	Direction
1	NAROS_REQUEST	Host → Server
2	NAROS_RESPONSE	Server → Host
3	NAROS_DNS_REQUEST	Host → Server
4	NAROS_DNS_RESPONSE	Server → Host
5	NAROS_ERROR_RESPONSE	Server → Host

Table A.1. Message types

A.1.1 NAROS_REQUEST

A NAROS_REQUEST is sent by a NAROS client to request its connection parameters. The format of the message is illustrated on Figure A.2. The minimum length of a NAROS_REQUEST message is 40 bytes.

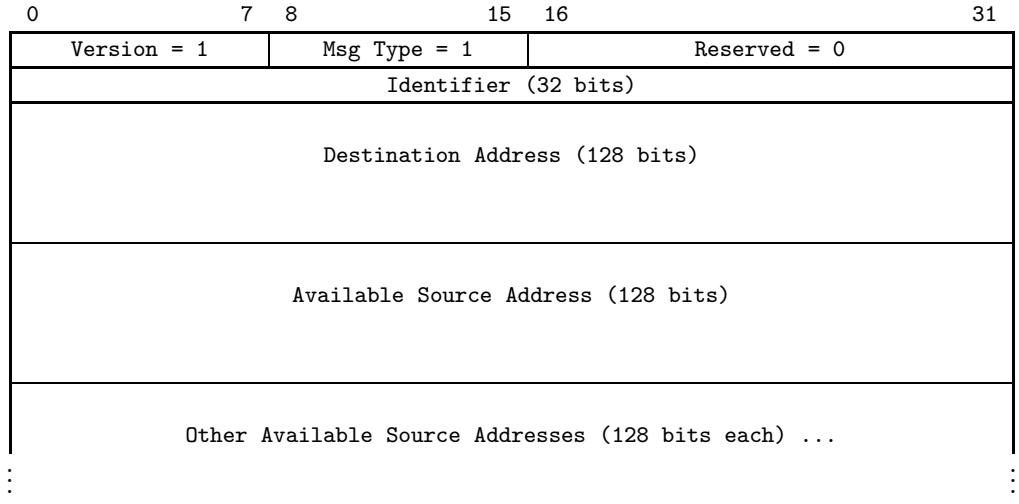


Figure A.2. The NAROS_REQUEST message

The Identifier field uniquely identifies the NAROS request. The Destination Address field is the address of the destination host that the client wants to contact. The Available Source Address fields are the source addresses currently allocated to the client. The client must specify in the message all the source addresses

it is able to use to reach the destination host, for instance its global-scope non-deprecated IPv6 addresses. However, the client should not put more than 16 IPv6 Available Source Addresses inside a NAROS_REQUEST message. If more than 16 source addresses are available, the client can freely select the 16 addresses that will be included in the NAROS_REQUEST message.

When selecting a source address, the client must first use the source address selection algorithm as defined in [73]. However, the 8th rule, i.e. use the longest matching prefix, must be superseded by the NAROS source address selection procedure. A NAROS_REQUEST message should be sent by the client before it initiates a communication with a remote host for the first time, except if the client already knows exactly which source address to use, for instance thanks to its policy table [73], or thanks to informations stored in the NAROS cache.

A.1.2 NAROS_RESPONSE

The NAROS_RESPONSE message is sent by a NAROS server and specifies which is the best source address to be used to contact the remote host, among the source addresses that were given in the previously described NAROS_REQUEST message. The format of the NAROS_RESPONSE message is given by Figure A.3. The overall length of a NAROS_RESPONSE message is 48 bytes.

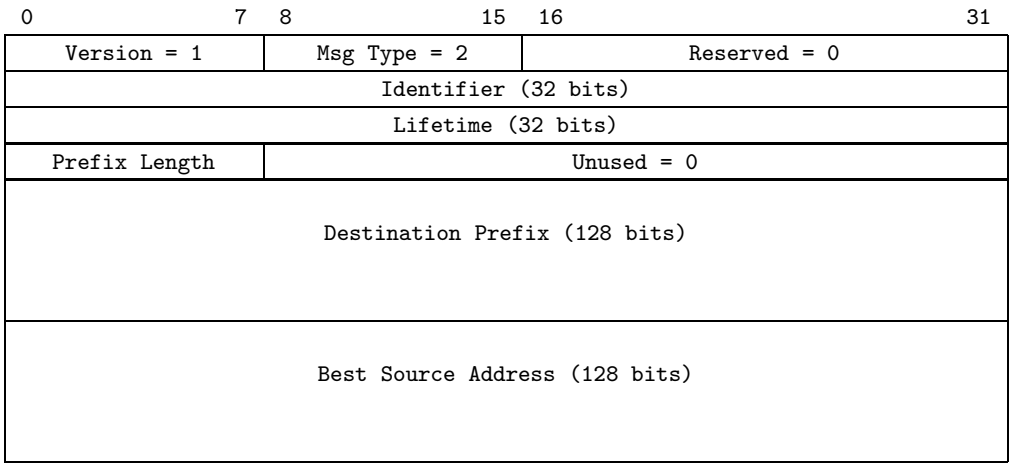


Figure A.3. The NAROS_RESPONSE message

The Identifier field specifies the request for which the NAROS_RESPONSE message is a reply. It must be the same identifier as the identifier contained in the corresponding request.

The Lifetime field defines how long the information contained in this message should be cached by the client.

The Prefix Length and Destination Prefix fields define the destination prefix for which the best source address contained in this message applies. The Prefix Length field indicates the length in bits of the destination prefix. A length of zero indicates a prefix that matches all IP addresses. The Destination Prefix field contains the destination prefix followed by enough trailing bits to fill the field. Note that the value of the trailing bits is irrelevant.

The Best Source Address field specifies the source IP address that the server determined to be the best to reach the destination prefix.

When receiving a NAROS_RESPONSE message, the client must first check the validity of the selected best source address, i.e. it must check that the source address is available and not deprecated. If the client determines that the best source address is not a valid choice, then it may use another available source address. Otherwise it should use the selected source address, for all communications towards any destination address covered by the destination prefix, until the lifetime expires. The selected source address for the destination prefix should be cached during the specified lifetime, for instance by inserting a new rule in the policy table of the host. When the lifetime expires, the rule is removed.

A.1.3 NAROS_DNS_REQUEST

A NAROS_DNS_REQUEST is used by a NAROS client to embed a DNS query within a NAROS request message. The format of the message is illustrated on Figure A.4.

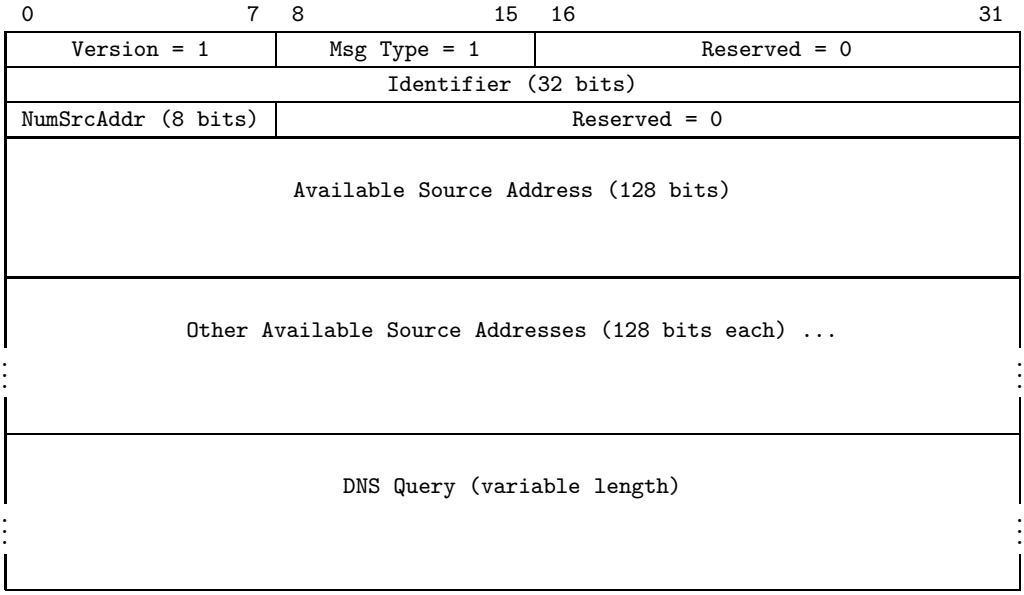


Figure A.4. The NAROS_DNS_REQUEST message

The Identifier field uniquely identifies the NAROS request. The NumSrcAddr field is an unsigned 8-bit number that indicates the number of available IPv6 source addresses that are contained in the NAROS_DNS_REQUEST. There should not be more than 16 IPv6 Available Source Addresses inside a NAROS_DNS_-REQUEST message. The DNS Query field contains a standard DNS query message, which is embedded *as is* in this NAROS message.

A.1.4 NAROS_DNS_RESPONSE

The NAROS_DNS_RESPONSE message is sent by a NAROS server in reply to a NAROS_DNS_REQUEST message. It specifies which is the best source address to contact the destination host, among the source addresses that were given in the preceding NAROS_DNS_REQUEST message. It also includes the DNS response to the DNS query that was embedded in the NAROS_DNS_REQUEST. The format of the NAROS_DNS_RESPONSE message is given by Figure A.5.

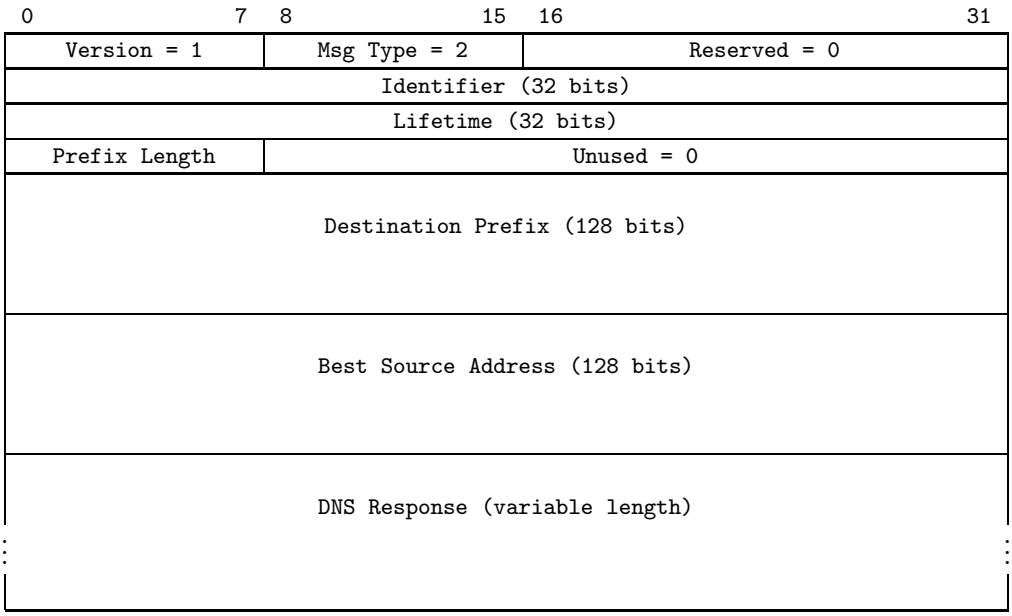


Figure A.5. The NAROS_DNS_RESPONSE message

The NAROS_DNS_RESPONSE plays the same role as the NAROS_RESPONSE message, except that it also includes the DNS response to the DNS query embedded in the preceding NAROS_DNS_REQUEST.

A.1.5 NAROS_ERROR_RESPONSE

A NAROS_ERROR_RESPONSE is used to provide error messages from a NAROS server to a NAROS client. Multiple errors must not be reported in the same NAROS_ERROR_RESPONSE. In situations where more than one error occurred, the NAROS server must choose only one error to report. The format of the message is illustrated by Figure A.6. The overall length of a NAROS_ERROR_RESPONSE message is 10 bytes.

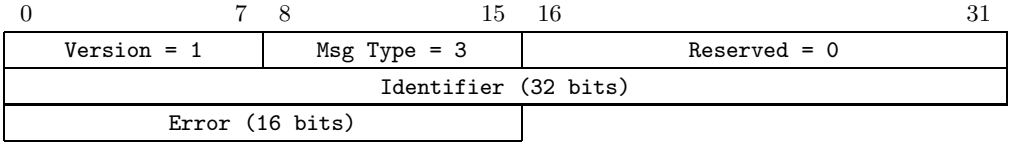


Figure A.6. The NAROS_ERROR_RESPONSE message

The Identifier field specifies the request for which the NAROS_ERROR_RESPONSE message is a reply. An NAROS_ERROR_RESPONSE message must only be transmitted by a NAROS server, and only in response to a request from a NAROS client. A NAROS client that detects an error in a message received from a NAROS server must silently discard the message.

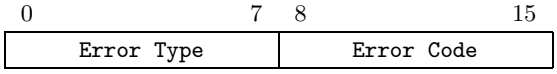


Figure A.7. The Error field of the NAROS_ERROR_RESPONSE message

The 2-byte length Error field is divided into a 1-byte Error Type and a 1-byte Error Code, as illustrated by Figure A.7. The currently defined error values are indicated in Table A.2 and Table A.3.

1 : General Errors

UNKNOWN_ERROR. An error occurred that cannot be identified. This error is used when all other error messages are inappropriate.

INTERNAL_SERVER_ERROR. The NAROS server detected an internal unrecoverable error.

UNSUPPORTED_NAROS_VERSION. A NAROS client sent a message with a version number not supported by the NAROS server.

Error Type	Description
1	General errors
2	Message errors
3	Routing errors

Table A.2. Values defined for the Error Type field

Error Type	Error Code	Error Name
1	1	UNKNOWN_ERROR
1	2	INTERNAL_SERVER_ERROR
1	3	UNSUPPORTED_NAROS_VERSION
2	1	ILLEGAL_MESSAGE
2	2	BAD_MESSAGE
3	1	NO_ROUTE
3	2	ADMINISTRATIVELY_PROHIBITED

Table A.3. Values defined for the Error Field

2 : Message Errors. The server should use these errors when it detects that a message is malformed, as well as when it does not understand a message.

ILLEGAL_MESSAGE. The message contains illegal values for one or more fields.

BAD_MESSAGE. The message is malformed and server parsing failed.

3 : Routing Errors. The server should use these errors when it is unable to select the best source address needed by the client to reach a destination.

NO_ROUTE. The server has no route towards the destination.

ADMINISTRATIVELY_PROHIBITED. The server has a route towards the destination but it is administratively prohibited.

Appendix B

Synthetic Location Information in the Domain Name System

B.1 Introduction

In Chapter 7, we proposed to use coordinates to help hosts to select, for each pair of source and destination stub ASes, the IPv6 addresses to be used that will lead to a low delay path. We suggested to store these coordinates in the Domain Name System (DNS).

Storing coordinates in the DNS is not new. Geographical location informations can already be expressed in the DNS by using the LOC resource record [134]. Geographical locations are useful to generate maps of routers or to perform "visual" traceroutes. They are however of little help for predicting round-trip times in the Internet, since the RTTs between Internet hosts are poorly correlated with their geographical distances. Hence several synthetic coordinates were developed so that the distance in the synthetic coordinate space between hosts x and y actually reflects the round-trip time between them. These synthetic coordinates are usually not expressed in terms of latitude/longitude, and are often defined in other coordinate spaces than the traditional two- or three-dimensional Euclidean spaces. For instance, we suggested in Chapter 7 to store 2D+Height synthetic

coordinates in the DNS. While the SVivaldi use of coordinates is intended for the prediction of Internet latencies of IPv6 paths, storing synthetic coordinates in the DNS can be useful for a wider range of applications, and, in particular, for many IPv4 applications.

This appendix details the DNS extensions needed for this purpose. It defines a new Resource Record (RR) for the Domain Name System, for experimental purposes. It describes a mechanism to allow the DNS to carry synthetic coordinates about hosts, networks, and subnets. The new resource record, called SLOC, can express synthetic locations for many different coordinate spaces, including the geographical coordinates. This appendix defines the format of the new SLOC RR for the DNS, and reserves a corresponding DNS type mnemonic (SLOC) and numerical code (51). The numerical code is to be confirmed by IANA.

This appendix assumes that the reader is familiar with the DNS [132] [133], and its IPv6 extensions [197].

B.2 RDATA Format of the SLOC Resource Record

Like every DNS resource records, the SLOC RR is formatted as illustrated by Figure B.1. The NAME, TYPE, CLASS, TTL, and RDLENGTH fields are all described in [133]. This section only details the RDATA field, whose format is specific to the SLOC resource record.

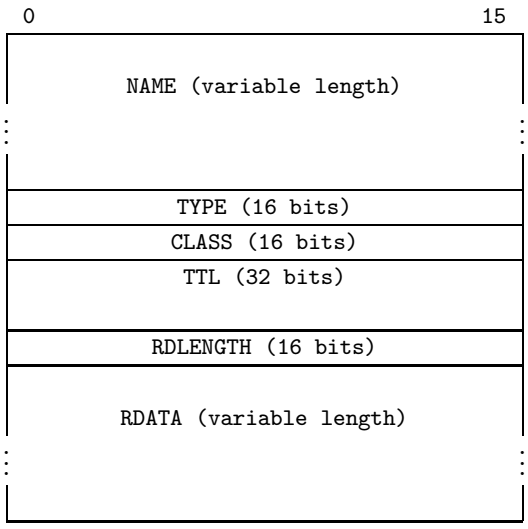


Figure B.1. The SLOC Resource Record format

The format of the SLOC RDATA field is detailed in Figure B.2. It is divided into three fields : the SLOC_CLASS, SLOC_ID, and LOCATION fields.

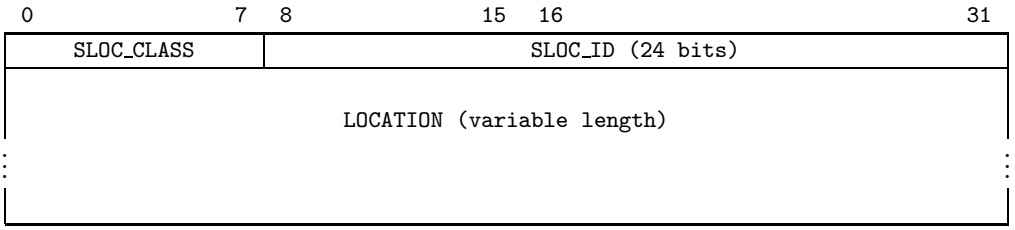


Figure B.2. The SLOC RDATA format

B.2.1 The SLOC_CLASS Field

The SLOC_CLASS field contains an unsigned 8-bit value that specifies how the coordinates are identified in the SLOC_ID field. The identification can be a standard identification (SLOC_CLASS = 1), or a vendor-specific identification (SLOC_CLASS = 2), or a private identification (SLOC_CLASS = 3). Standard identifications are destined for coordinates that are supposed to be used by hosts in the global Internet. For instance, a standard identification can be used to express geographical coordinates. Vendor-Specific identifications aim to be used by vendors wishing to define their own proprietary coordinates. They can also be used to experiment new coordinates. Typically, vendor-specific coordinates are useful to only a fraction of hosts in the Internet. Finally, private identifications can be used by anyone who wants to express coordinates that do not have any meaning outside the network where they are used.

B.2.2 The SLOC_ID Field

The format of the SLOC_ID field depends on the SLOC_CLASS value, i.e. the format of the field is different for standard, vendor-specific, and private coordinates. The value of the SLOC_ID field specifies both the nature of the coordinates, and the algorithm that can interpret and possibly compute these coordinates.

The SLOC_ID Field for Standard Coordinates

When the identification is standard (SLOC_CLASS = 1), the SLOC_ID field is divided into an 8-bit value SLOC_ALG, an 8-bit value SLOC_SPACE, and an 8-bit value SLOC_DIM, as illustrated by Figure B.3.

Figure B.3. The SLOC_ID field for standard coordinates.

The SLOC_ALG field specifies the algorithm that interprets and computes the coordinates. The SLOC_SPACE field uniquely identifies the nature of the space where the coordinates are defined, for instance an Euclidean, a spherical, or an hyperbolic space. Finally, the SLOC_DIM defines the number of dimensions of the space.

SLOC_ALG	Description
0	Reserved
1	Geographical coordinates [56]
2	GNP synthetic coordinates [141]
3	Lighthouse synthetic coordinates [157]
4	Vivaldi synthetic coordinates [55]
5	SVivaldi synthetic coordinates [70]
6	BBS synthetic coordinates [181]
7	NPS synthetic coordinates [142]
8	PIC synthetic coordinates [52]
9 - 255	Unassigned

Table B.1. Values for the SLOC_ALG field

Some SLOC_ALG field values currently defined for well-known synthetic coordinate algorithms are indicated in Table B.1.

SLOC_SPACE	Description
0	Reserved
1	Uninterpreted
2	Euclidean
3	Spherical
4	Cylindric
5	Hyperbolic
6	Height Vector
7-127	Unassigned
128-255	Algorithm-Specific

Table B.2. Values of the SLOC_SPACE field defined for typical coordinate spaces

Values currently defined for the SLOC_SPACE field are shown in Table B.2. It assigns values to popular coordinate spaces. When the algorithm does not make use of one these coordinate spaces, it may use the *uninterpreted* coordinate space (SLOC_SPACE = 1), or use one of the algorithm-specific values in the range [128..255].

Table B.3 shows the values defined for the SLOC_DIM field. Values in the range [1..63] indicate the number of dimensions of the coordinate space. Value 0 is reserved and must not be used.

SLOC_DIM	Description	LOCATION Min Size
0	Reserved	Undefined
1	1 dimension	1
2	2 dimensions	2
3	3 dimensions	3
:	:	:
63	63 dimensions	63
64-254	Reserved	
255	Variable	1

Table B.3. Values of the SLOC_DIM field

If SLOC_DIM equals 255, the LOCATION field contains a variable-length sequence of one or more 32-bits numbers. This value is typically used when SLOC_SPACE = 1, i.e. an uninterpreted coordinate space, or when the size of the LOCATION field suffice to indicate the number of dimensions used.

The third column of table B.3 defines the minimum number of 32-bit values required in the LOCATION field for this type of dimensions. The LOCATION field may contain more than this minimum number of 32-bit values. In this case the meaning of the additional values is algorithm-specific.

The tuple (SLOC_ALG, SLOC_SPACE, SLOC_DIM), i.e. the whole SLOC_ID field, uniquely defines a coordinate space and how to interpret it.

The SLOC_ID Field for Vendor-Specific Coordinates

When the identification is vendor-specific (SLOC_CLASS = 2), the SLOC_ID field contains the vendor-specific ID assigned by IANA (24-bit value encoded in network standard byte order), as illustrated by Figure B.4. If a vendor implements several algorithms, or use several coordinate spaces, then the SLOC_ID field may not suffice to uniquely identify a couple *(algorithm, coordinate space)*. Therefore, it is suggested that the vendor uses the first dimension(s) of its coordinates to identify the type of coordinates. However, in any cases, this implementation choice is left to the vendor.

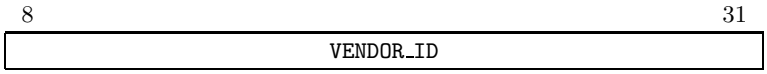


Figure B.4. The SLOC_ID field for vendor-specific coordinates.

The SLOC_ID Field for Private Coordinates

If the coordinates are private (SLOC_CLASS = 3), then the semantic and meaning of the SLOC_ID field depends on the domain where it is found. Such coordinates should not be transmitted to a host that is not inside the network where the coordinates are defined.

B.2.3 The LOCATION Field

The LOCATION field contains a sequence of 1 or more 32-bit unsigned integers, most significant octet first (network standard byte order). These integers express the coordinates of a node. The exact meaning of these integers depends on the SLOC_CLASS and SLOC_ID fields.

In the remaining of this document, we focus only on standard coordinates (SLOC_CLASS = 1).

B.2.4 SLOC RDATA Examples

Table B.4 shows some examples that illustrate the use of standard coordinates (SLOC_CLASS = 1). The table presents some meaningful combinations of SLOC_ALG, SLOC_SPACE and SLOC_DIM values. The content of the LOCATION field is written here as a sequence of 32-bit values separated by a colon ":".

SLOC_ALG	SLOC_SPACE	SLOC_DIM	LOCATION	Description
3	2	3	25:35:2	Lighthouse Euclidean 3-D
3	2	255	25:35:2	Same as above
4	6	3	5:3:1:100	Vivaldi 2D+H + error
4	2	3	5:3:1:100	Vivaldi 3D + error
5	6	3	5:3:1:100	SVivaldi 2D+H + error
1	3	3	0:0:0:1184274	Lat+Long+Alt+Extra field

Table B.4. SLOC RDATA examples

In the table above, the examples that use the Vivaldi network coordinate system (3d and 4th lines) illustrate the need for the SLOC_SPACE field. In these cases, the same algorithm (SLOC_ALG = 4) and the same number of dimensions (SLOC_DIM = 3) are used, with an identical number of 32-bit values in the LOCATION field. The meaning of the coordinates is then given by the SLOC_SPACE field.

The 3rd and 5th examples illustrate the need for the SLOC_ALG field. They both contain the same values in the SLOC_SPACE, SLOC_DIM and LOCATION fields. However, the coordinates of the 3rd example are computed by the Vivaldi algorithm, while the coordinates of the 5th example are computed by the SVivaldi algorithm.

The 6th example illustrates the coding of geographical coordinates in the SLOC RDATA. The extra field is used to encode the size and precision values for the coordinates, as detailed in [56].

B.3 Master File Format

The SLOC resource record is expressed in a master file in the following format :

```
<owner> <TTL> <class> SLOC {1 sl_alg sl_space sl_dim | sl_class sl_id} coord
```

where:

```
sl_class:   [1 .. 3]           (unsigned 8-bit value)
sl_alg:     [1 .. 255]         (unsigned 8-bit value)
sl_space:   [1 .. 255]         (unsigned 8-bit value)
sl_dim:     [1 .. 255]         (unsigned 8-bit value)
sl_id:      [0 .. 16777215]    (unsigned 24-bit value)

coord = value [ ":" coord ]
value:      [0 .. 4294967295]  (unsigned 32-bit value)
```

Numbers can be written in the decimal notation, or in the hexadecimal notation, as illustrated by the examples in the next section.

B.3.1 Examples of Master File

```
;; A SVivaldi 2D+H synthetic coordinate for domain example.net
example.net        SLOC   1 5 6 3     5:3:1:100

;; Host A has a standard and a vendor-specific coordinates
A.example.net      SLOC   1 3 2 3     0x11111111:0xABCDEF:9
A.example.net      SLOC   2 0x00005E  10:20:30:40

;; Geographical coordinates can also be expressed with SLOC RR
A.south.pole.net   SLOC   ( 1 3 3
                            0:0:10:0x00121212 )
```

The first example illustrates SVivaldi coordinates assigned to the domain `example.net`. These coordinates use the standard SLOC identification (SLOC_CLASS = 1), the SVivaldi algorithm (SLOC_ALG = 5), a Height Vector coordinate space (SLOC_SPACE = 6) with 2D+H = 3 dimensions (SLOC_DIM = 3).

The coordinates for SVivaldi are $x = 5$, $y = 3$, $h = 1$. There is one extra value $e = 100$, which stands for an error estimation of the coordinates [70].

The second example illustrates the use of coordinates for an individual host A inside the `example.net` network. This host has two kinds of coordinates. The first one is a standard (SLOC_CLASS = 1) Lighthouse (SLOC_ALG = 3) Euclidean (SLOC_SPACE = 2) 3D (SLOC_DIM = 3) coordinate. The second one is a vendor-specific (SLOC_CLASS = 2) coordinate with vendor ID = 0x5E.

The third and last example illustrates the coding of geographical coordinates using the SLOC resource record format. The coordinate values stand for the south pole. They respect RFC 1876 [56].

B.4 Application use of the SLOC Resource Record

B.4.1 Suggested Use

A suggested use is that an algorithm regularly computes the network coordinates of a (sub)domain or host, and securely updates the SLOC resource records in the DNS. This secure update can be performed using DNS extensions such as [210, 79, 208]. Using synthetic network coordinates, it is possible to evaluate the round-trip time that exists between two distant domains by computing the distance in the coordinate space between the coordinates of those domains. By storing the coordinates of hosts in the DNS using the SLOC RR, any host in the Internet is able to predict the latencies between any two hosts. This can be used for peer-to-peer distributed systems such as KaZaA [180] or BitTorrent [31]. It was also used in Chapter 7 as a scalable an efficient way to help hosts in selecting, for each pair of source and destination hosts, the IPv6 addresses that will lead to a low delay path.

B.4.2 Search Algorithms

This section specifies how to use the DNS to translate domain names and/or IP addresses into location informations. This section is similar to the section 5.2 of [56] describing the search algorithm for the LOC RR. The text is slightly adapted for the SLOC resource record, and for an IPv6 use of this record, so that its description is self-contained.

As already said, the location information may refer to a host, or to a network. When specifying the location of a network or subnet, network administrators are not required to specify the SLOC RR data for each individual host. For example, a computer lab with 24 workstations, all of which are on the same subnet and in basically the same location, would only need a single SLOC RR for the subnet. However, if the file server's location has been more precisely measured, a separate SLOC RR for it can be placed in the DNS.

The default behaviour of an application is to query the DNS for the location of an individual host. Sometimes, a host has no associated SLOC RR data, while its network has one. In this case, the application may have a *fallback* behaviour, and use the SLOC RR data of the network to get a less precise or larger area. The application may support the use of the algorithm in Section B.4.2, as noted in Section B.4.2 and Section B.4.2. If fallback is desired, this behaviour is the recommended default, but in some cases it may need to be modified based on the specific requirements of the application involved. This search algorithm is designed to allow network administrators to specify the location of a network or subnet without requiring SLOC RR data for each individual host.

Searching by Name

If the application is beginning with a name, rather than an IP address, it must check for a SLOC RR associated with that name. (CNAME records should be followed as for any other RR type.)

If there is no SLOC RR for that name, all A or AAAA records (if any) associated with the name may be checked for network (or subnet) SLOC RRs using the *Searching by Network or Subnet* algorithm (Section B.4.2). If multiple A or AAAA records exist and have associated network or subnet SLOC RRs, the application may choose to use any, some, or all of the SLOC RRs found, possibly in combination. It is suggested that multihomed hosts have SLOC RRs for their name in the DNS to avoid any ambiguity in these cases.

Note that domain names that do not have associated A or AAAA records must have a SLOC RR associated with their name, so that their location information is accessible.

Searching by Address

If the application is beginning with an IPv4 or IPv6 address, it must first map the address to a name using the IN-ADDR.ARPA or IP6.ARPA namespace (see [132] section 5.2.1, and [197] section 2.5), then check for a SLOC RR associated with that name.

If there is no SLOC RR for the name, the address may be checked for network (or subnet) SLOC RRs using the *Searching by Network or Subnet* algorithm (Section B.4.2).

Searching by Network or Subnet

Even if the name of a host does not have any associated SLOC RRs, the network(s) or subnet(s) it is on may. If the application wishes to search for such less specific data, the following algorithm should be followed to find a network

or subnet SLOC RR associated with the IP address. This algorithm is adapted slightly from that specified in RFC 1101 [134], sections 4.3 and 4.4.

Since subnet SLOC RRs are (if present) more specific than network SLOC RRs, it is best to use them if available. In order to do so, we build a stack of network and subnet names found while performing the RFC 1101 [134] search, then work our way down the stack until a SLOC RR is found. The search algorithm is presented here using an IPv6 example[1]. The search algorithm is similar for IPv4. An IPv4 example for the LOC resource record can be found in RFC 1876 [56].

1. Create a host-zero address using the network portion of the IP address. For example, for the host `2001:6a8:3080:1::1/32`, this would result in the address `2001:6a8::`.

2. Reverse the octets, suffix `.IP6.ARPA`, and query for PTR and AAAA records. Retrieve:

```
0...0.8.a.6.0.1.0.0.2.IP6.ARPA.        PTR   ipv6.ucl.ac.be.
                                       AAAA  FFFF:FFFF:FFFF::/48
```

   Push the name `ipv6.ucl.ac.be` onto the stack of names to be searched for SLOC RRs later.

3. Since an AAAA RR was found, repeat using mask from RR (`FFFF:FFFF:FFFF::/48`), constructing a query for `0...0.8.0.3.8.a.6.0.1.0.0.2.IP6.ARPA`. Retrieve:

```
0...0.8.0.3.8.a.6.0.1.0.0.2.IP6.ARPA.  PTR   info.ipv6.ucl.ac.be
                                       AAAA  FFFF:FFFF:FFFF:FFFF::/64
```

   Push the name `info.ipv6.ucl.ac.be` onto the stack of names to be searched for SLOC RRs later.

4. Since another AAAA RR was found, repeat using mask `FFFF:FFFF:FFFF:FFFF::/64`, constructing a query for `0...0.1.0.0.0.0.8.0.3.8.a.6.0.1.0.0.2.IP6.ARPA.` Retrieve:

```
0...0.1.0.0.0.0.8.0.3.8.a.6.0.1.0.0.2.IP6.ARPA.
                                       PTR  subnet.info.ipv6.ucl.ac.be
```

   Push the name `subnet.info.ipv6.ucl.ac.be` onto the stack of names to be searched for SLOC RRs later.

[1] In the IPv6 example, for readability purposes, the IPv6 addresses represented as names in the IP6.ARPA domain do not strictly follow the RFC 3596 guidelines [197] : lengthy sequences of zeros are replaced here by `0...0`. The number of zeros actually included in the sequence is such that the number of nibbles equals 32.

5. Since no AAAA RR is present at `0...0.1.0.0.0.0.8.0.3.8.a.6.0.1.0.0.2.IP6.ARPA.`, there are no more subnet levels to search. We now pop the top name from the stack and check for an associated SLOC RR. Repeat until a SLOC RR is found.

   In this case, assume that `subnet.info.ipv6.ucl.ac.be` does not have an associated SLOC RR, but that `info.ipv6.ucl.ac.be` does. We will then use `info.ipv6.ucl.ac.be`'s SLOC RR as an approximation of the location of this host. Note that even if `ipv6.ucl.ac.be` has a SLOC RR, it will not be used if a subnet also has a SLOC RR.

B.5 Applicability to non-IN Classes and non-IP Addresses

Like for the LOC record, the SLOC record is defined for all RR classes, and may be used with non-IN classes such as HS and CH. The semantic of such use is not defined in this work.

B.6 Implementation of the SLOC RR for the Bind Name Server

An implementation of the SLOC resource record is available for the popular Internet Systems Consortium's BIND (Berkeley Internet Name Domain) DNS server [115]. It is available at [60].

Bibliography

[1] J. Abley, B. Black, and V. Gill. Goals for IPv6 Site-Multihoming Architectures. RFC 3582, IETF, August 2003.

[2] J. Abley, K. Lindqvist, B. Black, and V. Gill. IPv4 Multihoming Practices and Limitations. Internet Draft, IETF, January 2005. <draft-ietf-multi6-v4-multihoming-03.txt>, work in progress.

[3] S. Agarwal, C.-N. Chuah, and R. H. Katz. OPCA: Robust interdomain policy routing and traffic control. In *Proceedings of OPENARCH*, 2003.

[4] A. Akella, B. Maggs, S. Seshan, R. Sitaraman, and A. Shaikh. A Measurement-Based Analysis of Multihoming. In *Proceedings ACM SIGCOMM'03*, August 2003.

[5] A. Akella, S. Seshan, and A. Skaikh. Multihoming Performance Benefits: An Experimental Evaluation of Practical Enterprise Strategies. In *Proceedings of USENIX Annual Technical Conference*, 2004.

[6] P. Akkiraju and Y. Rekhter. A Multihoming solution using NATs. Internet Draft, November 1998. <draft-akkiraju-nat-multihoming-00.txt>, work in progress (expired).

[7] D. Allen. NPN: Multihoming and Route Optimization: Finding the Best Way Home. *Network Magazine*, February 2002.

[8] G. Almes, S. Kalidindi, and M. Zekauskas. A Round-trip Delay Metric for IPPM. RFC 2681, IETF, September 1999.

[9] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proceedings of ACM SOSP'01*, October 2001.

[10] APNIC. Allocation sizes within APNIC IPv4 address ranges. `https://www.apnic.net/db/min-alloc.html`, May 2005.

[11] APNIC, ARIN and RIPE NCC. IPv6 Address Allocation and Assignment Policy. Document ID: ripe-267, `http://www.ripe.net/ripe/docs/ipv6policy.html`, January 2003.

[12] ARIN. IP Address Space Allocated to ARIN. `https://www.arin.net/reference/ip_blocks.html`, May 2005.

[13] J. Arkko. Failure Detection and Locator Selection Design Considerations. Internet Draft, IETF, January 2005. <draft-ietf-shim6-failure-detection-00>, work in progress.

[14] I. S. Association. Guidelines for 64-Bit Global Identifier (EUI-64) Registration Authority. `http://standards.ieee.org/regauth/oui/tutorials/EUI64.html`, March 2005.

[15] R. Atkinson and S. Floyd. IAB Concerns and Recommendations Regarding Internet Research and Evolution. RFC 3869, IETF, August 2004.

[16] T. Aura. Cryptographically Generated Addresses (CGA). RFC 3972, IETF, March 2005.

[17] T. Aura, P. Nikander, and G. Camarillo. Effects of Mobility and Multihoming on Transport-Protocol Security. In *IEEE Symposium on Security and Privacy*, Berkeley, California, May 2004.

[18] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and Principles of Internet Traffic Engineering. RFC 3272, IETF, May 2002.

[19] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. RFC 2702, IETF, September 1999.

[20] M. Bagnulo. Hash Based Addresses (HBA). Internet Draft, IETF, July 2005. <draft-ietf-shim6-hba-00.txt>, work in progress.

[21] M. Bagnulo, A. García-Martínez, and A. Azcorra. Efficient Security for IPv6 Multihoming. *ACM SIGCOMM Computer Communications Review*, vol. 35, no. 5, April 2005.

[22] M. Bagnulo, A. García-Martínez, A. Azcorra, and C. de Launois. An Incremental Approach to IPv6 Multihoming. *To appear in Computer Communications*, 2005.

[23] M. Bagnulo, A. García-Martínez, A. Azcorra, and D. Larrabeiti. Survey on proposed IPv6 multi-homing network level mechanisms. Internet Draft, July 2001. <draft-bagnulo-multi6-survey6-00.txt>.

[24] M. Bagnulo, A. García-Martínez, and I. Soto. Application of the MIPv6 protocol to the multi-homing problem. Internet Draft, February 2003. <draft-bagnulo-multi6-mnm-00.txt>, work in progress (expired).

[25] F. Baker and P. Savola. Ingress Filtering for Multihomed Networks. BCP 84, IETF, March 2004.

[26] A. Barábasi and R. Albert. Emergence of Scaling in Random Networks. *Science*, vol. 286:509–512, October 1999.

[27] J. Bartlett. Optimizing multi-homed connections. *Business Communications Review*, 32(1), January 2002.

[28] T. Bates, R. Chandra, and E. Chen. BGP Route Reflection - An Alternative to Full Mesh IBGP. RFC 2796, IETF, April 2000.

[29] S. Bellovin, R. Bush, T. G. Griffin, and J. Rexford. Slowing routing table growth by filtering based on address allocation policies. Available from `http://www.cs.princeton.edu/~jrex/`, June 2001.

[30] M. Bhatia. Advertising Equal Cost Multi-Path (ECMP) routes in BGP. Internet draft, <draft-bhatia-ecmp-routes-in-bgp-00.txt>, work in progress (expired), May 2003.

[31] BitTorrent, Inc. BitTorrent. `http://www.bittorrent.com/`, April 2005.

[32] O. Bonaventure and B. Quoitin. Common utilizations of the BGP community attribute. Internet Draft, IETF, June 2003. <draft-bq-bgp-communities-00.txt>, work in progress (expired).

[33] O. Bonaventure, P. Trimintzios, G. Pavlou, B. Quoitin, A. Azcorra, M. Bagnulo, P. Flegkas, A. García-Martínez, P. Georgatsos, L. Georgiadis, C. Jacquenet, L. Swinnen, S. Tandel, and S. Uhlig. *Quality of Future Internet Services, Cost263 final report*, chapter Internet Traffic Engineering, pages 118–179. Number 2856 in LNCS. Springer-Verlag, September 2003.

[34] M. Borella, D. Grabelsky, J. Lo, and K. Taniguchi. Realm Specific IP : Protocol Specification. RFC 3103, IETF, October 2001.

[35] M. Borella, J. Lo, D. Grabelsky, and G. Montenegro. Realm Specific IP : Framework. RFC 3102, IETF, October 2001.

[36] N. Bragg. Routing support for IPv6 Multi-homing. Internet Draft, IETF, November 2000. <draft-bragg-ipv6-multihoming-00.txt>, work in progress (expired).

[37] T. Bu, L. Gao, and D. Towsley. On Routing Table Growth. In *Proceedings of IEEE Global Internet Symposium*, 2002.

[38] R. Bush and D. Meyer. Some Internet Architectural Guidelines and Philosophy. RFC 3439, IETF, December 2002.

[39] R. Callon. Use of OSI IS-IS for Routing in TCP/IP and Dual Environments. RFC 1195, IETF, December 1990.

[40] K. Calvert, M. Doar, and E. Zegura. Modeling Internet Topology. *IEEE Communications Magazine*, June 1997.

[41] K. Calvert, J. Eagan, S. Merugu, A. Namjoshi, J. Stasko, and E. Zegura. Extending and Enhancing GT-ITM. In *Proceedings of ACM SIGCOMM Workshops*, Karlsruhe, Germany, August 2004.

[42] Z. Cao, Z. Wang, and E. W. Zegura. Performance of Hashing-Based Schemes for Internet Load Balancing. In *Proceedings of IEEE INFOCOM'00*, pages 332–341, 2000.

[43] B. Carpenter. Internet Transparency. RFC 2775, IETF, February 2000.

[44] B. Carpenter and K. Moore. Connection of IPv6 Domains via IPv4 Clouds. RFC 3056, IETF, February 2001.

[45] E. Chen and S. R. Sangli. Avoid BGP Best Path Transition from One External to Another. Internet Draft, IETF, July 2004. <draft-chen-bgp-avoid-transition-01.txt>, work in progress (expired).

[46] E. Chen and J. Yuan. AS-wide Unique BGP Identifier for BGP-4. Internet Draft, IETF, April 2005. <draft-ietf-idr-bgp-identifier-05.txt>, work in progress.

[47] Cisco Systems Inc. Enhanced IGRP. `http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/en_igrp.htm`, April 2002.

[48] Cisco Systems, Inc. Cisco IOS Optimized Edge Routing. `http://www.cisco.com/warp/public/732/Tech/routing/oer/`, November 2004.

[49] L. Coene. Multihoming issues in the Stream Control Transmission Protocol. Internet Draft, May 2002. <draft-coene-sctp-multihome-04.txt>, work in progress (expired).

[50] L. Coene. Stream Control Transmission Protocol Applicability Statement. RFC 3257, IETF, April 2002.

[51] L. Coene and J. Loughney. Multihoming: the SCTP solution. Internet Draft, IETF, January 2004. <draft-coene-multi6-sctp-00.txt>, work in progress (expired).

[52] M. Costa, M. Castro, A. Rowstron, and P. Key. PIC: Practical Internet Coordinates for Distance Estimation. In *International Conference on Distributed Systems*, Tokyo, Japan, March 2004.

[53] M. Crawford. Transmission of IPv6 Packets over Ethernet Networks. RFC 2464, IETF, December 1998.

[54] M. Crawford. Router Renumbering for IPv6. RFC 2894, IETF, August 2000.

[55] F. Dabek, F. Kaashoek, and R. Morris. Vivaldi: A Decentralized Network Coordinate System. In *Proceedings of ACM SIGCOMM'04*, Portland, Oregon, USA, August 2004.

[56] C. Davis, P. Vixie, T. Goodwin, and I. Dickinson. A Means for Expressing Location Information in the Domain Name System. RFC 1876, IETF, January 1996.

[57] C. de Launois. The NAROS Approach for IPv6 Multihoming with Traffic Engineering. `http://www.info.ucl.ac.be/people/delaunoi/naros/`, June 2003.

[58] C. de Launois. GHITLE. `http://openresources.info.ucl.ac.be/ghitle/`, April 2004.

[59] C. de Launois. Leveraging Internet Path Diversity and Network Performances with IPv6 Multihoming. Research Report RR 2004-06, Université catholique de Louvain - Department of Computer Science and Engineering, August 2004. `http://www.info.ucl.ac.be/people/delaunoi/diversity/`.

[60] C. de Launois. An Implementation of the SLOC RR for the Bind Name Server. `http://www.info.ucl.ac.be/people/delaunoi/`, June 2005.

[61] C. de Launois. RSIP for LINUX. `http://openresources.info.ucl.ac.be/rsip/`, May 2005.

[62] C. de Launois. Scalable Route Selection for IPv6 Multihomed Sites. `http://www.info.ucl.ac.be/people/delaunoi/svivaldi/`, November 2004.

[63] C. de Launois and M. Bagnulo. The Paths Towards IPv6 Multihoming. Submitted to IEEE Network Magazine, April 2005.

[64] C. de Launois and O. Bonaventure. NAROS : Host-Centric IPv6 Multihoming with Traffic Engineering. Internet Draft, May 2003. <draft-de-launois-multi6-naros-00.txt>, work in progress (expired).

[65] C. de Launois, O. Bonaventure, and M. Lobelle. The NAROS Approach for IPv6 Multihoming with Traffic Engineering. In *4th COST 263 International Workshop on Quality of Future Internet Services (QoFIS 2003)*, volume LNCS 2811, pages 112–121, Stockholm, Sweden, October 2003. Springer-Verlag.

[66] C. de Launois, A. Bonnet, and M. Lobelle. Connection of Extruded Subnets : a Solution Based on RSIP. In *Second International IFIP-TC6 Networking Conference (Networking 2002)*, volume LNCS 2345, pages 685–696, Pisa, Italy, May 2002. Springer-Verlag.

[67] C. de Launois, A. Bonnet, and M. Lobelle. Connection of Extruded Subnets : a Solution Based on RSIP. *IEEE Communication Magazine*, vol. 40, no. 9, September 2002.

[68] C. de Launois, G. Fauveaux, and J. Honlet. Routeur d'accès à adresse dynamique. Master's thesis, Université catholique de Louvain, June 2001.

[69] C. de Launois, B. Quoitin, and O. Bonaventure. Leveraging network performances with IPv6 multihoming and multiple provider-dependent aggregatable prefixes. In *3rd International Workshop on QoS in Multiservice IP Networks (QoSIP 2005)*, volume LNCS 3375, pages 339–352, Catania, Italy, February 2005. Springer-Verlag.

[70] C. de Launois, S. Uhlig, and O. Bonaventure. Scalable Route Selection for IPv6 Multihomed Sites. In *4th International IFIP-TC6 Networking Conference (Networking 2005)*, volume LNCS 3462, pages 1357–1361, Waterloo, Ontario, Canada, May 2005. Springer-Verlag.

[71] S. Deering, B. Haberman, T. Jinmei, E. Nordmark, and B. Zill. IPv6 Scoped Address Architecture. RFC 4007, IETF, March 2005.

[72] L. Deri. Passively Monitoring Networks at Gigabit Speeds Using Commodity Hardware and Open Source Software. In *Proceedings Passive & Active Measurement Workshop (PAM)*, April 2003.

[73] R. Draves. Default Address Selection for Internet Protocol version 6 (IPv6). RFC 3484, IETF, February 2003.

[74] R. Draves and R. Hinden. Default Router Preferences, More-Specific Routes. Internet Draft, IETF, June 2005. <draft-ietf-ipv6-router-selection-07.txt>, work in progress (expired).

[75] R. Droms. Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6. RFC 3736, IETF, April 2004.

[76] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315, IETF, July 2003.

[77] F. Dupont. Multihomed routing domain issues for IPv6 aggregatable scheme. Internet Draft, IETF, September 1999. <draft-ietf-ipngwg-multi-isp-00.txt>, work in progress (expired).

[78] A. Durand and C. Huitema. The Host-Density Ratio for Address Assignment Efficiency: An update on the H ratio. RFC 3194, IETF, November 2001.

[79] D. Eastlake. Domain Name System Security Extensions. RFC 2535, IETF, March 1999.

[80] K. Egevang and P. Francis. The IP Network Address Translator (NAT). RFC 1631, IETF, May 1994.

[81] P. Faltstrom. Design Choices When Expanding DNS. Internet Draft, IETF, June 2005. <draft-iab-dns-choices-02.txt>, work in progress.

[82] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. BCP 38, RFC 2827, IETF, May 2000.

[83] S. Floyd, M. Handley, and E. Kohler. Problem Statement for DCCP. Internet Draft, IETF, October 2002. <draft-ietf-dccp-problem-00.txt>, work in progress (expired).

[84] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: A Global Internet Host Distance Estimation Service. In *Proceedings of IEEE/ACM Transactions on Networking*, October 2001.

[85] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy. RFC 1519, IETF, September 1993.

[86] L. Gao. On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, December 2001.

[87] F. Georgatos, F. Gruber, D. Karrenberg, M. Santcroos, A. Susanj, H. Uijterwaal, and R. Wilhelm. Providing Active Measurements as a Regular Service for ISP's. In *Proceedings of PAM'01*, Amsterdam, April 2001. `http://www.ripe.net/ttm`.

[88] R. Gilligan and E. Nordmark. Transition Mechanisms for IPv6 Hosts and Routers. RFC 2893, IETF, August 2000.

[89] R. Gilligan, S. Thomson, J. Bound, J. McCann, and W. Stevens. Basic Socket Interface Extensions for IPv6. RFC 3493, IETF, February 2003.

[90] T. Griffin and G. Wilfong. Analysis of the MED oscillation problem in BGP. In *ICNP2002*, 2002.

[91] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating Latency between Arbitrary Internet End Hosts. In *Proceedings of IMW 2002*, Marseille, France, November 2002.

[92] J. Hagino and H. Snyder. IPv6 Multihoming Support at Site Exit Routers. RFC 3178, IETF, October 2001.

[93] T. Hain. Architectural Implications of NAT. RFC 2993, IETF, November 2000.

[94] T. Hain. Application and Use of the IPv6 Provider Independent Global Unicast Address Format. Internet Draft, IETF, August 2004. <draft-hain-ipv6-pi-addr-use-07.txt>, work in progress (expired).

[95] K. Harrenstien, M. Stahl, and E. Feinler. DoD Internet Host Table Specification. RFC 952, IETF, October 1985.

[96] K. Harrenstien, M. Stahl, and E. Feinler. HOSTNAME Server. RFC 953, IETF, October 1985.

[97] C. Hedrick. Routing Information Protocol. RFC 1058, IETF, June 1988.

[98] R. Hinden and S. Deering. Internet Protocol, Version 6 (IPv6), Specification. RFC 2460, IETF, December 1998.

[99] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 2373, IETF, July 1998.

[100] R. Hinden and S. Deering. Internet Protocol Version 6 (IPv6) Addressing Architecture. RFC 3513, IETF, April 2003.

[101] R. Hinden, S. Deering, and E. Nordmark. IPv6 Global Unicast Address Format. RFC 3587, IETF, August 2003.

[102] B. Huffaker, M. Fomenkov, D. Plummer, D. Moore, and K. Claffy. Distance Metrics in the Internet. In *Proc. of IEEE International Telecommunications Symposium (ITS)*, September 2002.

[103] C. Huitema. The H Ratio for Address Assignment Efficiency. RFC 1715, IETF, November 1994.

[104] C. Huitema. Multi-homed TCP. Internet Draft, IETF, May 1995. <draft-huitema-multi-homed-01.txt>, work in progress (expired).

[105] C. Huitema and B. Carpenter. Deprecating Site Local Addresses. RFC 3879, IETF, September 2004.

[106] C. Huitema, R. Draves, and M. Bagnulo. Host-Centric IPv6 Multihoming. Internet Draft, February 2004. <draft-huitema-multi6-hosts-03.txt>, work in progress (expired).

[107] C. Huitema, R. Draves, and M. Bagnulo. Ingress filtering compatibility for IPv6 multihomed sites. Internet Draft, IETF, October 2004. <draft-huitema-multi6-ingress-filtering-00.txt>, work in progress (expired).

[108] G. Huston. Interconnection, peering, and settlements. In *Proceedings of INET*, San Jose, California, USA, June 1999.

[109] G. Huston. Analyzing the Internet BGP routing table. *Internet Protocol Journal*, March 2001.

[110] G. Huston. Commentary on Inter-Domain Routing in the Internet. RFC 3221, IETF, December 2001.

[111] G. Huston. Architectural Approaches to Multi-Homing for IPv6. Internet Draft, IETF, October 2004. <draft-ietf-multi6-architecture-02.txt>, work in progress.

[112] G. Huston. BGP Routing Table Analysis Reports. `http://bgp.potaroo.net`, May 2004.

[113] G. Huston. IPv4 Address Space Report. `http://bgp.potaroo.net/ipv4/`, May 2005.

[114] IAB and IESG. IAB/IESG Recommendations on IPv6 Address Allocations to Sites. RFC 3177, September 2001.

[115] Internet Systems Consortium, Inc. ISC BIND. `http://www.isc.org/sw/bind/`, April 2005.

[116] J. Papen. Demystifying BGP. Linux Magazine, `http://www.linux-mag.com/2003-05/bgp_01.html`, May 2003.

[117] J. Jieyun. IPv6 Multi-Homing with Route Aggregation. Internet Draft, IETF, August 2000. <draft-ietf-ipngwg-ipv6multihome-with-aggr-01.txt>, work in progress (expired).

[118] C. Jin, Q. Chen, and S. Jamin. Inet: Internet Topology Generator. Technical Report CSE-TR-433-00, 2000.

[119] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. RFC 3775, IETF, June 2004.

[120] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, IETF, November 1998.

[121] E. Kohler. Datagram Congestion Control Protocol Mobility and Multihoming. Internet Draft, IETF, July 2004. <draft-kohler-dccp-mobility-00.txt>, work in progress (expired).

[122] E. Kohler, M. Handley, and S. Floyd. Datagram Congestion Control Protocol (DCCP). Internet Draft, IETF, November 2004. <draft-ietf-dccp-spec-09.txt>, work in progress.

[123] C. Labovitz, G. R. Malan, and F. Jahanian. Internet Routing Instability. In *Proceedings of ACM SIGCOMM'97*, September 1997.

[124] S. Lee, Z.-L. Zhang, and S. Nelakuditi. Impact of AS Hierarchy on Multihoming Performance : A Stub Network Perspective. In *Technical Report TR 04-008*, University of Minnesota - Department of Computer Science and Engineering, February 2004.

[125] K. Lindqvist. Multihoming in IPv6 by multiple announcements of longer prefixes. Internet Draft, IETF, December 2002. <draft-kurtis-multihoming-longprefix-00.txt>, work in progress (expired).

[126] A. A. B. Maggs, J. Pang, S. Seshan, and A. Skaikh. A comparison of Overlay Routing and Multihoming Route Control. In *Proceedings ACM SIGCOMM'04*, August 2004.

[127] G. Malkin. RIP Version 2. RFC 2453, IETF, November 1998.

[128] A. Matsumoto, T. Fujisaki, H. Matsuoka, and J. Kato. Source Address Selection Policy Distribution for Multihoming. Internet Draft, IETF, October 2004. <draft-arifumi-multi6-sas-policy-dist-00.txt>, work in progress (expired).

[129] A. Matsumoto, M. Kozuka, and K. Fujikawa. TCP Multi-Home Options. Internet Draft, IETF, October 2003. <draft-arifumi-tcp-mh-00.txt>, work in progress (expired).

[130] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An Approach to Universal Topology Generation. In *Proceedings MASCOTS '01*, August 2001.

[131] D. Meyer. Route Views Project. `http://antc.uoregon.edu/route-views`, August 2004.

[132] P. Mockapetris. Domain Names - Concepts and Facilities. RFC 1034, IETF, November 1987.

[133] P. Mockapetris. Domain Names - Implementation and Specification. RFC 1035, IETF, November 1987.

[134] P. Mockapetris. DNS Encoding of Network Names and Other Types. RFC 1011, IETF, April 1989.

[135] G. Montenegro and M. Borella. RSIP Support for End-to-end IPSEC. RFC 3104, IETF, October 2001.

[136] R. Moskowitz and P. Nikander. Host Identity Protocol Architecture. Internet Draft, IETF, June 2004. <draft-moskowitz-hip-arch-06.txt>, work in progress (expired).

[137] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host Identity Protocol. Internet Draft, IETF, February 2005. <draft-ietf-hip-base-02.txt>, work in progress.

[138] J. Moy. OSPF Version 2. RFC 2328, IETF, April 1998.

[139] A. Nagarajan and H. Tschofenig. Comparative Analysis of Multi6 Proposals using a Locator/Identifier Split. Internet Draft, IETF, October 2004. <draft-nagarajan-multi6-comparison-00.txt>, work in progress.

[140] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). RFC 2461, IETF, December 1998.

[141] T. S. E. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *Proceedings of IEEE INFOCOM'02*, New York, USA, June 2002.

[142] T. S. E. Ng and H. Zhang. A network positioning system for the internet. In *Proceedings of USENIX Conference*, June 2004.

[143] P. Nikander. Considerations on HIP based IPv6 multi-homing. Internet Draft, IETF, December 2003. <draft-nikander-multi6-hip-00.txt>, work in progress (expired).

[144] P. Nikander, J. Arkko, T. Aura, G. Montenegro, and E. Nordmark. Mobile IP version 6 Route Optimization Security Design Background. Internet Draft, October 2004. <draft-ietf-mip6-ro-sec-02.txt>, work in progress.

[145] P. Nikander, J. Arkko, and T. Henderson. End-Host Mobility and Multi-Homing with Host Identity Protocol. Internet Draft, IETF, February 2005. <draft-ietf-hip-mm-01.txt>, work in progress.

[146] E. Nordmark. Multihoming using 64-bit Crypto-based IDs. Internet Draft, IETF, October 2003. <draft-nordmark-multi6-cb64-00.txt>, work in progress (expired).

[147] E. Nordmark. Strong Identity Multihoming using 128 bit Identifiers (SIM/CBID128). Internet Draft, IETF, October 2003. <draft-nordmark-multi6-sim-01.txt>, work in progress (expired).

[148] E. Nordmark. Multihoming without IP Identifiers. Internet Draft, IETF, July 2004. <draft-nordmark-multi6-noid-02.txt>, work in progress (expired).

[149] E. Nordmark. Multi6 Application Referral Issues. Internet Draft, IETF, January 2005. <draft-ietf-multi6-app-refer-00.txt>, work in progress.

[150] E. Nordmark and M. Bagnulo. Multihoming L3 Shim Approach. Internet Draft, IETF, July 2005. <draft-ietf-shim6-l3shim-00.txt>, work in progress.

[151] E. Nordmark and T. Li. Threats relating to IPv6 multihoming solutions. Internet Draft, IETF, January 2005. <draft-ietf-multi6-multihoming-threats-03.txt>, work in progress.

[152] Nortel Networks. Alteon Link Optimizer. `http://www.nortelnetworks.com/products/01/alteon/optimizer/`, November 2004.

[153] M. Ohta. The Architecture of End to End Multihoming. Internet Draft, IETF, June 2003. <draft-ohta-e2e-multihoming-05.txt>, work in progress (expired).

[154] L. Ong and J. Yoakum. An Introduction to the Stream Control Transmission Protocol (SCTP). RFC 3286, IETF, May 2002.

[155] G. O'Shea and M. Roe. Child-proof Authentication for MIPv6 (CAM). *ACM Computer Communications Review*, vol. 31, no. 2, April 2001.

[156] P. Wilson and R. Plzak and A. Pawlik. IPv6 Address Space Management. Document ID: ripe-343, `http://www.ripe.net/ripe/docs/ipv6-sparse.html`, February 2005.

[157] M. Pias, J. Crowcroft, S. Wilbur, S. Bhatti, and T. Harris. Lighthouses for scalable distributed location. In *Proceedings of IPTPS'03*, February 2003.

[158] J. Postel and J. Reynolds. File Transfer Protocol. STD 9, RFC 959, IETF, October 1985.

[159] M. Py. Multi Homing Translation Protocol (MHTP). Internet Draft, IETF, November 2001. <draft-py-multi6-mhtp-01.txt>, work in progress (expired).

[160] M. Py. Multi Homing Aliasing Protocol (MHAP) intro. Internet Draft, IETF, March 2003. <draft-py-mhap-intro-00.txt>, work in progress (expired).

[161] M. Py and I. van Beijnum. GAPI: A Geographically Aggregatable Provider Independent Address Space to Support Multihoming in IPv6. Internet Draft, IETF, October 2002. <draft-py-multi6-gapi-00.txt>, work in progress (expired).

[162] B. Quoitin. C-BGP - An efficient BGP simulator. `http://cbgp.info.ucl.ac.be/`, March 2004.

[163] B. Quoitin. Towards a POP-level Internet topology. `http://cbgp.info.ucl.ac.be/itopo/`, August 2004.

[164] B. Quoitin and O. Bonaventure. A Cooperative Approach to Interdomain Traffic Engineering. In *1st Conference on Next Generation Internet Networks Traffic Engineering (NGI 2005)*, Rome, Italy, April 2005.

[165] B. Quoitin, C. Pelsser, O. Bonaventure, and S. Uhlig. A performance evaluation of BGP-based traffic engineering. *International Journal of Network Management (Wiley)*, vol. 15, no. 3, May-June 2004.

[166] B. Quoitin, S. Tandel, S. Uhlig, and O. Bonaventure. Interdomain traffic engineering with Redistribution Communities. *Computer Communications Journal*, vol. 27, no. 4:355–363, October 2003.

[167] B. Quoitin, S. Uhlig, and O. Bonaventure. Using redistribution communities for interdomain traffic engineering. In *3d COST 263 International Workshop on Quality of Future Internet Services (QoFIS 2002)*, volume LNCS 2511. Springer-Verlag, October 2002.

[168] B. Quoitin, S. Uhlig, C. Pelsser, L. Swinnen, and O. Bonaventure. Interdomain traffic engineering with BGP. *IEEE Communications Magazine*, May 2003.

[169] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1771, IETF, March 1995.

[170] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). Internet Draft, IETF, October 2004. <draft-ietf-idr-bgp4-26.txt>, work in progress.

[171] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. BCP 5, RFC 1918, IETF, February 1996.

[172] RIPE NCC. Smallest RIPE NCC Allocation / Assignment Sizes. Document ID: ripe-345, `https://www.ripe.net/ripe/docs/smallest-alloc-sizes.html`, April 2005.

[173] R. Rockell and R. Fink. 6Bone Backbone Routing Guidelines. RFC 2772, IETF, February 2000.

[174] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, IETF, June 2002.

[175] RouteScience Technologies, Inc. RouteScience Adaptive Networking Software. `http://www.routescience.com`, November 2004.

[176] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-To-End Arguments in System Design. *ACM Transactions on Computer Systems*, vol. 2, no. 4:277–288, November 1984.

[177] P. Savola. Examining Site Multihoming in Finnish Networks. Master's thesis, April 2003.

[178] P. Savola. Multihoming Using IPv6 Addressing Derived from AS Numbers. Internet Draft, IETF, January 2003. <draft-savola-multi6-asn-pi-00.txt>, work in progress (expired).

[179] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). RFC 2326, IETF, April 1998.

[180] Sharman Networks. KaZaA. `http://www.kazaa.com/`, April 2005.

[181] Y. Shavitt and T. Tankel. Big-Bang Simulation for embedding network distances in Euclidean space. In *Proceedings of IEEE INFOCOM'03*, San Francisco, CA, USA, April 2003.

[182] Y. Shavitt and T. Tankel. On the curvature of the Internet and its usage for overlay construction and distance estimation. In *Proceedings of IEEE INFOCOM'04*, April 2004.

[183] N. Spring, R. Mahajan, and D.Wetherall. Measuring ISP Topologies with Rocketfuel. In *Proceedings ACM SIGCOMM '02*, August 2002.

[184] P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663, IETF, August 1999.

[185] W. Stallings. *SNMP, SNMPv2 and RMON - Practical Network Management, Second edition.* Addison-Wesley, 1996.

[186] J. W. Stewart. *BGP4: Inter-Domain Routing in the Internet.* Addison-Wesley, 1999.

[187] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, and P. Conrad. Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. Internet Draft, IETF, February 2005. <draft-ietf-tsvwg-addip-sctp-11.txt>, work in progress.

[188] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol. RFC 2960, IETF, October 2000.

[189] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the Internet Hierarchy from Multiple Vantage Points. In *Proceedings of IEEE INFOCOM'02*, June 2002.

[190] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the Internet Hierarchy from Multiple Vantage Points. `http://www.cs.berkeley.edu/~sagarwal/research/BGP-hierarchy/`, May 2004.

[191] L. Tang and M. Crovella. Virtual Landmarks for the Internet. In *Proceedings of the ACM/SIGCOMM Internet Measurement Conference 2003*, pages 143–152, Miami Beach, Florida, October 2003.

[192] P. Tattam. Preserving active TCP sessions on Multi-homed networks, August 2001. `http://jazz-1.trumpet.com.au/ipv6-draft/preserve-tcp.txt`.

[193] R. Teixeira, K. Marzullo, S. Savage, and G. M. Voelker. Characterizing and Measuring Path Diversity of Internet Topologies. In *Proceedings of SIGMETRICS'03*, June 2003.

[194] R. Teixeira, K. Marzullo, S. Savage, and G. M. Voelker. In Search of Path Diversity in ISP Network. In *Proceedings of IMC'03*, October 2003.

[195] F. Terakoa, M. Ishiyama, and Kunishi. LIN6: A Solution to Mobility and Multi-Homing in IPv6. Internet Draft, December 2003. <draft-teraoka-multi6-lin6-00.txt>, work in progress (expired).

[196] R. Texeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of Hot-Potato Routing in IP networks. In *SIGMETRICS/Performance'04*, New York, NY, USA, June 2004.

[197] S. Thomson, C. Huitema, V. Ksinant, and M. Souissi. DNS Extensions to Support IP Version 6. RFC 3596, IETF, October 2003.

[198] S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration. RFC 2462, IETF, December 1998.

[199] K. Toyama and T. Fujisaki. Operational Approach to achieve IPv6 multihomed network. Internet Draft, IETF, February 2004. <draft-toyama-multi6-operational-site-multihoming-00.txt>, work in progress (expired).

[200] P. Traina, R. Chandrasekeran, and T. Li. BGP Communities Attribute. RFC 1997, IETF, August 1996.

[201] P. Traina, D. McPherson, and J. Scudder. Autonomous System Confederations for BGP. RFC 3065, IETF, February 2001.

[202] O. Troan and R. Droms. IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6. RFC 3633, IETF, December 2003.

[203] S. Uhlig. *Implications of Traffic Characteristics on Interdomain Traffic Engineering*. PhD thesis, Université catholique de Louvain, March 2004.

[204] S. Uhlig and B. Quoitin. Tweak-it: BGP-based interdomain traffic engineering for transit ASes. In *1st Conference on Next Generation Internet Networks Traffic Engineering (NGI 2005)*, Rome, Italy, April 2005.

[205] I. van Beijnum. *BGP*. O'Reilly, 2002.

[206] I. van Beijnum. Provider-Internal Aggregation based on Geography to Support Multihoming in IPv6. Internet Draft, IETF, October 2002. <draft-van-beijnum-multi6-isp-int-aggr-00.txt>, work in progress (expired).

[207] I. van Beijnum. Shim6 Reachability Detection. Internet Draft, IETF, July 2005. <draft-ietf-shim6-reach-detect-00.txt>, work in progress.

[208] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136, IETF, April 1997.

[209] D. Walton, D. Cook, A. Retana, and J. Scudder. Advertisement of Multiple Paths in BGP. Internet draft, <draft-walton-bgp-add-paths-01.txt>, work in progress (expired), November 2002.

[210] B. Wellington. Secure Domain Name System (DNS) Dynamic Update. RFC 3007, IETF, November 2000.

[211] R. White and A. Retana. *IS-IS : Deployment in IP Networks*. Addison-Wesley, March 2004.

[212] J. Ylitalo, V. Torvinen, and E. Nordmark. Weak Identifier Multihoming Protocol (WIMP). Internet Draft, IETF, January 2004. <draft-ylitalo-multi6-wimp-00.txt>, work in progress (expired).

[213] B. Zhang, R. Liu, D. Massey, and L. Zhang. Collecting the internet AS-level topology. *ACM SIGCOMM Computer Communications Review*, vol. 35, no. 1:53–61, 2005.