# BGP-based Interdomain Traffic Engineering

Bruno Quoitin

*Thesis submitted in partial fulfillment of the requirements for the Degree of Doctor in Applied Sciences*

August 4, 2006

Faculté des Sciences Appliquées
Département d'Ingénierie Informatique
Université catholique de Louvain
Louvain-la-Neuve
Belgium

# BGP-based Interdomain Traffic Engineering

by Bruno Quoitin

Computer Science and Engineering Department
Université catholique de Louvain
Place Sainte-Barbe, 2
1348 Louvain-la-Neuve
Belgium

This work was supported by the European Commission within the ATRIUM project and by the Walloon Government under the WIST TOTEM project. This work was also partially supported by the European Union under the E-Next project FP6-506869 and by a grant of France Telecom R&D.

*To my parents, wife and son*

# Preamble

In a few years, the Internet has rapidly evolved from a research network serving a handful of users to a huge interconnection of about 350 million hosts (June 2005 [ISC05]). In this way, the Internet is the largest distributed system ever built. The Internet is organized in a multitude of administratively independent networks called domains or Autonomous Systems (AS). For example, an AS can be an Internet Service Provider (ISP), a University campus or a corporate network. At the time of this writing, there are more than 21,000 ASs in the Internet [Hus06]. Over this huge infrastructure, there is a growing trend to deploy new applications such as the transmission of Voice or Video over IP and new services such as Virtual Private Networks (VPNs). These new applications and services require better or strict guarantees of quality while the Internet has been designed to provide a best-effort service. This evolution puts a lot of pressure on the ISPs for whom the ability to offer better than best-effort service or support tight Service Level Agreements (SLAs) [FE04] is a key differentiating factor.

Network engineers rely on Traffic Engineering (TE) to adapt the configuration of their network in order to support the evolution of the traffic demand and the customer SLAs. Traffic Engineering is defined by the IETF TE Working Group as the process of evaluating and enhancing operational IP networks performance [ACE$^+$02]. The objectives of Traffic Engineering can be summarized in avoiding congestion, providing resilience and supporting Quality of Service (QoS). Most of the Traffic Engineering complexity comes from hop-by-hop destination-based IP forwarding, i.e. each router on the path selects the next router to forward the packet to based on the packet destination only. There is no way to explicitly determine the path followed by IP datagrams to reach their destination. Moreover, the routing decisions are taken in a distributed manner by each hop along the path. One of the main difficulties of Traffic Engineering comes thus from routing. Inside a single domain, routing is done thanks to link-state protocols such as IS-IS or OSPF. From the perspective of Traffic Engineering, these protocols have the twofold advantage of propagating information on the whole topology and optimizing a single global objective (least-cost-path). Intradomain Traffic Engineering is a well understood problem and solutions exist [FT00, FRT02].

In contrast, when Traffic Engineering has to be performed over the boundaries of multiple domains, things are far more difficult. Central to the problem is the Internet routing system itself. Internet routing is currently built around the Border Gateway Protocol (BGP). BGP is a path-vector protocol, that is it propagates only a limited view of the topology. A BGP router will advertise to its neighbors a single route per reachable destination. Given the size of the Internet, this has serious advantages in terms of scalability and stability. A second characteristic of BGP is that each domain is administered independently. For this reason, BGP in each domain is configured to optimize local objectives. The objectives of one domain might be very different from those of another one. Moreover, each domain is allowed to filter the routes advertised to other domains. This poses serious challenges for interdomain Traffic Engineering. First, the limited view of the topology due to the path-vector nature of BGP and due to the local routing policies decrease the diversity of interdomain paths and subsequently the freedom of an AS to direct traffic along alternative paths. Second, an AS is not eager to let other ASs control the routing in its own network. BGP provides very limited control on the routing decisions taken by other domains.

The main purpose of this thesis is to study the Internet-scale selection of routes performed by BGP and the performance of the current BGP-based routing control techniques that could support interdomain Traffic Engineering. There are four main contributions in this thesis:

1. The **design and implementation of a BGP modeling tool**, C-BGP (Chapter 2). Our tool can compute the BGP routes in large-scale network topologies. This tool has two main applications. It can be used by ISP network operators to better understand their routing. It can also be used by researchers to study the macroscopic characteristics of Internet routing.

2. A **methodology to evaluate routing what-if scenarios in ISP networks** (Chapter 3). Due to the interaction between two types of routing protocols (link-state and path-vector), predicting the impact on the routing of an ISP network of topological and configuration changes is a complex task. We show how C-BGP can handle a large part of this complexity. We apply this methodology on a real transit network.

3. A **large-scale performance evaluation of current BGP-based traffic engineering techniques** (Chapter 4). We first survey the existing BGP-based routing control mechanisms. Then, using C-BGP on an Internet-scale topology, we evaluate two of these mechanisms: AS-Path prepending and Redistribution Communities. We show that these mechanisms are coarse and non-deterministic.

4. We propose **Virtual Peerings**, a new mechanism to engineer the traffic exchanged between two cooperating, but non adjacent ISPs (Chapter 5). This mechanism is deterministic, scalable and is almost readily deployable. We evaluate the utilization of virtual peerings to solve two traffic engineering objectives: load-balancing and improving end-to-end latency.

## Road map

The thesis is organized in three parts. The first part provides the background required to understand the thesis. This part can be skipped by readers who are familiar with Internet routing and Traffic Engineering. In the second part, we introduce the BGP modeling tool we have designed and implemented. We also show how to apply this tool to model the network of an ISP. The last part surveys BGP-based traffic engineering mechanisms and presents an evaluation of their performance. This part also proposes and evaluates a new Traffic Engineering mechanism called *Virtual Peerings*. The last part can be read independently from the others since understanding the internals of the modeling tool is not required to understand the simulation results.

### Part I - Background

In **Chapter 1**, we provide the background notions required to understand the thesis. We first give an overview of the organization of the Internet. We explain that the Internet is composed of domains that provide transit and domains that don't. The latter are called stub domains and they represent approximately 85% of the Internet domains. Secondly, we describe how routing is done between Internet domains. In particular, we give a detailed description of BGP, the current interdomain routing protocol. BGP is not based on the optimization of a single metric as in link-state protocols but on a complex decision process composed of several rules. Then, we give a brief overview of the Traffic Engineering process. We explain why it is difficult when performed across the boundaries of multiple domains. Finally, we describe the practice of multi-homing which is increasingly deployed by stub domains to improve the performance and robustness of their Internet access. We indicate that these stub domains have to face the lack of mechanisms provided by BGP to control their interdomain traffic.

### Part II - Modeling BGP Routing

In **Chapter 2**, we discuss the modeling of BGP routing and the evaluation of interdomain traffic engineering techniques. We show that this evaluation is challenging for two main reasons. First, the Internet topology is large which makes simulations computationally expensive. Second, the route selection performed by BGP is complex and modeling cannot rely on shortcuts as used for link-state protocols. We survey the tools that are traditionally used to study BGP routing and we conclude that none of them is currently suitable to efficiently study the BGP route selection in the global Internet. We define a new approach to the problem that we call a BGP routing solver. Then, we describe C-BGP, our implementation of such a BGP routing solver. This tool is used throughout the thesis.

In **Chapter 3**, we apply C-BGP to modeling a single Internet domain. C-BGP allows a network operator to build a model of its network and inject configuration, routing and traffic data for the purpose of investigating what-if scenarios. We describe the technical issues related to this modeling. Then, we apply our methodology to a real ISP network. We evaluate two different what-if scenarios. In the first one, we investigate the impact of adding or removing peerings on the routing. In the second scenario, we evaluate the impact of single link and single router failures on the routing. Based on such a study, a network operator can determine which links should be protected in order to provide a service resilient to failures.

### Part III - BGP-based Interdomain Traffic Engineering

In the third part of the thesis, we do not limit the modeling to a single domain, but we extend it to a large interconnection of networks. We apply the modeling tool to an Internet-like network in order to study the efficiency of current BGP-based traffic engineering techniques.

In **Chapter 4**, we describe how traffic engineering is performed today. We explain that controlling the traffic that leaves an ISP is feasible since only a local control is needed. In contrast, controlling the traffic coming in the reverse direction is harder since the routing in distant domains must be influenced. Based on large-scale simulations of BGP, we show that many routing decisions in the Internet model are taken randomly. In addition, a network in the Internet has a very limited view of the whole topology. This limits the applicability of traffic engineering techniques such as AS-Path prepending or Redistribution Communities. We show that these techniques do not provide a predictable, fine-grained control on the interdomain traffic flows. We conclude that there is thus a need to develop a new traffic engineering mechanism.

In **Chapter 5**, we setup the requirements for such a new technique: it must be predictable, scalable and deployable in today's Internet. We propose the Virtual Peerings which are a mean for two cooperating networks to better control the paths between each other. The Virtual Peerings have the following advantages. First, they provide a deterministic control on the interdomain paths. Second, they can be deployed in the current Internet since they are transparent to the intermediate domains. We use Virtual Peerings to solve two different traffic engineering problems faced by multi-homed stub domains. The first one is balancing the load of the interdomain traffic received by a stub domain on its access links. We show that a small number of Virtual Peerings is required to reach the objective of a near-perfect balance. The second problem that we explore is the utilization of Virtual Peerings to forward traffic along interdomain paths with a lower latency than the default BGP routes.

## Bibliographic Notes

Most of the work presented in this thesis appears in previously published conference proceedings and journals. The list of related publications is shown hereafter:

- S. Uhlig, B. Quoitin, S. Balon and J. Lepropre, *Providing public intradomain traffic matrices to the research community*. In ACM SIGCOMM Computer Communication Review, Vol 36(1), January 2006.

- B. Quoitin, *Topology generation based on network design heuristics*. In Proceedings of the CoNEXT Student workshop, Toulouse, France, October 2005.

- C. de Launois, B. Quoitin and O. Bonaventure, *Leveraging Network Performances with IPv6 Multihoming and Multiple Provider-Dependant Aggregatable Prefixes*. Computer Networks, Vol 50(8), p. 1145-1157, June 2006.

- B. Quoitin and S. Uhlig, *Modeling the routing of an Autonomous System with C-BGP*. In IEEE Network Magazine, Vol 19(6), November 2005, p. 12-19, ISSN: 0890-8044.

- B. Quoitin and S. Tandel, *A BGP Solver for Hot-Potato Routing Sensitivity Analysis*. In Proceedings of EUNICE 2005, Colmenarejo, Spain, July 2005.

- B. Quoitin and O. Bonaventure, *A cooperative approach to interdomain traffic engineering*. In Proceedings of the 1st Conference on Next Generation Internet Networks Traffic Engineering (NGI 2005), Rome, Italy, April 2005.

- S. Uhlig and B. Quoitin, *Tweak-it: BGP-based interdomain traffic engineering for transit ASes*. In Proceedings of the 1st Conference on Next Generation Internet Networks Traffic Engineering (NGI 2005), Rome, Italy, 2005.

- S. Uhlig, C. Pelsser, B. Quoitin, and O. Bonaventure, *Vers des réflecteurs de routes plus intelligents*. In Proceedings of the Colloque Francophone sur l'Ingénierie des Protocoles (CFIP 2005), Bordeaux, France, March 29-April 1st 2005.

- C. de Launois, B. Quoitin, and O. Bonaventure, *Leveraging network performances with IPv6 multihoming and multiple provider-dependent aggregatable prefixes*. In Proceedings of the 3rd International Workshop on QoS in Multiservice IP Networks (QoSIP 2005), Catania, Italy, February 2-4th 2005.

- B. Quoitin, C. Pelsser, O. Bonaventure, and S. Uhlig, *A performance evaluation of BGP-based traffic engineering*. In International Journal of Network Management (Wiley), Vol 15(3), p. 177-191, May/June 2005.

- G. Leduc, H. Abrahamsson, S. Balon, S. Bessler, M. D'Arienzo, O. Delcourt, J. Domingo-Pascual, S. Cerav-Erbas, I. Gojmerac, X. Masip, A. Pescaph, B. Quoitin, S.F. Romano, E. Salvatori, F. Skivée, H.T. Tran, S. Uhlig, and H. Hümit, *An open source traffic engineering toolbox*. In Computer Communications Journal (Elsevier), Volume 29, Issue 5, March 2006, p. 593-610.

- B. Quoitin, S.Tandel, S.Uhlig and O. Bonaventure, *Interdomain Traffic Engineering with Redistribution Communities*. In Computer Communications Journal (Elsevier), Volume 27, Issue 4, March 2004, p. 355-363.
- O. Bonaventure, P. Trimintzios, G. Pavlou, B. Quoitin (Eds.), A. Azcorra, M. Bagnulo, P. Flegkas, A. Garcia-Martinez, P. Georgatsos, L. Georgiadis, C. Jacquenet, L. Swinnen, S. Tandel and S. Uhlig, *Internet Traffic Engineering*. Chapter of Quality of Future Internet Services, COST263 final report, Springer-Verlag, LNCS2856, September 2003.
- S. Uhlig, O. Bonaventure and B. Quoitin, *Interdomain traffic engineering with minimal BGP configurations*. In Proceedings of the 18th International Teletraffic Congress, September 2003.
- B. Quoitin, S. Uhlig, C. Pelsser, L. Swinnen and O. Bonaventure, *Interdomain Traffic Engineering with BGP*, In IEEE Communications Magazine, Volume 41, Issue 5, May 2003, p. 122- 128, ISSN: 0163-6804.
- B. Quoitin, S. Uhlig and O. Bonaventure, *Using redistribution communities for Interdomain Traffic Engineering*, In Proceedings of QoFIS'02 Springer-Verlag, LNCS2511, October 2002.

# Acknowledgments

The fulfillment of this thesis would not have been possible without the contribution of a large number of persons. The very first persons I am deeply indebted to are my parents Marie-Christine Janne and Jacques Quoitin. They did not contributed to this thesis directly, but I would like to thank them for their support and love. I would also like to thank my wife, Valérie Baudoux, for her support. I am grateful to her for being patient when I was more obsessed by my work than by real life...

I owe a big thank-you to my thesis advisor, Olivier Bonaventure, who proposed me to work in his team a few years ago. I must concede that at that time, I was not really imagining what kind of experience I was running in. I appreciated very much Olivier's very pragmatic approach to networking. Olivier is also an inexhaustible source of networking references. Through his enthusiasm and unlimited support, he helped me to complete that thesis. Besides that, a large part of the ideas presented in this thesis belong to Olivier.

The second person I would like to thank is Steve Uhlig. Steve has an impressive overall view on the mathematics and computer science literature. I learned a lot from the references he pointed me to and from the many discussions we had. We collaborated on a lot of problems and my understanding of the interdomain routing system has been reshaped many times as we discussed. Steve provided the discussion on the characteristics of interdomain traffic in Section 4.3.2.

I am grateful to the external members of my thesis committee, Tim Griffin, Anja Feldmann and Guy Leduc, for accepting to read my dissertation and for providing interesting comments. I would especially like to thank Tim Griffin for his detailed comments and his recommendations regarding the description of Traffic Engineering. Let me also acknowledge the other members of the thesis committee for their comments: Bernard Fortz, Marc Lobelle and Yves Deville.

Let me also thank the other members of Olivier's team for their collaboration and their contribution to the pleasant ambiance that reigns in our office. Cristel Pelsser helped me a lot to improve my poor English by proof-reading some of my publications and the first chapter of this thesis. In addition to our collaboration on interdomain traffic engineering studies, Cristel has also been a valuable beta tester for the software tools I produced. She was not afraid of the bugs that remained in it. Cédric de Launois and I collaborated on characterizing the path diversity obtained with BGP and we had several discussions on the generation of realistic Internet topologies. Pierre François provided me some comments on Chapter 6

that I have finally not included (thank you anyway !). Sebastien Tandel has applied C-BGP to a large french tier-1 and hacked a little bit in the code. He helped me to identify a few bugs in C-BGP – well, not so few actually. Thank you also to Virginie Vandenschrieke who might give a second life to C-BGP by implementing a model of HLP [SCE$^+$05].

I would also like to acknowledge the work done in the TOTEM project [TOT05] by the Research Unit in Networking at University of Liège. They made an important effort to put the TOTEM toolbox framework on the tracks. It is a great pity that the research community considers ungratefully the effort to provide open-source implementations and reproducible experiments. My thanks go to Jean Lepropre, Fabian Skivée, Simon Balon, Gael Montfort, Olivier Delcourt and Guy Leduc for their work. Thanks also for their friendship during our collaboration.

The persons who provide data about their network also deserve a special attention. This thesis would probably not have been possible without them. In particular, I would like to thank the people at RIPE and University of Oregon for their public BGP data. My thanks also go to Nicolas Simar (Dante), Richard Gass (Intel Research) and Timothy Griffin (formerly with Intel Research, now with University of Cambridge) for the data they collected in the GEANT network. The Abilene observatory is another remarkable source of data. Jan Torreele from Belnet also provided data on BELNET and the Belgian Internet eXchange (BNIX).

The collaborations and contacts with the following persons were also both interesting and pleasant: Marcelo Yannuzzi from Universitat Politècnica de Catalunya, Stefano Iasi from Politecnico di Torino, Olaf Maennel and Wolfgang Müehlbauer from the Technical University of Munich. Olaf gave me a lot of valuable comments on earlier versions of Chapter 2. I have also discussed with or received interesting comments from Stefaan Decnodder (Alcatel), Sharad Agarwal (University of Berkeley, now with Microsoft Research), Marc de Berg (Technical University of Eindhoven) and Piet van Mieghem (TU Delft) during my research work.

I would finally like to take my hat off to all the persons who contribute to the development of open source software. I have used a large number of systems and tools that worked pretty well. It is almost impossible to name in this limited space all the people who were involved in the development and maintenance of these tools. I will rather cite some of the systems and tools I used: the Linux kernel, gcc, make, gnuplot, emacs, vi, perl, latex, ispell, ghostview, R, gnumeric, openoffice, firefox, thunderbird, dia, ...

Bruno Quoitin
August 4, 2006

# Contents

# List of Figures

# List of Tables

# Part I

# Background

# Chapter 1

# Internet, Routing and Traffic Engineering

## 1.1 Introduction

Initially developed as a network that connects a small number of research networks, the Internet has become a world-wide data network that is used for mission critical applications such as Voice over IP (VoIP) or Virtual Private Networks (VPNs). Supporting such applications across the global Internet implies several important challenges. The first challenge is the **size of the Internet**. The Internet is a large decentralized network that already connected about 350 million hosts in June 2005 [ISC05]. Furthermore, these hosts are organized in about 21,000 distinct domains [Hus06], a domain corresponding roughly to a company, an Internet Service Provider (ISP) or a campus network. All these domains are interconnected to form the global Internet. Over this large interconnection of networks, ISPs run two different families of routing protocols. Intradomain routing protocols are used within the ISP while an interdomain routing protocol is used across the ISP boundaries.

The second challenge is the evolution of the Internet in terms of **quality of service requirements**. The initial research Internet was designed with a best-effort service in mind where connectivity was the most important issue. Today, connectivity is considered to be granted but the architecture initially designed to provide a best-effort service is used for more demanding applications, and sometimes with Service Level Agreements (SLAs) [FE04]. To meet the requirements of these applications and/or to ensure the Quality of Service (QoS) required by SLAs, several ISPs rely on a process called Traffic Engineering (TE) [ACE$^+$02]. Traffic Engineering covers the evaluation and the improvement of the performance of operational IP networks. However, if performing Traffic Engineering inside a single AS is a well understood problem, it is far more difficult when performed across the boundaries of multiple ASs. The main limitation of interdomain Traffic Engineering comes from the current Internet routing architecture.

This chapter is organized as follows. We first give an overview of the Internet architecture in Section 1.2. Secondly, we introduce how routing is done in the Internet in Section 1.3. In particular, we detail in Section 1.3.2, the operation of BGP, the de facto standard interdomain routing protocol. Then, we describe in Section 1.4 the Traffic Engineering process and what currently limits its performance when performed at the interdomain level. Finally, we give in Section 1.5 an overview of the practice of multi-homing and the problems faced by multi-homed ASs. We conclude in Section 1.6.

## 1.2 Internet architecture

The Internet is a network composed of a huge collection of smaller networks, themselves containing a myriad of end systems and routers. The end systems are hosts such as personal computers or servers. They are usually the sources or sinks of data packets transiting on a network. The routers are the intermediate systems that intervene in the transport of data from an end system to another. Since the many networks that form the Internet are operated by a lot of independent institutions, the Internet is organized in **two levels**.

The first level is the **intradomain** level. A set of routers that is under a single administrative authority form a domain. A domain can be the network of a company, an Internet Service Provider (ISP) or a single campus network[1]. An example ISP is represented in Fig. 1.1. The routers of a domain are usually interconnected using multiple Synchronous Optical Networking links (SONET/SDH) and/or Ethernet. We distinguish the *core links* that interconnect the routers within the domain and the *edge links* that cross the domain boundaries. Since the edge links connect to routers lying outside of its network, a domain only manages one side of the edge links.

Through the edge links, the domain is connected to different kinds of neighbor networks. On one side, the *access links* mainly connect to customer networks. For example, an access link could connect to a DSLAM[2] for DSL users or a university or corporate campus network. On the other side, the *peering links* connect to other domains. For example, peering links could connect to neighboring ISPs. The routers where edge links are terminated are called the domain's *border routers*. The different geographical locations of border routers and access routers are usually called the Points of Presence (PoPs) of the domain.

The second level of the Internet is the **interdomain** level. It designates the interconnections between the different domains. In the Internet, a domain is also called an Autonomous System (AS). Most ASes are uniquely identified by an Autonomous System Number (ASN). Note that all domains need not to have a public ASN. This is usually the case for small to medium size university or corporate campus networks that buy connectivity from a single ISP. We show in Fig. 1.2 the

[1]A campus network is an interconnection of Local Area Networks (LAN)

[2]Digital Subscriber Line Access Multiplexer

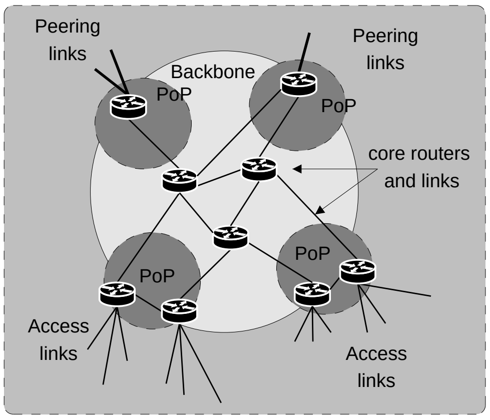


Figure 1.1: Topology of an ISP network.

sketch of a small imaginary Internet[3] composed of 8 different AS domains: *Carrier&Wireless*, *Level3*, *Belnet*, *Janet*, *Geant*, *Google*, *ISPx* and *ISPy*. In addition, there are 4 customer networks that do not have their own AS: *UCL.be* and *UCL.uk* are campus networks of universities while *apple.com* and *m$.com* are corporate networks.

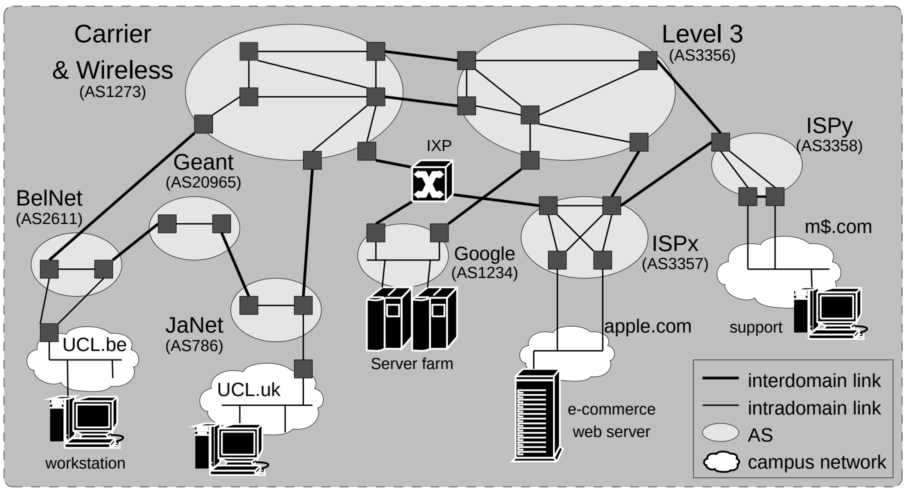


Figure 1.2: Sketch of the Internet architecture.

In the example Internet of Fig. 1.2, not all domains play an equal role. They can first be distinguished based on their connectivity. In [Hus99, Gao00], Huston and

[3]The organization of this example Internet is imaginary even if there are similarities with real domain names.

Gao have shown that there are two major types of interconnections between distinct domains: the **customer-provider** and the **peer-to-peer** relationships. In the customer-provider relationship, a customer domain purchases connectivity from a larger domain, called the provider. In this case, the provider agrees to forward the packets received from the customer to any destination. It also agrees to forward the packets destined to the customer. In Fig. 1.2, ISPx and ISPy are examples of customer ASs that buy connectivity from Level3.

On the other hand, the peer-to-peer relationship is used between domains that agree to share the cost of a private peering link. This private peering link is only used to exchange traffic between the peers and their own customers. No transit traffic will flow through the private peering links. Usually, private peerings are established at public Internet eXchange Points (IXPs). An IXP is a collocation crafted with networking equipment where ASs that participate can connect to each other. An example of such situation is the connection between Google and ISPx in Fig. 1.2. Negotiating the establishment of those *peer-to-peer* relationships is often a complicated process since technical and economical factors need to be taken into account, as exposed in [Bar00].

According to a study performed by Subramanian et al in 2002 [SARK02], the customer-provider relationship was used for about 95 % of the domains interconnections in the Internet. The classification of interdomain relationships in customer-provider and peer-to-peer leads to an interesting view of the Internet as a graph where money is ascending along customer-provider links (Fig. 1.3) [Hus99].

Relying on this classification of interdomain relationships, Subramanian et al [SARK02] made a first characterization of domains. There are basically two types of domain: **transit domains** and **stub domains**. Transit domains constitute the core of the Internet and their purpose is mainly to carry packets from a neighbor domain to another. In the example of Fig. 1.2, Carrier&Wireless and Level3 are example of large transit ASs. According to [SARK02], the core corresponds to about 15 % of the domains in the Internet and can be divided in three different subtypes (*dense*, *transit* and *outer core* depending on the connectivity of each domain). On the other hand, stub domains are regional ISPs or customer networks that do not provide transit. Stub domains correspond to 85 % of the Internet and they maintain only a few customer-provider relationships with domains in the core and some peer-to-peer relationships with other small domains. In the example of Fig. 1.2, BelNet, JaNet, Google, ISPx and ISPy are stub ASs.

In addition, domains can also be distinguished based on the type of service they provide to their customers. This is interesting mostly for stub domains. For instance, a stub domain can be a small regional ISP providing Internet access to Small/Medium Enterprises (SME) and/or dialup/xDSL/CaTV users. In this case, it will often receive more traffic than it sends. We call this kind of domain a **content-consumer**. In Fig. 1.2, BelNet and JaNet are examples of such domains since they only provide Internet connectivity to universities campus networks. In contrast, a stub domain that hosts video streaming servers or the web servers of a large company will often have more outgoing traffic than incoming traffic. This kind of

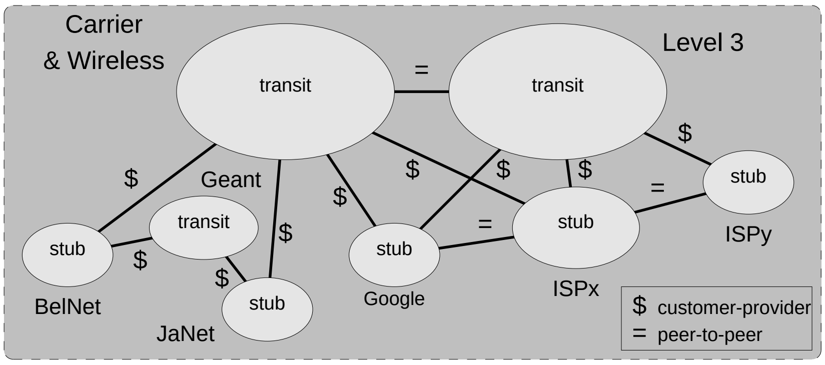


Figure 1.3: Internet business relationships.

domain is called a **content-provider**. An example of such domain in Fig. 1.2 is Google who hosts a farm of servers containing a lot of information accessed from everywhere in the Internet.

## 1.3 Routing in the Internet

To be uniquely identified in the Internet, each end system and router receives one or more Internet Protocol (IP) addresses. In the current version of the IP protocol (IPv4), an **IP address** is a 32-bits integer number. It is usually represented in the dotted format $A.B.C.D$. An example of IP address is $66.249.93.99$. Each AS in the Internet is often being allocated blocks of contiguous IP addresses that they can use for their own network or delegate to their customers. Throughout this thesis, we will refer to such a block as a **network prefix**. A network prefix represents the set of IP addresses that start with the same first bits. For example, the IP address $66.249.93.99$ belongs to the network prefix $66.249.64.0/19$ since its 19 most significant bits are equal to those of the prefix. In this case, we say that the IP address $66.249.93.99$ matches the prefix $66.249.64.0/19$. Network prefixes are also sometimes referred to as subnets.

The physical topology of the Internet defines the feasible paths that can be used to cross the network. The role of routing consists in determining for a given Internet device the path to be used to reach a destination IP address. In order to determine these paths, all the routers in the Internet usually exchange information about the network topology. These exchanges are supported by a **routing protocol**. In the Internet, routing is handled by two distinct protocols with different objectives. An intradomain routing protocol is used inside each domain and a single interdomain routing protocol is used between domains.

There are three main reasons for this schism. The first one is the need for scalability. An intradomain routing protocol usually has a very detailed knowledge of the whole domain topology. It handles routes towards any destination within the domain. To the contrary, an interdomain routing protocol has a limited view of the Internet topology, restricted to the interconnection between domains. An interdomain routing protocol also handles routes towards large aggregates of IP addresses. This avoids having to handle routes towards any destination. The second reason for having two distinct routing protocols is the Independence of domains. Each domain is allowed to setup its intradomain routing in an independent manner. Each domain is also allowed to perform policy routing. For example, a domain can refuse to serve as a transit domain for another domain.

### 1.3.1 Intradomain routing

Inside its network, an AS runs an Interior Gateway Protocol (IGP) such as OSPF [Moy98] or IS-IS [Ora90] in order to compute the interior paths from any AS's router towards the AS's other routers and prefixes. The IGP is typically a **link-state protocol**, that is it floods information about the state of the adjacencies between all routers in the whole AS. The objective of the intradomain routing is to find the shortest paths according to a selected metric assigned by the network administrator. ISPs usually use a metric that is proportional to the propagation delay along the path or to the bandwidth. Many network operators use the Cisco default metric, which is one over the bandwidth [HP00]. Some large ASs use a hierarchical IGP, where the AS is divided into different areas. Inside an area, all the adjacency information is flooded. Between areas, only aggregated information is exchanged.

In addition to the IGP, an AS sometimes uses static routing. Static routes are often used on the edge links since routers on both side of these links are not operated by the same authority. Static routes are also used to setup access to small customers that do not have their own AS.

### 1.3.2 Interdomain routing

In order to learn routes towards destination located outside their own domain, the routers run the Border Gateway Protocol (BGP) [RL04, Ste99, HP00]. BGP is the de facto standard routing protocol for the selection of the interdomain paths. The rationale behind the design of BGP was to provide **reachability** among domains and the ability for any domain to enforce its own **routing policies**, i.e. controlling what traffic enters and leaves the domain, and where. To the contrary of the intradomain routing protocol, BGP does not optimize a single global metric but relies on a *decision process* composed of a sequence of rules.

BGP is a **path-vector protocol** that works by sending route advertisements. BGP routers exchange routing information by means of BGP sessions. Each BGP session is established between a pair of routers over a TCP connection. External BGP (eBGP) sessions are established over the edge links while internal BGP (iBGP) sessions are established between the routers of the AS. There is a full-mesh (a clique) of iBGP sessions between the routers of the AS. We show in Fig. 1.4 the BGP sessions running over the topology of Fig. 1.2. In some ASs, the number of iBGP sessions can be quite large. Indeed, there is on the order of $n^2$ iBGP sessions in an AS of $n$ routers. For this reason, the ASs sometimes deploy route-reflectors [BCC00] in their network. Route-reflectors are special BGP routers that make possible an hierarchy of iBGP sessions, therefore reducing the number of iBGP sessions. It is also possible to reduce the number of iBGP sessions by using BGP Confederations [HP00].

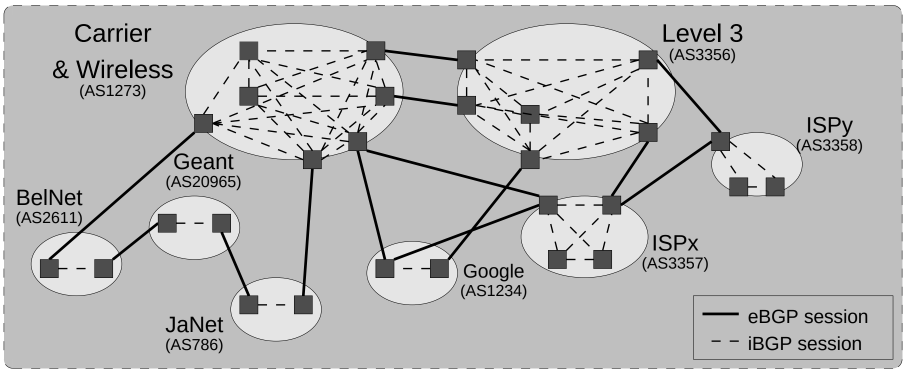


Figure 1.4: Internal and external BGP sessions.

A route advertisement indicates the reachability of a network. A route advertisement contains the prefix of the destination network as well as the complete interdomain path that the route follows. The interdomain path is the list of all the ASs that must be crossed in order to reach the AS of the destination. This list is called the **AS-Path** of the route. The AS-Path is used to avoid interdomain level routing loops[4]. In addition to the AS-Path, a route contains a **next-hop** attribute. The next-hop of the route is the IP address of the router to which packets must be sent in order to reach the destination network. The route also contains several additional attributes.

A router sends a route advertisement for a network if this network belongs to the same AS as the advertising router or if this network is reachable from the router through a neighboring AS. An important point to note about BGP is that if a BGP router $A$ in $ASx$ sends to a BGP router $B$ in $ASy$ a route advertisement for a network $N$, this implies that $ASx$ accepts to forward the IP packets to destination $N$ on behalf of $ASy$.

To better understand the operation of BGP, it is useful to consider a simplified view of a BGP router as shown in Fig. 1.5. The router is composed of 4 main components. First, route **input and output filters** can be configured for each BGP session. The role of a route filter is to deny the routes received or sent by the router or to manipulate their attributes. An example filter would be to only accept the routes with an AS-Path containing a set of trusted ASs. The route filters are

[4]When it receives a route, a router checks that the AS-Path of the route does not contain its own ASN. If this is the case, the route is dropped

configured by the network operator. The second component of a BGP router is the **BGP routing table**. This routing table contains all the routes received by the router and accepted by the input filters. The attributes of the routes stored in the routing table may have been updated by the input filters. The third component of a BGP router is its **decision process**. It is responsible for selecting among the routes stored in the routing table a single best route for each destination prefix. When a route is selected as best, it is installed in the **forwarding table** and it is sent to the neighboring routers. The forwarding table is the fourth component of the router. Each time a packet is received, this table is looked up and it indicates the outgoing interface that must be used to forward the packet to the destination.

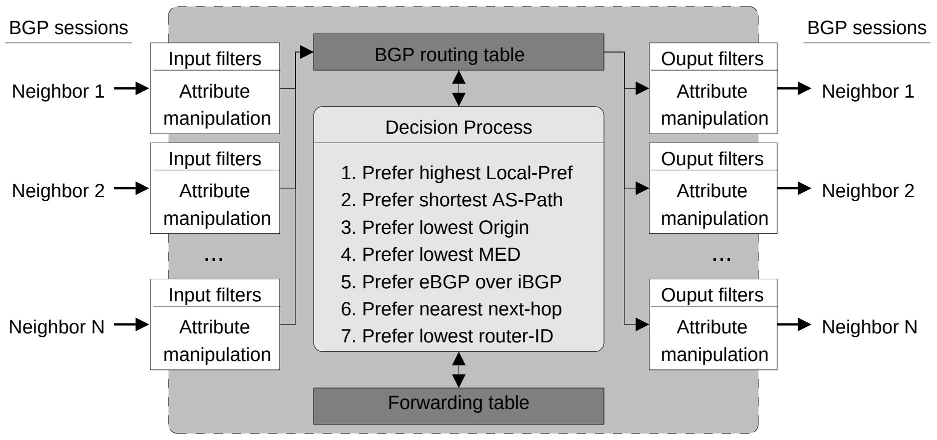


Figure 1.5: Sketch of a BGP router.

Through its BGP sessions, each router receives BGP routes towards destination prefixes. Since there might be multiple routes towards the same destination prefix, a choice must be made. Each router uses its **decision process** on a per-prefix basis to select the routes it will use. The BGP decision process is a sequence of rules applied to a set of routes towards the same destination prefix to **select a single route** called the *best route* towards this prefix. Basically, the BGP decision process ranks the routes according to their attributes. Each rule of the decision process keeps the routes that it prefers. The surviving routes are then submitted to the next rule, until a single route remains. A summary of the BGP decision process is shown in Fig. 1.5.

The BGP decision process considers several of the BGP route's attributes. The first attribute is the Local-Pref which corresponds to a local ranking of the route. It is usually attached to the route upon reception by a border router and it is never propagated outside the AS. The decision process prefers the routes having the **highest value of the Local-Pref** attribute. The second attribute is the AS-Path. The AS-Path contains the sequence of ASs that the route crossed to reach the local AS. In the decision process, the AS-Path is used as a distance metric in AS hops. The decision process prefers the routes with the **shortest AS-Path**. The third attribute is the Multi-Exit-Discriminator (in short, the MED). This attribute is used to rank routes received from the same neighboring AS. Usually, the MED attribute is set by the neighbor AS to indicate the preferred peering link to use (based on the IGP cost in the neighboring AS for instance). The decision process prefers the routes with the **smallest value of the MED**.

If there are still more than a single route at this step, the decision process will consider the BGP next-hop attribute of the route. The BGP next-hop is often called the *egress* of the route, i.e. the exit point of the AS. Note that the BGP next-hop may be different from the immediate IP next-hop. When a BGP router receives a route, it first checks that the next-hop is reachable before considering it in the decision process. The decision process uses the IGP cost of the intradomain path towards the next-hop to rank the routes. It prefers the routes with the **smallest IGP distance to the next-hop**. This rule implements hot-potato routing [TSGR04]. Its aim is to hand over packets to a neighboring AS as quickly as possible in order to consume as few network resources as possible in the local AS. In addition, it automatically adapts routing to topology changes that affect the IGP distance to the egress points inside the AS. This step within the BGP decision process is where the IGP and BGP protocols interact.

Finally, if there are multiple routes remaining, the decision process will break the ties by preferring the route announced by the neighbor router that has the lowest router-ID. The router-ID is the highest IP address of the router. Another tie-breaking rule that is sometimes deployed in BGP routers consists in preferring the older route [CS05].

## 1.4 Traffic Engineering

In order to evaluate and enhance the performance of their network, operators rely on Traffic Engineering (TE) [ACE$^+$02]. This section first briefly describes the concepts of Traffic Engineering. Then the section discusses why performing Traffic Engineering across the boundaries of multiple domains is so difficult.

To understand Traffic Engineering, it is interesting to have a look at the lifecycle of an operational IP network (see Fig. 1.6). Typically, an operational IP network has been designed and deployed at some point in order to satisfy predefined network communications objectives. For instance, the network must interconnect a given number of end-sites and the traffic demand between these sites has been forecasted. Through **network design** and **capacity planning** [Cah98, Gro04], the initial network infrastructure has been laid out. That means that a certain amount of networking devices and links have been deployed. This infrastructure has a limited capacity which is often higher than the forecasted traffic demand. It is a common practice to **over-provision** the network by a factor of 50 or 100% ([Tel02]) so as to accomodate future traffic demand growth.

Anyway, during the network lifecycle, the traffic demand will eventually grow

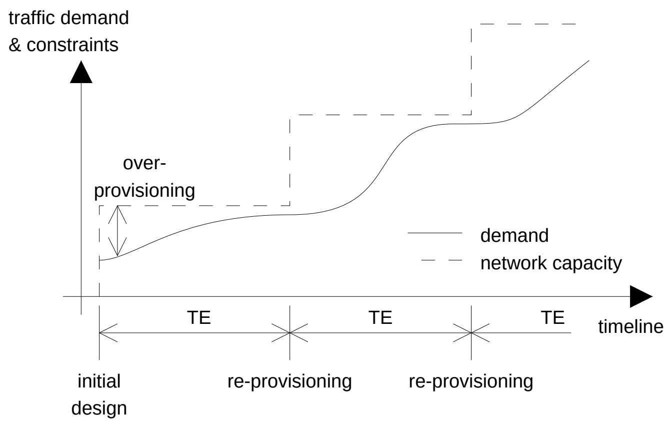


Figure 1.6: Network lifecycle.

to a point where the maximal capacity of the network is almost reached. This increases the risk of congestion and prevents any traffic growth. In this case, the network engineers need to **re-provision** the network. This is done by upgrading the networking equipment, by changing the underlying technologies and/or by deploying new links.

Traffic Engineering typically comes into play to bridge the gap between two re-provisioning phases. It is therefore an every day network engineering process that aims at evaluating and enhancing operational IP networks performance [ACE$^+$02]. The main objectives of traffic engineering can be summarized as (1) shifting traffic away from congested links, (2) better spreading the traffic load over the network resources in order to increase the amount of traffic that can be carried by the network,(3) quickly reacting to failures or errors by directing traffic away from the faulty networking resources and (4) efficiently supporting Quality of Service (QoS) requirements. The goal of the two first objectives is to avoid to re-provision the network every time the traffic demand changes. The third objective aims at providing more resilient IP networks. Finally, the fourth objective is to meet the requirements of the Service Level Agreements (SLA) contracted with the customers.

The Traffic Engineering process can be seen as a **closed-loop iterative optimization process** which takes as input the current operational state of the network, including the traffic matrix. The operational state of the network is continually determined by measuring the network performance. The typical output of the traffic engineering process is an adjustement of the network parameters, i.e. an on-line re-configuration of network equipment. The Traffic Engineering process can also be performed *off-line*. In this case, its output is a proposal of parameters adjustments and it is up to the network engineers to implement these adjustments in the network.

Traffic Engineering in general is a difficult process. The main reason for this is the manner in which datagrams are forwarded in IP networks. Datagrams are forwarded hop-by-hop based on the destination address only. There is no means in pure IP networks to explicitly determine the entire path the datagrams traverse. In addition, routing in IP networks is performed in a distributed manner. Inside a single domain, routing is performed using a link-state protocol. This means that all the routers in the domain share a unique view of the topology[5]. Moreover, link-state protocols such as OSPF or IS-IS optimize a single global objective (least-cost-path). From the Traffic Engineering perspective, knowing the whole topology and dealing with a single optimization objective is a neat advantage. For this reason, the problem of intradomain Traffic Engineering can be considered as well understood. Techniques for performing Traffic Engineering inside a single IP network have been widely discussed in the networking litterature (see [FT00, FRT02] and the references therein for example).

Contrasting with that, performing Traffic Engineering accross the boundaries of a single network is a much more difficult task. Most of the difficulties of interdomain Traffic Engineering are due to the current Internet infrastructure. A first issue comes from the utilization of a path-vector protocol. To the opposite of link-state protocols, path-vector protocols do not propagate the complete Internet topology across domain boundaries. BGP routers only redistribute a single route towards a destination. The **lack of visibility** on the whole topology reduces the number of alternative paths that can be exploited to reach a remote destination [dLQB06]. A second issue comes from the administrative independance of Internet domains. Each domain configures its routing to optimize **local objectives**. For this reason, each domain is allowed to enforce local routing policies. The routing policies of one domain may not be compatible with those of other domains. In order to perform interdomain Traffic Engineering it is often required to influence the routing choices performed in distant domains, but these domains will often not allow such control.

To perform Interdomain Traffic Engineering, there is an important need for the control of Internet routing. However, BGP has been designed to provide reachability, not to allow to control routing decisions. A handful of BGP-based techniques are currently used by IP network operators to control their interdomain routing [QUP$^+$03]. These techniques rely on tweaking the attributes of the BGP routes they receive and send. However, the performance of these techniques is still limited and they need to be applied in a trial and error manner [ACE$^+$02]. We show an evaluation of these techniques in Chapter 4.

## 1.5 Multi-homing

Today, a common method used by stub domains to engineer their interdomain traffic consists in buying their Internet connectivity from at least two providers [ACK03]. This increasingly used practice is called multi-homing. According

[5]This is not the case when multiple areas are configured in a single domain.

to Subramanian et al [SARK02], a large fraction of the stub domains are multi-homed. Among the 16,921 different domains seen in their analysis, 13,872 (82%) were stub domains. Among these stub domains, 8453 (61%) have at least 2 different providers. We show in Fig. 1.7 a breakdown of the stub domains in function of the number of their providers. We observe that the majority of multi-homed stubs are dual-homed. Stub domains that have more than 2 providers are also frequent.

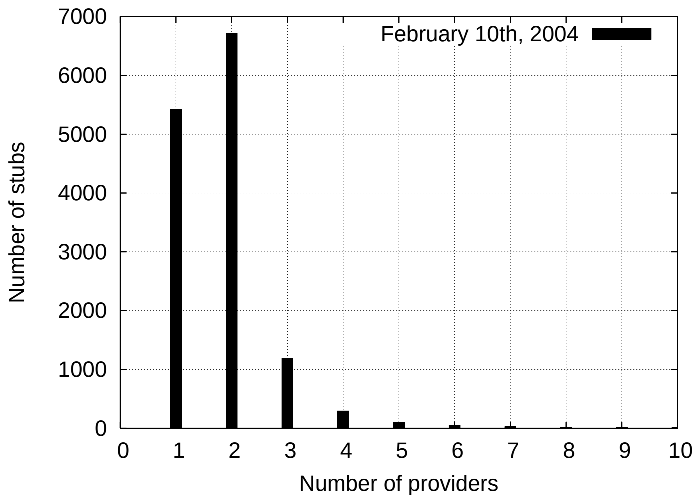


Figure 1.7: Breakdown of stub ASs by the number of their providers.

The reasons for stub AS to become multi-homed are various [ALD+05]. The first reason is to improve the **robustness** of their Internet access. In this case, a second (backup) access to the Internet is bought from an additional provider. This link is used in case of failure of the primary link. The second reason to become multi-homed is to improve the Internet access **performance**. Increasing the number of access links potentially increases the access bandwidth. In addition, an improvement of the access performance is also possible by buying connectivity from a provider that is closer to the domains with which critical services are running or with which a lot of traffic is exchanged. A third reason for ASes to become multi-homed is for **business** reasons. Buying connectivity from multiple providers introduces competitivity among them. Finally, some stub ASes have **no choice** but to connect through two providers in order to get full Internet connectivity.

Unfortunately, BGP has not been designed to efficiently accommodate such configurations. As a consequence, when a stub AS has become multi-homed, it has often to face new issues. The first problem is the **imbalance of the inter-domain traffic** on the different access providers. It is frequent for a dual-homed stub domain to exchange a large fraction of its traffic over one access link and a very small fraction on the other link. This imbalance is due to the routing choices made by BGP both in the local and remote routers. Another problem faced by multi-homed stub domains is the management of the **cost of their Internet connectivity**. A common way to bill for the Internet traffic exchanged over an access links is to rely on the maximum volume of traffic exchanged in either direction. This can be unfavorable if for instance a stub domain sends the majority of its traffic through one link and receives the majority of its traffic through the other link. Finally, a stub domain might want to control how its traffic enters and leaves its network for **policy reasons**. When a stub domain is connected to multiple providers offering **different qualities of services**, it has to select which one is best suited to reach a particular destination or for a particular service level. For example, it might want to send premium traffic through an high quality provider and best-effort traffic through a lower quality provider.

To better control the flow of their interdomain traffic, ISP often rely on BGP-based traffic engineering. However, BGP has not been designed with traffic engineering in mind. Today, the only solution is to tune the configuration of the BGP routing protocol. This tuning is often done on a trial-and-error basis and suffers from limitations.

## 1.6 Conclusion

In this chapter, we presented the context of the thesis. We first described the organization of the Internet in two levels. We explained that the Internet is composed of a large interconnection of independent domains. These domains do not play an equal role in the Internet. Some of them provide a transit service while others don't. The former are the transit domains and they represent approximately 15% of the Internet domains. The large majority of the Internet is thus composed of stub domains that do not provide transit.

Then, we explained that routing in each level of the Internet is performed by a different routing protocol. Inside a single domain, all the routers usually know the whole topology. In contrast, BGP, the routing protocol used across the domain boundaries, only carries a **limited view of the Internet topology**. This view can be further limited by the routing policies enforced by each domain. In addition, the route selection performed by BGP does not rely on the optimization of a single metric as in intradomain routing protocols. To the contrary, BGP relies on a complex decision process composed of several rules. This decision process is often configured to optimize local objectives.

The organization of the current Internet infrastructure in two levels has an impact on the efficiency of interdomain Traffic Engineering. The limited view of the Internet topology provided by BGP reduces the availability of diverse interdomain paths and subsequently the freedom of using alternative paths for Traffic Engineering purposes. Moreover, BGP lacks the routing control mechanisms that are needed to perform efficient interdomain Traffic Engineering. A typical example of the lack of routing control provided by BGP is exhibited in the case of multi-homed stubs. These stubs buy connectivity from multiple providers to improve the robustness and the performance of their Internet access. However, they face difficulties to fully exploit their improved connectivity.

This thesis tackles the problem of BGP-based routing control as a means to perform interdomain Traffic Engineering. Our first step is to better understand the large-scale selection of interdomain routes by BGP. For this purpose, we build a modeling tool for BGP routing. This tool and its applications are described in Part II. In Part III, we apply this modeling tool to the evaluation of current BGP-based routing control mechanisms.

# Part II

# Modeling BGP Routing

# Chapter 2

# A Routing Solver for Large Scale Topologies

## 2.1 Introduction

In this chapter, we describe the model and the tool that we use throughout the whole thesis to simulate the interdomain routing system. Such a tool is needed to better understand the current Internet infrastructure and its routing protocol, BGP. On one side, understanding BGP is required for network operators that want to better manage their network and prepare it to support new Internet-based services. On the other side, a good understanding of BGP is also required for researchers that want to characterize the behavior of BGP in the global Internet. The motivation behind the development of this tool is to make possible “playing” with BGP for the purpose of evaluating the existing Internet infrastructure as well as to design the future traffic engineering mechanisms. The tool focuses on the selection of routes by BGP, not on a model of the convergence of BGP.

Reproducing the behavior of BGP is a challenging problem. In this chapter, we will explain why this is a difficult task. The Internet is composed of an interconnection of hundreds of thousands of routers running BGP. Since the real Internet now supports critical applications, it is not possible to run large experiments or to deploy modified versions of BGP on the production routers. Due to the number of routers in the Internet, it is also not possible to reproduce the behavior of such a large network on genuine routers in a lab. A natural first step is thus to turn to simulations [FP01]. However, building an efficient simulation model of BGP is a difficult task since it poses serious scalability constraints. In this chapter, we describe the hypothesis we take and their impact on the efficiency of the simulator.

The chapter is structured as follows. We first survey the state-of-the-art in modeling BGP in Section 2.2. We describe the different approaches and tools available today and we indicate why they are not suitable to perform large-scale simulations of BGP. Then, in Section 2.3, we define the requirements that a BGP model must satisfy in order to be successfully used for Internet-wide simulations.

In Section 2.4, we present C-BGP, a BGP routing solver and the hypothesis that allow us to build a suitable BGP model. In the same section, we describe the main components of C-BGP and its implementation. We define the BGP routing model implemented in C-BGP in Section 2.5. We subsequently validate the tool, discuss its convergence properties and evaluate its performance. We conclude in Section 2.6.

## 2.2 Modeling BGP

Reproducing the behavior of BGP in a large topology with thousands of routers poses serious scalability constraints. Indeed, the BGP decision process is complex by nature because of its rules which define different sometimes contradictory orderings on the routes. Another difficulty comes from the distributed nature of BGP. The BGP decisions are taken in a distributed manner by the BGP routers, without explicit concertation. Though, a single local decision can affect the information available to all the other routers. There is therefore no easy shortcut in simulating BGP as it is the case for a link state protocol like OSPF where an enhanced Dijkstra algorithm can be used as a satisfactory model. The most efficient and straightforward method to simulate BGP is to build a realistic implementation of the decision and filtering processes and simulate the propagation of messages.

### 2.2.1 Daemons

A first natural way to experiment with BGP is to run open-source BGP daemons in a testbed. There are several open-source implementations of BGP daemons available today. Zebra [Ish96] is probably the most famous one and also the most mature implementation. Zebra supports a lot of different intra- and interdomain routing protocols: RIP, OSPF and BGP. Quagga [Qua03] is a recent fork of Zebra that seems to be more actively maintained. Finally, BIRD [FMMF05] and OpenBGPD [BJ05] are younger yet promising implementations of routing daemons supporting BGP. BIRD runs under Linux while OpenBGPD has been developed for the OpenBSD platform.

It is possible to run several instances of these BGP daemons on a single workstation, using emulation environments such as User Mode Linux [Dik], VNUML [GF04, FdMG04] or Netkit [BPP$^+$]. Another approach implemented by [MPDZ, Zec03] consists in virtualizing the networking stack instead of running different instances of OS kernels.

Using such daemons provides accurate models of BGP. There are however two important limitations to this approach. First, these routing daemons require a huge amount of computational resources if used to study large networks. The reason is that they keep too much state about the network. For instance, routing tables are maintained in each router instance. Another issue of this approach is that the modeling is done in real-time. Indeed, the timers used in the BGP daemons will run as if they were used in a production network.

### 2.2.2 Packet-level simulators

Another approach to studying the BGP protocol is to turn to simulation. Network protocol simulation is often done with packet-level simulators. Such simulators can be used to model a large set of different network protocols such as TCP, routing protocols or multicast protocols over different types of communication media. The modeling is done at a fairly detailed level since they usually model the packets on the physical transmission lines of the network. The most famous network simulators are ns-2 [MRG], SSFNet [CNO99] and J-Sim [Tya02].

The heart of packet-level simulators is a discrete-event scheduler which models the time-line of events that occur in the simulated network. Examples of such events are the reception of a packet by a router or the expiration of a timer used in a protocol such as TCP. The advantage of discrete-event simulators is that they can skip portions of time where there is no event, going straight to the next event to process. Discrete-event schedulers maintain the simulator events in a priority queue. The priority usually represents the time when the event will occur. Priority queues used to require an high computational complexity for the insertion of elements, usually on the order of log(n) [Knu00]. Recent discrete-event schedulers use calendar queues in order to gain in efficiency. Calendar queues follow the same principle as the human desk calendars. They group small sets of events into a single page usually corresponding to a day or a week. The same page (day) can be used for multiple years without causing confusion. The events are labeled with the year they correspond to. The time to insert a new event is composed of the time to find the page and the time to insert into the page. The time to find the page is constant and depends on the number of days per year. The page insertion time depends on the number of events already in the page. The idea behind calendar queues is to keep the number of events per page small in order to keep the insertion time low. Modern calendar queues are able to adjust this parameter automatically [Bro88].

Packet-level simulators are usually packaged with a handful of network models. The TCP protocol for instance is supported in all of them. The main advantage of using a model of BGP in a packet-level simulator such as SSFNet, J-Sim or ns-2 is the possibility to use the other protocol models packaged with the simulator. It is therefore possible to simulate applications using interdomain paths computed by the BGP model. Support for the BGP protocol is more recent and still highly experimental in some simulators.

The first BGP model available in a packet-level simulator, and also the most popular, came with SSFNet [Pre01, Pre02]. This model of BGP was developed from scratch and is fairly accurate. It has been used to perform a large number of published BGP experimentations [GP01, MGVK02, PAMZ04]. SSFNet's BGP model relies on the model of the TCP protocol coming with SSFNet to implement the BGP sessions. It models the BGP Finite State Machine and the various BGP protocol timers such as the Minimum Route Advertisement Interval (MRAI). There are however some limitations in this BGP model. First, the BGP decision process does not take into account the IGP cost to reach the next-hops when it compares two routes. This prevents performing simulations of the interaction between BGP and an IGP. Second, the length of the Cluster-ID-List is not taken into account in the decision process. Realistic simulations with complex iBGP hierarchies are therefore not possible. Third, the BGP policies that are supported by SSFNet are quite limited. Among those limitations are the inability to set the Communities attribute (which is useful for traffic engineering purposes [BQ03]) and the inability to combine predicates with another boolean operator than AND.

The J-Sim support for BGP [Quo03a] is an adaptation of the SSFNet implementation. It has an improved support for BGP policies allowing the manipulation of the Communities attribute. In addition, its decision process was modified in order to take into account during the decision process the IGP cost to reach the BGP next-hops of routes. This implementation has been used to perform large-scale studies of traffic engineering [BTP$^+$03] and QoS routing [YFMB$^+$04].

Developed in parallel to our own work, the BGP support provided with ns-2, called BGP++ [DR04] is more original since it relies on the adaptation of Zebra [Ish96]. BGP++ is probably the most detailed implementation of BGP in a packet-level simulator. It allows to build complex routing scenarios including route-reflectors or confederations. This simulator has not yet been widely used. Another recent BGP implementation for SSFNet has been presented by Hao and Koppol in [HK03]. According to the paper describing this simulator, it tackles the scalability problem of BGP simulation. Nevertheless, the simulator is not publicly available and it is incompletely described.

The main issue with models of BGP in packet-level simulators is the memory consumption. When used to study large network topologies, they require a huge amount of memory, mainly to store the BGP routing tables. They currently lack techniques to efficiently store routing tables and to aggregate the redundant routes. For example, in SSFNet and J-Sim models, the BGP routing tables (Loc-RIB, Adj-RIB-in and Adj-RIB-out) are based on non-compressed tries (radix-trees) while more efficient techniques such as Patricia trees [Mor68] or level-compressed tries [NK99] have been described in the literature [Var05]. On the side of ns-2, BGP++ inherits the efficient route storage capabilities from Zebra. However, it does not take advantage of routing information that could be shared between the router instances. This would lead to reduced memory consumption[1].

A second issue with packet-level simulators is that they are too detailed for our purpose. A packet-level simulator will usually reproduce the finite state machines of all the protocols. In the case of a BGP simulation for example, many different protocols are involved. For example, multiple TCP connections support the BGP sessions. TCP stacks must therefore be modeled which includes all the TCP timers.

[1]Scalability improvements have been brought recently to the BGP++ implementation [DR06]. It was however not available at the time we performed our evaluations of BGP-based traffic engineering.

A model of an IGP might also be needed to compute the routes between the BGP routers. In a packet-level simulator, such a model will reproduce the flooding of LSPs in each IGP domain. Such a level of details allow to study the dynamics of routing protocols, but it does not scale when only the outcome of the BGP decision process is required. Too much details add burden on the simulations completion time.

### 2.2.3 Emulation

In [FWR04], Feamster et al have proposed another approach: a BGP emulator that computes the outcome of the BGP route selection process for each router in a single AS. This tool was developed for the purpose of evaluating the traffic engineering actions performed in a single AS. The aim of the BGP emulator is to compute the routes selected by each router in the AS given only the configurations of the routers as well as the external routes learned by the AS while considering each AS's available routes only once. In order to do this, the tool does not model the flow of BGP routes inside the AS. Hence it does not reproduce the route filtering process occurring within an AS.

In addition, this BGP emulator [FWR04] is targeted at studying a single AS. Its main purpose is to serve as an inner-loop of a tool that would help ISP operators to understand how BGP works in their domain, by playing with the routers configurations and observing how the outcome of the BGP decision process would be affected. It is therefore also stuck to studying what-if scenarios that will affect the outbound traffic. Indeed, to take the inbound traffic into account, the model must be enriched with at least the neighbors of the domain under study, if not the whole Internet.

Finally, due to the assumptions taken by the model proposed in [FWR04], it has been designed to study the current BGP protocol. Taking into account new extensions to the BGP protocol that would affect the route selection or the filtering processes would need to re-design the inner algorithm.

## 2.3 Modeling requirements

From the previous section, we can conclude that today, there is no model of BGP that together contains the complete decision process, versatile route filtering features, and that can efficiently support large-scale topologies. In addition, some models are limited to the simulation of single domains or are not publicly available. In this section, we describe which features are required by a BGP model suitable to perform large-scale evaluation of interdomain traffic engineering. We call this model a *BGP routing solver*.

The purpose of a BGP routing solver is to accurately and deterministically compute the paths that routers will select given the configuration of the network without reproducing all the details of the protocol dynamics. That is, we are interested by the outcome of the BGP decision process but not by the transient states of the protocol convergence.

In addition to the computation of routes, we want that a BGP routing solver meet the following requirements:

- **Accurate model of the decision process**. The BGP decision process in our model must be as accurate as possible, taking into account all the possible BGP attributes associated with the routes. For this purpose, each step of the decision process must be modeled. In addition, techniques deployed to improve the scalability of the BGP protocol such as route-reflectors have to be modeled as well.

- **Accurate model of the filtering process**. It must be possible to model the policies enforced by ISP operators. For instance, the BGP routing solver must support the business relationships implemented between the different ASs. These policies are a key feature of BGP which are enforced through the use of route filters. The routing model must therefore allow the definition of complex route filters. Today's BGP implementations allow many different filters to be applied to routes. Our BGP model must support the relevant filters, e.g. those that deny routes or change their attributes and finally affect the outcome of the decision process.

- **Interaction between multiple domains**. The BGP routing solver must not be limited to a single domain. Since we want to be able to study both the routes that a particular domain has selected to reach external domains as well as routes that other domains will use to reach this particular domain. This is required to study both inbound and outbound traffic engineering techniques for instance.

- **Determinism**. The BGP routing solver must be deterministic. That means that its outcome must only depend on the initial configuration of the simulated network as well as on the advertised routes. The motivation behind this requirement is the ability to reproduce and compare simulation results. With a non deterministic decision process, this is not always possible. The requirement for determinism comes at a cost. Indeed, it is well-known that some BGP configurations have multiple solutions [GW99]. In this case, a deterministic routing solver will always lead to one of the solutions or not converge (see Section 2.5.7). The exploration of all the solutions of a BGP configuration is another problem that we do not solve in this thesis, though we will discuss the behavior of our routing solver in these situations in more details in Section 2.5.7.

  Note that certain implementations of BGP propose a knob that can trigger a decision process that is intentionally non-deterministic [CS05]. We do not model this behavior, hence the decision process of the solver must be deterministic.

- **Scalability**. The routing solver must be able to handle very large topologies with a size of the same magnitude as the Internet, i.e. composed of thousands of ASs themselves containing hundreds of routers.

- **Extensibility**. The model of BGP which is implemented in the solver must be easily extensible. Adding BGP extensions or studying the impact of changes in the protocol must not require to rewrite the whole model.

## 2.4 C-BGP: a BGP routing solver

In this section, we describe our approach to build a BGP routing solver meeting the requirements described in Section 2.3 and the hypothesis that are taken in order to make it efficient. Our implementation of this routing solver is called C-BGP. It is open-source and publicly available [Quo03b].

There are currently two main approaches to computing the BGP routes known by a router. The first one consists in resolving the dependencies between the routing decisions taken by the different routers in a distributed manner. This is the approach followed by [FWR04]. Its main limitations are (1) that it leads to a very specialized model of BGP even if it might be slightly more efficient[2] and (2) that it is limited to a single domain. The other approach is to *reproduce the propagation of BGP messages between the routers and the route selection performed by each router*. This is the approach used in C-BGP. However, in contrast with BGP daemons and packet-level simulators, C-BGP models the propagation of BGP messages across routers in a static way. We can do that since we are not interested in the transient states of routing, but only in its outcome. Hence, some aspects of the BGP protocol which are modeled in traditional packet-level simulators must not be considered in C-BGP. This allows to improve the efficiency of the simulator. We explain the simplifications we made in the BGP routing solver in the following paragraphs.

First, **we do not model the TCP connections that support the BGP sessions**. In the genuine BGP implementations, TCP connections are used to provide a reliable transport of the BGP messages as well as to keep them in sequence. In the routing solver, we assume that the BGP sessions between the routers are reliable. Therefore, we do not need to model TCP mechanisms such as the retransmission of presumably lost packets. In addition, the solver guarantees that the ordering of BGP messages exchanged between two routers will be unchanged.

The second simplification is in the **simplified model of the Finite State Machine** used by the BGP protocol to establish and maintain the sessions with neighbor routers. Basically, these FSMs rely on a small number of messages to initiate or close a session as well as to detect errors. For instance, both sides of a BGP session exchange keep-alive messages in order to notify each other that they are up and running. This is not required in our model. The session establishment is

[2]We could not compare the efficiency of our model to the BGP emulation proposed in [FWR04] since their implementation is not publicly available.

replaced by checking that each side can reach the other one, based on the routing information it currently has.

Then, we **do not model various timers** that are used by BGP in order to minimize BGP messages churns. Two timers are concerned. First, the Minimum Route Advertisement Interval (MRAI) timer is used to prevent a router to send to a neighbor too frequent updates for the same destination prefix. The second timer is used by a technique called Route Flap Dampening [MGVK02] which will avoid too frequent updates received from neighbor routers for the same prefix to be taken into account by the decision process. We do not model these timers since we do not care about routes that are flapping. This is reasonable since the large majority of Internet routes are stable over time [RWXZ02, UMB$^+$04].

Finally, we use a **steady-state model of the IGP protocol** to compute the intradomain routes without exchanging messages. A model of the IGP protocol is needed to compute the paths between BGP routers for instance.

All these simplifications do not affect the outcome of the decision process. However, they allow us to build a routing solver where we do not need to model the time as in discrete-event simulators. At the end, this approach determines the paths that routers have selected once the BGP routing has converged [GW99]. Another advantage of this approach is that it provides more information about the router states inside the studied domain since it determines the list of alternative routes that each router knows. This is very useful if one wants to study the diversity of BGP routes for instance.

### 2.4.1 Architectural overview

In this section, we give an overview of the architecture of C-BGP. As shown in Fig. 2.1, the architecture is articulated around 3 main conceptual components: the *network configuration*, the *network state* and the *scheduler*.

The first component of the BGP routing solver is the *network configuration*. It contains the static information about the network. The network configuration can be further refined in two parts. The first one is the network topology composed of the nodes and the links that interconnect them. The second part is composed of the configuration of the routers and links. This includes for instance the link attributes such as the IGP weights, and the addresses of the routers. The network configuration can be defined through the Command-Line Interface (CLI) of the routing solver or by mean of scripts.

The second component is the *network state*. It contains the current state of the nodes and links. For instance, the links can be up or down, meaning that they can currently be or not be used as part of the routes computed by the solver. The network state also includes the various routes that are currently known by each router. The network state is by essence a dynamic part of the routing solver. The network state can be changed manually (i.e. through the CLI or scripts) or it can be changed by events occurring within the routing solver.

Finally, the third component is the *scheduler*. The scheduler is the main part of

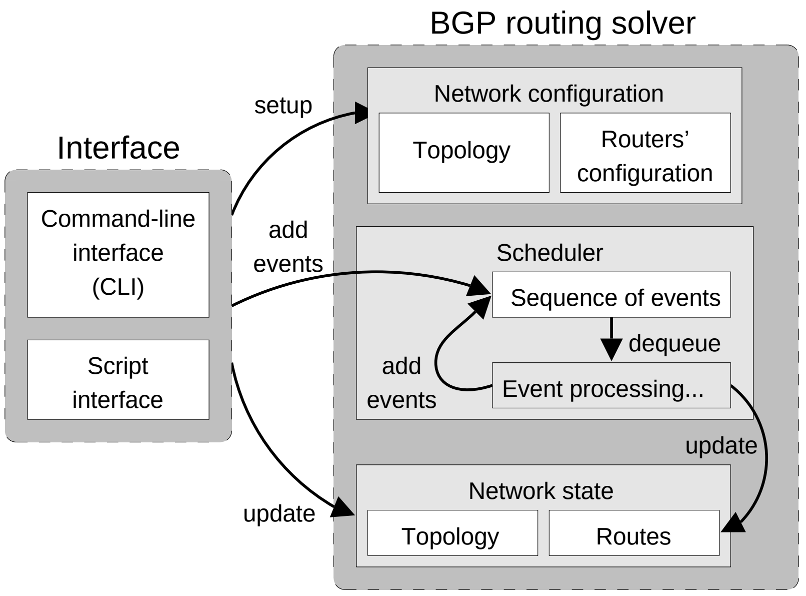


Figure 2.1: Architecture of the BGP routing solver.

the routing solver. It contains a sequence of scheduled events that will eventually update the state of the network when they are processed. A typical event in the routing solver is the reception of a BGP message by a router. The reception of a BGP message will usually trigger the re-computation of routes in the destination routers. The events are placed manually in the list of the scheduler or they are generated in reaction to another event.

We describe the network configuration, the network state, the scheduler and the message passing model of C-BGP in more details in the following sections.

### 2.4.2 Topology database

The topology database represents the topology of the simulated network. This network can be composed of several ASs. In the real world, each AS is composed of a collection of routers that are interconnected. These routers are usually interconnected using multiple SONET/SDH links, Ethernet LANs and switches. In our routing solver, the topology model does not include the physical- and facility-level details. It also does not contain a representation of end-systems. Indeed such details are not required in order to be able to accurately model how the route selection is performed in the AS. Therefore, it is sufficient to model the network topology with a graph where the nodes represent IP routers and the edges represent layer-three links. In Fig. 2.2, we show an example topology containing 2 domains. Domain 1 contains 4 routers and Domain 2 contains 2 routers. The label on a link represents the IGP weight of the link.

The graph is implemented by using two different data structures. In Fig. 2.3, we show the internal representation of the topology of Fig. 2.2. First, a **global trie** references all the routers of the graph. Each router is identified by a 32-bit integer. For instance, router R1 in the example of Fig. 2.2 has the identifier 1.0.0.1. This identifier can be thought of as the IP address of the loopback interface of the router. Each router can also have several other addresses that identify one endpoint of the links it is adjacent to. The identifier of each router is used as a key in the trie. The trie can thus have a depth which is at most 32. Hence, the complexity of an insertion or lookup in this trie is $O(32)$. We use a compressed unibit trie [Var05], so that the lookup time will often be far lower, as shown in Fig. 2.3.

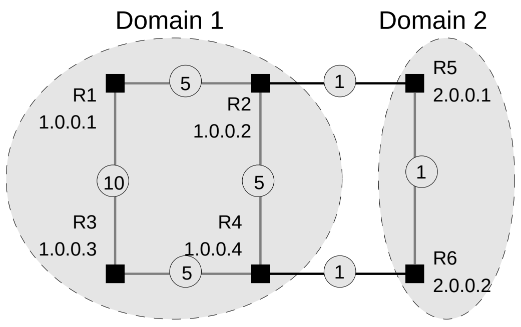


Figure 2.2: A simple network topology with 2 domains.

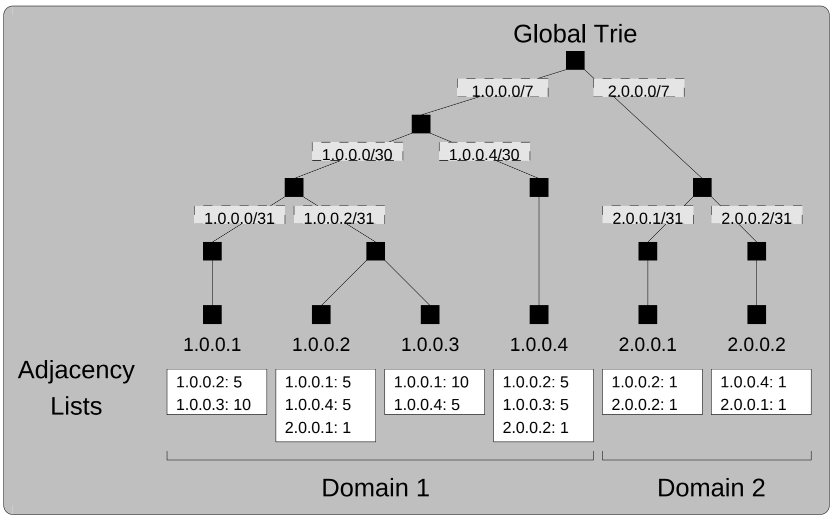


Figure 2.3: The internal representation of a topology inside C-BGP.

The second part of the graph consists in multiple **adjacencies lists** which represent the adjacencies of each router. For instance, router R2 in the example of

Fig. 2.2 has 3 adjacencies: R1, R4 and R5. Its adjacency list shown in Fig. 2.3 the identifier 1.0.0.1 (R1), 1.0.0.4 (R4) and 2.0.0.1 (R5). Each adjacency list is implemented as a sorted heap which allows lookups with a complexity of $O(log_2(n))$. The reason why we choose an adjacency-list based representation for the graph is the low density of the networks we expect to represent. Usually, network topologies have a degree distribution which looks like a power-law [FFF99]. This means that most nodes will have a low degree and a few nodes will have a large degree. In addition, since we focus on the backbone and border routers and not on the access routers, most nodes will have a low degree. According to [Gro04], the average node degree of North American carrier networks, is $\bar{d} < 2.5$, while the average node degree of European networks is closer to $\bar{d} > 3.5$. The consequence is that the average size of an adjacencies list will be small and the average lookup time will be short. In addition, the memory consumption is kept low even for very large graphs.

For each vertex and each edge in the graph, we can store attributes. For instance, the current routing solver associates an IGP metric with each direction of the links (in Fig. 2.2 and Fig. 2.3, we assume that the IGP weight of each link is the same in both directions). This metric is used when the intradomain routes are computed. The state of the links (up/down) is also stored in the graph.

The network topology in our routing solver can be built in many different ways. The first one consists in manually building a representation of an existing network. We discuss the modeling of a real ISP network in Chapter 3. Another possibility is to extract information from an IGP protocol trace captured in an existing network. A tool such as [Bon05] can be used for the ISIS protocol. Finally, it is also possible to build synthetic networks using topology generators such as BRITE [MAMB01] or GT-ITM [CDZ97].

### 2.4.3 Network state and IGP model

In this section, we describe the network state. The network state is mainly composed of the routes currently available for each node. In contrast with the topology database, the information stored in the network state is dynamic. In addition, the topology database describes the possible paths between each pair of routers while the network state contains the paths that are actually used to go from a router to another.

The routes contained in the network state at the end of the BGP routing solver execution are the results of the computation. On the other hand, to perform its computation, the BGP routing solver also needs some initial network state. This initial knowledge is required to allow two neighboring BGP routers to talk to each other. A BGP router must know the path to reach its neighboring routers. In real world networks for instance, an IGP protocol is used to distribute a view of the topology of a domain to all the routers in this domain. This is the same in C-BGP and the routes stored in the network state can initially be defined statically, as part of the routers configurations (through the CLI or script) or from a model of the intradomain protocol (IGP).

The selection of paths by the intradomain routing protocol is modeled without exchanging messages. The real IGP protocols such as OSPF or IS-IS are link-state protocols and rely on flooding information about their adjacencies to their neighbors in the same domain, in the form of Link State Packets (LSPs). Based on the LSPs it has received, each router can build a map of the network topology of its domain. Using this topology, each router computes in a distributed manner the shortest paths towards each node in the domain.

In our routing solver, we do not model the exchange of those LSPs. We are not interested in the dynamics of the IGP protocol. Moreover, exchanging these LSPs and computing the intradomain routes in a distributed manner would require additional computing resources in the routing solver. Indeed, we would need to build a database containing the received LSPs for each router. This would be equivalent to storing the topology of the domain multiple times. Instead of modeling the exchange of LSPs, we use the global knowledge of the adjacencies in each domain (contained in the topology database) to determine the intradomain routes. This is done by computing the shortest paths based on the IGP weights associated with each link. We compute the shortest paths in the solver using Dijkstra's SPF algorithm. The model we have implemented currently supports a single area, the most common type of IS-IS deployment in large ISP networks. However, an accurate model of the OSPF model supporting multiple areas for our routing solver has been developed by [Ias05].

In a typical simulation scenario, the intradomain routes are computed once after the topology has been defined. The paths computed by the IGP model do not change until there is a weight change or a link/router failure. When there is such a change in the topology, the re-computation of intradomain paths is not triggered automatically since the routing solver is not event-driven. In this case, the re-computation of intradomain routes must be asked explicitly (through the CLI or script).

The routes known by each router are stored in its routing table. Each routing table is implemented based on a compressed unibit trie and route lookup is done on a longest-match basis as in plain IP.

### 2.4.4 Scheduler and message passing model

The main feature of C-BGP is that the computation of the interdomain routes relies on the exchange of reachability information between nodes. This exchange takes the form of BGP messages. In this section, we describe how the nodes exchange BGP messages, although the message passing model we present here is more generic and is not limited to exchanging routing information.

The message passing model of our BGP routing solver is similar to packet forwarding in the IP protocol. The first similarity is in the **structure of the exchanged messages**. A message contains the following mandatory fields: a *Source Address* that identifies the node that sent the message, a *Destination Address* that identifies the node to which the message is destined, a message *Type* which indicates how the message must be processed and a *Payload* which carries the message content. Both the Source and Destination Addresses are 32-bit integers as in IPv4. The message *Type* is similar to the destination port field in IP packets in that it tells how the message must be processed on the destination node. The semantic of the message payload depends on the message type. In the BGP routing solver, the typical message type is BGP and in this case the payload can contain a BGP Update or a BGP Withdraw.

The characteristics of the message passing model are as follows. It is a simplified model of a transport protocol that **preserves the ordering** of messages, such as TCP. In addition, it is **reliable**, meaning that there is no loss of message. The propagation of messages from a source node to a destination node is done **hop-by-hop**. We detail these characteristics in the following paragraphs.

The message passing model is based on the scheduler of C-BGP. The scheduler is responsible for running the simulation by propagating messages between routers. Basically, the scheduler is based on a single, global linear queue (FIFO) to hold all the messages that are currently being propagated (see Fig. 2.4). The FIFO queue of C-BGP is implemented as a circular buffer. The size of the queue grows dynamically in order to hold more messages if required. When the size is stable, the *enqueue* and *dequeue* operations have a complexity of $O(1)$. In this way, C-BGP can handle events faster than in event-driven simulators where the scheduler is based on a priority queue. In the latter, the insertion operation can be as complex as $O(log_2(n))$.

As shown in Fig. 2.4, each message in the FIFO queue contains an additional field, the *Next-Hop*, which identifies the node to which the message must be delivered. In the example of Fig. 2.4, the queue contains 2 messages. Since MSG1 has been inserted before MSG2, it will be delivered first. MSG1 will be delivered to node 1.0.0.2 and MSG2 will be delivered to node 1.0.0.3.

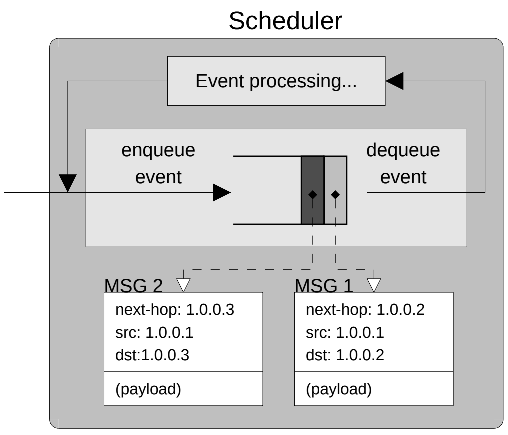


Figure 2.4: The global linear queue at the center of C-BGP.

The operation of the scheduler's algorithm, shown in Alg. 1, is simple to understand. The algorithm keeps running until the FIFO queue is empty. At each iteration, the algorithm first dequeues the message at the top of the queue. Doing so, it reduces the depth of the queue. Secondly, it finds the router identified by the Next-Hop field of the message by looking up in the global trie of the topology database. Finally, it delivers the message to this router. The routing solver uses a single thread to run the scheduler. When it delivers a message to a router, the scheduler waits until the router has finished processing the message. When a router is active, it only processes the received message and updates the network state if required. The router then stops and returns the control to the scheduler which will look for further messages in the global queue.

**Alg. 1** Simplified algorithm for the scheduler

```
1: while $!msg\_queue.empty()$ do
2:    /* Get next message to process */
3:    $msg = msg\_queue.dequeue()$
4:    /* Find next-hop router */
5:    $router = topology.find\_router\_by\_addr(msg.next\_hop)$
6:    /* Process message in destination router (dst) */
7:    $router.recv(msg)$
8: end while
```

During its processing of a received message, a router may produce additional messages that it pushes onto the queue. One can thus see that at each iteration of the loop of Alg. 1) the depth of the queue is reduced by one at first and subsequently increased by $n$ during the processing of the message. The value of $n$ depends on the behavior of the router and the content of the message. For instance, if the message is to be delivered locally, it is possible that no additional message will be produced ($n = 0$). To the contrary, if the message is to be forwarded to another router, the same message will be re-pushed onto the queue with an updated Next-Hop ($n = 1$). It is also possible that $n > 1$ messages will be produced after the reception of a message. For instance, a BGP router may receive a BGP UPDATE message and redistribute it to all its neighbors.

### 2.4.5 Messages ordering

The idea of using a global linear queue is not new. [LABJ00] already used a single linear, global queue to model the exchanges between single-node ASs and [GW00] proposed but did not implement this model.

Using a FIFO queue to model a transmission line guarantees that the message ordering will not be changed. That is, if a message A is issued by router R1 towards router R2 before a message B, R2 will receive message A before message B (see left part of Fig. 2.5). In C-BGP, there is a single FIFO queue. That means that the ordering of events is also preserved globally. Let's take the example shown in the right part of Fig. 2.5. If router R1 issues a message A destined to router R2 before router R3 issues a message B destined to router R4, then router R2 will receive message A before router R4 receives message B. If the ordering is kept globally, it is also kept on each transmission line. In addition, acting this way, is like if all the transmission lines have the same delay. This is unrealistic if one tries to model the dynamics of a protocol, but it is an efficient model to compute the outcome of the routes selection process.

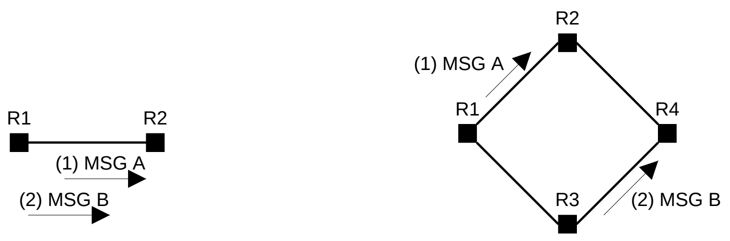


Figure 2.5: Preservation of the ordering along a single link (left) and along all links (right).

C-BGP forwards packets **hop-by-hop**. That means that if R1 sends a message to R3 in the example shown in Fig. 2.6, this message will first be delivered to the intermediate router R2 before being delivered to R3. There are two motivations for using hop-by-hop message passing. The first motivation is that using hop-by-hop propagation allows to check the validity of the path at each intermediate node. The routing solver can thus avoid to deliver a message to a router that is not reachable. This is useful for instance to check that a BGP session can be established between two routers. The second motivation is that it allows us to easily dig into the network state. For instance, we can directly discover the router-level path between two nodes by performing a trace-route between them. This makes possible for instance to check for forwarding deflection due to the IGP protocol [Ias05] or the BGP protocol.

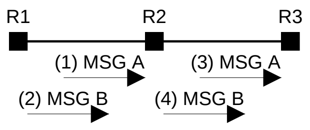


Figure 2.6: Preservation of the ordering along a path.

Messages are propagated by the scheduler on a hop-by-hop basis until they reach the destination. At each hop, the current router checks the destination address of the received message. If the router has an address that corresponds to the destination address of the message, it processes the message according to the message type field. Otherwise, the router has to find in the network state the adjacent router (next-hop) to which the message has to be forwarded. Two problems may occur during the forwarding of a packet. The first problem is the unreachability of the destination of the message. This occur when no route can be found to reach the message destination. In this case, the message is lost. The second problem is a forwarding loop. It is possible that due to misconfiguration the sequence of next-hops forms a cycle. This would cause the message to be forwarded forever and this would prevent the routing solver from terminating. For this reason, each message has a *Time-To-Live (TTL)* field that is decreased at each hop. The message is dropped when the TTL reaches 0. This ensures that the message will not loop forever and guarantees that the solver will not be prevented from terminating.

There is an additional requirement in order for this hop-by-hop message passing model to preserve the ordering of messages propagated along a path. This requirement is that **the path must not change during the propagation**. Hence, the routes used to forward messages must not change during the propagation. This can not occur for static routes since they can not be changed during the execution of the scheduler. This can not occur for intradomain routes since they cannot be recomputed during the execution of the scheduler. This can only occur for interdomain routes computed by BGP. Fortunately, these routes will not be used to forward messages over iBGP session or eBGP session. The only problem is in the case of multi-hop eBGP sessions. Indeed, in this case, the eBGP neighbor can be reached by a BGP route. Let's take the example of Fig. 2.7. In this example, there are 3 ASs. R1 in AS1 has a multi-hop eBGP session with R3 in AS3. R1 first sends a message A over this session to R3. The current route used by R1 to reach R3 is through R2. Therefore, message A is sent over the link R1-R2. Then, the route used by R1 is updated and goes directly to R3. This can occur if R1 has received a BGP update from R3 for instance. R1 will then send a second message, B, to R3. This message will be sent on link R1-R3. The scheduler will now deliver message A to R2 which will forward it on link R2-R3. The same is done for message B which is delivered to R3, its destination. Finally, the scheduler will deliver message A to R3. The consequence is that while message A has been sent earlier than message B by R1 to R3, message B has been delivered first.

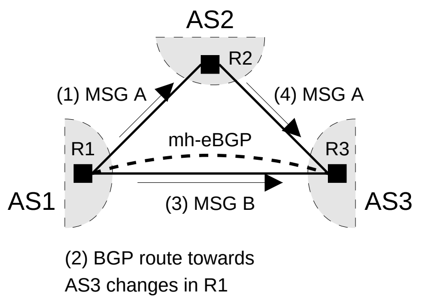


Figure 2.7: Multi-hop eBGP sessions may break the ordering.

### 2.4.6 Example

In order to clarify how the scheduler works, let's take the example topology shown in the right part of Fig. 2.5. Assume that R1 sends a message to all the other routers: R2, R3 and R4. We show in Table 2.1 the evolution of the scheduler's queue during the propagation of the messages. The first column is the iteration of the scheduler's loop. The second column describes the actions taken by the scheduler and the routers. The third column shows the content of the queue after the actions are taken. The third column only shows the next-hop, source and destination fields of the enqueue messages.

| **Step** | **Action** | **Queue** |
|---|---|---|
| **init** | R1 pushes 3 messages | (1) n-h:R2, src:R1, dst:R2<br>(2) n-h:R3, src:R1, dst:R3<br>(3) n-h:R2, src:R1, dst:R4 |
| **1** | scheduler pops message, delivers to R2. | (2) n-h:R3, src:R1, dst:R3<br>(3) n-h:R2, src:R1, dst:R4 |
| **2** | scheduler pops message, delivers to R3. | (3) n-h:R2, src:R1, dst:R4 |
| **3** | scheduler pops message, delivers to R2.<br>R2 pushes message (forward). | (4) n-h:R4, src:R1, dst:R4 |
| **4** | scheduler pops message, delivers to R4. | |

Table 2.1: Example of message propagation and evolution of the scheduler's queue.

First, router R1 pushes three messages onto the queue. One destined for R2, the second destined for R3 and the third one destined to R4. Since R2 and R3 are adjacent to R1, the next-hop for these routers is equal to their destination. To the opposite, R4 is not adjacent to R1. We suppose that R1 knows that it must send its message to R2 in order to reach R4.

After this initialization step, the queue contains three messages. The simulation starts. During the first loop iteration, the scheduler pops the first message from the queue. The next-hop of this message is R2. The scheduler delivers the message to R2. Since it is locally destined, there is no subsequent enqueuing of message. The queue now contains 2 messages. The scheduler starts the second iteration by poping the first message of the queue. The next-hop of this message is R3. The scheduler delivers the message to R3 which is the final destination of the message. No message is added to the queue during this iteration. The queue now contains a single message. The scheduler now enters its third iteration. It pops the message from the queue. The next-hop of this message is R2. The scheduler delivers the message to R2. The final destination of the message, R4, is not R2. R2 will thus forward the message to R4. For this purpose, it pushes the message onto the queue with the next-hop field now containing R4. At the end of this iteration, the queue still contains one message. The scheduler pops the message from the queue during the fourth iteration. It delivers it to router R4 which is the final destination of the message. At the end of this iteration, the queue is empty and the simulation is finished.

## 2.5 BGP Routing Model

In this section, we describe the BGP routing model implemented in the solver. We discuss the components of BGP routers that are modeled and those that were simplified or left over since they have little or no impact on the computation of routes. We first present the BGP model from the perspective of the routers, e.g. which components of a BGP router are represented in the solver. Then, we describe how the BGP sessions between the routers are modeled. We then show the basic operations of a router in the solver. We give more details on the filtering processes that are implemented in the model in Sec. 2.5.4 as well as a full description of the decision process in Sec. 2.5.5. We validate C-BGP in Section 2.5.6 and we discuss its convergence properties in Section 2.5.7. Finally, we evaluate its performance in Section 2.5.8, the scalability issues and related solutions in Section 2.5.9.

In order to model the full BGP decision and filtering processes, the routing solver must contain most components of genuine BGP routers. As shown on Fig. 2.8, each router has one **Loc-RIB** where all the best routes are stored. In addition, each router has a **list of neighbors**. For each neighbor, the router knows the identifier of the corresponding node, i.e. the IP address of the neighbor, as well as the domain the neighbor belongs to. Based on this, it is able to determine if the BGP session it has with the neighbor is internal (iBGP) or external (eBGP). Each router also has a pair of adjacent RIBs for each neighbor router: an **Adj-RIB-In** is used to store the routes received from this router and an **Adj-RIB-Out** is used to store routes redistributed to this router. In addition to providing information on routes advertised to the neighbors of a router, the Adj-RIB-Outs allow a stateful BGP that do not send the same UPDATE/WITHDRAW multiple times. The Loc-RIB, Adj-RIB-ins and Adj-RIB-outs are implemented with compressed unibit tries in order to allow fast lookups and limited memory consumption.

### 2.5.1 BGP sessions and Finite State Machine

For the purpose of maintaining the BGP sessions with their neighbors, genuine BGP routers have a Finite State Machine (FSM) for each session. An FSM controls which messages can be exchanged over the session. It is also responsible for establishing a BGP session and maintaining it. For this purpose, genuine BGP routers regularly exchange KEEPALIVE messages to inform the other side that the session is still running. In the routing solver, each router has an FSM per session. However, the FSMs used in the model are simplified: they are only used for the es-

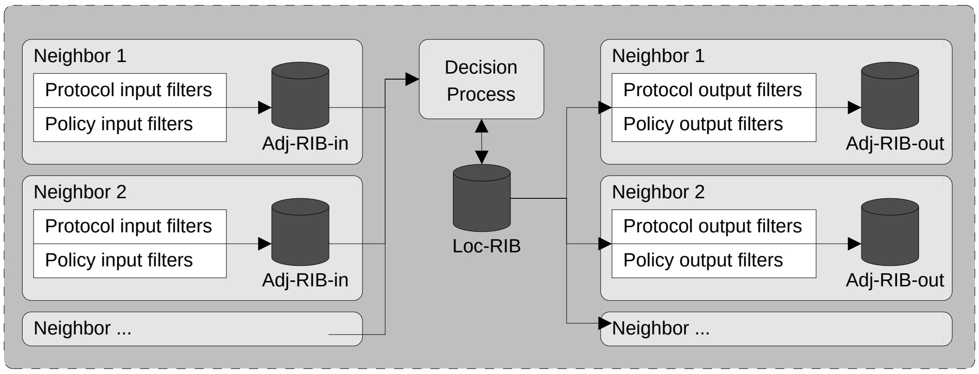


Figure 2.8: Model of a BGP router in C-BGP.

tablishment of sessions and to control if reachability information can be exchanged or not. Indeed, in the routing solver, there is no need to exchange KEEPALIVE messages since the solver is not event-driven. Once a session is established, it will not fail during the routes computation.

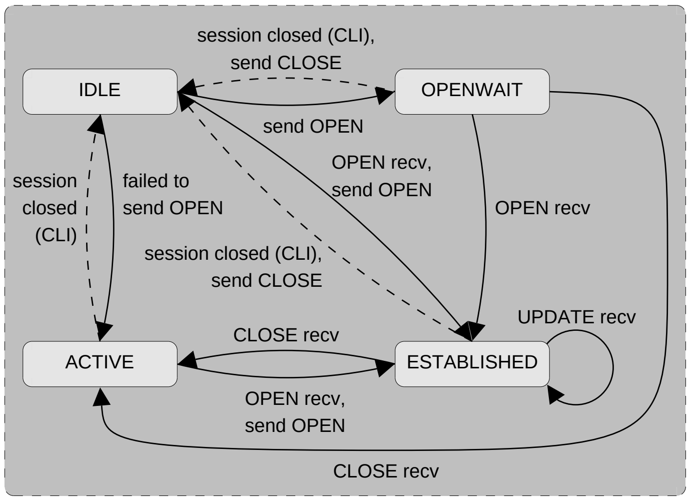


Figure 2.9: Simplified Finite State Machine in C-BGP.

The FSMs of the model (shown in Fig. 2.9) are mainly used to help detecting problems with the establishment of sessions and to detect the failure of sessions after topology changes. For this purpose, an FSM can only be in 4 different states: *IDLE*, *ACTIVE*, *OPENWAIT* or *ESTABLISHED*. The IDLE state means that the session is administratively disabled. The ACTIVE state means that the session is administratively enabled but it could not be established. One possible reason for this is that there was no route available to reach the neighbor router. The OPEN-

WAIT state means that the FSM has started establishing the session and is awaiting an answer from the other side. If the session remains in OPENWAIT state, that can mean that the other side of the session is not configured, not enabled or that the neighbor router can not reach the other side. The ESTABLISHED state means that the session can be used to exchange reachability information.

Four message types are used in the BGP routing solver: *OPEN*, *CLOSE*, *UPDATE* and *WITHDRAW*. The OPEN and CLOSE message types are used to respectively setup and shutdown a BGP session. The UPDATE message type is used to advertise the reachability of a network prefix and the WITHDRAW message type is used to cancel such reachability. UPDATE and WITHDRAW messages can only be exchanged over a session that is in the ESTABLISHED state. KEEPALIVE and NOTIFY messages are not modeled.

### 2.5.2 BGP route attributes

An UPDATE message carries a BGP route composed of a destination IP prefix and route attributes. In the routing solver, we model the BGP routes accurately. We show in Table 2.2 the BGP route attributes and indicate in the second column if the attribute is standard or experimental, in the third column if the attribute is mandatory and in the fourth column if the attribute is supported by the solver. The current version of the routing solver supports the following mandatory route attributes: *Next-Hop*, *Local-Pref* and *AS-Path*. The AS-Path is composed of a sequence of segments of different types. The solver models the *AS_SEQUENCE* that contains an ordered set of AS numbers and the *AS_SET* that contains an unordered set of AS numbers. The solver currently does not support the segment types related to BGP confederations [TMS01]: *AS_CONFED_SEQUENCE* and *AS_CONFED_SET*.

The standard optional attributes such as the *MED* and the *Communities* [TCL96] are supported. In particular, the well-known community values *NO_EXPORT* and *NO_ADVERTISE* are supported. To the opposite, the *NO_EXPORT_SUBCONFED* value is not supported since Confederations are currently not modeled. The *Extended Communities* [STR06] are also supported. The attributes related to route-reflection [BCC00] such as the *Originator-ID* and the *Cluster-ID-List* are supported. In addition, experimental optional attributes such as the *Redistribution Communities* [BCH$^+$03] are supported. The solver currently does not support the attributes related to route aggregation: *Atomic-Aggregate* and *Aggregator*.

### 2.5.3 BGP router model

We show in Alg. 2 the simplified algorithm that models a BGP router. The algorithm only shows the processing of the UPDATE and WITHDRAW messages when the FSM is in state ESTABLISHED. First, the router tests if the received BGP message is an UPDATE or a WITHDRAW. If the BGP message is an UPDATE, the router checks if the route contained in the message is accepted by its input filters. If so, the route is stored in the Adj-RIB-in corresponding to the neigh-

| **Attribute** | | **Standard** | **Mandatory** | **In C-BGP** |
|---|---|---|---|---|
| **Next-Hop** | | X | X | X |
| **Local-Pref** | | X | X | X |
| **AS-Path** | *AS_SEQUENCE* | X | X | X |
| | *AS_SET* | X | X | X |
| | *AS_CONFED_SEQUENCE* | X | | |
| | *AS_CONFED_SET* | X | | |
| MED | | X | | X |
| **Communities** | any value | X | | X |
| | *NO_EXPORT* | X | | X |
| | *NO_ADVERTISE* | X | | X |
| | *NO_EXPORT_SUBCONFED* | X | | |
| **Atomic-Aggregate** | | X | | |
| **Aggregator** | | X | | |
| **Originator-ID** | | X | | X |
| **Cluster-ID-List** | | X | | X |
| **Ext-Communities** | | X | | X |
| **Red-Communities** | *PREPEND* | | | X |
| | *NO_EXPORT* | | | X |
| | *IGNORE* | | | X |

Table 2.2: List of route attributes supported by C-BGP.

bor that sent the message. In addition, the router's decision process is run. The decision process retrieves from the Adj-RIB-ins all the reachable routes for the considered prefix, compares them and selects the best one. The best route is installed in the router's forwarding table. The next-hop interface is found by looking up in the forwarding table the interface used to reach the BGP next-hop. Then, the router propagates the new best route to its neighbors according to its output filters. The propagation is done by pushing new BGP messages on the global linear queue. The solver continues until the message queue is empty, which means that BGP has converged.

**Alg. 2** Simplified algorithm for handling a BGP UPDATE message $M$ received by a router $R$

```
if $(M.type == UPDATE)$ then
   if $(R.in\_filter(M.src, M.route) == DENY)$ then
      $R.adj\_rib\_in[M.src].remove(M.route.prefix)$
   else
      $R.adj\_rib\_in[M.src].replace(M.route.prefix, M.route)$
   end if
   $R.decision\_process(M.route.prefix)$
   if $(R.best\_has\_changed(M.route.prefix))$ then
      $R.FT.install(R.best)$
      for all $neighbor$ in $(R.neighbors)$ do
         if $(R.out\_filter(neighbor, M.route) == ACCEPT)$ then
            $msg\_queue.push(BGP\_UPDATE, M.src, neighbor, M.route)$
         end if
      end for
   end if
else if $(M.type == WITHDRAW)$ then
   /* ... */
end if
```

If the message is a WITHDRAW, the router removes from the corresponding Adj-RIB-in the route towards the withdrawn prefix. If this route was selected as best, the router runs the decision process in order to select a new best route. The new best route is then installed in the forwarding table and redistributed to the neighbors of the router in a similar way than after the reception of an UPDATE message.

In the following sections, we describe with more details the filtering and decision processes implemented in the solver.

### 2.5.4 Route Filtering

We showed in Alg. 2 that a BGP router has route filters that it applies on received routes (input filters) as well as on outgoing routes (output filters). The input filters are used to decide which routes received from the neighbors will be accepted for selection by the decision process while the output filters are used to decide which best-routes from the Loc-RIB will be advertised to the neighbors. There are two types of filters implemented in the model: filters that are required for the BGP protocol itself and filters that implement routing policies. Filters of the first category are available in all the routers of the model. To the opposite, filters implementing routing policies are configurable and can be defined independently in each router, as in genuine BGP routers. In addition, the input and output filters of the second category can be defined on a per session basis for the same router. This allows the modeling of very flexible routing policies.

The filters belonging to the first category (protocol filters) are mandatory in the BGP routing solver since they are part of the BGP protocol and its extensions. For instance, there is an input filter that checks that the AS-Path of received routes does not already contain the local AS. If it does, the route cannot be selected since it would cause an AS-Path loop otherwise. Another input filter rejects routes that contain the local Cluster-ID in their Cluster-ID-List. When routes are about to be redistributed, protocol output filters are also applied. For instance, if the router is not a route-reflector, it is not allowed to redistribute a route learned over an iBGP session to another iBGP peer. If the router is a route-reflector, additional rules apply (see [BCC00]). Another output filter that is supported by our routing solver is the Sender Side Loop Detection (SSLD) which checks if advertising a route to an eBGP neighbor will cause an AS-Path loop or not. In the latter case, the route can be redistributed. Using SSLD decreases the number of messages to propagate during the simulation.

The filters belonging to the second category must be highly flexible in order to allow the evaluation of various policies and traffic engineering methods. These filters can manipulate the attributes of one route or simply reject the route. The input and output filters work in the same manner. The principle of filters is to check a set of conditions on the route and if these conditions are met, to apply a set of actions to the route. More formally, a filter is composed of a sequence of rules. Each rule is made of two distinct parts: a predicate and a set of actions. The purpose of the predicate is to express the conditions that the route must meet before being applied the set of actions. The predicate is a logical combination of checks on the route attributes. The logical combination of these checks is based on the binary operators *AND* and *OR* and the unary operator *NOT*. The basic predicates that are currently supported by the routing solver are listed in Table 2.3. The basic predicates allow for instance to test the value of the `Local-Pref` attribute, to match the `AS-Path` with a regular expression[3] or to test if a particular value is contained in the `Communities` attribute.

On the other hand, the set of actions modifies the route attributes. Special actions can be used to simply accept or reject the route without further processing.

[3]The predicate that checks if an AS-Path matches a regular expression was added by Sebastien Tandel.

| **Predicate** | **Description** |
|---|---|
| $Communities \ni C$ | Test if the value $C$ belongs to the *Communities* attribute. |
| $NextHop \in P$ | Test if the next-hop is included in the prefix $P$. |
| $NextHop = A$ | Test if the next-hop is equal to the address $A$. |
| $ASPath \equiv R$ | Test if the AS-Path attribute matches the regular expression $R$. |
| $Prefix \in P$ | Test if the destination prefix is included in the prefix $P$. |
| $Prefix = P$ | Test if the destination prefix is equal to the prefix $P$. |

Table 2.3: List of basic filtering predicates supported by C-BGP.

Table 2.4 contains the list of actions currently supported by C-BGP.

| **Action** | **Description** |
|---|---|
| **accept** | Accept the route (skip the subsequent filter rules). |
| **prepend AS-Path** | Prepend the local AS-number to the AS-Path attribute. |
| **append** $V$ **to Communities** | Add the value $V$ in the Communities attribute. |
| **remove** $V$ **from Communities** | Remove the value $V$ from the Communities attribute. |
| **clear Communities** | Empty the Communities attribute. |
| **deny** | Reject the route. |
| **set Local-Pref to** $V$ | Set the value of the Local-Pref attribute to $V$. |
| **set MED to** $V$ | Set the value of the Local-Pref attribute to $V$. The solver also allows $V$ to be determined automatically based on the IGP cost. |
| **append** $RC$ **to Red-Community** | Add the value $RC$ to the Redistribution Communities attribute. $RC$ can request to prepend $N$ times the AS-Path of the route redistributed to... |

Table 2.4: List of filtering actions supported by C-BGP.

Alg. 3 shows the algorithm which is applied each time a filter is invoked on a route. For each rule that composes the filter, the algorithm tests if the predicate of the rule matches the route. If so, the actions of the rule are applied in sequence to the route. If an action is ACCEPT or DENY, the route is immediately either accepted or denied. In this case, the subsequent actions of the current rule and the subsequent matching rules are not applied. Otherwise, the remaining actions are applied. Once all the actions of the current matching rule have been applied, the algorithm treats the second rule in the same way. The algorithm finishes once all the rules have been applied.

**Alg. 3** Algorithm for filtering route $R$ with filter $F$.

```
1: for all $rule$ in $F$ do
2:   if $(rule.predicate$ matches $ROUTE)$ then
3:     for all $action$ in $rule.actions$ do
4:       if $(action == ACCEPT)$ then
5:         return $ACCEPT$
6:       else if $(action == DENY)$ then
7:         return $DENY$
8:       else
9:         $apply(action, ROUTE)$
10:        return $ACCEPT$
11:      end if
12:    end for
13:  end if
14: end for
```

Fig. 2.10 shows how a single rule is encoded in C-BGP. The rule contains a predicate (on the left) and a sequence of actions (on the right). The predicate is encoded in a tree that represents the logical combination of basic predicates. Here, the tree represents the predicate $(Communities \ni 1) \vee ((Prefix \in 130.104/16) \wedge (Communities \ni 2))$. This means that the set of actions will be applied to a route if its Communities attribute contains the Community value 1. The actions will also be applied if the Communities attribute contains the Community value 2 and the advertised prefix belongs to 130.104/16. In the example of Fig. 2.10, the sequence of actions consist in first setting the Local-Pref to 100 and then accepting the route.

In a large-scale simulation, there can be a large number of BGP sessions. Since the filters can be specified on a per-session basis, the number of filters stored in the solver can become large and require a significant amount of memory. However, the filters applied on the sessions are often the same. There is often only a limited number of different filters. For instance, all the customers of a domain will often be treated in the same manner. The filters used on the sessions with these customers will therefore be identical. For this reason, C-BGP has a mean to share filters across multiple sessions. This helps to ensure the scalability of the model[4].

[4]Sebastien Tandel implemented methods to share filter rules among multiple filters and successfully applied the routing solver to a large transit domain.

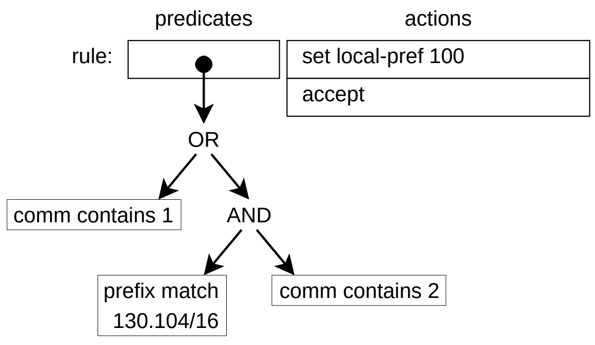


Figure 2.10: Structure of a single filtering rule: a predicate and a list of actions.

### 2.5.5 Decision Process

In this section, we describe the BGP decision process implemented in the solver. Basically, the decision process takes a set of routes as input and selects a single one based on multiple criteria. We list in Table 2.5 the criteria that compose the solver's model of the BGP decision process. It slightly differs from the decision process implemented in genuine BGP routers for one reason: we want a decision process that is deterministic. That means that each time it is run on the same set of routes it must select the same best route. Some genuine BGP routers allow an intended nondeterminism in their implementation of the decision process. These routers use a tie-break function that depends on the time when routes are received [CS05]. Basically, the route received first is preferred. The motivation behind this tie-break function is that the route received first is more stable than the routes with the same quality (preference, length of AS-Path, etc.) received later. The problem with such a tie-break function is that it is not always possible to predict the outcome of the route selection. We rely on the fact the the large majority of Internet routes are stable [RWXZ02] along time to motivate the use in the solver of a decision process which does not depend on time.

We now precisely define the decision process implemented in the solver. The decision process is an ordered set of rules. Each rule is applied sequentially to a set of eligible routes towards the same prefix until the set contains at most one route. The 1st rule of the decision process is a filtering rule. It ignores the routes for which there is no route IGP to reach the BGP next-hop. This rule appears in the decision process and not in the protocol filters since the accessibility of the next-hop is not a route attribute. The accessibility may change across time, e.g. the next-hop may be temporarily inaccessible due to a link failure. The 2nd rule prefers routes towards networks that are locally originated. The reason for this rule is to prevent the decision process to select an external route towards a network belonging to the local AS. In genuine BGP routers, the decision process also has a rule that prefers the local aggregates [Sys05a]. Since we currently do not support aggregation, this

| Rank | Rule |
|---|---|
| **1** | Ignore if the next-hop is inaccessible |
| **2** | Prefer locally originated networks |
| **3** | Prefer highest Local-Pref |
| **4** | Prefer shortest AS-Path |
| **5** | Prefer lowest Origin |
| **6** | Prefer lowest MED |
| **7** | Prefer eBGP over iBGP |
| **8** | Prefer nearest next-hop |
| **9** | Prefer lowest router-ID or Originator-ID |
| **10** | Prefer shortest Cluster-ID-List |
| **11** | Prefer lowest neighbor address |

Table 2.5: Decision process implemented in C-BGP.

rule is not implemented.

The 3rd rule of the decision process prefers the routes which have the highest value of the Local-Pref attribute. The 4th rule prefers the routes which have the shortest AS-Path. Note that the AS_SET segments have a length of 1 while AS_SEQUENCE segments have a length which depends on the number of AS number they contain. The 5th rule prefers the routes which have the lowest value of the Origin attribute. The Origin attribute can take the values IGP, EGP and INCOMPLETE. The preference is as follows: $IGP < EGP < INCOMPLETE$. The 6th rule prefers the routes which have the lowest value of the MED. The 7th rule prefers the routes learned through an eBGP session over routes learned through an iBGP session. The 8th rule prefers routes which have the lowest IGP cost to their next-hop.

The remaining rules are used to break the ties if there are still multiple possible routes after the above decision criteria. The 9th rule prefers the routes received from the neighbor with the lowest router-ID if the route has not been reflected or the route with the lowest Originator-ID if the route has been reflected. The 10th rule prefers the routes with the shortest Cluster-ID-List. And finally, the 11th rule prefers the routes with the lowest neighbor address. The rules 9 to 11 define a deterministic tie-breaking function.

### 2.5.6 Validation

In this section, we describe how we validated C-BGP. By validation, we understand checking that the outcome of the BGP decision process computed by C-BGP is conform to what genuine BGP would have computed. Usually, the validation of a protocol implementation is performed with a large suite of test cases covering various aspects of the protocol. For BGP, there is no general agreement on which test suite should be used. We rely on a suite of validation tests inspired from the

SSFNet BGP implementation [Pre]. Among these tests, we only considered those that concern the routing decisions, not the dynamics. We detail some of the tests in this section[5]

The first validation we use is **route-distrib**. The aim of this validation is to test the ability of two routers $R1$ and $R2$ in two separate domains to exchange their routes over an eBGP session. The configuration of the test is shown in Fig. 2.11. $R1$ and $R2$ advertises the prefixes $P1$ and $P2$ respectively. We check that the routers receive the route from each other. $R1$ receives a route towards $P2$ with an AS-Path equal to `2`. In a similar manner, $R2$ receives a route towards $P1$ with an AS-Path equal to `1`. We also check that the routes are selected as best and installed in the forwarding table. The simulation generates 4 events. The 2 first events are OPEN messages used to initiate the eBGP session. The next events are UPDATE messages containing the network prefixes advertised by each router.

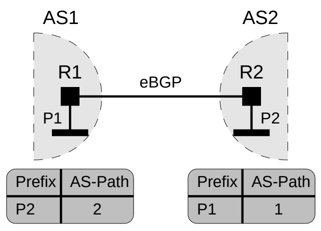
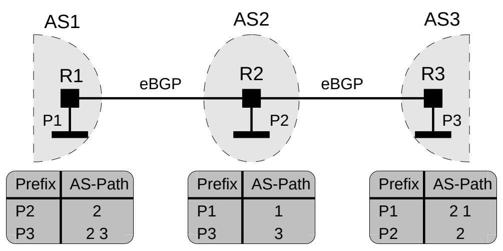


Figure 2.11: Validation: route-distrib.

The second test we perform is **propagation**. The aim of this validation is to test that a router that receives an eBGP route propagates this route to its neighbors. The configuration is shown in Fig. 2.12. Three routers $R1$, $R2$ and $R3$ in three separate domains exchange their routes over two eBGP sessions (in line). The central router $R2$ propagates the UPDATE messages received from its neighbors. We check that all three routers have received the routes from the others. In addition, we check that the AS-Path of the propagated routes have been updated. For example, $R3$ has a route to $P1$ with an AS-Path equal to `2:1`. The simulation generates 10 events. The 4 first messages are OPEN messages used to initiate the eBGP sessions. The next 4 events are the UPDATE messages containing the network prefixes advertised by each router. The last 2 events are UPDATE messages propagated by $R2$ to $R1$ and $R3$.

The third test we perform is called **select**. The aim of this test is to check that when a router has received multiple routes with the same preference, it selects the one with the shortest AS-Path. The configuration of this test is shown in Fig. 2.13. Three routers $R1$, $R2$ and $R3$ in three separate domains are connected in a full-mesh of eBGP sessions (a triangle). A single router, $R1$, sends its own network prefix $P1$ to the other routers. We check that $R2$ receives the route from $R1$ and propagates it to $R3$ with an AS-Path equal to `2:1`. Similarly, $R3$ propagates to $R2$

[5]A more complete validation test suite is available from C-BGP's web site [Quo03b].



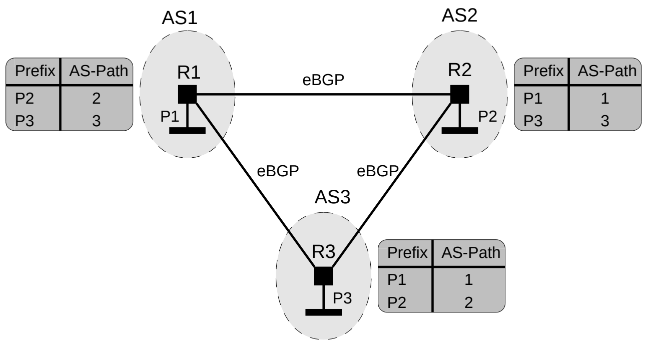
| Prefix | AS-Path |
|---|---|
| P2 | 2 |
| P3 | 2 3 |

| Prefix | AS-Path |
|---|---|
| P1 | 1 |
| P3 | 3 |

| Prefix | AS-Path |
|---|---|
| P1 | 2 1 |
| P2 | 2 |

Figure 2.12: Validation: propagation.

a route towards $P1$ with an AS-Path equal to `3:1`. We check that when $R2$ has two routes at disposal it selects the one with the shortest AS-Path. We check the same for $R3$. The simulation generates 10 events. The 6 first messages are OPEN messages used to initiate the eBGP sessions. The next 2 events are the UPDATE messages containing the network prefix advertised by router $R1$. The last 2 events are UPDATE messages propagated by the $R2$ and $R3$. Since the routes contained in the latter messages have longer AS-Paths, they are not selected as best and not redistributed. The simulation does not generate additional events in this case and terminates.



| Prefix | AS-Path |
|---|---|
| P2 | 2 |
| P3 | 3 |

| Prefix | AS-Path |
|---|---|
| P1 | 1 |
| P3 | 3 |

| Prefix | AS-Path |
|---|---|
| P1 | 1 |
| P2 | 2 |

Figure 2.13: Validation: select.

The fourth validation tests the behavior of a router when a route is withdrawn. The test is called **withdrawals**. In this test, two routers in two separate domains are connected by an eBGP session. One router X advertises its own prefix to the other router, Y. Y receives the route and installs it in its forwarding table. Then, router X withdraws the previously advertised prefix. Y receives the withdraw and removes the route from its forwarding table. The simulation generates 4 events. The 2 first events are OPEN messages used to initiate the eBGP session. The next event is the UPDATE message containing the network prefix advertised by router X. The last event is the WITHDRAW message for the prefix previously advertised by router X.

The fifth test called **ibgp** involves three domains. This situation is depicted in Fig. 2.14. One domain (AS2) is composed of two routers R2 and R3 connected through an iBGP session. The other domains, AS1 and AS3, contain only one router, R1 and R4 respectively. The three domains are connected in a line using eBGP sessions. The router in AS1 advertises its own prefix, P1. We check that prefix is propagated across all the sessions. The simulation generates 9 events. The 6 first events are OPEN messages used to initiate the eBGP and iBGP sessions. The next 3 events are the UPDATE messages propagated across the first eBGP session between AS1 and AS2, then across the iBGP session inside AS2 and finally across the eBGP session between AS2 and AS3.

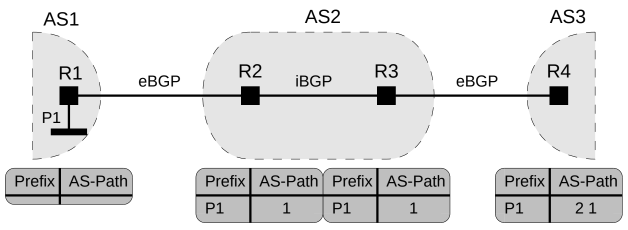


Figure 2.14: Validation: ibgp.

The sixth test, called **reflection**, involves four domains (see Fig. 2.15). One of them, AS1, contains 6 routers connected with an hierarchy of iBGP sessions. R13, R14 and R15 are three route-reflectors connected by a full-mesh of iBGP sessions. R13 has two clients, R11 and R12. R15 has a single client, R16 and R14 has no client at all. The three other domains contain a single router and are connected to a different non-route-reflector router in AS1. R2 in AS2 originates a single prefix P1. We check that all the routers in the topology receive one route towards this prefix. The simulation generates 26 events. 18 events are due to the exchange of OPEN messages to establish the 9 BGP sessions. The other 8 events concern the propagation of the UPDATE message originated by R2. This message is first propagated to R11, then to R13 which reflects it to R12, R14 and R15. R14 cannot reflect the UPDATE to R15. R12 propagates the UPDATE to R3. R15 reflects the UPDATE to R16 which subsequently redistribute it to R4.

We summarize in Table 2.6 the validation tests inspired from the SSFNet implementation. We compare the number of events required by the SSFNet simulator to perform these validations to the number of events required by C-BGP. An event in the SSFNet simulator can represent the reception of a message or the expiration of a timer. In C-BGP, an event only corresponds to the reception of a message. The number of events required to complete a simulation is a measure of the complexity of the simulation. The more events are required, the more the simulation will last.

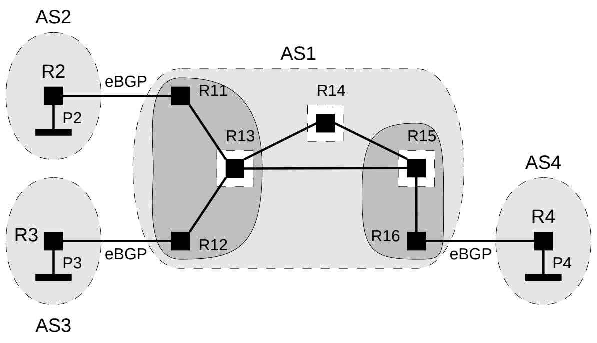


Figure 2.15: Validation: reflection.

| | **SSFNet** | **C-BGP** |
|---|---|---|
| **route-distrib** | 44 | 4 |
| **propagation** | 6729 | 10 |
| **select** | 7120 | 10 |
| **withdrawals** | 73 | 4 |
| **ibgp** | 927 | 9 |
| **reflection** | 6370 | 26 |

Table 2.6: Comparison of the number of events required for simple simulation scenarios.

### 2.5.7 Convergence properties

In this section we discuss the convergence properties of C-BGP in presence of naughty BGP configurations. It has been shown that the BGP protocol will not lead to a stable solution in certain configurations [GW99]. We say that the protocol does not converge in these cases. The BGP protocol may also lead to non deterministic solutions in certain cases. Griffin and Wilfong have explained in [GW99] some simple situations where BGP is not able to find a solution or will not always lead to the same solution. In [GH05], a more complex situation involving the communities attribute has been shown to lead to unstable solution. Since C-BGP does not exactly reproduce the dynamics of BGP, it is important to ask what will be its behavior when it is used to solve such configurations.

The message propagation model of C-BGP relies on a single global linear queue. This queue guarantees that the ordering of messages is kept on each BGP session. In addition, messages that are issued at a given time are delivered before messages issued at a later time. To the opposite of discrete-event simulators, the propagation of messages in C-BGP is deterministic. Any execution of a simulation script will lead to the same outcome while in discrete-event simulators, the outcome of the simulation may depend on the seed of the pseudo random number generator[6]. This has a consequence on the convergence of the simulations performed with C-BGP.

In this section, we discuss how C-BGP will behave when used to solve the simple configurations described in [GW99]. In the case where a BGP configuration has multiple stable solutions (see the DISAGREE case for instance), the simulation will not converge. In the case of a BGP configuration without a stable solution (such as the BAD-GADGET), the behavior of C-BGP will be the same as with discrete-event simulators.

## DISAGREE

The most simple configuration described by [GW99] is called DISAGREE. This configuration is shown in Fig. 2.16. There are three routers R1, R2 and R3, each one in a distinct AS. A prefix is originated by R1. R2 prefers the routes received from R3 over the routes received from R1 (indicated by the "+" on the arrow). Similarly, R3 prefers the routes from R2 over those received from R1. In this situation, there are two possible stable solutions. The first solution is R2 selects the route through R3 and R3 uses its direct route to R1. The second solution is R2 uses its direct route towards R1 and R3 uses the route through R2.

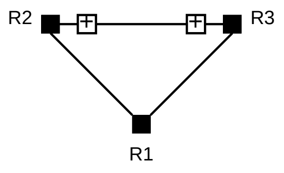


Figure 2.16: DISAGREE: a configuration that admits multiple solutions.

In real routers, BGP will often find a solution to this configuration. It will thus eventually chose one of the two solutions, but we do not know which one. With discrete-event simulators, the simulation might converge to one of the solutions, in a non-deterministic manner. To the contrary, if we run C-BGP with such a configuration, it will never find a solution. It will keep sending BGP messages between R2 and R3 forever. The reason for this is due to the model of the propagation of BGP messages in C-BGP. Let's take a deeper look at what will happen in C-BGP.

In Fig. 2.17, we show the DISAGREE configuration and 8 steps of the C-BGP convergence. Each step corresponds to the processing of the message that was at the top of the queue in the previous step. We show with an arrow the pair of routers that exchange a message, as well as the content of the global FIFO queue after the message has been processed. The messages shown with the darkest color are the messages enqueued during this step. For instance, step 1 represents the initial

[6]This is especially true in discrete-event simulators relying on calendar queues where random jitter is introduced in order to better balance the number of events in each of the calendar's pages.

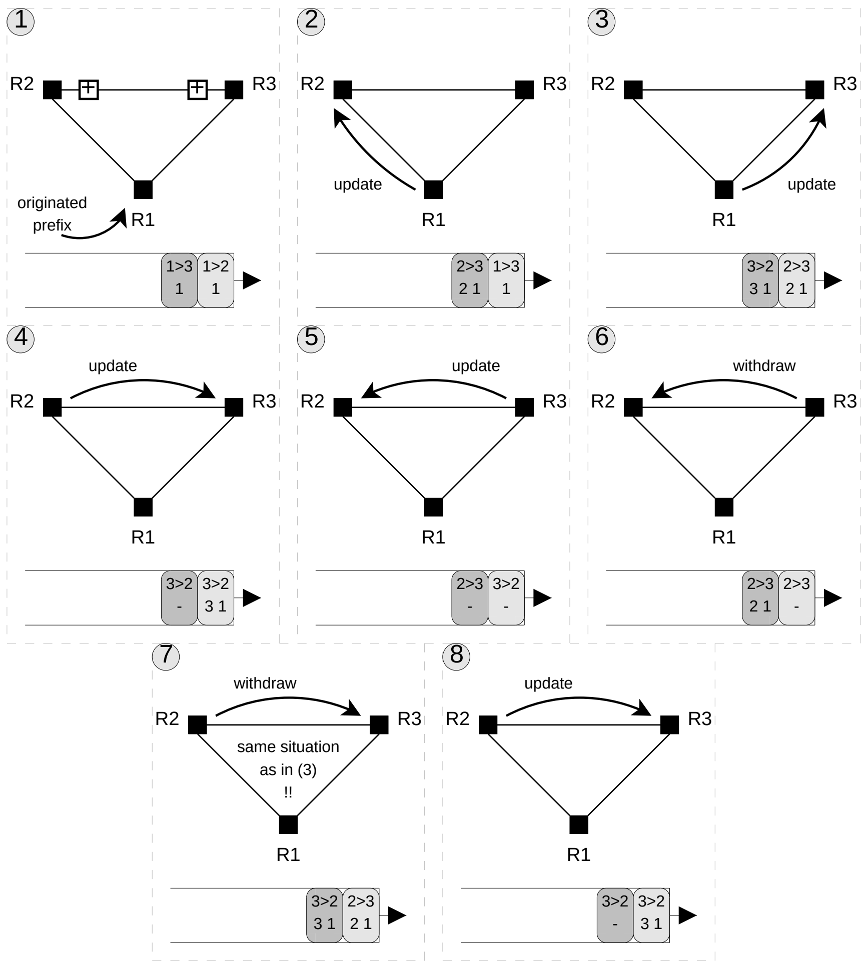


Figure 2.17: DISAGREE: divergence with C-BGP.

origination of a prefix by R1. This results in the addition of two messages in the FIFO queue, destined for R2 and R3. During step 2, the first message, destined to R2, is extracted from the queue and delivered to R2. Upon the reception of this message, R2 pushes a new message onto the queue, destined to R3. We do not describe the other steps, but we note that the state of the network as well as the content of the FIFO queue are equal at the end of step 3 and step 7. This means that the simulation will never converge.

The divergence of the above system when it is run with C-BGP is due to the fact that the message passing model in C-BGP does not consider the propagation delay along the links. The time to cross any link is equal. Hence, both R2 and R3 receive the initial UPDATE messages of R1 at consecutive steps. The same situation occurs for the UPDATE messages they propagate to each other. These UPDATE messages cause both R2 and R3 to select the route advertised by the other and subsequently withdraw the routes they have previously advertised. These WITHDRAW messages are also received at consecutive steps. Finally, both routers switch to their respective direct route and the cycle starts over.

To clarify the above experiment, let's take a modified version of the original DISAGREE system that we call DISAGREE2 (shown in Fig. 2.18). The BGP configuration of the DISAGREE and DISAGREE2 systems are equal. However, we have introduced two additional routers N1 and N2 between R1 and R2. These routers do not run BGP. Their purpose is only to introduce a delay in the propagation of the messages exchanged between R1 and R2. With this system, C-BGP will find one of the solutions of the system[7].

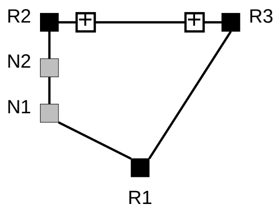


Figure 2.18: DISAGREE2: a configuration that admits multiple solutions.

The reason for the convergence in this case is that R2 receives the route towards the prefix from R3 before the routes from R1. When it later receives the route from R1, no new messages are generated since this new route is not preferred and the best route of R2 does not change. The whole convergence for DISAGREE2 is shown in Fig. 2.19.

We can conclude that in the case of a BGP system that admits more than one solution, **C-BGP might converge to one of the solutions** (example of DISAGREE2) or **it might diverge** as well (example of DISAGREE). It will depend on the path explored by C-BGP in the evaluation graph of the system [GW99]. For the DISAGREE system, C-BGP follows a valid activation sequence that leads to divergence. However, for a given system, C-BGP will always produce the same outcome: divergence or one particular solution. The outcome will not change from one run to another.

The DISAGREE situation can occur in the real world. However, we can consider that it will always converge because it is very unlikely that the BGP messages are propagated and processed with exactly the same sequence as shown in the example of Fig. 2.17. Though it is very likely to converge, this configuration is undesirable. The reason is that its outcome is unpredictable. For network operators that want to predict how their traffic will be forwarded, this situation has to be avoided. Determining if a BGP configuration admits a unique solution has

[7]Note that in reality, a single additional router would suffice to make C-BGP converge for this system because OPEN messages are first exchanged. The explanation with OPEN messages is similar but more complex since more messages and more steps are required.

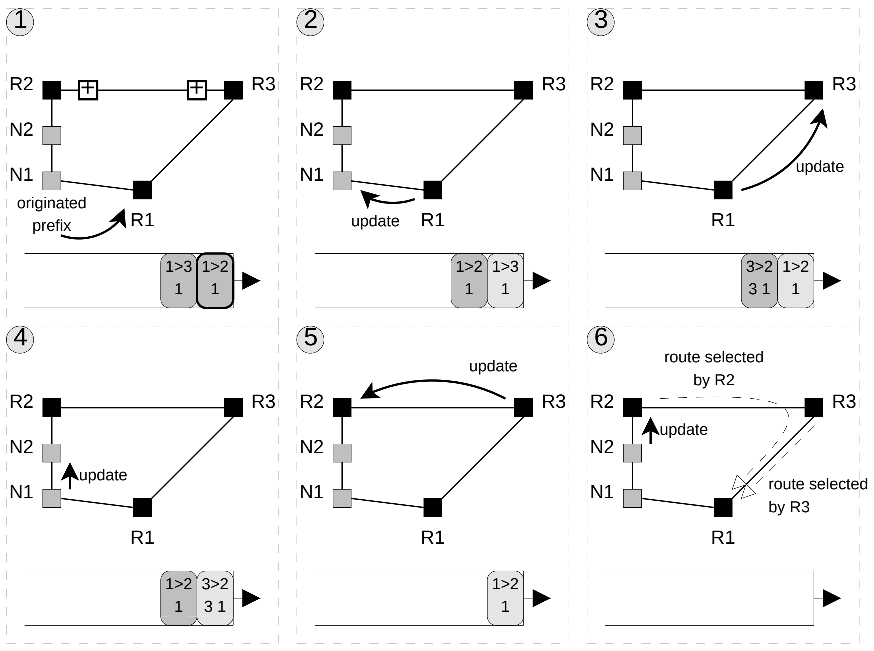


Figure 2.19: DISAGREE2: convergence with C-BGP.

been shown to be NP-hard [GW99]. In [NCM02], Nykvist et al have proposed a tool that allows to stochastically explore some of the possible solutions of a BGP configuration admitting more than a unique solution.

## BAD-GADGET

The second configuration described by [GW99] is named BAD-GADGET. It is depicted in Fig. 2.20. In this configuration, there are four routers: R0, R1, R2 and R3. R0 advertises one prefix. Each router that is not directly connected to the originated prefix prefers the route learned from its counter-clockwise neighbor. In this situation, the BGP protocol is unable to find a solution.

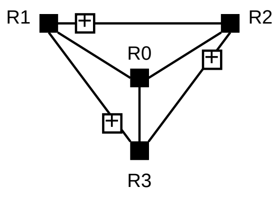


Figure 2.20: BAD-GADGET: a configuration without a solution.

We show in Fig. 2.21, the first steps of the simulation of the BAD-GADGET

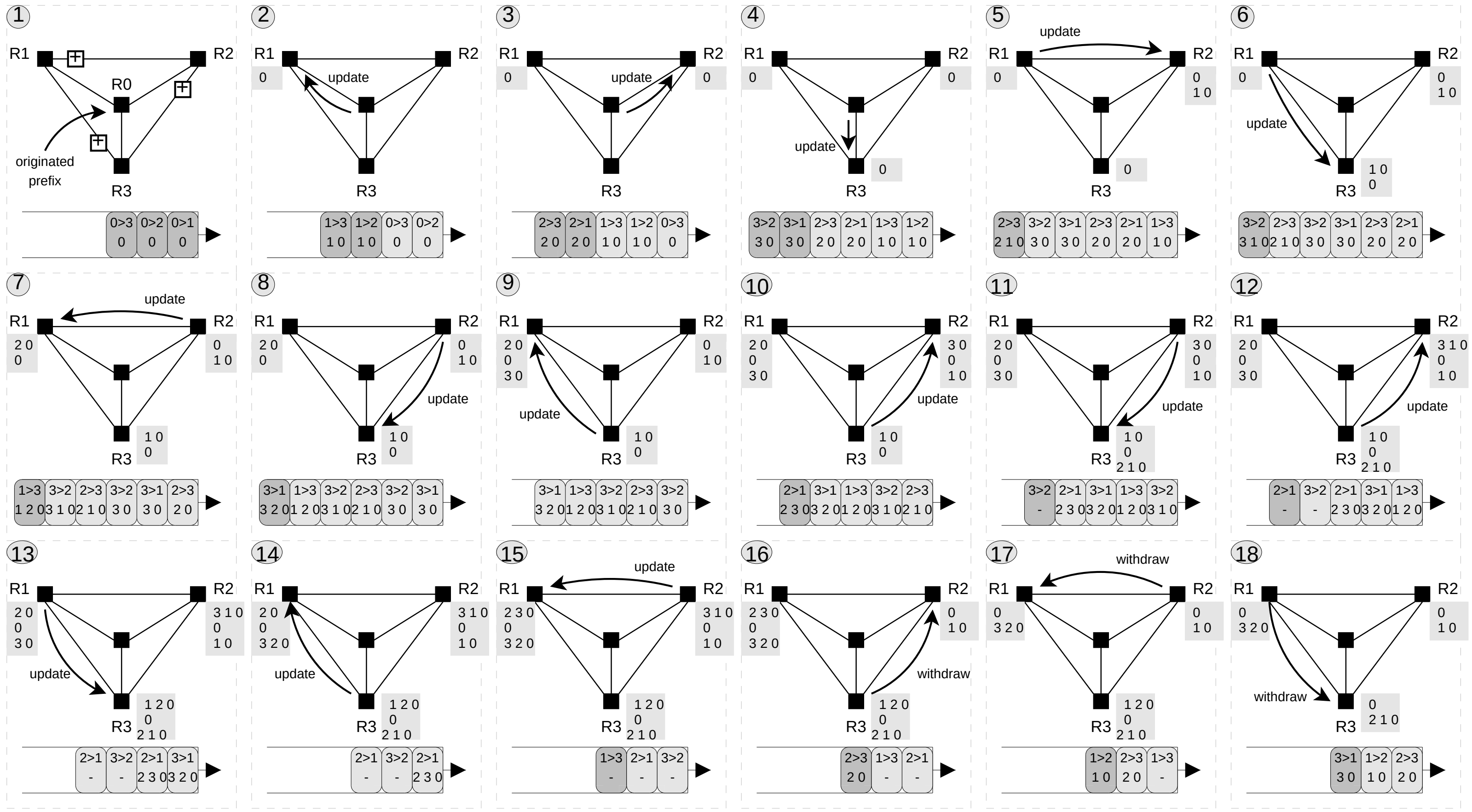


Figure 2.21: BAD-GADGET: divergence in C-BGP.

system with C-BGP. We use the same convention than for the DISAGREE and DISAGREE2 systems. However, in this example, we also show the content of the Loc-RIB and Adj-RIB-ins in the gray box near each router. The top route represents the current best route. We observe that each router will progressively select a route with an increasing AS-Path length, until it can not advertise its best route to its neighbors (since it would cause a routing loop). Withdraws are then exchanged and each router falls back to the direct route through R0. Then, the cycle starts over...

In the case of a BGP system that admits no solution (such as BAD-GADGET), **C-BGP will always diverge** because the corresponding evaluation graph does not contain a path leading to a stable state [GW99]. The behavior of C-BGP will not depend on the number of steps required to exchange a message over a BGP session. It will also be consistent across multiple runs.

Note that the BAD-GADGET configuration is a situation that must be avoided in real configurations since it will not reach a stable solution and cause an endless exchange of messages. The problem of detecting such misconfigurations in a network configuration is difficult. It has been shown that determining if a BGP system is solvable is NP-hard [GW99]. Other BGP configurations involving the MED attribute can also lead to oscillations as shown in [GW02b, GW02a].

### 2.5.8 Performance

In this section, we evaluate the performance of C-BGP. We rely on different topologies and we measure the execution time as well as the maximum memory consumption of C-BGP. For this purpose, we rely on the memtime utility [Ben02]. To measure the memory consumption of a process, memtime uses sampling and keeps the maximum amount of memory among the samples. The sampling rate is fixed to 100ms.

Our first performance evaluation relies on **synthetic AS-level topologies** generated by BRITE [MAMB01]. We use the Waxman [Wax88] and Barabasi-Albert [BA99] probability models. For the Waxman model, we rely on the default parameters $\alpha = 0.15$ and $\beta = 0.2$. The Barabasi-Albert model has no parameter. BRITE has an additional parameter $m$ which defines the number of links added per new node. We use $m = 2$ and $m = 4$. For each model and value of $m$, we generate topologies with the following numbers of nodes: 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000 and 20000. For each generated topology, we perform 10 simulations using C-BGP, where we inject a prefix at one single router. The prefix is originated at a different router each time. These simulations were performed on an Intel P4 ©CPU running at 2.66 MHz, with 512 MB of RAM.

We show the results in Fig. 2.22 for the execution time and in Fig. 2.23 for the memory consumption. A first observation is that for a given topology generation model and for the $m$ parameter fixed, the execution time grows linearly in function of the number of nodes. Note that in the topologies generated by BRITE, the number of edges is equal to $m$ times the number of nodes. So the simulation time

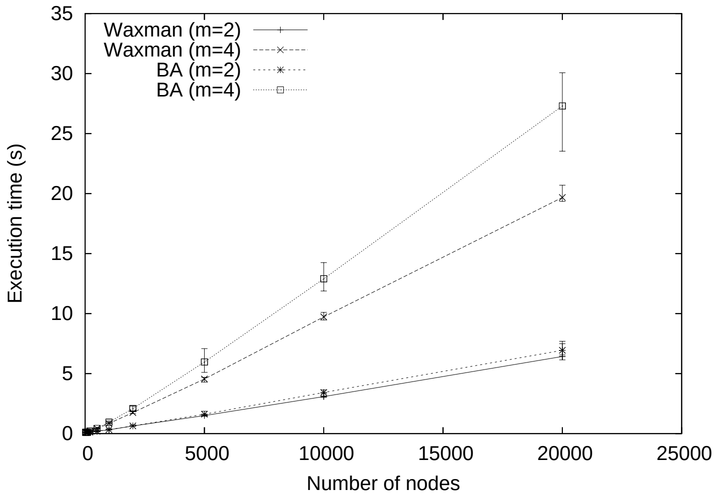


Figure 2.22: Execution time for Waxman and Barabasi-Albert topologies.

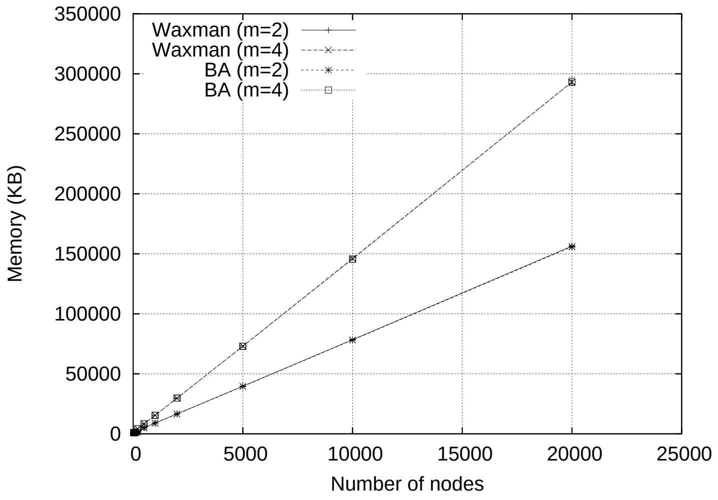


Figure 2.23: Memory consumption for Waxman and Barabasi-Albert topologies.

is also proportional to the number of edges. Secondly, the structure of the topology influences the execution time. For the same number of nodes/edges, the simulation time differs from one model to another. We looked at the number of decision process executions in each router for both models (see Fig. 2.24). We observe that for the same number of nodes and edges in the topology, the Barabasi-Albert has a significantly higher number of routers that require a larger number of decision process executions during the simulation. One possible explanation is that the routers with an higher node degree require more decision process executions. In the

Barabasi-Albert topology, for the same average node degree, there are more nodes with an higher degree. Finally, we observe that the origination point for a prefix has an impact on the simulation time. We show for each topology the minimum and maximum execution time. For the Waxman model, there is small variability. For the Barabasi-Albert model, the execution time varies more. For example, for the 20000 nodes topology with $m = 4$, the execution time is comprised between 23.53s and 30.07s.

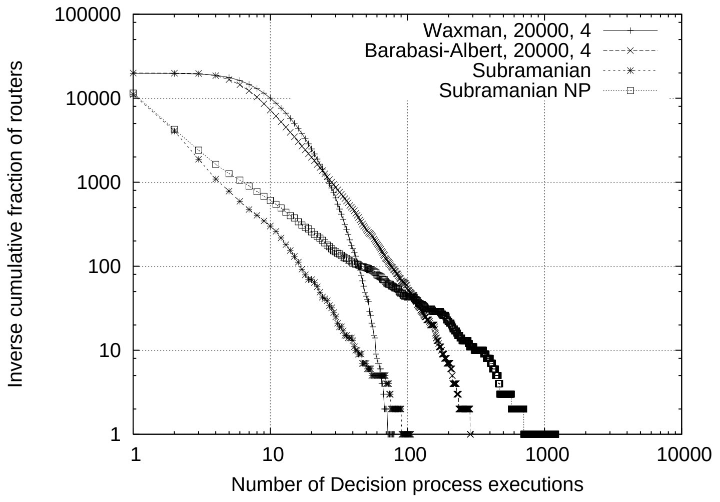


Figure 2.24: Number of BGP decision process executions per router.

On Fig. 2.23, we observe that the memory consumption is proportional to the number of nodes/edges. There is no difference between the topologies generated with the Waxman model or the Barabasi-Albert model for a given value of $m$ (the curves are mixed up on the figure). The memory consumption is mainly proportional to the number of BGP routes stored in the RIBs. In these simulations, no policies were used, hence a route is sent on most BGP sessions.

Our second performance evaluation relies on an **AS-level topology** inferred by Subramanian et al from multiple BGP routing tables collected from multiple vantage points [SARK02]. The topology we use is dated from the 10th of February 2004. It contains 16921 nodes and 37271 edges. The advantage of this topology is that it contains the business relationships between any two AS. We enforce the policies underlying these business relationships in C-BGP using route filters (see Section 2.5.4). We perform 100 simulations where a single prefix is originated from a different AS each time. In order to compare the performance of the simulator with and without policies, we also perform the same simulations without setting up the route filters.

We show the results of these simulations in Fig. 2.25. We distinguish the simulations with policies and without policies (NP). We also distinguish the setup time

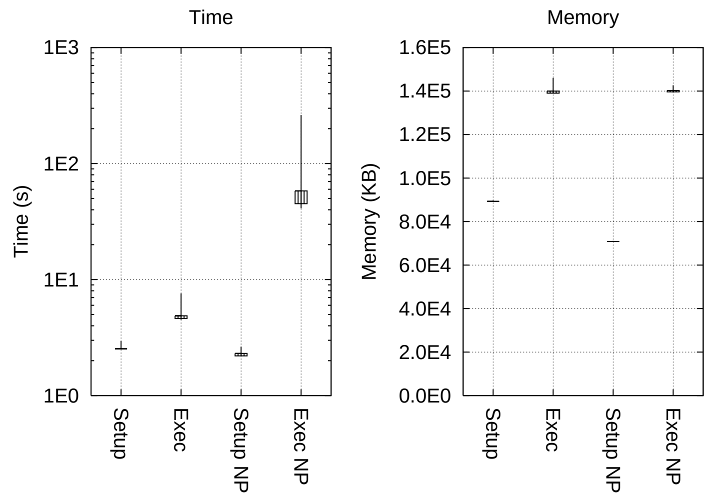


Figure 2.25: Execution time and memory consumption for Subramanian et al topology.

from the complete execution time. Note that the execution time includes the setup time. We can first observe that the setup time is slightly increased when route filters are added. There is one filter for each side of a BGP session, i.e. 74542 filters to setup. Second, the execution time is smaller when policies are enforced. This is due to the fact that route filters prevent some routes to be propagated. Fewer BGP messages are thus propagated. The average execution time is 5.2s for simulations performed with policies and 73.9s for those without policies. In addition to this, there is an high variability of the simulation time among the various simulations. This means that the origin of the prefix in the topology has an important impact on the simulation time, as we already noticed for the synthetic topologies. For the simulations without policies, the total execution time ranges from 41.14s. to 262.02s while the mean execution time is 73.9s.

Concerning the memory usage, we observe that the simulations with policies already consume more memory after the setup phase (89.3 MB on average) than the simulations without policies (70.1 MB on average). This is due to the 74542 route filters that are created. We can therefore estimate that a route filter consumes on the order of 257 bytes on average. To the contrary, the memory consumption after the complete execution does not differ much between the simulations with and without policies (140,312 KB and 140,306 KB respectively, on average). Again, the simulations with policies propagate less routes than simulations without policies since some routes are filtered. This accounts for a difference of memory usage on the order of 19 MB.

Finally, we evaluated the performance of C-BGP on cliques composed of $N$ routers. Cliques are the worst topologies for our performance measurements since their connectivity is the highest. Above the clique, we perform simulations with a **full-mesh of iBGP sessions** and with a **full-mesh of eBGP sessions without policies**. A full-mesh of iBGP session is the typical iBGP deployment in small to moderate size networks. A full-mesh of eBGP session is less frequent, but it can be found in the dense core of the Internet topology, between the tier-1 ASs. For each simulation scenario, we vary the number of routers, $n$, and the number of prefixes advertised per router, $p$. Each router originates $p$ different prefixes. We performed our measurements on an Intel P4 ®CPU running at 2.66 GHz, with 2 GB of RAM.

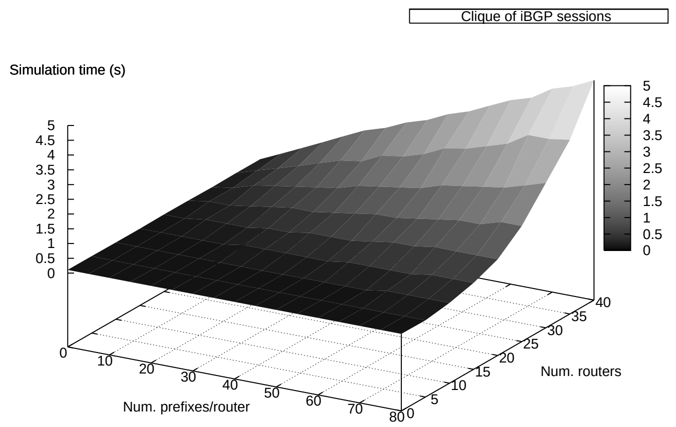


Figure 2.26: Simulation time for full-meshes of iBGP sessions.

We show the results of the iBGP full-mesh simulations in Fig. 2.26 and Fig. 2.27. We observe that both the simulation time and the memory consumption grow linearly with the number of prefixes originated ($p$). This was expected since this only increases the number of prefixes propagated and the number of entries in the RIBs. On the other side, these metrics increase quickly as a function of the number of routers. This was also expected since a router cannot propagate to an iBGP neighbor a route that it has received from another iBGP neighbor. Each router will only advertise its own prefixes to the $n-1$ other routers. The total number of advertisements for $n$ routers each advertising $p$ prefixes is therefore $p.n.(n-1)$. The execution time and the memory consumption should evolve as $O(n^2)$.

We show the results of the eBGP clique simulation in Fig. 2.28 and Fig. 2.29. As for the iBGP full-mesh, the simulation time and the memory consumption are linear functions of the number of originated prefixes $p$. To the opposite of the iBGP full-mesh, the routers can propagate over an eBGP session the routes they have received through another eBGP session. At the end of the convergence, each prefix originated by a router will be propagated to $n-1$ neighbors that will select this route as best since it has the shortest AS-Path length. These neighbors will subse-

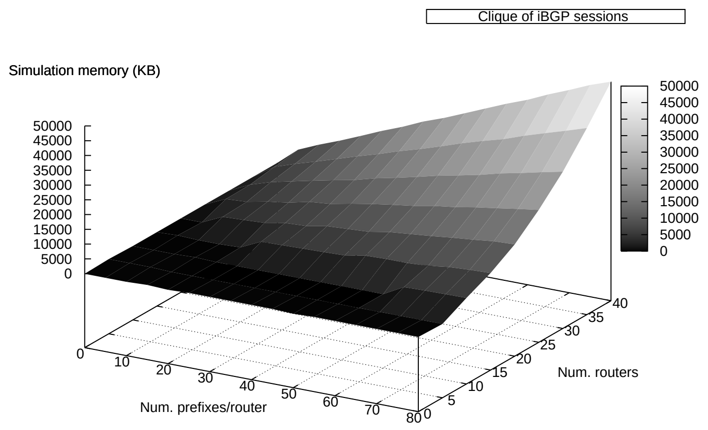


Figure 2.27: Memory consumption during the simulation of full-meshes of iBGP sessions.

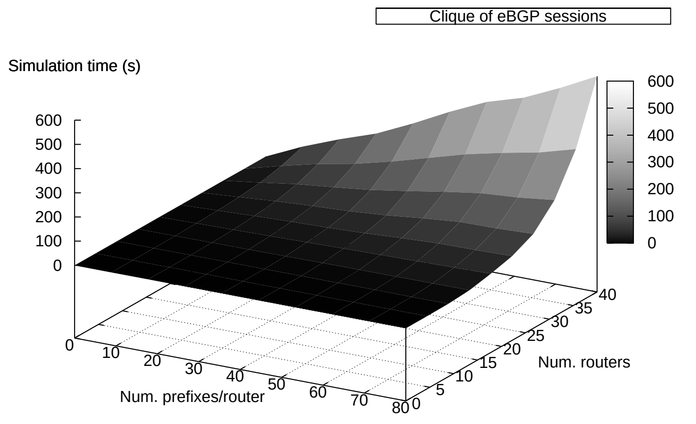


Figure 2.28: Simulation time for full-meshes of eBGP sessions.

quently propagate this route to their own $n-2$ other neighbors. These neighbors will not select this route as best and the propagation stops. The total number of advertisements is therefore $p.n.\left((n-1)+(n-1).(n-2)\right) = p.n.(n-1)^2$. The memory consumption and the execution time should therefore evolve as a cubic function of $n$.

Note that theoretically, path exploration can occur and a path-vector protocol such as BGP can explore as much as $O\left((n-1)!\right)$ alternative paths in a clique

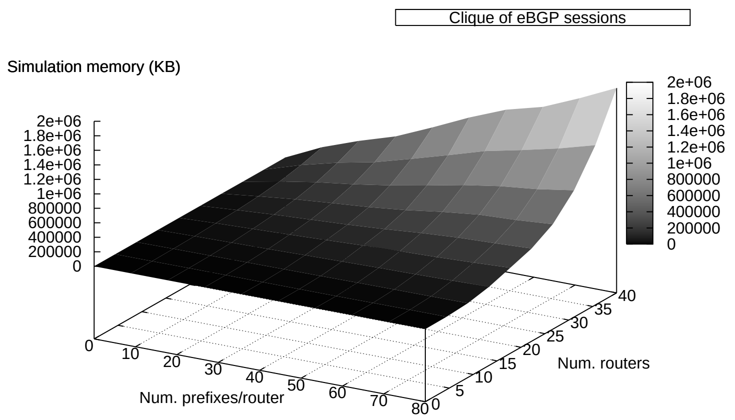


Figure 2.29: Memory consumption during the simulation of full-meshes of eBGP sessions.

composed of $n$ routers before converging [LABJ00, PAMZ04]. This occurs when routers receive several routes to a destination before receiving their final best one. In the case of our simulations, there is no path exploration for two reasons. First, no policies are applied. Second, due to the message passing model of C-BGP, the routes are forwarded along each link as if the links have the same propagation delay. This means that the first route received by each router is the best one.

### 2.5.9 Scalability

One of the most important resources consumed by BGP simulations is the memory. Indeed, a single BGP routing table may contain up to hundreds of thousands of routes [Hus05]. Each BGP route is stored in memory with a lot of attributes: destination prefix, Local-Pref, MED, next-hop, AS-Path, Communities, Originator-ID and Cluster-ID-List. Some of these attributes like the AS-Path or the Communities can require a lot of space to be stored depending on how many ASs have been crossed by the route and how many times it has been tagged with a Community value.

Storing these routes in a router requires a large amount of memory, on the order of tenths of megabytes. The memory consumption of a single router depends on different factors. First, the number of peerings determines how many different routes towards a given prefix could be received by the router. Second, the number of prefixes originated in the simulation determines how many different entries will appear in a routing table. The number of destination prefixes appearing in a full BGP table was above 176.000 [Hus06] at the time of this writing. If $m$ is the number of peers of one router and $p$ is the number of prefixes originated in the simulation, then the upper bound on the number of entries in its routing table (including Loc-RIB and Adj-RIB-in/out) will be on the order of $O(p.(m-1).2)$. This can become quite important when the size of the topology increases. Especially inside large domains, when a full-mesh of iBGP sessions is used, a router will peer with all the other BGP routers in the domain. In a domain containing $n$ BGP routers, the number of internal BGP sessions of one router will therefore be $n-1$, each session potentially carrying a full BGP table, leading to a worst case of $2.(n-1).p$ routes to be stored in a single router. Fortunately, all iBGP sessions will usually not carry a full BGP table.

In this section, we characterize BGP routes collected in real world BGP routers. We observe that they contain a lot of redundancy. Based on this observation, we discuss the possible reductions of the memory footprint of C-BGP. The scalability improvements presented hereafter still make possible to mine the routing data efficiently. The storage of routing information in the solver is based on a trade-off between an optimal data structure to handle BGP routes and an optimal data structure that allows fast lookups.

#### AS-Path and Communities redundancy

We analyzed real BGP routing tables from Abilene, GEANT, RouteViews and RIPE and observed that many routes share the same AS-Path. Fig. 2.30 and Fig. 2.31 show the frequency of AS-Paths in routing tables collected in GEANT routers and collected by the RIPE NCC project[8]. The figures show that nearly half the AS-Paths found in a routing table are present in at least 2 different routes. Moreover, AS-Paths that appear in more than 5 different routes are frequent. The figures also show that for a single peer, there are about 25.000 different AS-Paths in the RIPE routing tables. This number is closer to 20.000 routes in the GEANT routing tables. The maximum frequency shown below each plot indicates how many time the most frequent AS-Path was used. It is around 1400 for both GEANT and RIPE RIBs.

The above observation suggests that it is possible to exploit the redundancy in routing tables to reduce the memory footprint of C-BGP. One simple technique would be to keep a single instance of each different AS-Path in memory. All the routes sharing the same AS-Path would then have a reference to the instance of the AS-Path. This can easily be done by storing the AS-Paths in an hash table. In addition, each route has a pointer to the AS-Path stored in the hash table. This is not a new technique: BGP implementations such as GNU Zebra and Quagga [Ish96, Qua03] or commercial routers [ZB03] already encode their routing tables this way. In contrast with a Zebra/Quagga BGP daemon, in C-BGP, we can share the global AS-Path repository among multiple routers, further taking on the redundancy. For example, in a single domain configured as a full-mesh of iBGP sessions, there will be an high AS-Paths redundancy since this attribute is not changed when

[8]We show an analysis of additional routing tables in Appendix A.

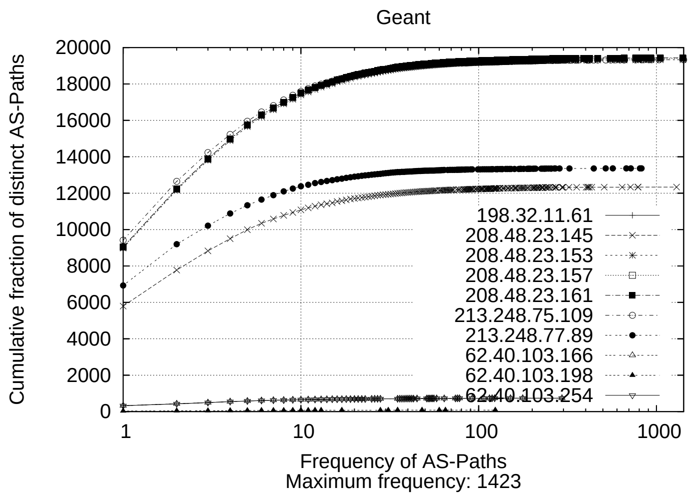


Figure 2.30: Frequency of AS-Paths in GEANT routing tables.

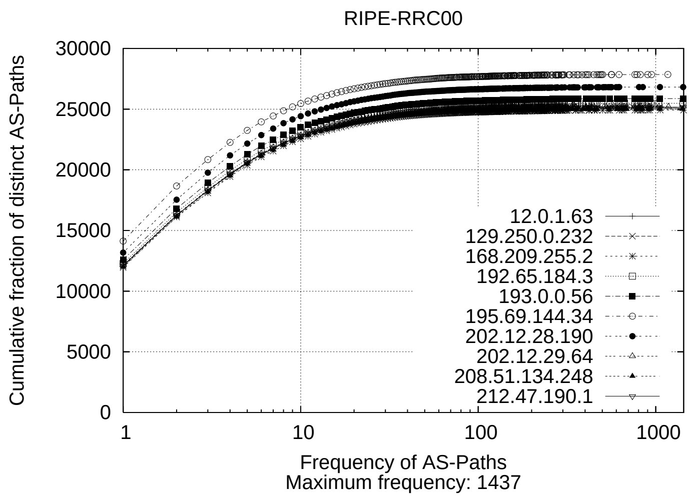


Figure 2.31: Frequency of AS-Paths in RIPE routing tables.

propagated over iBGP sessions. A single route received from an eBGP peer and selected as best by the domain's border router will be redistributed with the same AS-Path to the $n-1$ iBGP neighbors. Encoding of BGP routing tables is implemented using this technique in the latest versions of C-BGP.

We performed the same analysis of routing tables and looked at the frequency of occurrence of the Communities value. We took the Communities as they appeared in the routing tables, without taking into account the fact that Communities with the same values in different orderings should be considered as equal. We show our results in Fig. 2.32 and Fig. 2.33. The situation is even better here than with AS-Paths since the redundancy is higher. We found that more than two thirds of the Communities are used in at least two routes. The maximum frequency is also far higher than for AS-Paths. For Communities, it ranges from 114,905 in GEANT RIBs to 153,387 in RIPE RIBs.

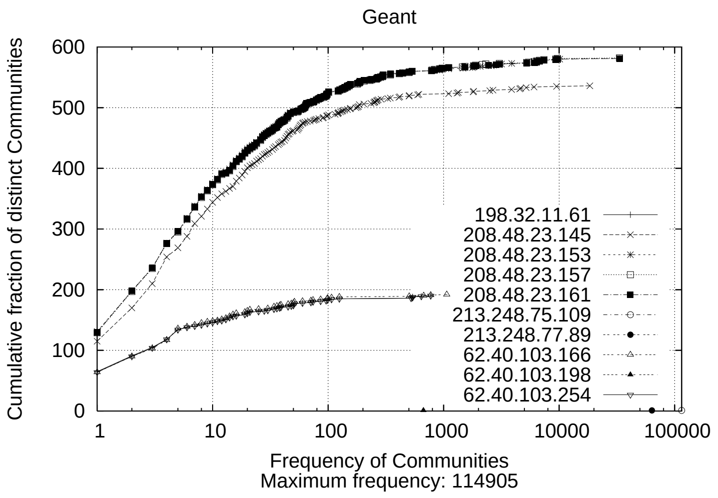


Figure 2.32: Frequency of Communities in GEANT routing tables.

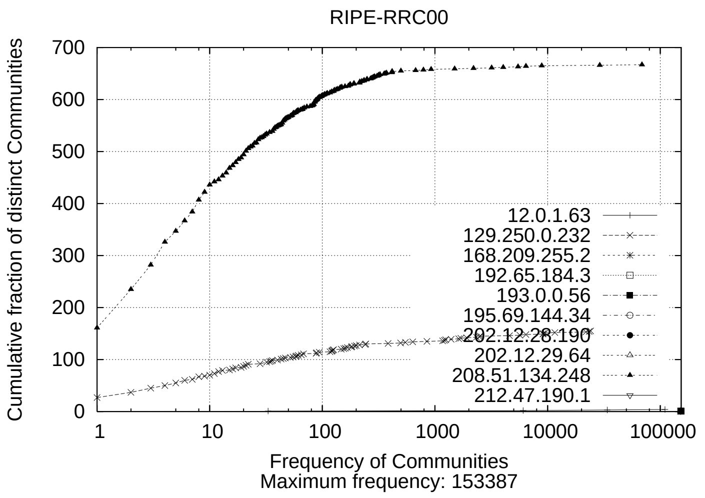


Figure 2.33: Frequency of Communities in RIPE routing tables.

The same technique as for AS-Paths can therefore be used to reduce the memory consumption due to the Communities attribute. The same observation as for

AS-Paths can also be done for routes redistributed over iBGP sessions since the Communities attribute will seldom be changed inside the iBGP. At least changing the BGP attributes of routes propagated over iBGP sessions is not a recommended practice.

#### Further memory reduction

In addition to the redundancy of attributes in a single RIB or across multiple RIBs, there is also an high redundancy of routes across multiple RIBs. In the real world, each BGP router keeps its best routes in its own RIB. In addition to this, each router also keeps the routes it has sent to its neighbors in its Adj-RIB-outs. The neighbors keep the same routes in their Adj-RIB-ins. In such a situation, the routes are duplicated in C-BGP. In a full-mesh of iBGP sessions, the duplication is even worse since the same route will be copied $n-1$ times in the Adj-RIB-outs and $n-1$ times in the Ad-RIB-in of the iBGP peers. In genuine BGP router implementations this problem is solved with two different approaches.

In CISCO IOS for instance, the notion of peer groups [Sys05b] has been introduced. The aim of a peer group is to reduce the amount of system resources used in the generation of an UPDATE message. It also allows to share a single Adj-RIB-out among all the neighbors belonging to the same peer group. In Zebra and Quagga, peer-groups are also supported but they need not be specified in order to benefit from a reduced memory footprint. In a similar way to AS-Paths and Communities, Zebra and Quagga route attributes are also stored in an hash table allowing equal attributes to be shared among multiple routes.

In the routing solver, we can go further since route attributes can be shared among multiple routers allowing multiple routers to share equal route attributes. This is especially efficient in the case of the iBGP full-meshes where attributes are seldom changed over iBGP sessions. This technique is not yet implemented in C-BGP.

Another scalability improvement that might prove useful has been proposed in [HK03] by Hao and Kopol. It consists in storing the AS-Path of the routes computed during a simulation in a global RIB. This global RIB would be composed of multiple trees rooted at the origination router of each prefix. Hao and Kopol did not evaluate this technique.

## 2.6 Conclusion

In this chapter, we have first setup the requirements for a model of BGP suitable to evaluate the routing decisions taken by BGP in large topologies. We surveyed the tools currently available to simulate such BGP routing decisions. We explained that BGP daemons such as Zebra or Quagga, though they are the most accurate and versatile open-source implementations of BGP, require too much time for the computation of routes. On the side of packet-level simulators, the most mature implementation of BGP comes with the SSFNet simulator but it does not support the complete BGP decision process. In addition, packet-level simulators currently require too much computer resources to be used on large topologies. We concluded that at the time of this writing, no suitable tool was available.

We have therefore introduced our own model, that we call a *BGP routing solver*. Its purpose is the computation of the outcome of the BGP decision process in multiple routers. The underlying hypothesis of this BGP routing solver are as follows. Firstly, we do not model the time in our solver since we are not interested in the transient states of routing convergence but only in its outcome. Secondly, we do not model the various timers that are used in BGP routers. Thirdly, we do not model the TCP protocol used to support the BGP sessions. Instead of this, we rely on a single global linear queue to guarantee the message ordering. Finally, we use a steady-state model of an IGP protocol to compute the paths that underly the BGP sessions. On top of this model, we reproduce accurately how the BGP UPDATE and WITHDRAW messages are exchanged between routers. We also use a complete implementation of the BGP decision process to select one best route per destination. Finally, our routing solver supports versatile input and output filters that can be configured on a per-session basis. This makes possible the definition of complex routing policies such as in real world networks. We have developed C-BGP, an implementation of our BGP routing solver that we made publicly available [Quo03b].

We have validated C-BGP by performing basic behavior tests such as the exchange, the propagation and the selection of routes. We also tested more advanced behaviors such as the convergence in an environment composed of several route-reflectors forming a complex hierarchical iBGP configuration. In addition to this, we have studied the behavior of C-BGP in presence of naughty BGP configurations. We have shown that due to its deterministic exploration of the evaluation graph, C-BGP behaves consistently from one run to another which might not be the case in packet-level simulators. The determinism of C-BGP is an advantage for evaluating routing what-if scenarios and traffic engineering techniques since it allows to perform reproducible experiments.

Finally, we evaluated the performance of C-BGP on several synthetic topologies. We measured both the execution time of the simulations and the maximum memory consumed during their execution. We observed that the main limitation of C-BGP is its memory consumption. We are still improving C-BGP in order to enhance its scalability. We discussed how the memory consumption of C-BGP can be reduced by taking on the redundancy in a single BGP RIB and across multiple RIBs. Some of the techniques discussed are already implemented in C-BGP at the time of this writing.

We conclude that simulating BGP on large scale topologies is feasible with C-BGP at a reasonable cost on a single PC or a small cluster[9]. We apply C-BGP

[9]Distributing a C-BGP simulation on multiple CPUs is possible using a simple data-decomposition pattern [MSM05]. If no prefix aggregation is done, it is possible to allocate to each

to the modeling of a single domain in Chapter 3 and to large scale topologies in Chapters 4 and 5.

CPU a subset of the destination prefixes. Then, each CPU computes the BGP routes towards its prefixes only.

# Chapter 3

# Modeling the routing of an ISP

## 3.1 Introduction

The main service provided by Internet Service Providers (ISPs) is best-effort. Today, customers are asking for a better than best-effort service as well as guaranteed performance and reliability. New widely deployed services such as VPN or VoIP require increased performance guarantees from the network. For this reason, ISPs are very sensitive to the resilience and performance of their network. They try to provide quality assurance to their customers through Service Level Agreements (SLAs) [FE04]. Therefore, ISPs seek to build networks that will accommodate varying traffic loads and be robust to link and router failures. To satisfy the tight constraints of the SLAs, ISPs engineer their network to ensure the best performances, like minimizing the delay across the network or preventing congestion to occur on access links.

Today, the complexity of ISPs' networks make it difficult to investigate the implications of internal or external changes on the distribution of the traffic across their network. Several studies have recently examined the interaction between the IGP and BGP protocols. Based on the AT&T network, Teixeira et al have shown [TSGR04] that IGP changes in IP networks could cause shifts in the traffic and externally-visible BGP updates. Such BGP updates can lag more than 60s after the corresponding IGP event, causing extra-delay in the forwarding-plane convergence. In [ACBD04], Agarwal et al studied the impact of BGP updates on the intradomain traffic matrix of Sprint. They concluded that although a lot of BGP updates were received, the traffic matrix remained stable. This was predicted by Uhlig et al [UMB$^+$04] who showed that the routes towards the majority of Internet destinations are stable. In [ANB05], Agarwal et al examined how hot-potato influences the choice of IGP metrics in the European part of Sprint and concluded that hot-potato could lead to sub-optimality by as much as 20% of link utilization. In [TAR05], Teixeira et al present a consolidated view of the results of several measurement studies performed in AT&T and Sprint. They conclude that building a model of a large tier-1 network is difficult and advocate the development of a model of the interplay between the network topology, the routing protocols, the traffic and the network events.

No modeling tool currently fully captures both the diversity of the routes announced by the neighboring domains nor the details of the routing configuration inside an AS. In [FGL+00], Feldmann et al described Netscope, a tool to study and visualize the flow of traffic in a backbone network. Netscope focuses on intradomain only[1]. In [FWR04], Feamster et al described a BGP emulator that allows to compute the BGP routing decisions in a single AS. Unfortunately, this tool is proprietary and not publicly available. Recently, Teixeira et al proposed a model of the sensitivity of hot-potato changes on the BGP route selection process [TGVS04]. The analytical formulation proposed in [TGVS04] however reproduces neither the full BGP decision process nor the complexity of the working of BGP inside an AS [HP00]. This analytical model is not available as a tool that can be used by network operators.

On the side of commercial products, it is unclear whether tools really take into account the BGP information and how they do it. The Cariden MATE framework [Tec05a] is a capacity planning and traffic engineering product. It recently started to support interdomain routing, but this is not documented yet. The BGP support in WANDL IP/MPLSView [Lab05] is currently limited to simple BGP configuration checks. Product informations from OPNet about their SPGuru tool [Tec05b] claim that BGP is taken into account in some way. We were unable to get more precisions from OPNet.

As we will show in this chapter, understanding the routing of large ASs not only requires to model the routing inside the AS, but more importantly taking into account the routing information received from neighbor ASs. We describe in Section 3.2 how to model the routing of an ISP network. We explain which information is required in order to build such a model and why it is so complex. We apply our methodology on a transit network and provide two applications of our model to study the behavior of this AS in Section 3.5. The first scenario studies the impact of changes in the Internet connectivity of the transit network. The second investigates the impact of single internal link and router failures. Finally, we conclude in Section 3.6.

## 3.2 Modeling an Autonomous System

In this section, we describe how to build a model of an ISP network that is suitable for the routing solver described in Chapter 2. We show that building such a model is a task that includes several aspects, starting from understanding the AS's architecture, gathering network data, building a representation of the AS's network and ending up with a model suitable for the routing solver.

[1]Netscope has a support for external destinations reachable through multiple egress points. It does not support BGP though.

In order to model the routing of an ISP, we need to build a model that resembles the example of Fig. 3.1. First, we need to model the network topology. In the example, the topology is composed of routers R1 to R6 and the links between them. Since we are interested in interdomain routing, we also need to include in the topology the interdomain links that connect the border routers of the ISP to the border routers of other ISPs. In our example, the model contains 7 interdomain links that connect the ISP to 5 neighbors. Then, we need to obtain the configuration of the intradomain routing protocol to be able to compute the intradomain routes of each router. We also need to define the BGP sessions between the routers. This includes the iBGP sessions between the routers of the modeled ISP. These iBGP sessions may be organized in an hierarchy using route-reflectors. This also includes the eBGP sessions between the border routers and the external peers. In particular, these eBGP sessions will allow us to feed the model with the interdomain routes learned by the ISP. Finally, if we want to study the impact of routing changes on the traffic, we need to include in the model "flows" of traffic entering at one ingress point and exiting at a computed egress point.

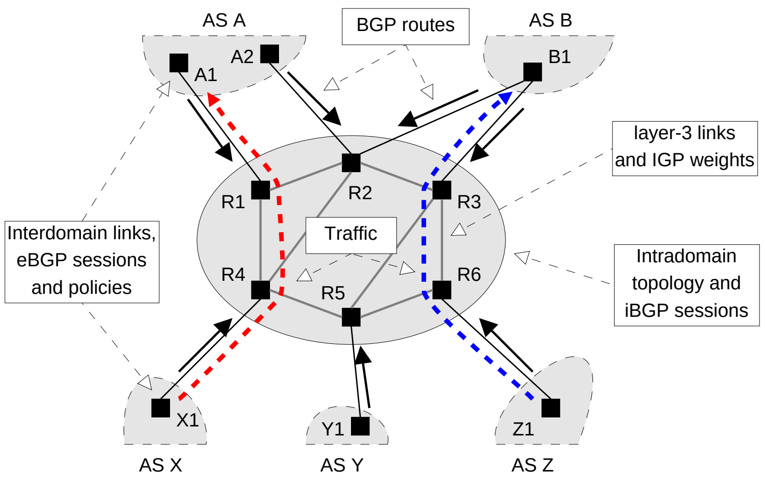


Figure 3.1: Overview of the model of an ISP.

We discuss in the following sections how the information required to build this model can be obtained. We also explain what are the current technical challenges in gathering these data.

### 3.2.1 Gathering routers' configuration

The first part towards building an AS's model consists in retrieving its configuration. The configuration of the network is spread all over its routers. The configuration of a router includes the following elements which are relevant for a routing model. First, it contains the **mapping between the physical links and the layer-**

**three links**. Note that IP tunnels are also important in practice and they can be seen as additional layer-three links. Each physical interface (or group of physical interfaces in the case of link bundles) will receive one or more IP addresses identifying the link endpoint. Second, the router configuration contains the **IGP metric** associated with a layer-three link. If the IGP is hierarchical, the configuration of a router also contains the definition of the areas to which it belongs. Finally, the router configuration contains the definition of the **BGP sessions** the router has with its neighbors. For each BGP session, the configuration indicates whether it is internal or external. If the router is a route-reflector, the configuration will also indicate for which session reflection is allowed. This configuration usually also includes the **BGP policies** that are enforced on each session. The configuration of the routers also contains many other parameters that we do not discuss here since they are not required to build our model.

Handling the routers' configuration in a large network is difficult. First, in a large IP network, the volume of information found in the routers' configurations is far too large for a human to be able to deal with it manually. Second, the configurations of the routers vary and there are frequently inconsistencies between different routers in the same network [FR01, QN04, FB05]. For instance, it is not rare to find half-defined BGP sessions, that is the session is defined in the configuration of one router but not in the configuration of its peer. Finally, many networks use heterogeneous equipment. The configurations are thus written in different configuration languages. Sometimes, some options even depend on the version of the network equipment's operating system. There is therefore a need to automate the process of analyzing the network configuration and to properly report inconsistencies.

Most of the time, discussion with the operator as well as cross-checking the files will be required in order to exploit the network configuration.

### 3.2.2 Representing the topology

The second step of the model construction consists in building a model of the network topology. As explained in Chapter 2, we do not include the physical/facility-level details since we do not need them in order to be able to accurately model how the route selection is performed in the AS. We need to build a graph of IP routers and layer-three links. To build this topology, we can rely on the configuration of the routers as described above. Another possibility is to dump the topology database built by one router[2]. Indeed, since with link-state routing protocols such as OSPF or IS-IS, each router in the domain builds its own database of the adjacencies between routers, it is possible to rely on the database of adjacencies built by a single router. In practice, a workstation running an IGP daemon such as Quagga [Qua03] will be used for this purpose and will have an adjacency with a router in the domain. This method requires to configure a new adjacency in a router of the network. In the case of an hierarchical IGP, a capture of the topology database

[2]This can be done using the `show isis database verbose` for IS-IS in CISCO routers.

in each area is needed. Some network operators prefer to rely on wiretapping the IGP on an Ethernet link [Bon05] since it requires less configuration efforts. This method has the inconvenient of missing the history of past LSPs and it may take some time to have a correct view of the topology.

One difficulty that can be encountered in this part is mapping the nodes and edges of the model with the real networking equipment. Various IP addresses might be used to identify various parts of the equipment. Routers might have multiple IP addresses corresponding to different physical interfaces and different loopback addresses. In certain configurations, the IP address used to identify the router in the IGP differs from the IP address that identifies the router in the BGP protocol. One solution to this consists in mapping all the addresses of one router to a single IP address [Tan06]. This must be done carefully since routing protocols may take decisions based on this address. This is the case for BGP, for instance, where the IP addresses of the routers may be used to break the ties at the last step of the decision process.

### 3.2.3 Routing data

The third part of the modeling of an AS consists in feeding it with routing data. Concerning the intradomain routing, the routes may be computed based on the adjacencies found in the graph of the model and on the IGP metric of the existing edges. For the interdomain routing, additional information needs to be inserted in the model. The AS's routers will perform their route selection based on the external routes received through BGP by the border routers. These routes have to be captured and to be injected into the AS's model. To be able to perform very accurate predictions with the model, **all the eBGP routes learned by the AS should be collected**.

Collecting all the BGP routes learned by an AS can be an operational problem. Technically speaking, it is possible to capture all the BGP routes that are received on the peering links of the AS. It is also possible to log in on all the border routers and ask them to dump all their eBGP routes. In practice however, due to current limitations in the routers' software and to the reluctance of operators to perform these operations on the production routers, collecting the eBGP routes is not that simple. The technique used to collect the BGP data depends on the AS's network, but one common technique is to rely on a dedicated workstation running a software implementation of BGP such as Quagga [Qua03] that has passive BGP sessions with the BGP routers of the AS (see Fig. 3.2). In a small or medium size network with an iBGP full-mesh, all routers will have an iBGP session with the workstation and all the best routes of the routers will be learned. In large networks, each router cannot maintain an iBGP session with the route collector and only a subset of the routes will be collected. Typically, large networks rely on an iBGP hierarchy and routes will be collected on important route reflectors.

It is important to notice that using a subset of the BGP routes may lead to inaccuracies in the model since possible egresses for some destination prefixes

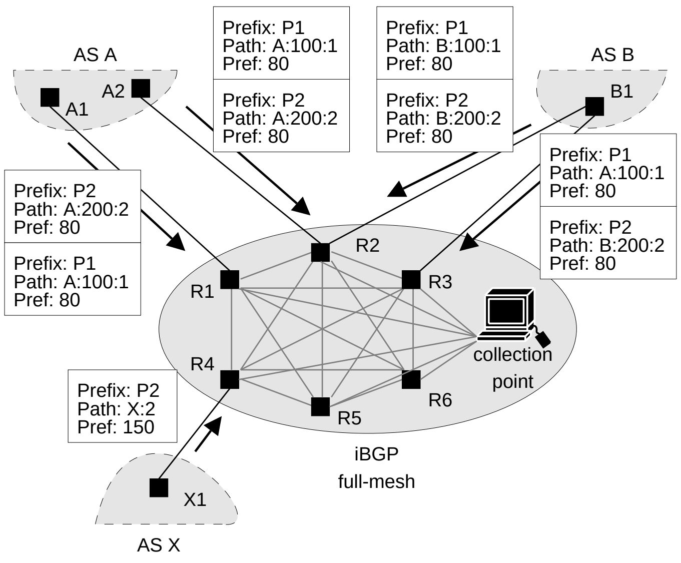


Figure 3.2: Limited view on eBGP routes.

will be unknown. For example, let's have a look at Fig. 3.2. We first consider the case of the routes learned towards $P1$. Two routes are received by $R1$ and $R2$ from $ASA$. Two routes are also received by $R2$ and $R3$ from $ASB$. Each border router ($R1$, $R2$ and $R3$) **propagates a single best route in the iBGP**. $R2$ has received two eBGP routes and advertises a single one in the iBGP. Which one is advertised depends on the outcome of the decision process in $R2$. A second possible case concerns $P2$. Routes towards this prefix are received from $ASA$, $ASB$ and $ASX$. But the preference of $150$ given to the route received from $ASX$ is higher than the preference given to routes received from $ASA$ and $ASB$ (80). This means that $R4$ propagates this route in the iBGP and it is subsequently selected as best by $R1$, $R2$ and $R3$ as well. The consequence is that $R1$, $R2$ and $R3$ do not advertise their eBGP routes towards $P2$ in the iBGP and the collection point is thus not aware of them.

New approaches such as the BGP monitoring Protocol (BMP) [Scu05] are currently discussed at the IETF. BMP allows an easier access to the BGP routes known by a router (including the Adj-RIB-in). Juniper routers also have a hidden feature that allows them to advertise in the iBGP the best route learned over eBGP sessions if their current best route was learned from the iBGP [3].

[3]The magic JunOS command to enable the advertisement of the external best route is `bgp advertise-best-external-to-internal`

### 3.2.4 Traffic statistics

The fourth part required to model an AS concerns the traffic. For an AS, the traffic information raises serious problems [VE04]. In an intradomain model of the AS, only the router-router traffic matrix needs to be considered. This level of detail is sufficient since changes in the intradomain routing will only change the paths from router to router, not the volume of data sent from one router to another. In this case, one can rely on SNMP measurements on the external interfaces of the AS and use techniques such as tomo-gravity [ZRDG03] to infer a router-router traffic matrix. The accuracy of these techniques is questionable [GJT04]. Another technique that can be used in ASs where MPLS is deployed is to collect per-LSP statistics.

When considering a model of an AS that provides transit service, **the router-router matrix is not sufficient**. **One must consider the prefix-prefix matrix** since (1) the egress router selected by an ingress router to reach a destination prefix may change and [TGVS04] (2) the ingress router where the traffic from a prefix was received may also change. The techniques described above are not applicable here since they do not provide information on the sources and destinations of the traffic flows.

One solution is to rely on Netflow statistics [SF02] collected on the border routers. Collecting such statistics is still an operational issue today [VE04]. The problems faced by network operators are the following. First, the size of a prefix-prefix matrix is significantly larger than a router-router matrix. The number of source and destination prefixes is on the order of 180,000 [Hus06]. Second, activating Netflow can put an important burden on the border routers. Finally, setting up such a measurement infrastructure requires a significant investment in configuration time and equipment. Consequently, Netflow will usually only be activated on the peering interfaces that carry a significant fraction of the traffic. In addition, Netflow sampling [CB05] is also used in order to decrease the volume of the collected statistics.

## 3.3 The GÉANT Model

In this section, we describe how we have used our routing solver to model the routing of a transit network. We use the GÉANT network as a case study. GÉANT is the pan-European research network and it is operated by Dante. It carries research traffic from the European National Research and Education Networks (NRENs) connecting universities and research institutions. GÉANT has PoPs in all the European countries. All the routers of GÉANT are border routers. Fig. 3.3 shows an overview of the GÉANT backbone.

### 3.3.1 Topology

We obtained the layer-three topology of GÉANT from a one-day IS-IS trace captured on November 24th, 2004. We cross-checked the obtained topology with a

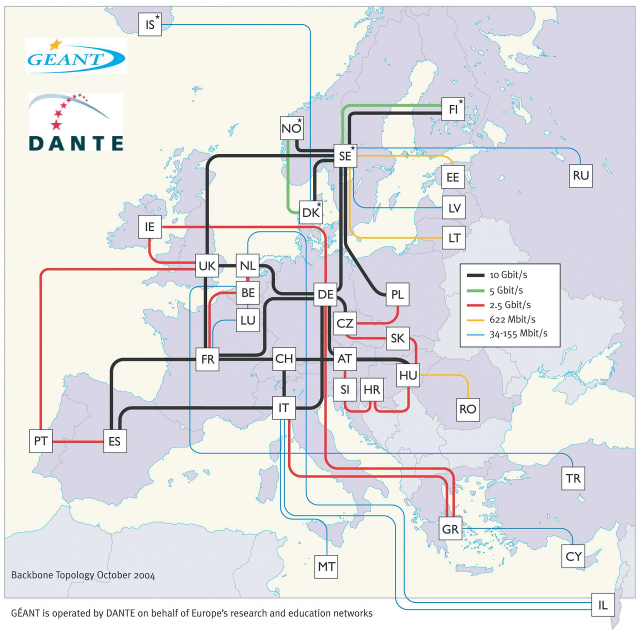


Figure 3.3: Overview map of GÉANT (source: http://www.geant.net).

map of the network provided by Dante. We model GÉANT with a graph composed of 23 routers, 38 core links and 53 edge links.

## 3.3.2 Routing data

In GÉANT, the BGP routes were collected using a dedicated workstation running GNU Zebra, a software implementation of BGP. The workstation has an iBGP session with 22 of the 23 border routers of the network. Using this technique, it is possible to collect all the best BGP routes selected by the border routers of the AS. We used a snapshot collected on November 24th, 2004 and obtained the 640,897 BGP routes propagated in the iBGP. Thus, we know all the best routes currently selected by each router of GÉANT. This is a subset of all the eBGP routes learned by GÉANT on its 53 peerings since we do not know the eBGP routes currently not selected as best. Having only the eBGP routes currently selected as best may affect the results of the experiments in the case where an unknown eBGP route would have been selected instead of the current best one. Note that collecting all the eBGP routes received by GÉANT would have required the capture of BGP messages received on the 53 peering links.

The computational complexity of the routes computation is directly proportional to the number of prefixes in the routing tables. A full BGP routing table can contain more than 180.000 prefixes. However, when considering the routes announced by all the neighbors of one domain, it appears that a lot of prefixes are learned from the same neighbors, with the same BGP quality (Local-Pref, length of the AS-Path, MED, next-hop) [HP00]. The outcome of the decision process in the whole network will be the same for these prefixes. For example, on Fig 3.4, the network receives eBGP routes towards 2 different prefixes: $P1$ and $P2$. These prefixes are learned from the same 3 neighbors and for each neighbor, the BGP quality is the same for both prefixes. For $P1$, the AS-Path received from $A1$ is `A:100:10` and for $P2$, the AS-Path is `A:100:20`. The length of the AS-Path is the same for both routes. It is the same for the routes received from $A2$ and from $Z1$. The egress point chosen by the routers in the modeled network is the same for both prefixes (see dotted arrows in Fig. 3.4). Therefore, we can limit to prefix $P1$ the computation of the outcome of the BGP decision process in the model since the outcome will be the same for $P2$.

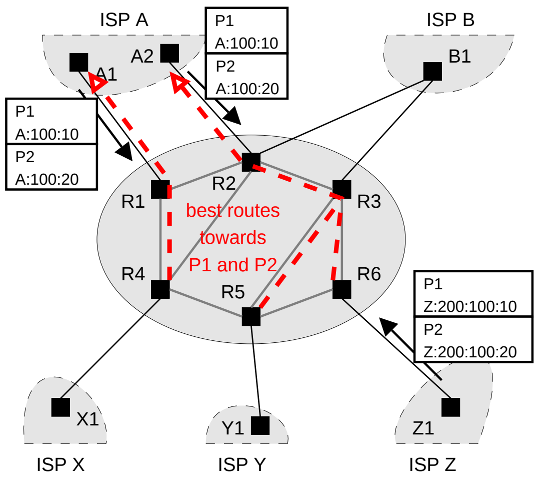


Figure 3.4: Clustering of BGP prefixes.

To assess the applicability of this clustering method on GEANT, we looked at the eBGP routes collected on November 24th, 2004. We grouped together the prefixes announced with the same BGP quality by the same neighbor routers. We ended up with 406 clusters for 105,071 prefixes. This represents a clustering ratio of 0.27%. We show in Fig. 3.5 the distribution of the number of prefixes in the obtained clusters. 80% of the clusters contain less than 67 prefixes. The remaining clusters can contain up to thousands of prefixes. We also looked at the stability of the number of clusters along the time. During the period going from the 16th to the 30st of November, 2004, the number of prefixes ranged from 149,438 to 150,234. For this period, the number of clusters was 405.8 on average with a small standard deviation of 16.5, meaning that the number of clusters did not change a lot during this period. We observed the same behavior for more recent datasets.

This means that for GEANT, instead of injecting 105.071 prefixes in the simulator (representing 640.897 different routes), we could only inject 406 prefixes

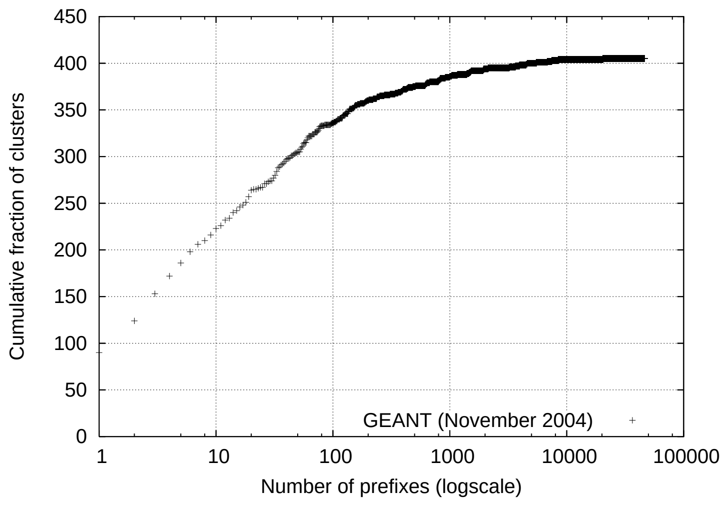


Figure 3.5: Cumulative frequency of clusters with the same number of prefixes.

(representing 986 routes) in the model without loosing any information on the outcome of the decision process. Computing the BGP routes towards the 406 clusters of GÉANT using C-BGP required only 68s on an Intel P4 running at 2.66GHz and the memory footprint was only 69MB.

### 3.3.3 Traffic statistics

To build an accurate model of the traffic, we obtained the Netflow statistics collected on all the edge links of the GÉANT network. In order to limit the volume of Netflow traces, a Netflow sampling rate of 1/1000 is used. This still generates in the order of 150 GB of gzipped traces per month of traffic. Hence, we further summarized the Netflow information by aggregating the raw Netflow flows on a (source prefix, destination prefix) basis, keeping only the byte volume for each (source prefix, destination prefix) pair for each Netflow file. We call the result of the summarization process the "aggregated netflow". We used a single BGP Routing Information Base (RIB) to aggregate the raw Netflow flows, as the purpose is only to limit the size of the traces.

## 3.4 Characterization of routing

In this section, we characterize the routing in GÉANT. We identify in the following subsection what are the factors that influence both the intra- and interdomain routings. These factors will help us to understand the results of the “what-if” scenarios studied in Section 3.5.

### 3.4.1 Intradomain routing

In order to characterize the intradomain routing, we rely on a measure of the centrality of the routers and links in the graph. This measure indicates how the intradomain routing is sensitive to the failure of a router or link. That is, the more a node or link is central, the more its failure will cause intradomain path changes.

The measure usually used to perform this characterization is the **betweenness-centrality** [BE04]. Basically, the betweenness-centrality computes the amount of shortest-paths that go through a vertex or an edge. The centrality of a vertex $v$ is computed as:

$$c(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

i.e. the sum for all pairs of sources and destination $(s, t)$ of the fraction of shortest-paths from $s$ to $t$ that pass through $v$. $\sigma_{st}$ denotes the number of shortest-paths from $s$ to $t$ and $\sigma_{st}(v)$ denotes the number of shortest-paths from $s$ to $t$ that go through $v$. A similar definition is used to compute the centrality of edges:

$$c(u, v) = \sum_{s \neq u,v} \frac{\sigma_{st}(u, v)}{\sigma_{st}}$$

where $\sigma_{st}(u, v)$ denotes the number of shortest-paths from $s$ to $t$ that go through edge $(u, v)$. Note that the direction matters.

We show the betweenness-centrality of GÉANT routers in Fig 3.6 and the betweenness-centrality of GÉANT links in Fig 3.7. Note that for edges, we summed the centrality for both directions. The first observation that can be done is that the centrality distribution is skewed. This means that from the intradomain routing viewpoint, some nodes and some links are more critical than others. If these nodes or links fail, their impact on the intradomain paths will be higher than others. For instance, about half the nodes have a centrality which is close to zero. This means that from the intradomain routing viewpoint, these nodes are seldom transit nodes. The other half supports a varying number of shortest-paths. DE2, DE1 and AT1 have the highest centrality. We expect that their failure will have a significant impact on routing. For what concerns links, we also observe that a few links seem to be more critical than others: DE2-DE1, DE2-AT1 and AT1-HU1 are the links with the highest centrality. We can conclude that **the intradomain routing in GÉANT is mostly sensitive to the failure of a small number of nodes and links**.

### 3.4.2 Interdomain routing

In order to characterize the interdomain routing in GÉANT, we first analyzed the diversity of interdomain routes. For this purpose, we counted the number of interdomain routes learned by GEANT over its eBGP sessions towards each destination prefix. This gives information about the number of possible egress points for each destination prefix. We show the breakdown of the prefixes versus the number of

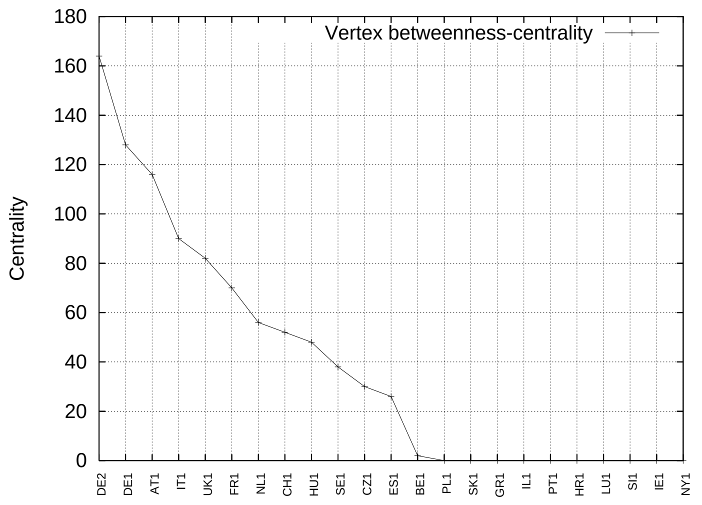


Figure 3.6: Betweenness centrality of GÉANT routers.

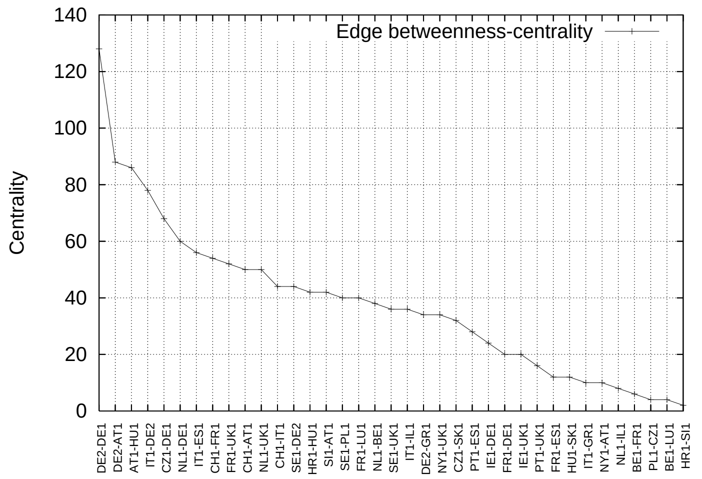


Figure 3.7: Betweenness centrality of GÉANT links.

eBGP routes in Fig. 3.8. The second information shown in Fig. 3.8 concerns the number of routes received from the same neighboring AS for each prefix. Multiple routes are learned from the same neighboring AS when there are parallel peering links with this AS. We counted for each destination prefix and for each neighboring AS the number of routes learned. We consider for each prefix the maximum of the number of routes learned from the same AS. We show the breakdown of prefixes versus the maximum number of *"parallel routes"* in Fig. 3.8.

A first observation is that GÉANT has learned at least 2 routes for the ma-

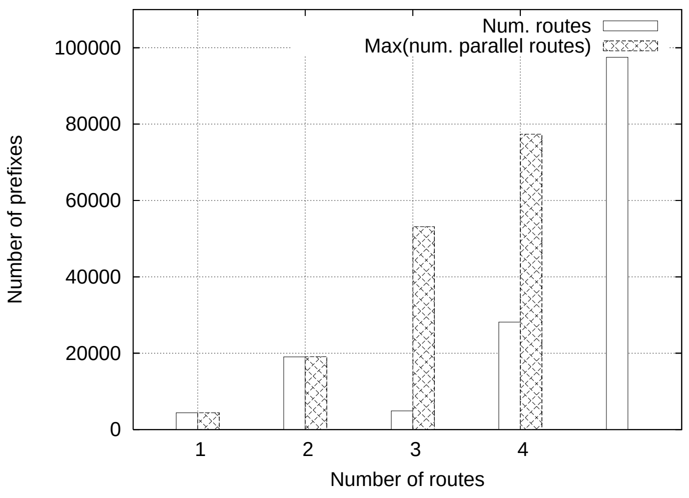


Figure 3.8: Breakdown of prefixes by number of received routes in GEANT.

jority of destination prefixes (more than 97%). For 63% of the prefixes, it has even learned 5 routes. Moreover, the maximum number of routes learned from a neighboring AS is between 3 and 4 for the majority of destination prefixes. This was expected since GÉANT has up to 4 peering links with some of its neighbors. There is thus an **high route diversity** in GÉANT.

Given the route diversity shown in Fig. 3.8, it is interesting to ask how each router will select its best route. For this purpose, we looked into each modeled router, what was the rule of the BGP decision process that was used to select the best route towards each destination prefix. We show our results in Fig. 3.9. The x-axis of the figure represents each modeled router identified by its name. The y-axis represents the destination prefixes known by this router. For each router, the figure shows the rule of the BGP decision process that was used to select a route towards a fraction of prefixes. We show only the 4 main rules. The rule 'Single route' means that the router has no choice because it received only a single route for the destination prefix. The meaning of the other rules is straightforward.

We observe that the impact of the routes diversity on the BGP decision process is high. In GEANT most destination prefixes (more than 97%) are reachable through at least 2 different routes (see Fig.3.8). As a consequence, the impact of the internal structure of GEANT on the interdomain route selection is important. The interesting point here is that **for most destination prefixes, the rules related to hot-potato routing are used**. We see that the border routers receiving a large number of eBGP routes select the best route towards a large fraction of prefixes using the rule "eBGP over iBGP". This is the case for IT, SE, UK, CH and DE1. This means that the location of peering points is important for the model. In addition, the rule "Nearest next-hop" is used for the majority of route selection. This means

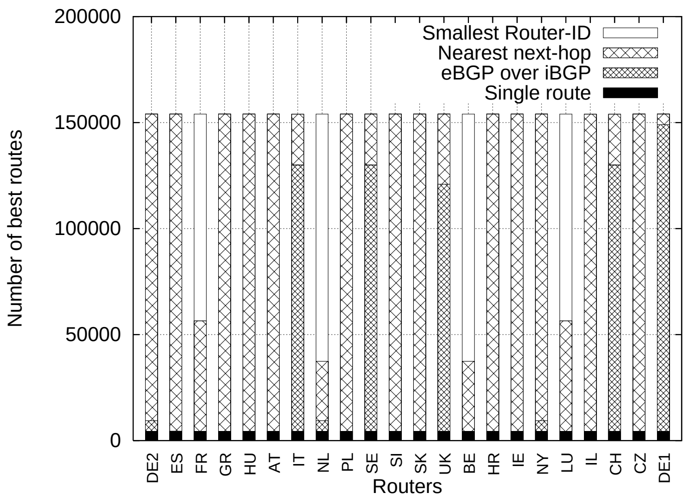


Figure 3.9: Importance of the tie-breaking rules in GÉANT.

that for these routers and for these destinations, the route selection was partially based on the IGP cost to reach the egress routers. We also observe that 4 routers use the rule “Smallest Router-ID” to select a large fraction of their routes: FR, NL, BE and LU. The reason is that these routers are at an equal IGP distance from egress routers announcing a large fraction of routes.

## 3.5 What-if scenarios

In this section, we present two case studies that we performed on our model of the GÉANT network. The first one investigates the addition or removal of peerings on the flow of the traffic. The second studies the routing impact of link failures. All the PoP names and peerings that appear in the following studies have been anonymized in the following case studies upon request of Dante.

### 3.5.1 Optimized peering

An important problem faced by ISPs is about finding the optimal location of peering points. An ISP will search for new peering points in order to improve the efficiency of its interdomain traffic and/or to decrease the cost of its peerings. Finding the optimal peering location is a non-trivial problem [AAM98], that depends on both technical and economical factors [Nor02]. In [AAM98], the authors focused on the location of peerings with a single neighbor AS. With our tool, we can investigate the problem of modifying the interdomain connectivity of an AS since we take into account the BGP information. To our knowledge, existing approaches have not taken BGP into account.

In practice, an AS can choose to peer with many different ASs and at several locations. However, all the possible locations do not satisfy the ISP's requirements and the ISP must decide which one best fits its goals. Let us take the example of a transit provider. Assume that its network is composed of $n$ PoPs. Now, suppose that the provider serves new customer ASs that connect at some PoP $x$. These customers cause an increased amount of traffic to cross the topology before exiting at other PoPs. To prevent the traffic to cross the whole network before reaching the egress points, the transit provider might prefer to add a peering close to the PoP $x$ that generates more traffic so that the traffic received by this PoP exits the network as early as possible. It is thus expected that a given amount of traffic will exit the network through the new peering, but it is difficult to predict how much.

Adding a peering has the potential effect of modifying the best routes of the BGP router connected to this new peering. It is likely that this router will select routes learned through its new peering as best routes. It will then redistribute its new best routes to the other BGP routers through the iBGP sessions. If the latter BGP routers choose to use some of the routes learned through the new peering, it is possible that more traffic than originally planned will exit the network at the new peering. Models of an AS that do not consider BGP routing cannot predict the exact change in the traffic matrix in such a case.

With our modeling tool, we are able to predict what will happen to an AS's traffic when a new peering is added or removed. In order to compare the impact of the various scenarios, we use different metrics. First, we compute the distribution of the traffic over the peering links. Adding a new peering may attract some traffic from the already existing peering links and decrease the likeliness of congestion to occur on these links. Then, we compute the IGP cost that the traffic undergoes when the new peering is added. If this cost decreases for a significant number of ingress-egress pairs, it means that the traffic follows intradomain paths that are shorter in term of the IGP weights assigned by the ISP.

We performed this evaluation on the GÉANT network. Actually, Dante who operates GÉANT, is currently designing the next version of its network. GÉANT2 will have an increased number of customers, mainly in eastern Europe. GÉANT2 will provide transit to additional NRENs including the Russian NREN, JSCC, for instance. In this context, it is important for Dante to know which locations will benefit from additional peerings. In our evaluation, we consider the six most important peerings of the GÉANT network that we call *PR1*, ..., *PR6*. Today, all these peerings use OC-48 links with a 2.4 Gbps capacity. We study the addition and removal of peerings and its impact on the traffic coming from all the GÉANT customers.

Fig. 3.10 shows the impact of the removal or the addition of a peering, in terms of the distribution of the outgoing traffic over the considered peerings of GÉANT. The x-axis of Fig. 3.10 gives the different scenarios we simulated. The one labeled "default" gives the distribution of the traffic if we leave the current peerings unchanged. Those labeled "remove-X" concern the scenarios where we removed the existing peering X. The others labeled with "add-PRX" are the scenarios where a

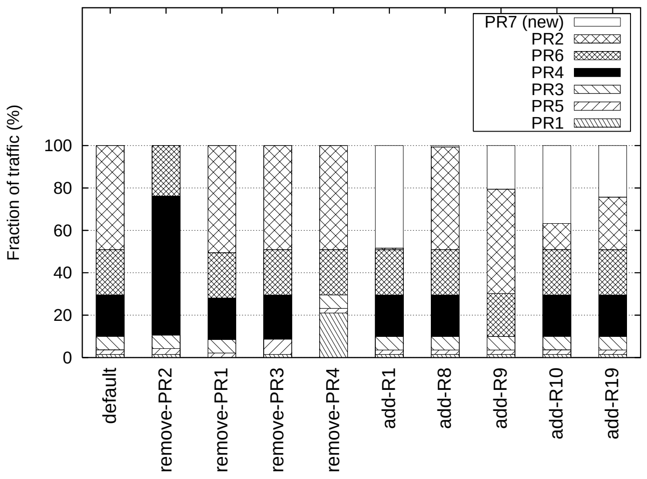


Figure 3.10: Impact of addition/removal of peering on the distribution of the traffic among the peering links.

peering was added at the PoP X. The y-axis of Fig. 3.10 shows the distribution of the percentage of the total outgoing traffic carried by the considered peerings.

The "default" scenario on Fig. 3.10 shows that almost 50% of the traffic is carried by a single eBGP peering (*PR2*) and two other peerings carry each about 20%. The traffic is thus unevenly balanced over the considered peering links. Removing a peering does not change this uneven distribution. When the peering *PR2* is removed, almost all the traffic exits at *PR4* and *PR6*. Removing the *PR1* or *PR3* peerings has little effect. Removing the *PR4* peering shifts its traffic to the *PR1* peering. Now let's consider the addition of peerings at PoPs *R1*, *R8*, *R9*, *R10* and *R19*. Adding a peering link at *R1* absorbs all the traffic that previously exited through *PR2*. The explanation is that most of the traffic sent through the *PR2* peering was coming from eastern Europe. To reach the *PR2* peering, these packets now have to pass through the *R1* router and thus leave the GÉANT network there. If the purpose of adding this peering is to change the uneven distribution of the traffic among the peering links, then adding a peering in *R1* does not help. The situation is similar when adding a peering at *R9*, as it absorbs most of the traffic that previously exited through the *PR4* peering. Adding a peering at *R8* is worthless since the distribution of the traffic is left unchanged. Adding a peering in *R10* or *R19* on the other hand improves the balance of the traffic over the considered peering links.

Modifying the peerings of an AS not only changes the distribution of the traffic among the peerings, but also how traffic crosses the intradomain topology. Fig. 3.11 shows the impact of adding or removing a peering on the IGP cost suffered by the traffic to cross the network. On the x-axis of Fig. 3.11, we show the difference between the IGP cost the traffic is subject to in the default situation and the one in each scenario. A positive difference means an improvement since the IGP cost has been decreased. A negative difference means a deterioration. On the y-axis of Fig. 3.11, we show the cumulative fraction of the traffic that perceives a change in the IGP cost.

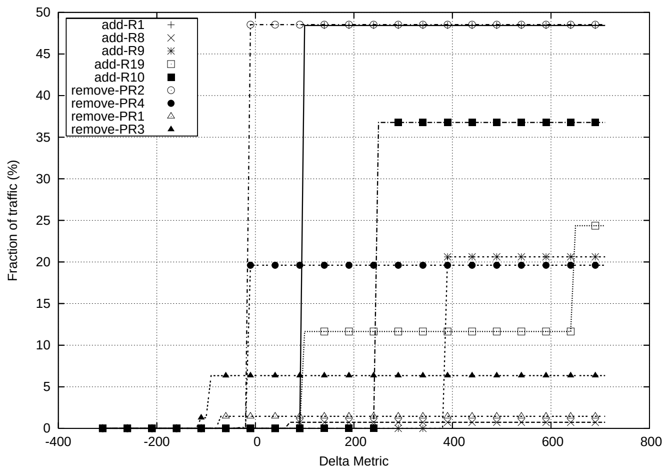


Figure 3.11: Impact of addition/removal of peering on IGP cost seen by traffic.

We have seen in Fig. 3.10 that removing the peerings *PR1* and *PR3* does not impact much the balance of the traffic on the peering links. However, we can see that removing the peering *PR3* has an impact on the IGP cost seen by 5% of the traffic. The IGP cost for this traffic has been increased by 100. Another observation is that although removing the peering *PR2* has a significant impact on the distribution of the traffic on the peering links, it has a small impact on the IGP cost ($\Delta$=-5) seen by this traffic. The most interesting scenarios are the addition of the peering in *R10* or *R19* that improves both the IGP cost and the distribution of the traffic among the peering links as showed above. If the purpose of adding a single peering link is to improve both the distribution of the traffic over the peering links without worsening the delay across the network, then we know that the solution is to add a peering either in *R10* or in *R19*.

In Fig. 3.11, we show the impact of adding or removing a peering on the delay seen by the traffic to cross the network. The results are presented in the same way as for the IGP cost difference shown in Fig 3.10. On the x-axis of Fig. 3.12, we show the difference between the delay the traffic is subject to in the default situation and the one in each scenario. A positive difference means an improvement since the delay has been decreased. A negative difference means a deterioration. On the y-axis of Fig. 3.12, we show the cumulative fraction of the traffic that perceives a change in the delay.

We observe in Fig. 3.12 that most of the time the IGP cost improvement is correlated with the improvement in delay. There is an exception with the removal

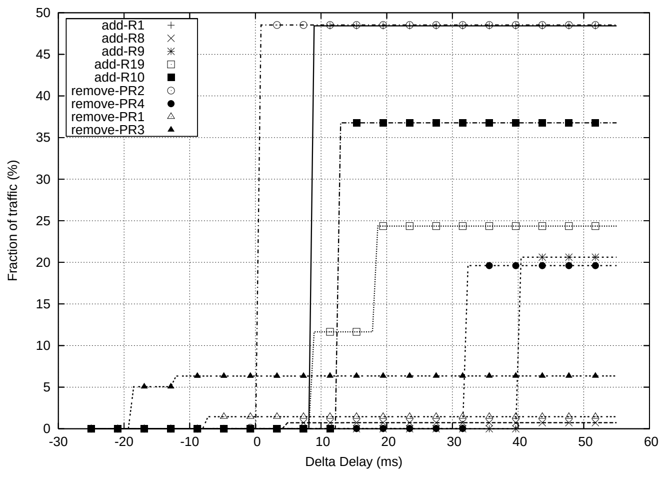


Figure 3.12: Impact of addition/removal of peering on the delay seen by traffic.

of the PR2 peering which causes a small improvement in the IGP cost but a small degradation of the delay. This may seem surprising but it can be explained by the way the IGP weights have been assigned in GÉANT. Dante explains that by the fact that they first use IGP weights that are inversely proportional to the links capacities. Then, manual tuning of the weights is done to prefer the lowest delay paths. Finally, the IGP weights are assigned in order to avoid equal-cost paths. There is thus not necessarily a correlation between the delay and the IGP cost. It is also important to note that Fig 3.12 only gives the intradomain delay. To measure the impact on the end-to-end delay, active probing should be performed on a per destination basis, which is expensive.

### 3.5.2 Link and router failures

Evaluating the impact of link and router failures on the network is another non-trivial problem. In a large network, determining which link and router failures will change the outcome of the egress selection performed by BGP is a difficult problem. This is an important problem since routing changes can cause traffic shifts and lead to congestion [TSGR04]. For an operator, it is important to check that the network will be able to accommodate the traffic load even in the case of single link or router failures. If not, it is useful to identify which network links should be protected by the addition of parallel links, SONET-SDH protection or the use of MPLS protection tunnels [VPD04].

Our methodology for studying the impact of intradomain changes on the path selection is as follows. First, we build a representation of the network inside the routing solver. We let the solver compute the routes in each router, then we store a snapshot of the selected routes. This snapshot corresponds to the state of routing when everything is up and running. Then, we remove the failing link or router and we let the routing solver recompute the paths.

In order to provide a synthetic view of the impact of each failure, we partition the set of routing changes in four different classes: *Peer change*, *Egress change*, *Intra cost change* and *Intra path change*. The *Peer change* class corresponds to changes in the next-hop AS. If the next-hop AS has not changed but the egress router has changed, we speak of an *Egress change*. When the egress is unchanged but the IGP cost of the ingress-egress path has changed, the routing change is classified as an *Intra cost change*. Finally, if an ingress-egress path with the same IGP cost has been found, the routing change is put in the *Intra path change* class. This can only occur if there are multiple equal cost paths between an ingress and an egress routers in the network.

We simulated all the single-link failures in GÉANT and observed the impact on the BGP routes selected by each GÉANT router. We show our results in Fig. 3.13. On the x-axis, we show all the internal links of GEANT. On the y-axis, we show the number of routing changes cumulated on all the GÉANT routers. The routing changes are classified in *Peer change*, *Egress change*, *Intra cost change* and *Intra path change* as explained above. The links on the x-axis are ordered according to the total number of routing changes caused by their failure. We observe that most of the time, a single link failure causes many egress changes. **Nearly 60% of the GÉANT links cause more than 100,000 routing changes when they fail.**

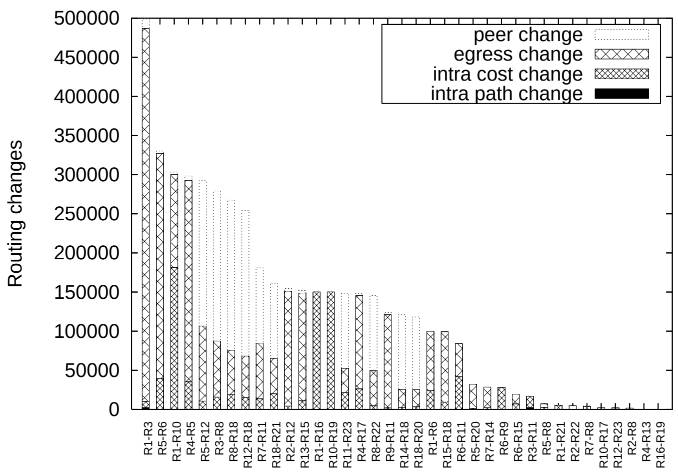


Figure 3.13: Single link failure analysis: impact on BGP.

We can also observe that, in GÉANT, there are few pure intradomain re-routings. That is, there are few routing changes in the *Intra cost change* and *Intra path change* classes. These results indicate that **a pure intradomain model of the GÉANT network would not capture most of the routing changes that occur under single link failures**. This shows clearly that to accurately model a transit network like GEANT, it is necessary to take the interdomain routes and the topology into account.

The same method can be used to perform the single-router failure analysis or to study the impact of changing the IGP cost of one link. Similar results to those found for the link failures have been obtained. We show them in Fig. 3.14. On the x-axis, we show all the routers of GÉANT. On the y-axis, we show the number of route changes cumulated on all the GÉANT routers. The routers on the x-axis are ordered according to the total number of route changes caused by their failure. We observe that the failure of nearly half the GÉANT routers cause route changes. The failure of a single router can be seen as the joint failure of all its links. The consequence is that the routers whose failure cause the largest number of route changes are the routers that connect to the most critical links identified in Fig. 3.13. Routers R5, R3, R18, R11 and R15 peers with the commercial providers and thus receive BGP routes for nearly all the Internet prefixes. Routers R1 and R6 are critical for the Internet connectivity of a large number of GÉANT PoPs, mainly in eastern European countries.

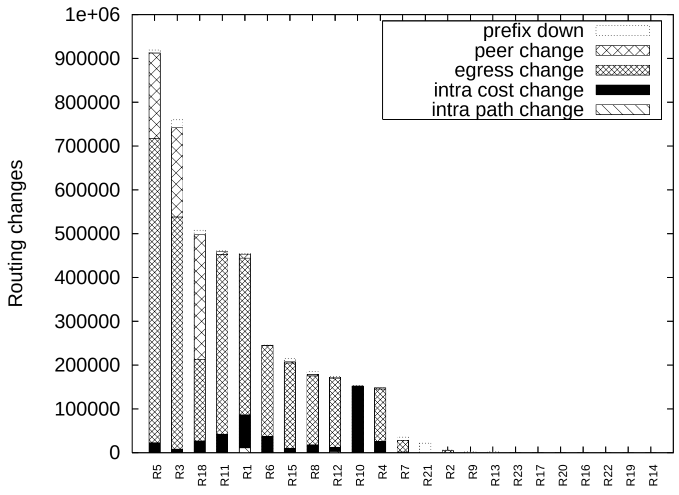


Figure 3.14: Single router failure analysis: impact on BGP.

We can also observe on Fig. 3.14 that the class *Prefix down* is not empty. This happens for destination prefixes for which a single eBGP route is known in the model. This means that based on our model, we could conclude that the failure of some routers can lead to the unreachability of some destinations. Actually, this conclusion is a bit hasty. A first reason is that we do not know all the external routes received by GEANT routers. We only know the best routes selected by each border router. In the real GEANT network, other routes might become available when the current best route is withdrawn, as we said in Section 3.2.3. In addition, even if a prefix is not reachable anymore, it is possible that a less specific prefix still is. In this case, the destination would still be reachable.

### 3.5.3 Impact on traffic

In addition to studying the impact of failures on routing, we evaluated the importance of traffic shifts caused by link failures. Indeed, when routes change some traffic flows may be forwarded along different intradomain paths. This will occur for traffic flows that are forwarded on a path that has been affected by the failure. Traffic shifts will modify the distribution of the traffic inside the network and change the load of some links. As a consequence, some links can even become congested.

For the purpose of evaluating the traffic shifts caused by link failures, we used an interdomain (prefix-prefix) traffic matrix from Géant as described in Section 3.3.3. One interdomain traffic matrix of GÉANT is a set of triples (ingress router, destination prefix, traffic volume). Each triple represents the traffic volume that has been received by an ingress router and to be sent towards the destination prefix over a given period of time.

To compute the intradomain (router-router) traffic matrix, we replay the flow of the traffic across GÉANT. For this purpose, we take each triple from the interdomain (prefix-prefix) traffic matrix, one at a time. Then, we perform a longest-matching in the routing table computed by the routing solver for the considered ingress router, in order to find the prefix that contains the destination. We then use the route associated with this prefix to route the traffic. We repeat this step on a hop-by-hop basis until the egress router is reached. The intradomain traffic matrix is obtained by summing the volume of traffic exchanged between all the pairs of ingress and egress routers for the period of time.

We use the above methodology to compute the intradomain traffic matrix after each link failure. We show a subset of the results in Fig. 3.15 and Fig. 3.16. These figures show the traffic volume carried by each link after a single link failure (links R1-R3 and R5-R6 are shown). We distinguish the links directions since each direction may carry a different volume of traffic. On the x-axis, we show all the directed links ordered based on the volume of traffic they carry before the failure. The y-axis shows the amount of traffic carried by the corresponding link. In each figure, we show two curves. The first one, labeled “default”, represents the links load when all the links are up and running. The second one represents the links load after the link failure. In the default situation shown in Fig 3.15, we can observe that the traffic load is unevenly distributed on the links. The most loaded link is R5-R6 followed by R1-R3.

Fig. 3.15 shows the links load after the failure of link R1-R3. We have shown in Fig. 3.13 that this link was the most important from the routing viewpoint. We observe that, as expected, the traffic originally carried by the R1-R3 link now passes through other links. We also observe that the load of the previously most loaded link, R5-R6, has nearly doubled. This indicates that a large fraction of the traffic initially going through R1-R3 is now forwarded along paths that go through R5-R6.

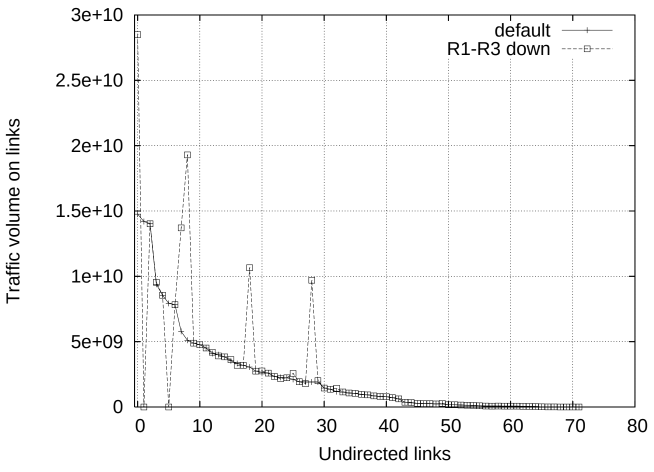


Figure 3.15: Impact of the failure of R1-R3 on the links load.

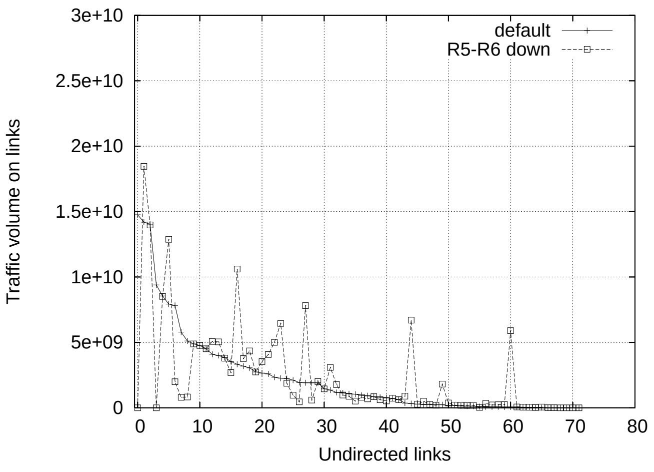


Figure 3.16: Impact of the failure of R5-R6 on the links load.

In Fig. 3.16, we show the links load following the failure of link R5-R6. We observe that the traffic shifts are less localized than with the failure of R1-R3. More links have their load changed. This may seem surprising since Fig. 3.13 shows that the number of routing changes due to the failure of R5-R6 is lower than the number of routing changes due to the failure of R1-R3. As explained in Section 3.5, this is due to the complex interaction between IGP, BGP and the traffic. The failure of R1-R3 and the failure of R5-R6 do not have the same impact on the intradomain routes computed by the IGP. These routes are used by ingress routers to reach egress routers. Then, the ingress-egress routes are used by the BGP routes to cross Géant. Depending on which routes are used to forward large amounts of traffic, the impact will differ.

Finally, Fig. 3.17 shows a summary of the impact of all the link failures on the traffic. The x-axis shows all the link failures ordered based on the load of the most loaded link. The y-axis provides the following statistics: the median, 5th- and 95th-percentile as well as the mean load and the load of the most loaded link. First, we observe that in GÉANT, the failure of a single link can cause a large increase of the maximum link load. There is even a link failure that causes the maximum link load to nearly double. Second, we observe that in GÉANT, the shape of the links load distribution is not impacted much by the links failures. Indeed, the median and the 5th and 95th percentile does not change much.

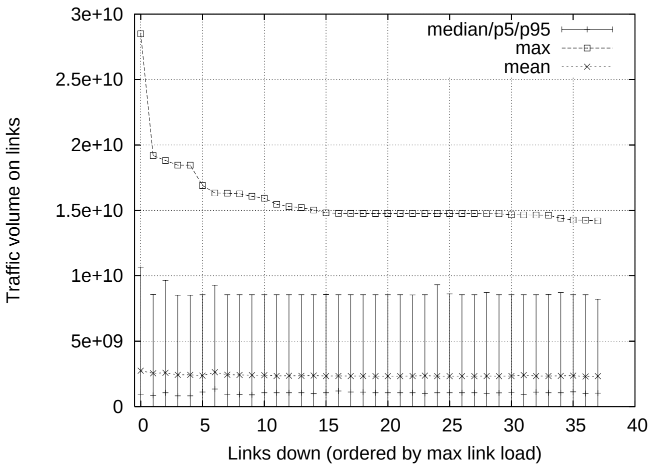


Figure 3.17: Most sensitive single link failure.

We have shown in this section that **link failures can cause important traffic shifts**. For instance, the failure of link R1-R3 has doubled the load of link R5-R6. During our analysis of the GÉANT network, no congestion was provoked by the link failures. This is due to the particular situation of GÉANT that is composed of links having a very high capacity compared to the traffic volume that they usually carry. However, in other networks, this analysis can help to determine which links are important for the traffic. We have shown that the interdependence between the intra- and interdomain routing combined with the traffic distribution is difficult to understand in a medium-sized transit network. A methodology such as the one used in this chapter can help operators to evaluate their design or to evaluate how their network will behave when the routing and/or traffic conditions change.

## 3.6 Conclusion

In this chapter we have described how to build a model of the routing of a large AS. We have described the essential factors that need to be taken into consideration when building a model of the routing of an AS. In particular, we have shown that building an accurate model of an AS requires the network topology, the traffic, the routing protocols and their configuration. Obtaining the related data can still be an operational issue. We illustrated the use of the routing solver described in Chapter 2 through two different case studies. The first case study was an analysis of the impact of changing the peers of a transit AS on its traffic. We have shown that using a model of an AS makes possible to explore various peering solutions. The second study investigated the impact of link failures on the routing changes inside the AS. This is important since network events such as failures and maintenances are frequent. The two case studies have shown the importance of taking into account the interdomain routing information to understand the routing of a large AS.

In this chapter, we considered routing in a single domain. In the following chapters, we will study the interaction between multiple interconnected ASs. C-BGP can be used to compute the outcome of the BGP route selection when there are multiple domains. However, this requires a knowledge of the structure and policies of the other domains. In order to study the impact of changes in one domain on its inbound traffic for instance, we need to have a knowledge of nearly all the Internet domains.

In addition, we are still evolving our tool. A possible future improvement consists in operating the model on a continuous feed of topology, routing data and traffic data. We believe that our approach to integrate the topology, the routing data and the traffic data can serve the ISP operators to better understand the behavior of an AS and help them to investigate improvements in the design of their network.

# Part III

# BGP-based Interdomain Traffic Engineering

Chapter 4

# Current Interdomain TE Techniques

## 4.1 Introduction

Initially developed as a research network, the Internet has been optimized to provide a service where the network does its best to deliver packets to their destination. In the research Internet, connectivity was the most important issue. During the last years, the Internet has undergone a rapid growth and it is increasingly used to carry services such as e-commerce, Virtual Private Networks or Voice/Video over IP. To efficiently support those services, several Internet Service Providers (ISP) rely on Traffic Engineering (TE) [ACE$^+$02] to better control the flow of packets inside their network. Traffic engineering encompasses several techniques that ISP operators can use to evaluate and enhance the performance of their network. Common objectives of traffic engineering consist of shifting traffic away from congested links, distributing the traffic inside the network in order to increase the amount of traffic that can be carried by the network, quickly reacting to failures by directing traffic away from the faulty links or efficiently supporting Quality of Service (QoS) requirements.

While intradomain traffic engineering is a well understood problem, performing traffic engineering across the boundaries of multiple domains is complex. This complexity is mainly due to the current Internet routing architecture. BGP propagates only a subset of the Internet topology among routers. This limits the visibility an AS has on the Internet topology. In addition, BGP does not optimize a global objective, but rather allows each AS to independently select and redistribute interdomain routes in order to satisfy its own local objectives.

Controlling the selection of interdomain routes with BGP in order to support the interdomain traffic engineering process is difficult since BGP was not designed for this purpose. Anyway, some primitive BGP-based routing control mechanisms are currently used by ASs. These mechanisms rely on tuning the attributes of BGP routes. In this chapter, we survey the BGP-based routing control mechanisms used by ASs nowadays and we describe their operation. We focus on unicast traffic and consider the engineering of large aggregates of traffic, typically originated/destined from/to network prefixes. We do not consider a finer control such as between specific pairs of sources and destinations or a per flow engineering.

The chapter is organized as follows. We start by describing in Section 4.2 the techniques that can be used by an ISP to control the traffic inside its own network. Then, we describe in Section 4.3 the problem of engineering the interdomain traffic, i.e. the traffic that crosses interdomain boundaries. We explain the issues inherent to the interdomain routing architecture and the path-vector nature of BGP. In Section 4.4, we describe the current BGP-based routing control mechanisms and we discuss their performance. We focus on two techniques currently used by ISPs to control inbound routes. We present detailed simulation studies of these techniques in Section 4.5 and Section 4.6. For the sake of completeness, we briefly present in Section 4.7 proposals for providing an increased interdomain routing control without relying on tuning the BGP route attributes. Finally, we conclude in Section 4.8.

## 4.2 Intradomain Traffic Engineering

Inside a single ISP (or domain), the whole topology of the network is known by all the routers due to the utilization of link-state routing protocols. Moreover, the intradomain routing protocol usually optimizes a single global objective. Therefore, several techniques can be used to control the flow of the IP packets. They can be divided in two classes. The first class contains the techniques usable in pure IP networks, i.e. dealing with hop-by-hop destination-based forwarding. The second class contains the techniques that can be used in circuit-switched networks. In particular, we discuss the techniques relying on the emulation of ciscuit-switched networks on top of packet-switched networks, such as with Multi-Protocol Label Switching (MPLS) [BZB$^+$97, DR00].

### 4.2.1 IP-based solutions

In a pure IP network, the flow of the IP packets can be controlled by tuning the intradomain routing protocol (also called Interior Gateway Protocol - IGP). Inside a domain, the routing protocol will compute the best path to reach each destination. The best path is usually the path with the smallest cost where the cost of a path is the sum of the weights of all the links that compose the path. The cost associated to each link is usually administratively assigned by the network operators depending on their optimization objective. If each link has a unitary cost, then the routing protocol will favor paths with the smallest number of hops. If the cost of each link is a function of the link transmission delay, then the routing protocol will select paths with the shortest delay. If the cost associated to a link is a function of the link bandwidth[1], then the routing protocol will favor high bandwidth paths.

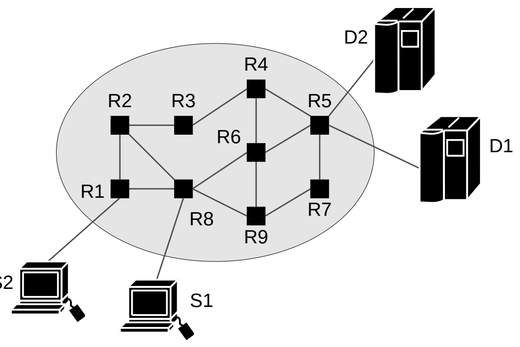


Figure 4.1: A simple ISP network

For example, in figure 4.1, if all links have a unitary cost, the shortest path from the source node $S_1$ to the destination node $D_1$ is the path $(S_1, R_8, R_6, R_5, D_1)$ having a total cost of 4. The path used between source $S_2$ and destination $D_2$ is $(S_2, R_1, R_8, R_6, R_5, D_2)$. If there is a lot of traffic from S1 to D1 and from S2 to D2, then the $R_8 - R_6$ and $R_6 - R_5$ links might become the bottleneck since these two links are used by both flows. A common method to redirect traffic away from congested links is to tune the cost of key links [FRT02]. In the example above (figure 4.1) it is possible to force the traffic flow $S_2 - D_2$ (resp. $S_1 - D_1$) to follow the path $R_1 - R_2 - R_3 - R_4 - R_5$ (resp. $R_8 - R_9 - R_7 - R_5$) by using a cost of 2 instead of 1 on link $R_1 - R_8$ and $R_6 - R_9$ and 3 instead of 1 on link $R_8 - R_6$. In fact, if the traffic demand between each source-destination pair on the network is known, the setting of the link costs can be converted into an optimization problem [FT00] that can be solved by using appropriate mathematical methods. However, these methods are only useful in practice if the traffic demand is relatively stable. If the traffic demand changes frequently, it will be difficult to dynamically recompute the optimal setting of the link costs and dynamically reconfigure all routers without affecting the current traffic inside the network. The robust optimization of IGP weights [FT03] is a possible solution.

### 4.2.2 MPLS-based techniques

MPLS is often used for traffic engineering purposes inside ISP networks. MPLS relies on the label switching technique as in ATM or Frame Relay networks. The main advantage of MPLS for Traffic Engineering, compared to classical IP routing, is that MPLS can be used to explicitly determine the path followed by packets

[1]The default IGP metric on CISCO routers, for instance, is a function of the inverse of the capacity.

inside the domain. MPLS can thus easily bypass the shortest path selected by IP routing. This is done by establishing Label Switched Paths (LSPs) between pairs of routers in the network.

A first utilization of MPLS is to solve the traffic engineering problem statically. Based on a known traffic demand between all source-destination pairs, it is possible to compute an optimum layout of the LSPs in the network in order to spread the load among all the available links. This utilization is similar to the setting of the link costs in a pure IP network although MPLS offers a greater flexibility. With only the link costs, a change in the cost of one link may potentially affect the traffic distribution throughout the entire network. With MPLS, it is possible to force any flow of packets to follow a particular path. For example, it would be possible to force the packets from the $S_1 - D_1$ flow to follow the $R_8 - R_1 - R_2 - R_3 - R_4 - R_6 - R_5$ path while the packets from the $S_2 - D_2$ flow would follow the $R_1 - R_8 - R_9 - R_7 - R_5$ path in figure 4.1. This type of traffic distribution would not be possible by selecting the link costs in a pure IP network.

MPLS is also useful in a more dynamic environment where the traffic demand changes slowly (e.g. the demand during the business hours is not the same as during the evening). In this case, MPLS is used in combination with enhanced intradomain routing protocols (OSPF-TE [KKY03] or ISIS-TE [SL04]). These protocols are extensions to the classical link-state intradomain protocols (OSPF and ISIS) that are able to distribute to all routers together with the entire topology of the network additional information like the bandwidth and the utilization of each link. Based on this information, each router can determine the most heavily loaded links inside the network. To understand this utilization of MPLS, let us consider figure 4.1 again and assume that there are two important flows of packets ($S_1 - D_1$ and $S_2 - D_2$) and that links $R_8 - R_6$ and $R_6 - R_5$ are becoming congested while the other links are lightly utilized. Based on the information distributed by the constrained routing protocol, routers $R_1$ and $R_8$ are aware of the load on these links. If these routers notice large flows from $S_2$ and $S_1$, they could redirect those flows to other links by establishing new LSPs. To establish such an LSP, $R_1$ will compute a constrained path for the new LSP towards $D_2$. To select the best path, router $R_1$ will specify a set of constraints that must be fulfilled by the chosen path. For example, $R_1$ could request a path that avoids the congested links ($R_8 - R_6$ and $R_6 - R_5$). If $R_1$ knows that the $S_2 - D_2$ flow requires 100 Mbps on average, it could also select a path where all the links have at least 100 Mbps of unused bandwidth. The selected constrained path will be used by the RSVP-TE signaling protocol [ABG$^+$01] that will establish the LSP. This signaling protocol can also be used to reserve resources (e.g. bandwidth) on the selected path if required. An additional advantage of MPLS is that it is possible to re-route an existing LSP over another (e.g. less congested) path before breaking the existing LSP. This utilization of MPLS combined with constrained routing protocols and a signaling protocol allows the network to be traffic engineered in a more dynamic manner than with a pure IP solution.

## 4.3 Interdomain Traffic Engineering

A large fraction of the Internet traffic does not remain inside a single domain but instead crosses its interdomain boundaries. The source and destination of this interdomain traffic might even be several domains away. This makes interdomain traffic engineering an important matter for ISPs.

The interdomain traffic engineering requirements are diverse. They depend on the connectivity of an AS with others (see Chapter 1) but also on the type of business handled by this AS. Typically, content-providers that host a lot of web or streaming servers and usually have several customer-to-provider relationships with transit ASs will try to optimize the way traffic leaves their networks. Secondly, access-providers that serve small and medium enterprises, dial-up or xDSL users typically wish to optimize how Internet traffic enters their networks. Finally, a transit AS will try to balance the traffic on the multiple links it has with its peers.

Therefore, the main objective of interdomain traffic engineering is to control through which peering links traffic flows will enter or exit the network. This can be for the purpose of balancing the traffic on multiple links with other ASs, to shift traffic away from a congested peering link or to move part of the traffic to less expensive peering links. Another interdomain traffic engineering requirement is the ability to direct traffic to alternative interdomain paths with different properties. An example would be to select a path with better end-to-end performance (lower latency or higher available bandwidth), even at higher cost. Another purpose would be the provision of a backup path completely disjoint from the main one.

### 4.3.1 Limitations of the current Internet routing architecture

Unfortunately, reaching these objectives with the current Internet routing architecture is difficult. There are two main issues.

First, the **path-vector** nature of BGP imposes limitations on the selection of interdomain routes. Each BGP router only redistributes a single “best” route to its neighbors. Even if the router knows many candidate routes, the alternative routes are not propagated. This limits the visibility each router has on the Internet topology and reduces the path diversity. This is in contrast with the intradomain routing protocols (link-state) that distribute the complete topology to all participant routers. The implication for traffic engineering is a limited freedom for directing traffic to alternative paths. In addition, there is inherently less control on the routing decisions, since the routes one router receives depend on the decisions of other routers downstream.

Second, BGP relies on a **complex decision process** which does not optimize a global metric. This is in contrast with the selection of least-cost paths by intradomain routing protocols. Instead, the operators of each AS can independently configure their own routers so as to optimize local objectives and enforce local route filtering policies. For example, each AS can configure a local ranking of the routes towards a given destination using the Local-Pref attribute. The implication for traffic engineering is that the configuration of one AS might cancel out the route control effort of another AS.

### 4.3.2 Characteristics of the interdomain traffic

Another difficulty of interdomain traffic engineering comes from the large number of sources and destinations involved. At the intradomain level, the number of destinations announced by the IGP protocol is limited to the routers and subnets of the domain. Depending on the size of the domain, its order of magnitude ranges from tens to thousands of destinations. At the interdomain level, the number of sources and destinations includes remotely advertised networks. This number is far larger. For example, the current number of destination prefixes in a BGP routing table is on the order of 180,000 [Hus06].

A first point to note however is that although an AS might exchange traffic with a large fraction of the Internet domains, it **does not exchange the same amount of traffic with each remote AS**. On Fig. 4.2, we show the cumulative distribution of the interdomain traffic received or sent by three ISPs. The figure concerns the traffic received by BELNET, the Belgian National Research and Education Network (NREN) during one entire week in December 2000; the traffic received by YUCOM, a Belgian ISP, during 5 consecutive days in April 2001; and the traffic sent by the Pittsburgh Super-Computing Center (PSC) during one day in March 2002. More details on the data collected can be found in [UB02]. We observe that the 100 largest sources of traffic for BELNET contribute more than 64 % of the traffic received during one week. Similarly, the 100 largest sources of traffic for YUCOM contribute more thatn 72 % of the traffic received by this ISP. For PSC, the concentration of the traffic sinks is even more important as the 100 largest destinations receive 78 % of the total traffic sent by PSC. [FBR03] mentions a similar distribution for the interdomain traffic of a large *tier-1* ISP.

Another important point to mention about the interdomain traffic exchanged by the studied ISPs is the distance (measured in AS hops) between the remote ASs and each studied ISP. Fig. 4.3 shows, for each ISP, the percentage of its interdomain traffic that was produced by or sent to remote ASs as a function of their distance measured in AS-hops. This analysis shows that the studied ISPs only exchange a small fraction of their traffic with their direct peers (AS-hop distance on 1). **Most of the packets are exchanged with ASs that are only a few AS hops away**. For BELNET, most of the traffic is produced by sources located 3 and 4 AS hops away while YUCOM mainly receives traffic from sources that are 2 and 3 AS hops away. PSC on the other hand sends traffic to ASs located at up to 4 AS hops away.

This implies that an AS willing to engineer its interdomain traffic could move a large amount of traffic by influencing only a small number of distant ASs, typically, the most popular sources/destinations. In addition, since the sources and/or destinations of interdomain traffic are located only a few AS hops away, interdomain traffic engineering solutions should be able to influence ASs a few hops beyond their upstream providers or direct peers.

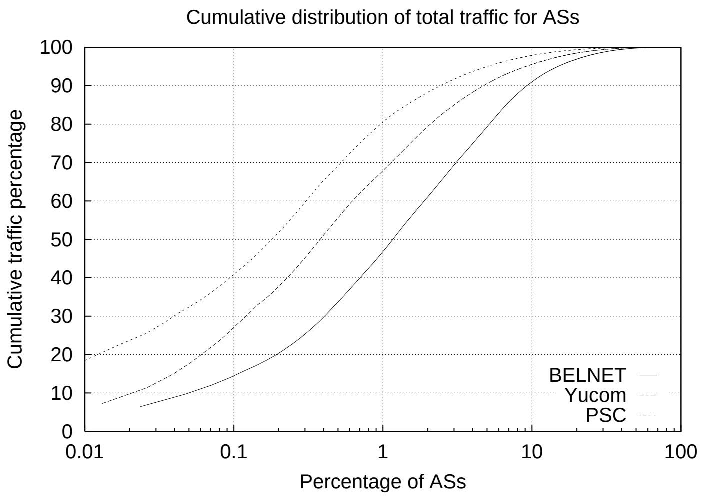


Figure 4.2: Cumulative distribution of the traffic for each studied ISP.

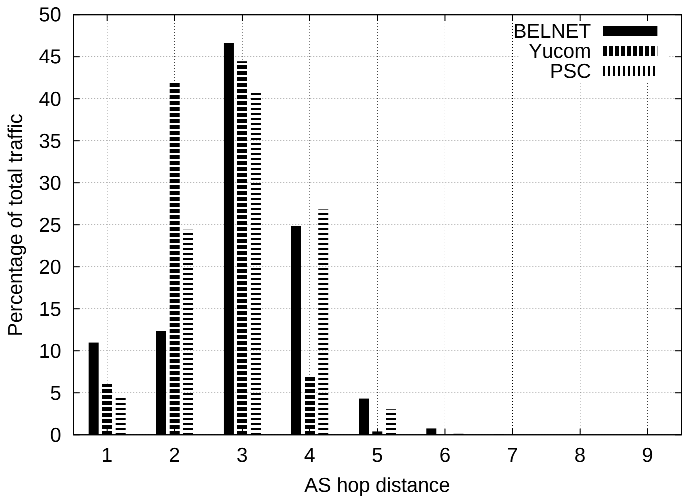


Figure 4.3: Per-AS hop distribution of the traffic for each studied ISP.

## 4.4 BGP-based Traffic Engineering

Optimizing the way traffic enters or leaves a network means to favor one link over another to reach a given destination or to receive traffic from a given source. This type of interdomain traffic engineering can be performed by tweaking the configuration of the BGP routers of the AS. Indeed, a key feature of BGP is the decision process used by each BGP router to select, among all the received advertisements, the best path to reach each destination. In order to understand how BGP can be used to control the way traffic enters, leaves or crosses an AS, a better understanding of the BGP decision process is required. We repeat the BGP decision process in Fig. 4.1. A BGP router receives one route toward each destination from each of its peers. To select the best route among this set of routes, a BGP router relies on a set of rules called the decision process. Most BGP routers apply a decision process similar in principle to the one shown in figure 4.1. The set of routes with the same destination are analyzed by the rules in the sequence indicated in figure 4.1. These rules act as filters and the $N^{th}$ rule is only evaluated if more than one route has passed the $N-1^{th}$ rule. It should be noted that most BGP implementations allow the network administrator to optionally disable some of the rules of the BGP decision process.

| **Rank** | **Rule** |
|---|---|
| **1** | Prefer highest Local-Pref |
| **2** | Prefer shortest AS-Path |
| **Tie-breaking rules** | |
| **3** | Prefer lowest Origin |
| **4** | Prefer lowest MED |
| **5** | Prefer eBGP over iBGP |
| **6** | Prefer nearest next-hop |
| **7** | Prefer lowest router-ID or oldest route |

Table 4.1: The BGP decision process.

The first attributes used to compare routes are the Local-Pref and the AS-Path (Fig.4.1). In the initial design of the BGP decision process [RL95], the purpose of the Local-Pref attribute was to let the network operator choose the most desirable route while the AS-Path played the role of a route metric. The remaining rules were mainly used to break the ties when the above attributes where not sufficient to elect a single best route. Today, many attributes are used to influence the decision process [QUP+03, CR05]. Different mechanisms can be used to control the outgoing traffic and the traffic entering an AS. In the following subsections, we describe traffic engineering techniques that rely on the manipulation of BGP attributes in order to influence the outcome of the BGP decision process.

### 4.4.1 Control of the outgoing traffic

To control how the traffic leaves its network an AS must be able to choose which route will be used to reach a particular destination through its peers. Since an AS controls the decision process on its BGP routers, it can easily influence the selection of the best path. Two techniques are frequently used.

A first technique is to prefer some routes over others by **setting the Local-Pref** attribute. A common utilization of this attribute is to prefer routes learned from customers over routes learned from providers [GR00]. The Local-Pref attribute is optional and it is only distributed inside an AS. It can be used to rank routes and is the first rule of the BGP decision process (figure 4.1). For example, consider a stub AS with two links toward one upstream provider : a high bandwidth and a low bandwidth link. In this case, the BGP router of this AS could be configured to insert a low Local-Pref to routes learned via the low bandwidth link and a higher value to routes learned via the high bandwidth link. A similar situation can occur for a stub AS connected to a cheap and a more expensive upstream provider. The utilization of the Local-Pref attribute for the purpose of controlling the outgoing traffic of an ISP has been studied in [UBQ03a].

In practice the manipulation of the Local-Pref attribute can be based on passive or active measurements. Recently, a few companies have implemented solutions [Bor02, All02] that allow multi-homed stub ASs and content-providers to engineer their interdomain traffic. These solutions usually measure the load on each interdomain link and some rely on active measurements to evaluate the performance of interdomain paths. Based on these measurements and some knowledge of the Internet topology (either obtained through a central server or from the BGP router to which they are attached), they attach appropriate values of the Local-Pref attribute to indicate which route should be considered as the best route by the BGP routers. Some router vendors also provide this kind of automatic outbound route selection directly into routers. For instance, CISCO proposes its Optimized Edge Routing (OER) [Sys05c].

A second technique, often used by large transit ISPs, is to **rely on the intradomain routing protocol** to influence how a packet crosses the transit ISP. As shown in figure 4.1, the BGP decision process will select the nearest IGP neighbor when comparing several equivalent routes received via iBGP. For example, consider in figure 4.4 that router $R_{27}$ receives one packet whose destination is $R_{45}$. The BGP decision process of router $R_{27}$ will compare two routes towards $R_{45}$, one received via $R_{28}$ and the other received via $R_{26}$. By selecting router $R_{28}$ as the exit border router for this packet, AS2 will ensure that this packet will consume as few resources as possible inside its own network. If a transit AS relies on a tuning of the weights of its intradomain routing protocol as described in [FRT02], this tuning will indirectly influence its outgoing traffic [TSGR04, ANB05].

In addition to this, vendor specific techniques exist. CISCO routers for instance allow to assign an additional attribute called Weight to BGP routes. This attribute can be set on a per-router basis. The route with the highest value of the Weight is preferred by the decision process. The Weight attribute is taken into account before the Local-Pref attribute. However, to the opposite of the Local-Pref attribute, it is not propagated over iBGP sessions. The value of the Weight attribute can be set based on the content of the AS-Path, based on the originator of the route and so on. The setting can be done automatically based on input route filters.

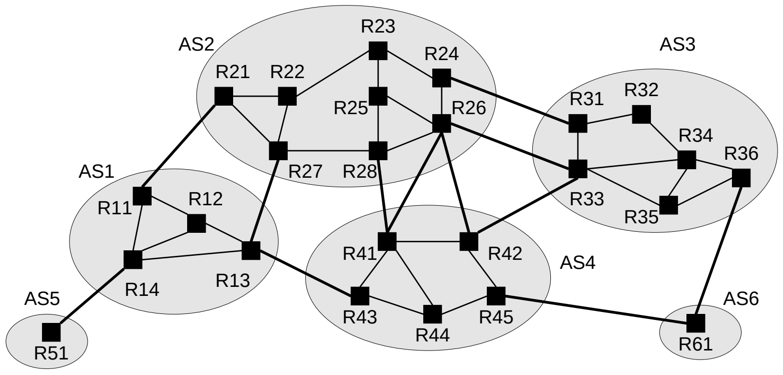


Figure 4.4: A simple Internet

### 4.4.2 Control of the incoming traffic

If an ISP needs to control the traffic in the reverse direction, i.e. entering its network, the situation is far more complex. Indeed, it is not enough for the ISP to control the decision process in its own routers. It needs to influence the decisions made by routers in other domains. A typical example where such control is needed is an access provider which usually has much more inbound traffic than outbound traffic. We describe in the following paragraphs several BGP-based techniques to control the incoming traffic.

The first method that can be used to control the traffic that enters an AS is to rely on **selective announcements** and advertise different route announcements on different links For example in figure 4.4, if AS1 wanted to balance the traffic coming from AS2 over the links $R_{11} - R_{21}$ and $R_{13} - R_{27}$, then it could announce only its internal routes on the $R_{11} - R_{21}$ link and only the routes learned from AS5 on the $R_{13} - R_{27}$ link. Since AS2 would only learn about AS5 through router $R_{27}$, it would be forced to send the packets whose destination belongs to AS5 via router $R_{27}$. However, a drawback of this solution is that if the link $R_{13} - R_{27}$ fails, then AS2 would not be able to reach AS5 through AS1. This is not desirable and it should be possible to utilize link $R_{11} - R_{21}$ for the packets toward AS5 at that time without being forced to change the routes that are advertised on this link.

A variant of the selective announcements is the advertisement of **more specific prefixes** (also known as **prefix splitting**). This kind of advertisement relies on the fact that an IP router will always select in its forwarding table the most specific route for each packet (i.e. the route with the longest matching prefix). For example, if a forwarding table contains both a route toward `16.0.0.0/8` and a route toward `16.1.2.0/24`, then a packet whose destination is `16.1.2.200` would be forwarded along the second route. This fact can also be used to control the incoming traffic. In the following example, we assume that prefix `16.0.0.0/8` belongs to AS3 and that several important servers are part of the `16.1.2.0/24` subnet. If AS3 prefers to receive the packets toward its servers on the $R_{24}$-$R_{31}$

link, then it would advertise both `16.0.0.0/8` and `16.1.2.0/24` on this link and only `16.0.0.0/8` on its other external links. An advantage of this solution is that if link $R_{24}$-$R_{31}$ fails, then subnet `16.1.2.0/24` would still be reachable through the other links.

Another method is to allow an AS to indicate a ranking among the various route advertisements that it sends. The technique called **AS-Path prepending** exploits the fact that the BGP decision process uses the length of the AS-Path to estimate the quality of a route (see Fig. 4.1). A natural way to influence the choice of a neighbor router is therefore to artificially increase the length of the AS-Path of certain routes to make them less preferable. Many network operators use AS-Path prepending on a backup line for instance or to deviate traffic from some neighbors without losing connectivity. Coming back to figure 4.4, assume that `AS3`'s primary interdomain is link $R_{61} - R_{45}$ while link $R_{61} - R_{36}$ is only used as backup primary link. In this case, `AS6` would announce its routes normally on the primary link (i.e. with an AS-Path of `AS6`), but would add its own AS number several times instead of once in the AS-Path attribute (e.g. `AS6 AS6 AS6`) on the $R_{61} - R_{36}$ link. The route advertised on the primary link will be considered as the best route by all routers that do not rely on manually configured settings for the Weight and Local-pref attributes. This technique can be combined with selective announcements. For example, an AS could divide its address space in two prefixes $p1$ and $p2$ and advertise prefix $p1$ without prepending and prefix $p2$ with prepending on its first link and the opposite on its second link.

The last method to allow an AS to control its incoming traffic is to **rely on the Multi-Exit-Discriminator** (MED) attribute. This optional attribute can only be used by an AS multi-connected to another AS to influence the link that should be used by the other AS to send packets toward a specific destination. It should however be noted that the utilization of the MED attribute is usually subject to a negotiation between the two peering ASs and some ASs do not accept to take the MED attribute into account in their decision process. The MED only provides a local control of inbound traffic. We do not study its efficiency in more details in this thesis. The engineering of traffic between neighboring domains has been studied by Winick et al in [WJR02].

### 4.4.3 Community-controlled Route Propagation

Another technique that is becoming very popular for controlling the incoming traffic is to rely on the BGP Communities attribute [TCL96, BQ03]. This optional route attribute is a set of 32 bits integers, each one identifying a BGP "community". A community design a group of routes that share common attributes or that should receive the same treatment. Community values are often used to attach optional information to routes such as a code representing the city where the route was received or a code indicating whether the route was received from a peer or a customer. The presence of certain BGP communities inside a BGP route can influence how this route will be processed by distant routers. Typically, an AS defines, in the configuration of its routers, a list of community values and the actions to perform when a route containing these community values is received. Customers of this AS may attach such communities to the routes they announce to this provider.

Several ISPs have been using the Communities attribute to give their customers a finer control on the redistribution of their routes. The customers were therefore given the possibility to better engineer their incoming traffic by attaching predefined Community values to routes. The typical traffic engineering actions supported by ISPs are listed in Table 4.2. These actions typically apply toward a large AS (e.g. tier-1 or tier-2 ISPs providing transit service), an interconnection point, a country or a continent. Note that although we give names such as NO_ANNOUNCE and PREPEND to these community values, no standard encoding exists and every ISP can define its own community values.

| **Community** | **Requested action** |
|---|---|
| NO_ANNOUNCE | Do not announce the route to specified peer(s) |
| PREPEND | Prepend the AS-Path when announcing the route to specified peer(s) |
| CHANGE_PREF | Set the Local-Pref value in the AS receiving the route [CB96] |

Table 4.2: Typical route redistribution actions available with Communities.

In the first case, the community is attached to a route to indicate that this route should not be announced to a specified peer or at a specified interconnection point. For example, in the left part of Fig. 4.5, `AS2` has configured its routers to not announce to `AS4` routes that contain the `2:1004` community. `AS2` has documented the utilization of this community to its peers so that `AS1` can attach this value to the routes advertised to `AS2` to ensure that it does not receive packets from `AS4` through `AS2`. In a detailed survey of the RIPE `whois` database [BQ03], we have shown that this type of communities was often used by ISPs.

The second type of community is used to request the upstream AS to perform AS-Path prepending for the associated route. The right part of Fig. 4.5 shows how `AS1` uses the `2:3003` and `2:2005` communities documented by `AS2` to request that the AS-Path of the route it announced be prepended twice when announced to `AS3` and `AS5`. To better understand the usefulness of such community values, let us consider again Fig. 4.4, and assume that `AS6` receives a lot of traffic from `AS1` and `AS2` and that it would like to receive the packets from `AS1` (resp. `AS2`) on the $R_{45}$-$R_{61}$ (resp. $R_{36}$-$R_{61}$) link. `AS6` cannot achieve this type of traffic distribution by performing prepending itself. However, this would be possible if `AS4` could perform the prepending when announcing the `AS6` routes to external peers. `AS6` could thus advertise to `AS4` its routes with the community `4:5202` (documented by `AS4`) that indicates that this route should be prepended two times

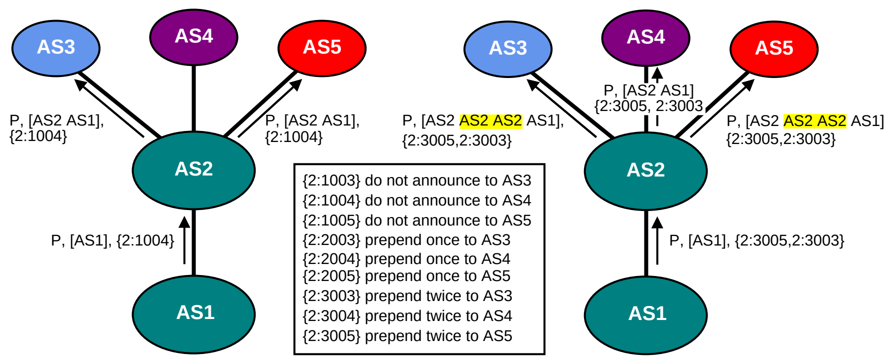

AS3
AS4
AS5
P, [AS2 AS1], {2:1004}
P, [AS2 AS1], {2:1004}
AS2
P, [AS1], {2:1004}
AS1
{2:1003} do not announce to AS3
{2:1004} do not announce to AS4
{2:1005} do not announce to AS5
{2:2003} prepend once to AS3
{2:2004} prepend once to AS4
{2:2005} prepend once to AS5
{2:3003} prepend twice to AS3
{2:3004} prepend twice to AS4
{2:3005} prepend twice to AS5
AS3
AS4
AS5
P, [AS2 AS1] {2:3005, 2:3003}
P, [AS2 AS2 AS2 AS1], {2:3005,2:3003}
P, [AS2 AS2 AS2 AS1] {2:3005,2:3003}
AS2
P, [AS1], {2:3005,2:3003}
AS1

Figure 4.5: Example of communities-controlled route redistribution.

when announced to AS2.

Finally, the third common type of community used for traffic engineering purposes is to set the Local-Pref attribute in the upstream AS as described in [CB96].

### 4.4.4 Discussion

The above sections have described several manipulations of the BGP attributes that are used by ISPs to engineer their interdomain traffic. However, there are some limitations to be considered when deploying those techniques.

A first point to note is that the control of the outgoing traffic with BGP is based on the selection, among the available routes, of a single best route. This selection can be performed on the basis of various parameters, but it is limited by the diversity of routes received from upstream providers which depends on the connectivity and the policy of these AS. However, as shown in [UBQ03a], outgoing traffic engineering based on the manipulation of BGP attributes is possible at a reasonable cost.

On the opposite, the control of the incoming traffic is based on a careful tuning of the advertisements sent by an AS. This tuning can cause several problems. For instance, announcing the prefixes selectively on peering sessions does not guarantee connectivity to the prefixes when a session fails. The selective announcements technique is thus not robust. One solution could be to announce more specific prefixes. However, an AS that advertises more specific prefixes or has divided its address space in distinct prefixes to announce them selectively will advertise a larger number of prefixes than required. All these prefixes will be propagated throughout the global Internet and will increase the size of the BGP routing tables of potentially all AS in the Internet. [BNC02] reports that more specific routes constitute more than half of the entries in a BGP table. Faced with this increase of their BGP routing tables, several large ISPs have started to install filters dropping advertisements for small prefixes, in order to avoid an unnecessary growth of

BGP routing tables [BBGR01]. At the time of this writing, filtering prefixes longer than 19 bits for some classes of addresses is a common and encouraged practice among ISP operators, making the technique of more specific prefixes less effective [Veg05]. The MED attribute can only be used when there are multiple physical links between two ASs and not in the case of stub ASs multi-homed to several providers, a very common situation today [ACK03].

The only remaining techniques for controling the inbound traffic of an ISP are AS-Path prepending and BGP communities. Despite the importance of traffic engineering for ISPs, there have been few studies on the efficiency of these techniques.

## 4.5 Evaluation of AS-Path prepending

According to a detailed analysis of BGP routing tables presented in [BNC02], AS-Path prepending is a widely used technique. Broido et al [BNC02] reveals that 6.5% of routes were affected by prepending in November 2001. Strangely, the efficiency of this technique has never been studied on a wide-scale basis. We performed a first simulation-based study of AS-Path prepending in [BTP$^+$03]. Later, measurement studies were performed in parallel in [QPBU05] and [CL05]. In [QPBU05], we have studied AS-Path prepending from a real stub AS and have shown that the granularity of AS-Path prepending is coarse. Prepending only once or twice can already shift a large fraction of the traffic. This means that in practice, AS-Path prepending can be used to indicate that a backup link should be avoided whenever possible, but using it for load-balancing purpose is difficult. In [CL05], Chang and Lo proposed an automated AS-Path prepending technique. They evaluated from a non commercial site. They concluded that their technique works but requires intrusive active measurements and is not fine grained. Finally, an optimized AS-Path prepending technique was proposed in [GDZ05]. A recent study by Yang et al [YXW$^+$05] has investigated uncoordinated traffic engineering and derived guidelines ensuring stability without global coordination.

However, the results obtained from small simulation studies or considering a single measurement point may depend on the actual location of the stub AS and cannot easily be generalized to the whole Internet. In this section, we present the results of a simulation study of the AS-Path prepending performance in a model of the Internet topology.

### 4.5.1 Evaluation model

To evaluate the efficiency of AS-Path prepending as a traffic engineering technique, we rely on simulations. We use the C-BGP routing solver described in Chapter 2 as well as an AS-level Internet topology. This topology was inferred from real BGP routing tables gathered from multiple vantage points by Subramanian et al. [SARK02]. The topology we used for this study is dated from January 9th, 2003. It contains 14695 domains and 30815 interdomain links. There is at most one link between two different domains. We model each domain with a single BGP router and routing policies based on the economical relationships, determined by [SARK02], which exist between the domains. To our knowledge, no simulation study has been able to analyze the impact of the routing policies on large networks composed of thousands of routers with routing policies. Most simulation studies only consider a few tens or sometimes a few hundred of routers. Given the importance of the routing policies, we choose to model them realistically. Memory constraints and the impossibility of inferring the internal topology of each AS from the available routing tables [SARK02] forced us to consider a single router inside each AS (see Fig. 4.6).

The routing policy of our BGP routers is composed of two parts. The first part is the so-called *selective export rule* [Gao00] which governs the provision of transit service. One domain provides a full transit service to its customers, a limited transit service between its customers and its peers but never between its providers and its peers. In our simulations, we configured each BGP router with the routing policies corresponding to the relationships with each of its peers. The second part of the policy introduces a preference among routes learned over different relations [Gao00]. The routes learned from customers are preferred over routes learned from peers which in turn are preferred over routes learned from providers. The reason for such preferences is that providers do not have to pay their customers to carry traffic. This also ensures that interdomain routing will converge [GR00].

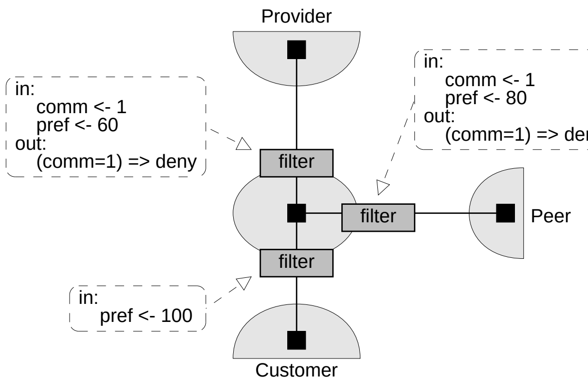


Figure 4.6: Model of an AS with policies.

These policies can be implemented easily with the help of filters. We show the detail of these filters in Fig. 4.6. Routers will be configured to mark each routes received from a peer of a providers with a Community. This Community is used to prevent the redistribution of these routes to other peers and providers. The preference for routes received from customers, then from peers and finally from providers is implemented in our simulations by relying on the `Local-Pref` attribute. A Local-Pref value of 100 is assigned to routes received from customers, a value of 80 for routes received from peers and a value of 60 for routes received from providers.

Although the January 9th topology was the most accurate publicly available map of the global Internet, it has several limitations. First, in this topology each AS is modeled as a single node connected to neighboring ASs. In reality, an AS may contain up to several hundred of routers and there may be more than ten different physical links between two ASs although these links appear as a single edge in the inferred topology. Furthermore, the heuristic used to infer the routing policies is limited by two factors. First, it relies on a small set of BGP tables, typically collected at large tier-1 ISPs and those tables do not contain all interdomain links. Second, the inferred routing policies are not always correct.

We compute the interdomain paths for each prefix independently. This does not change the outcome of the simulation since we do not consider BGP route aggregation. The selection of the interdomain paths towards each prefix is thus independent. This makes possible to easily distribute the computation on different CPUs, without adding much overhead.

### 4.5.2 Importance of the tie-breaking rules

Before analyzing the simulations of AS-Path prepending, it is important to understand how the BGP decision process selects the best path towards each destination.

For this purpose, we perform simulations with the model described in Section 4.5.1. We instrumented the simulator to record, for each best route selected by a BGP router, the specific rule of the BGP decision process which was locally responsible for its selection. We then use this information to determine the importance of the different rules in the BGP route selection.

We perform 14695 simulations. In each simulation, a different domain announces a single prefix. We then count for each domain the number of routes selected by each rule of the decision process to join the announced prefix. For each source domain and each destination prefix, there are 4 possibilities. First, the domain has received a single route toward the prefix and the decision process is not applied. Second, the domain has received one route with a highest preference, thus the rule *local-pref* is accounted. Third, there are more than one route with the highest Local-Pref but one among them has a shorter AS-Path, thus the rule *shortest-path* is accounted. And finally, if there is still more than one route after the *shortest-path* rule, the *tie-break* is accounted.

We classify these results based on the level of the domain in the Internet hierarchy as identified by [SARK02]. There are 5 levels in the Internet hierarchy. The first level is the core of the Internet and contains a full-mesh of large transit domains (tier-1's). The second level contains large transit networks (tier-2's) deeply interconnected. The third and following levels contain smaller regional providers and stub domains.

Fig. 4.7 presents the results of these simulations. On the x-axis we show the 5 levels of the Internet hierarchy identified by [SARK02]. The y-axis shows the relative importance of the rules, for each level of the hierarchy.x There is a bar for each considered rule: *local-pref*, *shortest-path* and *tie-break* as well as a bar for *single route*. The latter gives an idea of the importance of networks which only receive a single route to reach a destination and which thus do not apply the decision process.

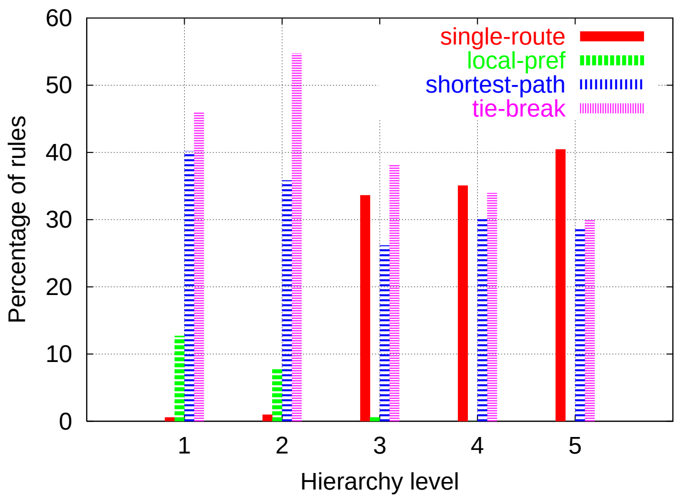


Figure 4.7: Importance of each rule of the BGP decision process at different levels of the Internet hierarchy.

The simulation results shown in Fig. 4.7 reveal several interesting results. First, we can observe that between 30 and 45% of the tier-3 to tier-5 ASs only receive a single route to each destination. Those ASs are singled-homed ASs. For the tier-3 to tier-5 ASs, about 30% of the BGP routes are selected on the basis of the length of their AS-Paths. The remaining 30% of the routes are selected on the basis of the tie-breaking rules. For stub ASs, those tie-breaking rules often correspond to the router ID step of the BGP decision process. Note that in reality, the *local-pref* rule may be used more frequently in stubs for backup or traffic engineering reasons, but this is not modeled in our simulations.

Concerning the tier-1 and tier-2 ASs, 10% of the routes selected by those ASs are chosen on the basis of their Local-Pref attribute. Then, about 40% of the routes are selected on the basis of their AS-Path. Finally, in the tier-1's and in the large national transit domains, about 50% of the routes are selected on a tie-break basis. This is due to the large number of interconnections that exist between all these domains and thus to a large number of alternative routes with a similar AS-Path length. For the transit domains, the tie-break rules correspond to the third, fourth and fifth step of the BGP decision process (Fig. 4.1).

A consequence of the importance of the tie-breaking rules in the BGP decision process is that it is difficult to predict which best route will be selected in a distant AS. The selection of the best route depends on information that is not available outside the local AS. Indeed, in the first tie-breaking rule, the value of the MED is only visible between neighboring ASs. In the second tie-break rule, if one eBGP route exists, the iBGP routes are removed from consideration. The outcome of the IGP metric rule depends on the internal IGP cost allocation policy of the considered domain. This information is usually confidential. Although some researchers have used *traceroutes* to infer the IGP costs of internal links in transit ISPs [SMW02], their accuracy appears to be limited [TMSV03]. In the final tie-breaking rule, we must know the Router-ID or the IP addresses of the involved routers, if the implementation relies on this information. Again, this is often kept secret by network operators. On the other hand, if the final tie-break keeps the oldest route, this decision is non-deterministic.

Since the tie-breaking rules are widely used in the BGP route selection, it is hard for an AS to evaluate how the traffic will enter the AS. Moreover, this also shows that the ASs often receive routes with the same AS-Path length for each destination prefix. We can already guess that this will influence the efficiency of AS-Path prepending. By increasing the length of the AS-Path for a route to one provider of a dual-homed stub, the route announced through the other provider is preferred by all ASs that used the tie-breaking rules for this destination. Consequently, a lot of incoming routes are likely to move to the preferred provider because the tie-break is used for more than 30% of the routes.

### 4.5.3 Evaluation of AS-Path prepending

The aim of this section is to generalize our observations on the control of the routes entering our experimental AS with AS-Path prepending. Therefore, we perform simulations with the topology described in Section 4.5.1, which captures a large portion of the real Internet. In the simulations we are not limited to a single dual-homed stub. We can obtain results similar to [QPBU05] for each dual-homed stub in the topology. For each dual-homed stub, we study the use of AS-Path prepending to control how the other ASs reach the stub.

We rely on dual-homed stub domains to easily evaluate the impact of prepending on the distribution of the routes on their two upstream providers. These stubs represent 82% of the multi-homed stub ASs in the considered topology. The 5841 dual-homed stubs consist of more than 39% of the ASs in the January 9th topology. Single homed stubs are not considered since they do not have the possibility to engineer their traffic on multiple interdomain links. Stubs with more than 2 providers are less frequent. We do not consider them in this study because it is difficult to present graphically the simulation results for such multi-homed stubs.

We use the simulation model presented in Section 4.5.1. For each considered stub, we determine how it is joined by all the other domains when no prepending is used. Then we compute for each stub the distribution of the paths via their two providers. We call it the “default” distribution. This distribution is plotted in Fig. 4.8. To present the results graphically, we defined an ordering among the providers. Each of our stubs has a well connected provider and a less connected provider. To determine what is the less connected provider of a dual-homed stub, we associate to each domain a ranking based on the following degrees: the number of providers of the domain, the number of peers and the number of customers. This ranking is a lexicographic order on ($\langle num\_prov,\ num\_peer,\ num\_cust \rangle$) to define the importance of a domain. This ordering has one exception for tier-1 domains in the core that do not have providers. When two domains have to be compared, if one is in the core and the other is not, the domain in the core is considered more important. Otherwise, the domain which has more providers is more important. If the number of providers is equal, we compare the number of peers, then the number of customers if required.

On the x-axis of Fig. 4.8, we show the percentage of paths that cross the less connected provider. The y-axis of Fig. 4.8 shows the number of stubs which have the same distribution of paths. We can observe that when no prepending is done, there is no clear tendency in favor of one of the two providers. Some stubs receive most of their interdomain paths via their most connected provider, others receive the same number of paths via each provider, while some stubs receive most of their paths through the less connected provider.

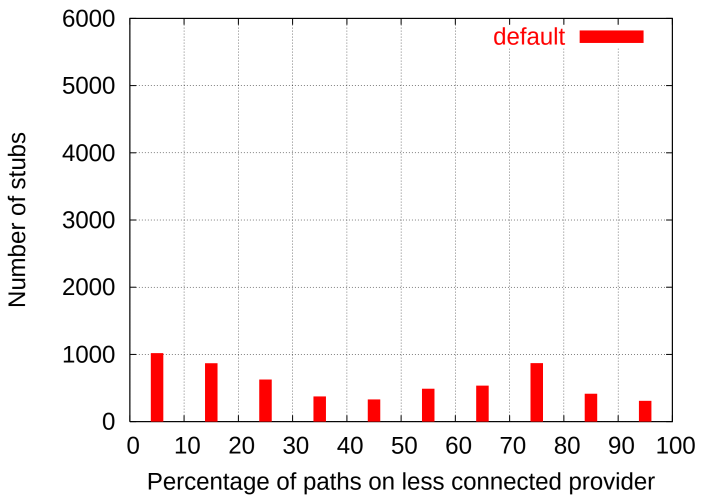


Figure 4.8: Default relative distribution of paths on the less connected provider.

We then perform simulations where each dual-homed stub selectively prepends the AS-Path toward its less connected provider and toward its more connected provider. For each stub and for each of their providers, we use three different amounts of prepending: 1, 2 and 7. The results of these simulations are shown in Fig. 4.9. The top plot shows the impact of prepending towards the less connected provider while the bottom plot shows the impact of prepending towards the most connected provider.

The first important result that one can draw from these simulations is that **the effect of prepending is coarse**. On average, prepending once toward one provider already moves a large fraction of the paths away from this provider. The granularity of AS-Path prepending is thus extremely limited. So is its interest for traffic engineering.

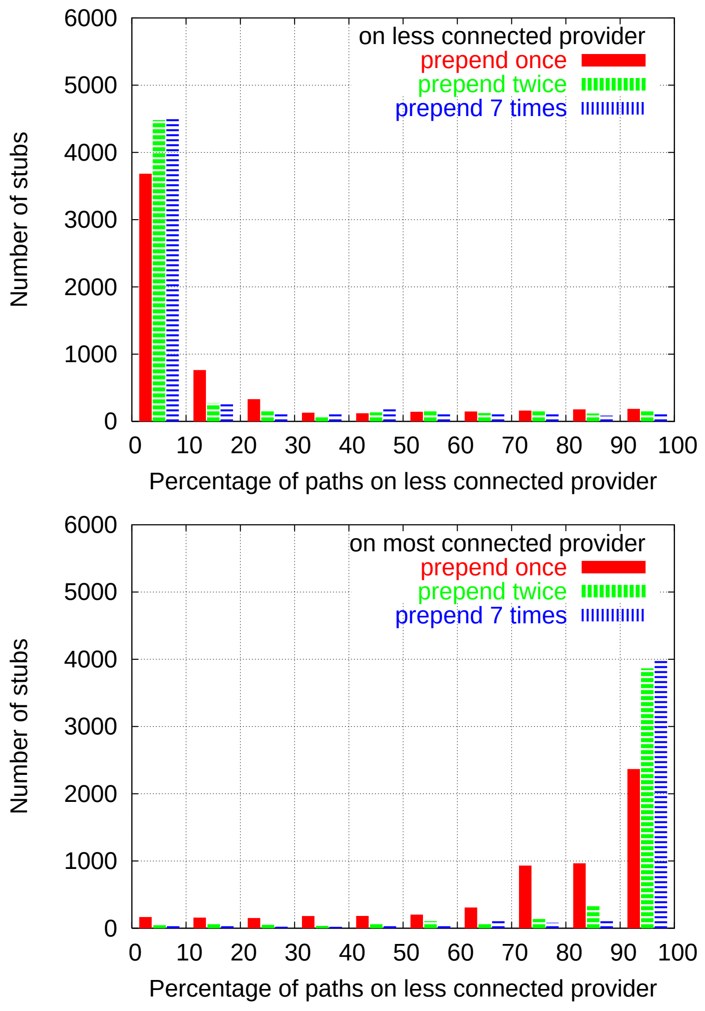


Figure 4.9: Percentage of paths on the less connected providers for various amount of prepending.

The second conclusion one can draw from these simulations is that **the marginal benefit of prepending decreases quickly**. One can see that prepending once moves a lot of paths. Prepending twice still moves a lot of paths away. But the difference is minor between prepending twice and prepending 7 times. Furthermore, prepending too much can be a problem because inflated AS-Paths require an increased amount of memory in routers.

Third, **the efficiency of prepending is highly uncertain** and depends on the location of the stub's providers as well as the relationships that these providers have with other domains. There are stubs for which paths can be moved easily from one provider to another, other stubs for which it is easier to move path from one provider to the other than the other way around and even stubs for which a very large part of the paths cannot be moved independently of the amount of prepending.

We further examine a few cases that happened in our simulation study to clarify how the topology and business relationships influence the efficiency of AS-Path prepending. Fig. 4.10 presents two different stubs from the topology we have used and shows how the connectivity of their providers constrain the efficiency of prepending. First, on Fig. 4.10(a), the stub `AS3748` has two providers, `AS3786` and `AS4766` which have a similar connectivity. They both have many *customer-to-provider* relations with domains in the core. The default distribution of incoming paths on the stub's access link is thus balanced: approximately 50% is received through each provider. This is due to the similar distance of the stub to the rest of the Internet through both providers. When prepending is used once toward `AS3786`, the percentage of paths which reach the stub through it decreases to 10%. This is explained by the distance of the stub which quickly becomes longer through provider `AS3786` making the alternate path preferred. When prepending twice, this percentage falls to nearly 0%. The behavior is similar when prepending is used toward `AS4766`. After prepending once, the percentage of paths through `AS4766` decreases to 3%. After prepending twice, it is close to 0%.

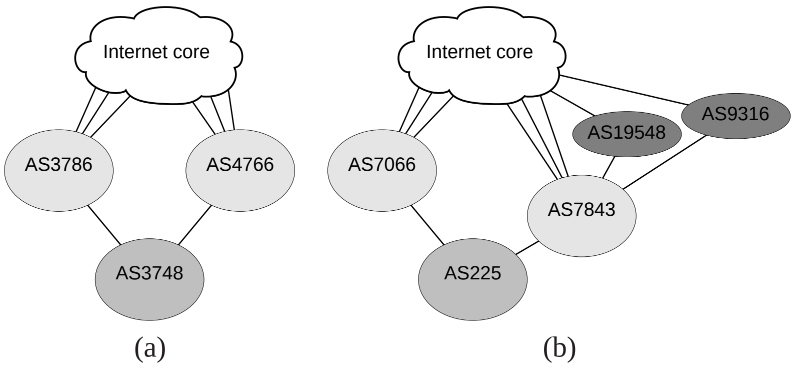


Figure 4.10: Topology and policies impede on the AS-Path prepending efficiency.

The second example, shown in Fig. 4.10(b), shows a stub which has providers of different importance. The less connected provider, `AS7066` has a single *customer-to-provider* link to `AS1239` in the core. It also has a few customers. On the contrary, the second provider, `AS7843` has three *customer-to-provider* relationships with `AS1`, `AS209` and `AS701`, all in the Internet core. It also has two other relationships with minor providers and a few customers. Here, the default incoming path distribution is already unbalanced: 15% pass through `AS7066` while 85% pass through `AS7843`. This is due to the choices of the domains in the core. They select the shortest path to the stub and re-advertise it to their clients and peers. In this case, the efficiency of AS-Path prepending differs when it is used toward the less connected provider or toward the more connected provider. After prepending once and twice toward AS7066, the percentage of incoming paths received through this provider becomes respectively 12% and nearly 0%. On the contrary, it is not possible to move all the incoming paths away from the other provider, AS7843. The results of prepending once, twice and 7 times give the following percentage of paths: 75%, 67% and 50%.

Another example is given in Fig. 4.11. Here, the stub, AS17049 is also connected to two providers of different importance. The less connected provider is a priori AS6467 because it is not in the core while the other provider, AS1239 is. However, AS6467 has an excellent connectivity with domains in the core, such as AS1, AS701, AS7018 and also AS1239. Moreover, these domains (except AS1239) will prefer the routes learned from AS6467 which is a customer over the routes received from AS1239 which is a peer whatever the AS-Path length is! This is why after prepending only twice toward AS1239, there are already no more paths passing through it. On the contrary, prepending toward the other provider, AS6467, hardly moves a lot of paths. Even after prepending 7 times, the percentage of paths which reach the stub through AS6467 is still more than 58%.

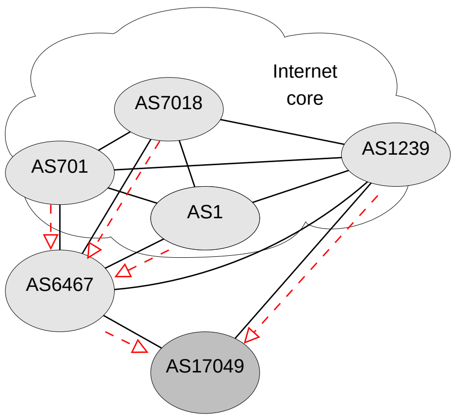


Figure 4.11: Topology and policies impede on the AS-Path prepending efficiency.

## 4.6 Evaluation of Communities

Using the Communities attribute to perform incoming traffic engineering can provide a finer granularity than AS-Path prepending or selective announcements. Although the Communities attribute is widely used in the Internet today as indicated in our surveys [BQ03], its utilization for the purpose of engineering the traffic of an ISP has not been studied yet. In [QTUB04], we have shown that the utilization of Communities for traffic engineering purposes relies on an ad hoc definition of

Community values and on manual configurations of BGP filters which makes it difficult to use and subject to errors. In addition Communities with a local meaning were often uselessly propagated to the entire Internet. To solve these problems a number of standard encoding of Communities have been proposed: Redistribution Communities [BCH$^+$03] and Proxy Communities [AG04]. More recently, we have published a measurement study of Communities-based traffic engineering performed from a single stub domain [QPBU05].

In practice, it can be expected that those Communities will be used to influence the redistribution of routes towards large transit ISPs with a large number of customers. Consider for example the case of YUCOM, a European dual-homed ISP. Like many other ISPs [SARK02], this ISP has two major upstream providers that allow it to reach the entire Internet. Fig. 4.12 provides a closer look at the tier-1 providers and peers on the basis of the BGP advertisements received by the studied ISP. In this figure, we show a small subset of the interdomain topology and the number of distinct AS advertised through the three largest tier-1 ISPs.

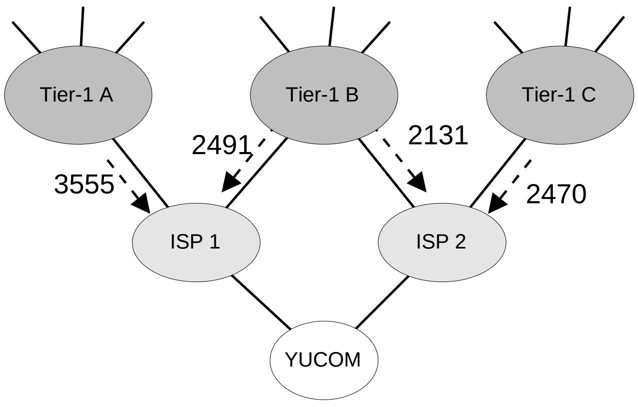


Figure 4.12: Part of the interdomain topology seen from YUCOM

Fig. 4.12 reveals two interesting informations. First, each tier-1 ISP provides connectivity and thus announces routes towards a large number of AS. In total, the three largest tier-1 providers announce prefixes from more than 8500 AS. Second, the studied ISP learns routes about more than 2000 AS attached to tier-1 $B$ via its two upstream providers. By using Communities targeted at those large tier-1 ISPs, our ISP could influence the redistribution of its routes to a large number of AS with only a few Communities. For example, the studied ISP could utilize a single Community to request its first upstream provider to announce its local routes with AS-Path prepending only towards tier-1 $B$. The result of this modified advertisement by the first provider will be that the traffic coming from AS attached to tier-1 $B$ would be received through the other provider.

This is a good news for Community-based traffic engineering. The classical BGP Communities or the Redistribution Communities being developed by the

IETF citedraft-red-comm-grow-00:2003 **can be used to achieve a finer control on the incoming traffic than AS-Path prepending**. However, it should be noted that they suffer from three important drawbacks.

The first drawback is that, given our limited knowledge of the Internet topology and the routing policies used by distant ASs, **it is difficult to predict the impact of a given Community value**. For example, consider Fig. 4.13 and assume that the stub AS attaches to its route advertised to ISP2 a Community indicating that ISP2 should not advertise the route to tier-1 B. In this case, tier-1 B will not use its link with ISP2 to reach the stub. From Fig. 4.13, tier-1 B will send its packets to either tier-1 C or tier-1 A. In the first case, the Community used by the stub does not have any effect on the packets received by the stub. Furthermore, the sources that are downstream of tier-1 B will recompute their best route to reach the stub and some of them may use tier-1 C instead of tier-1 B to reach the stub while others will utilize other paths. Given our limited knowledge of the Internet topology, it is very difficult to predict the decision that all those ASs will take. Note that if prepending towards tier-1 B was requested by the stub, the Community would not have any impact since tier-1 B prefers the direct routes received from its customers over routes through its peers.

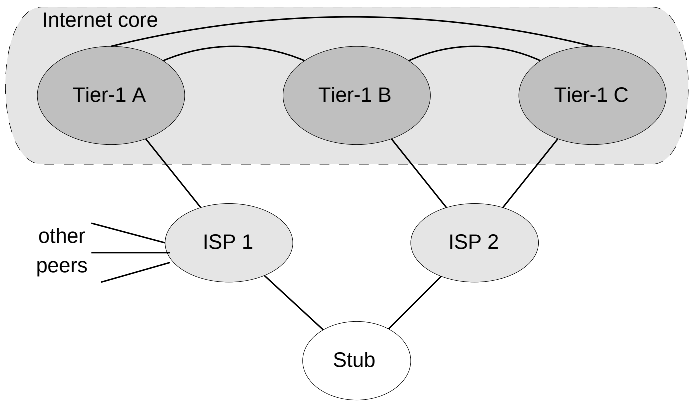


Figure 4.13: Influence of the topology and the business relationships on the efficiency of Communities.

A second drawback of the BGP communities is that **the impact of one Community on the incoming packet flow will depend on whether it is associated with other Communities or not**. For example, consider the right part of Fig. 4.10. Assume that AS17049 uses a Community to request AS6467 to not announce its route to AS701. In this case, AS701 may update its BGP routes and use its peering link with AS1 to reach AS17049 via AS6467. Thus, the Community has no effect on the packet flow as seen by AS17049. However, if this community is used together with a Community requesting AS6467 to not advertise the route to AS1, then both AS701 and AS1 will probably use AS1239 to reach AS17049.

Finally, the last drawback of the utilization of Communities is that a typical AS will need to choose among a **large number of different Communities**. For example, consider the Redistribution Communities [BCH$^+$03] that allow a stub to influence the advertisement of its routes to the peers of its peers. The number of available Redistribution Communities depends on the number of ASs that are two AS-hops away. For Belnet, there are 1729 distinct ASs at two AS-hops.

In practice however, it can be expected that Redistribution Communities will mainly be used on customer-provider links. Fig. 4.14 shows the cumulative distribution of the links at 2 hops for the multi-homed stub ASs in the topology from [SARK02], on January 9th, 2003. This gives us an idea of the number of Redistribution Communities that could be used by at a multi-homed stub. The first curve (on the left) gives the cumulative number of multi-homed stubs with a given number of links with providers at 2 hops. The second curve concerns the number of peer-to-peer links at two hops. The third curve shows the number of links, at 2 hops, with customers that are single-homed. The last curves give the number of peerings with single and multi-homed customers at 2 hops and the total number of peerings at 2 hops.

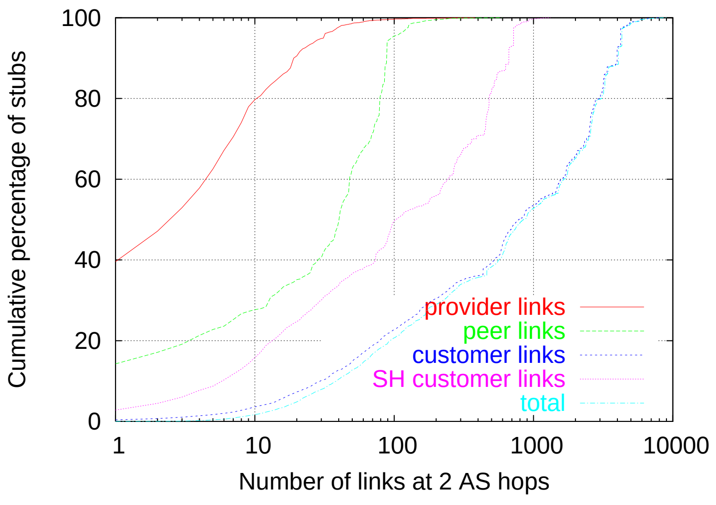


Figure 4.14: Importance of different business relationships at 2 AS hops from multi-homed stub domains.

Twenty percents of the multi-homed stubs have more that ten peerings providers at two hops. These stubs can use $2^{10} = 1024$ sets of Communities to engineer their incoming traffic with Communities influencing the redistribution of their route toward providers only. Sixty percent of stubs have more than 500 links at 2 hops. This implies that a a lot of combinations of Communities exist to engineer the traffic of these stubs even if we exclude the Communities targeting single-homed customers since this traffic cannot be moved with Redistribution Communities.

## 4.7 Approaches not relying on tweaking BGP

In addition to the traffic engineering techniques that we have discussed in the previous sections, there are other proposals to control the interdomain traffic that do not necessarily rely on tweaking BGP attributes.

A first example is the utilization of **end-systems-based overlay networks** such as RON [ABKM01]. In those approaches, overlays are established between end-systems based on collected measurements. The overlay is implemented by using IP tunnels. These approaches require that end-systems be modified. In addition, these approaches require the establishment of a large number of tunnels since they work at the level of flows or source/destination pairs. Another similar approach is the Detours proposed in [SAA+99]. Detours also relies on tunnels established between routers and assumes that end-systems are able to select the appropriate detour router.

In [GCLC04], a load-balancing system based on **dynamic Network Address Translation (NAT)** is proposed and evaluated. The source addresses of outgoing packets are translated to one of the external addresses of a NAT box connected to multiple providers. The external address chosen by the load-balancing system depends on which link the returning traffic is assigned. This system allows to control the incoming and the outgoing traffic for small enterprise networks. The offered control is fine-grained since it can be done at the level of layer-4 flows.

In addition, there are also proposals to bring drastic changes to interdomain routing. For instance, the **Overlay Policy Control Architecture (OPCA)** [ACK03] considered the use of a separate protocol to carry control information. The idea lying behind OPCA is to separate routing and routing policy. This leads to using another protocol in an overlay network to handle changes in routing policy. The applications of such an architecture encompasses improving the time of route failover as well as allowing multi-homed stubs to control their incoming traffic.

Finally, [Yan03] proposes a **New Internet Routing Architecture (NIRA)** to allow hosts to select the transit ISPs used to reach a destination. NIRA includes methods that allow individual hosts to discover the topology, i.e. the existing routes to the destination, as well as to discover the availability of these routes, i.e. whether they can currently be used or not. [Yan03] discusses the challenges of designing and deploying such a new interdomain routing protocol. Another approach is described in [KKW+03] where the BANANAS framework is proposed. This framework aims at providing means of exploiting the multiplicity of paths available in the Internet.

## 4.8 Conclusion

In today's Internet, ASs often need to control the flow of their interdomain traffic, for cost or performance reasons. In this chapter, we have explained why it is difficult for an Autonomous System to control the flow of its incoming traffic with the current Internet architecture based on BGP.

We started with a survey of BGP-based traffic engineering techniques. We have shown that these techniques rely on influencing the decision process of BGP routers. We distinguished the control of the outgoing traffic from the control of the incoming traffic. Both techniques are different in that they manipulate different attributes of the BGP routes, but also due to the scope of their influence. Techniques focusing on the outgoing traffic rely on decisions taken by BGP in the local domain while techniques trying to control the incoming traffic need to influence the BGP decision process in distant domains, which is far more difficult.

We summarize our conclusions in Table 4.3. We show for each technique which direction of the traffic it can control (2$^{nd}$ column) as well as the scope of the technique (3$^{rd}$ column). The remaining columns indicates different qualities of the techniques. Firstly, we indicate if the technique is *Predictable*, i.e. if its routing impact can be predicted. Indeed, to accurately control the flow of its incoming packets, an AS should be able to predict which route will be selected by distant ASs. Secondly, we indicate if the technique is *Scalable*, i.e. if the technique can be used given the number of routes in the Internet today and its expected growth. Thirdly, we indicate if the method is *Robust*, that is if it does not impede the current robustness of the interdomain routing system. The last column gives comments on the efficiency of the technique.

The BGP-based techniques available to control the outgoing traffic work by influencing the routers in the local domain. All these techniques are deterministic, scalable and robust.

The conclusions for the BGP-based techniques considering the incoming traffic are more variate. First, the selective announcements are not robust since the failure of an access link will cause the complete withdrawal of some prefixes. The announcement of more specific prefixes is not deterministic since it is sensitive to filtering by distant ASs. In addition, this technique is not scalable since it increases the number of prefixes in the BGP routing tables. The MED attribute can only be used to control the traffic received over multiple links with a neighbor AS. In addition, this technique requires an agreement with the neighbor AS. Finally, using the MED attribute might lead to oscillations difficult to debug [GW02a], so its robustness is questionable.

We showed by simulations that AS-path prepending, although a widely used technique, provides a too coarse and non predictable control on the incoming traffic. This prediction is difficult for two reasons. First, our knowledge of the Internet topology and the routing policies is incomplete. Second, even with a detailed topology, it would still be very difficult to predict the outcome of the tie-break rules of the BGP decision process. In practice, AS-Path prepending can be used to indicate that a backup link should be avoided whenever possible, but it is difficult to use it to balance the incoming traffic.

An alternative approach is to rely on techniques based on special BGP communities [QTUB04, AG04]. Those techniques provide a finer control on the incoming traffic. Unfortunately, they are difficult to use in practice due to the incomplete

| | **Traffic** | **Scope** | **Predictability** | **Scalability** | **Robustness** | **Efficiency** |
|---|---|---|---|---|---|---|
| **BGP-based approaches** | | | | | | |
| **Local-Pref** | Out | Domain | ✓ | ✓ | ✓ | |
| **IGP weights** | Out | Domain | ✓ | (✓) | ✓ | |
| **Sel. announcements** | In | Internet | ✓ | | | Not robust to access link failure. |
| **More spec. prefixes** | In | Internet | | | ✓ | Sensitive to filtering |
| **MED** | In | Neighbor(s) | ✓ | ✓ | (✓) | Requires bilateral agreement(s) |
| **AS-Path prepending** | In | Internet | | ✓ | ✓ | Limited granularity (given the diameter of the Internet). Impact difficult to predict. |
| **Communities** | In | Internet | | ✓ | ✓ | Impact difficult to predict. Large search space. |
| **Non BGP-based approaches** | | | | | | |
| **RON, Detours** | In/Out | Internet | ✓ | | ✓ | Require modifications to end-systems. Rely on a large number of IP tunnels. |
| **NAT** | In | Internet | ✓ | | | Target multi-homed enterprise networks. Poses problem when one access link fails. |
| **New architectures** | In/Out | Internet | ✓ | ✓ | ✓ | Difficult to deploy in the current Internet. |

Table 4.3: Summary of traffic engineering methods.

view of the whole Internet that an operator has and to the combinatorial explosion of possibilities given the number of distinct BGP communities that an AS can use.

Based on our analysis, the current BGP-based techniques are not appropriate to control the incoming packet flows. Changes to the Internet architecture are thus necessary to achieve such control. Different alternative architectures have been proposed in the literature such as OPCA, NIRA and BANANAS, but these architectures require that all the BGP routers in the Internet be updated which would require several flag days. The same conclusion can be drawn for end-systems-based alternatives such as RON or Detours, since they would require that all the endsystems be updated. In addition, these end-systems-based approaches scale less since they work at a finer granularity (flows or source/destination pairs). Finally, we do not consider the NAT-based system to be applicable for large stub ASs such as broadband access providers. NAT-based systems are targeted at small enterprise networks. In addition, they pose problems when an access link connected to the NAT-box fails since the DNS must be updated.

Therefore, we can conclude that no technique can be used today by a large stub domain to control its incoming traffic in a fine, scalable and predictable way.

# Chapter 5

# Cooperative Traffic Engineering

## 5.1 Introduction

In this chapter, we propose a novel approach to allow a stub AS to control its incoming traffic in a **deterministic** manner. Moreover, it is **scalable** and **can be deployed** in the Internet today with little effort. Our approach relies on the establishment of **Virtual Peerings between cooperating ASs**. A Virtual Peering allows a destination AS to request a source AS to send its packets via a chosen ingress router in the destination AS. Our solution can be used by ASs willing to load-balance their incoming traffic, use low-latency paths, high-bandwidth paths or even to decrease the cost of their interdomain traffic. We focus on stub ASs such as content-providers, enterprise networks and broadband access providers that produce and sink most of the traffic in the Internet. Though our solution could also be used in the case of transit ASs, controlling the incoming traffic in large transit ASs is a different problem that is outside the scope of this thesis.

Our solution slightly modifies the BGP protocol. To ensure that those modifications can be deployed incrementally, we do not require transit ASs to support our extensions. The only affected routers are located within the cooperating source and destination. This is a key contribution of our solution.

Virtual Peerings can be used to achieve various types of traffic engineering objectives such as balancing the load of traffic, preferring the lowest delay paths, the highest bandwidth paths or reducing the cost of traffic. We investigate the utilization of Virtual Peerings to solve two different traffic engineering problems. The first problem examined, is **balancing the load of traffic** received by a stub domain on its access link. As described in Chapter 1, the traffic of a multi-homed stub domain is often unbalanced. When a stub AS is connected to two transit providers it may, depending on the BGP configuration of its providers, receive 80 or 90% of its traffic through one provider. This imbalance may lead to congestion and packet losses on the access links. To avoid this congestion, stub ASs need to move some incoming traffic flows between providers to obtain a better traffic balance.

The second problem examined is the **selection of paths with a better quality**. As BGP does not use any metric besides the length of the AS-Path [HFP$^+$02, BNC02], the choice it makes is seldom the best when another QoS metrics matters. There are cases were two ASs would like to rely on another metric such as the delay or the available bandwidth to perform their route selection. For instance, two ASs may want to select the interdomain path with the lowest latency between VoIP gateways. Another example is the case of two National Research and Education Networks (NRENs) which host laboratories that must exchange huge data files such as telescope images. They might want to find the route that has the largest available bandwidth to proceed to the file transfer. The same requirement applies to GRID computing where large volumes of data must often be exchanged between centers that own computation power. In Section 5.5, we focus on the selection of paths with the smallest delay.

The chapter is organized as follows. First, we introduce in Section 5.2 the concept of Virtual Peerings and we give a brief sketch of the proposed approach. In Section 5.3, we present the architecture in more details. We define the components involved in the Virtual Peerings and the messages they exchange. At the end of Section 5.3, we discuss three important technical issues related to Virtual Peerings: security, robustness and deployment. We show that these issues can be overcomed with today's Internet. In Section 5.3.7, we present a simulation study that we performed to evaluate the benefit of Virtual Peerings in term of path diversity. In Section 5.4 and Section 5.5, we evaluate the utilization of Virtual Peerings for two traffic engineering objectives: load balancing and latency reduction. Finally, we conclude in Section 5.6.

## 5.2 Virtual peerings

Today, a common method used by ASs to engineer the flow of their interdomain traffic is to establish peerings with other ASs [Bar00]. These additional peerings provide diverse routes to the AS, increasing its path diversity and the chance to get better routes towards some destinations. Those peerings are established either through direct private links between the two ASs or over an interconnection point. An eBGP session is used over the peering link to advertise the prefixes that are reachable via each AS. BGP peerings are established manually by changing the routers configurations by hand. However, manual operations are error-prone and slow [MWA02]. In addition, the time of establishment of a new peering is often on the order of magnitude of several days or weeks.

We propose to solve these problems with Virtual Peerings, which automate the establishment of BGP peerings between cooperating ASs and extend them to distant ASs[1]. Virtual Peerings allow an AS to control the ingress point used by a non-adjacent AS. Therefore, Virtual Peerings represent a deterministic solution

[1]Note that some ASs already establish peering relations with non-adjacent ASes by relying on L2VPNs (see [BP05, LIN06], for example)

to the control of an AS's incoming traffic. Virtual Peerings are established by cooperating ASs based on the current traffic load or another property. We expect that the establishments and removals of Virtual Peerings will occur on a timescale of at least a few hours.

A *Virtual Peering* is a peering built on a dynamically established uni-directional IP tunnel between two cooperating, but non-adjacent, ASs. This tunnel is used by the source AS to send packets to the destination AS via a chosen ingress router in the destination AS. To control these Virtual Peerings, we propose to place, inside each cooperating AS, a *Virtual Peering Controller* (VPC) that will be responsible for the establishment and maintenance of Virtual Peerings. A VPC can for example be a dedicated workstation or a stand-alone BGP router.

Various types of IP tunnels can be used to carry the packets on the Virtual Peerings. The simplest solution is to use IP-in-IP encapsulation [Sim95] or Generic Routing Encapsulation (GRE) [FLH$^+$00] tunnels. Those solutions have a low overhead (20 bytes for IP-in-IP and 24 bytes for GRE) and are supported by most routers. Two other possible types of tunnels are Layer Two Tunneling Protocol (L2TP) [LTG05] and IPSec [Ken05] in tunnel mode. L2TP is often used to provide Virtual Private Network (VPN) services, but its overhead is larger than GRE. The main advantage of IPSec would be its authentication and encryption facilities that could be used to protect the Virtual Peering.

In the past, IP tunnels have often been criticized because of the cost of encapsulating/decapsulating packets and the risk of fragmentation. We would like to point out that those are not operational problems anymore. High-end routers are now capable of supporting line rate encapsulation and decapsulation, either on the normal interfaces or by using special interfaces [Net04]. Second, with Packet over SONET/SDH links that are widely deployed by ISPs, the Maximum Transmission Unit (MTU) is less stringent as it was earlier. Furthermore, almost all TCP implementations support PathMTU Discovery (PMTUD) [MD91] and the tunnel head-end could also perform PMTUD on the tunnel itself.

Another common type of tunnels used by ISPs are MPLS [DR00] tunnels. For Virtual Peerings, MPLS would offer a lower overhead as well as fast restoration, bandwidth reservation and traffic engineering capabilities. Unfortunately, those advantages come at a price: the transit domains must support MPLS and must allow other domains to use RSVP-TE [ABG$^+$01] to establish MPLS tunnels through their own network. While many large ISPs use MPLS inside their network , they are often reluctant to let their customers or peers send RSVP-TE messages to establish MPLS LPSs through their network.

To understand the operation and the usefulness of Virtual Peerings, let us consider the simple network shown in Fig. 5.1. In this network, assume that `ASD` would like to balance over its providers the traffic received from the two sources `AS1` and `AS2`. As the two sources are attached to the same provider, neither AS-Path prepending nor redistribution communities [QTUB04] would allow `ASD` to control its incoming traffic.

As the access router of `ASD` is attached to two providers, `P1` and `P2`, another

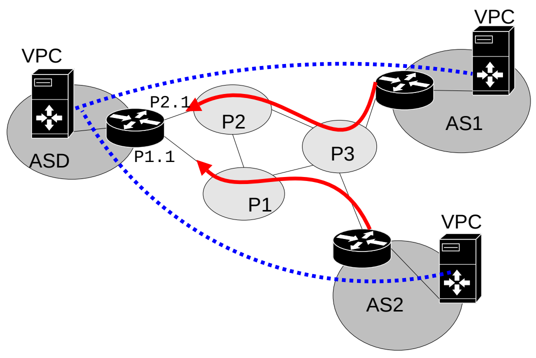


Figure 5.1: Sketch of the approach

solution is possible. When a provider establishes a link with one of its customers, it usually allocates two IP addresses on this link from one of its own CIDR blocks. The first one is for its own router and the second one is for the router on the customer side. A consequence of this common practice is that the access router of `ASD` can be reached via two distinct IP addresses: `P1.1` and `P2.1`. As `P1.1` belongs to the CIDR block advertised by `P1`, any packet sent on the Internet with `P1.1` as destination will reach ASD via `P1`. Based on this finding, `ASD` can balance its incoming traffic provided that it can convince `AS1` (resp. `AS2`) to send all the packets whose destination belongs to `ASD` inside an IP tunnel that terminates at `P2.1` (resp. `P1.1`). This tunnel can be established without any cooperation from the transit providers. It is transparent for `P1`, `P2` and `P3` and entirely controlled by `AS1` (resp. `AS2`) and `ASD`.

## 5.3 Architecture and protocol

Multiple components are involved in the establishment and operation of a Virtual Peering, as shown in Fig. 5.2. First, the two autonomous systems that will establish the Virtual Peering: the Requester AS (RAS) and the Source AS (SAS). The RAS is the AS that is willing to control its incoming traffic. One of its router will terminate the tunnel that will originate in one router of the SAS. Both the RAS and the SAS can be networks composed of several BGP routers. There must be at least one Virtual Peering Controller (VPC) in each SAS and RAS. The VPCs are responsible for monitoring the network and establishing the required Virtual Peerings. In order to monitor the traffic, VPCs are linked to a measurement infrastructure such as [VE04]. In addition, VPCs will collect the available eBGP routes from the border routers. In a full-mesh iBGP, the VPC will typically have an iBGP session with all the routers in the domain. Solutions such as BMP [Scu05] and Juniper's *external-best* could be used in the future to increase the number of eBGP routes learned. VPCs can be dedicated workstations or stand alone BGP routers. Due to their central position in an AS, it would be natural to implement the VPC features on BGP route-reflectors.

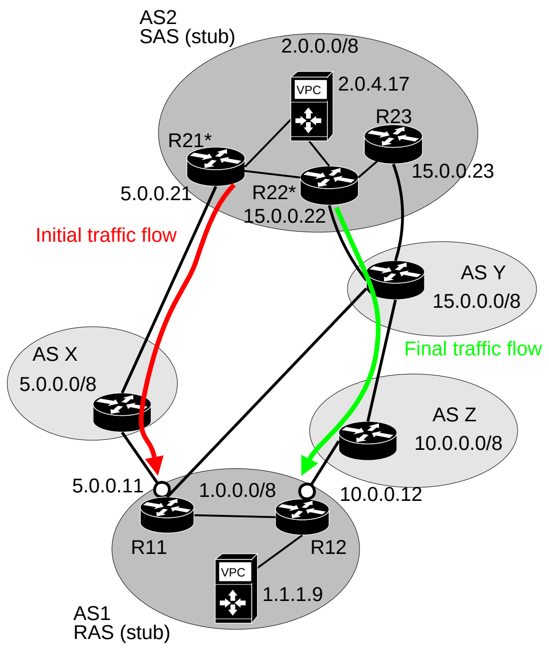


Figure 5.2: Interdomain network topology

To initiate the establishment of Virtual Peerings and to exchange the parameters between the involved VPCs, a protocol is required. Instead of defining a new signaling protocol, we propose in this chapter to rely on the already deployed BGP protocol as a mean to exchange Virtual Peerings requests. The reason for this choice is that such a protocol requires few modifications to BGP and that it can be deployed incrementally. Our extensions solve the following issues. First, the VPC in an RAS must learn the IP address of the VPC in the SAS. Second, the VPC in an RAS needs a secure mean of requesting from a VPC in a remote AS the establishment and removal of Virtual Peerings. Then, the VPC in the SAS must communicate with the border routers of its AS in order to setup the requested tunnels. Finally, routes must be distributed inside the SAS in order to advertise the tunnels.

### 5.3.1 Advertisement of VPC addresses

The **Virtual Peering Controller Advertisement (VPCA)** is used to advertise the IP addresses of the VPCs that serve the SAS. Indeed, to request the establishment of a Virtual Peering with the SAS, the RAS needs to know the IP address of the SAS's VPC. If a small number of ASs want to use Virtual Peerings, these addresses could be distributed in an offline manner, by e-mail or by other means. However, as the number of participants grows, an automatic solution will be required.

We propose to distribute the SAS's VPC IP address inside the BGP Update messages originated by the SAS. The VPC IP address is encoded in a transitive extended community value. The extended community attribute is an optional attribute supported by all BGP implementations. It is already used to encode various types of optional information [STR06]. We define the Virtual Peerings extended community that contains the IP address of the VPC that is responsible for the associated prefixes and a set of bit flags indicating the types of tunnels that can be used to establish a Virtual Peering. For redundancy reasons, an AS could use several VPCs. In this case, it simply attaches several Virtual Peerings extended communities to the BGP routes that it originates.

In the example of Fig. 5.2, the BGP routes towards the prefix `1.0.0.0/8` advertised by the RAS will contain the IP address `1.1.1.9` in the Virtual Peering community. The BGP routes advertised by the SAS for prefix `2.0.0.0/8` will contain `2.0.4.17`, the IP address of the SAS's VPC.

### 5.3.2 Establishment and removal of Virtual Peerings

To request the establishment of Virtual Peerings, some messages must be exchanged between the RAS's VPC and the SAS's VPC. Instead of defining a new signaling protocol from scratch to establish the Virtual Peerings, we use a multi-hop eBGP session between the two VPCs[2]. This session is used by the RAS's VPC to send messages to the SAS's VPC. We call this session the Virtual Peering Session. The messages exchanged over a Virtual Peering Session are not propagated to other BGP routers.

Two types of BGP messages are exchanged over a Virtual Peering Session: Virtual Peering Establishment and Virtual Peering Removal. A **Virtual Peering Establishment (VPE)** is a BGP Update message sent in order to request the establishment of a Virtual Peering or to change the parameters of an existing Virtual Peering. The VPEs sent by the RAS's VPC over the Virtual Peering session must contain both the destination prefixes (`1.0.0.0/8` in the case of Fig. 5.2) and the information required to establish the tunnels including the tunnel tail-end. This information can be encoded by using the tunnel Subsequent Address Family Identifier (SAFI) proposed in [NKTW05]. This proposal defines a new type of address family that allows to attach tunnel information to the advertised prefixes. The encoding proposed in [NKTW05] allows to specify the parameters for different types of tunnels. For example, a tunnel route indicating a GRE tunnel can contain the required key and the session ID [Dom00] while a tunnel route indicating an L2TPv3 tunnel will contain the required cookie. Furthermore, several types of tunnels can be attached to each tunnel route. When advertising tunnel routes, a

[2]We do not describe the details of a BGP session establishment, but refer the reader to the BGP specifications [RL04]

VPC may request distinct Virtual Peerings by advertising different prefixes with different associated tunnel tail-ends.

A **Virtual Peering Removal (VPR)** is a BGP Withdraw message sent in order to shutdown an existing Virtual Peering. The VPRs sent by the RAS's VPC only contain the prefix for which the Virtual Peering must be shutdown and concern the same address family. When a VPR is received by a VPC, it contacts the tunnel head-ends to shutdown the tunnels for the given prefix.

### 5.3.3 Distribution of Virtual Peering routes within the domain

The distribution within the domain of routes learned through the Virtual Peering is done by mean of a **Virtual Peering Tunnel Route (VPTR)**. A VPTR is a normal BGP Update message that the VPC sends to the border routers in order to distribute the tunnel routes received in a VPE. The VPTRs are sent over the genuine iBGP sessions that the VPC has established with the border routers. Each VPTR contains a *prefix*, a *tunnel tail-end* and the type of tunnel requested. The VPTR uses a different address family, so that both the normal IPv4 addresses and the tunnel routes can be advertised over a single BGP session.

The selection of the best border routers to serve as the Virtual Peering head-end in the SAS could depend on how the SAS wants to optimize its interdomain traffic. In practice, this decision will be taken by the VPC. Two approaches are possible. In the first approach, the VPC itself can learn the eBGP routes known by each border router. Since it has an iBGP session with each border router, this is easy. The VPC can then measure the quality of each eBGP route based on predefined criteria by requesting each border router to perform latency and bandwidth measurement using a technique such as [Sys04]. For instance, it can measure the latency of the routes or the maximum bandwidth available along the route. Based on the result of the measurement, the VPC can then select the most appropriate border router.

The second approach consists in establishing multiple tunnels. The VPC must then select multiple border routers that will serve as tunnel head-ends. This approach is interesting if the SAS has multiple peerings with its providers, located at very distant places. In this case, it can be interesting to setup tunnels departing at each of these peerings in order to favor hot-potato routing.

Upon reception of a VPTR each border router determines whether it has a best eBGP route to reach the *tunnel tail-end* in its BGP routing table. In that case, the border router can serve as a tunnel head-end for the packets sent towards this *prefix*. For instance, in the example of Fig. 5.2, `R22` has learned from `AS Y` an eBGP route towards the prefix of the tail-end router `R12`. The router `R12` has the IP address `10.0.0.12` that belongs to the prefix `10.0.0.0/8` originated by `AS Z`. Once the tunnel is established, the border router advertises via iBGP a new route indicating that it can reach the destination prefix, i.e. `1.0.0.0/8`. This route has an higher Local-Pref to force other iBGP neighbors to prefer it over routes received outside of the Virtual Peering. The AS-Path of this advertisement contains the AS-Path of the route that the border router uses to reach the tunnel tail-end, i.e. `Y:Z:1`. If there are no eBGP routes to reach the tunnel tail-end (this may be due to BGP policies), the border router will not serve as a tunnel head-end for the packets sent towards this prefix.

We assume that in a typical IP network, only a fraction of the border routers will be able to serve as tunnel head-ends. This could for example depend on the type of interfaces installed on each router. To allow the VPC to know the routers that are capable of establishing virtual peering links, we assume that each router indicates in the IGP link state packets that it originates the types of tunnels that it supports, if any. For IS-IS, this can be encoded by using the capability TLV defined in [VSA06]. Based on its link state database, the VPC can thus easily determine the capabilities of all the border routers inside its AS.

### 5.3.4 Security considerations

From a security viewpoint, the Virtual Peerings approach proposed in this chapter exhibits two major issues. First, the VPCA message advertises to the global Internet the IP addresses of the VPCs attached to a prefix. If an attacker could modify the content of the Virtual Peerings extended communities in BGP advertisements passing through a (transit) router, it could redirect Virtual Peering requests to another machine. This could lead to traffic redirection attacks. However, it should be noted that if an attacker is able to modify BGP messages, many types of attacks are possible with the standard BGP that is deployed today. In order to avoid this problem, the best solution is to use one of the BGP extensions proposed in [Whi03, KLS00], that allow to authenticate BGP advertisements. If those extensions cannot be used, a possible solution is to ensure that the IP address of the VPC belongs to the advertised prefix.

A second issue is due to the VPEs. When receiving a VPE, a VPC should be able to verify that the RAS is authorized to advertise those prefixes and tunnel tail-ends. Otherwise, an attacker could easily redirect packets sent by the SAS to its premises instead of a tail-end in the RAS. This verification could be based on publicly available address allocation registries such as ARIN or RIPE. Several major ISPs, notably in Europe, already use those databases to filter the routes advertised by their peers and customers. Those techniques can also be used by VPCs. In the long term, the BGP security extensions being developed by the IETF [Whi03, KLS00] will address this problem.

### 5.3.5 Deployment

Our rationale for designing the protocol described in this section was to make possible an incremental deployment. Since the protocol does not require modifications in the intermediate ASs, two domains can start to use it to negotiate Virtual Peerings. Moreover, inside a single domain, only a subset of the border routers must be updated in order to support the VPTRs. In addition, the VPC can initially be implemented in a separate workstation and later be deployed inside a genuine BGP router or route-reflector. Finally, while the BGP security extensions [Whi03, KLS00] have not yet been deployed in the global Internet, it would be easier to use them between VPCs as there will be fewer VPCs than normal BGP routers and the VPCs do not redistribute the VPEs received over the Virtual Peering sessions to other ASs.

A possible deployment scheme would be to initially start using Virtual Peerings between a small number of universities or research labs. Afterward, the solution can naturally be deployed by content and access providers as well.

### 5.3.6 Robustness

Once the virtual peering has been established, it will be used to carry packets. During the operation of the virtual peering, several events can occur. First, the source or the destination AS may wish, for any reason, to terminate the virtual peering. This can be achieved by terminating the Virtual Peering session and indicating the reason for the failure in the BGP NOTIFY message sent.

Second, failures and changes on the interdomain path followed by the virtual peering from head-end to tail-end may affect its operation. To cope with those events, the tunnel head-end should use a failure detection protocol such as Bidirectional Forwarding Detection (BFD) [KW06]. BFD can be used in various environments and is able to detect both link failures and tunnel failures. Once BFD on the tunnel head-end has detected a failure of the virtual peering, the tunnel head-end should withdraw the iBGP routes that announced the prefixes reachable via the virtual peering. This withdrawal will force the other routers of the AS to either switch to another virtual peering tail-end that is still able to reach the destination prefix or a normal BGP route.

Another event that can occur is the failure of the Virtual Peering session between the VPC in the RAS and the VPC in the SAS. This failure can be detected in the same way as a genuine BGP session, i.e. by relying on BGP KEEP_ALIVE messages. When the VPC in the SAS detects that the Virtual Peering session is failing, it acts as if a VPR was received. It withdraws the VPTRs previously advertised within the domain. One way to be robust to the failure of a Virtual Peering session is to have at least two VPCs in each domain and separate Virtual Peering session between them. Each VPC in the SAS will therefore receive a VPE for each destination prefix and advertise the corresponding VPTRs to the border routers. Each border router will receive two VPTRs. If one Virtual Peering session fails, one of the VPTRs will be withdrawn, but the other will remain. This solution work if the cause of the Virtual Peering session failure does not affect the other Virtual Peering session. If both sessions follow the same Internet route, they share the same risks.

Finally, to avoid a single point of failure, a good operational practice would be to place two redundant VPCs in an AS.

### 5.3.7 Evaluation

In order to show the potential benefit of using Virtual Peerings to exploit the routes towards the providers of the destination domain, we performed a simulation based on real BGP routing tables collected on December 1st, 2004 by the RouteViews [Mey05] and RIPE [RIP05] projects. The routing table from RouteViews contained 5,750,380 routes received from 34 different peers. In the simulation, we only considered the 32 peers that announced a full routing table, i.e. more than 140.000 routes. The RIPE routing table was collected by the RRC00. It contained 1,641,618 routes announced by 11 peers. All these peers announced more than 140.000 routes. We summarize the datasets in Table 5.1.

| **Dataset** | **Routes** | **Peers** | **Pairs** | **M-h stubs** | **M-h prefixes** |
|---|---|---|---|---|---|
| RouteViews | 5,750,380 | 32 (34) | 496 | 6,402 | 29,575 |
| RIPE RRC00 | 1,641,618 | 11 | 55 | 6,247 | 29,934 |

Table 5.1: Summary of RouteViews and RIPE datasets.

Among all the received routes, we identified, based on the AS-paths, 6,402 multihomed stubs advertising 29,575 different prefixes for the RouteViews dataset and 6,247 multi-homed stubs advertising 29,934 prefixes for the RIPE dataset. We then considered all the pairs of peers present in each dataset. There are 496 different pairs of peers in the RouteViews dataset and 55 different pairs in the RIPE dataset. We simulated a dual-homed stub domain connected to the selected peers. For each simulated stub, we counted the number of different paths learned through BGP towards all the considered destination prefixes. We consider that two paths are different if at least the provider in the source AS or the provider in the destination AS are different. Note that if two paths are different, that does not mean that they are completely disjoint.

We show the results of our simulations in Fig. 5.3. The figure shows the distribution of the number of different paths available when using BGP routes towards the destination domain and when using Virtual Peerings, for all the destination prefixes. On the x-axis, we show the number of different paths available and on the y-axis, the number of prefixes that could be reached with the corresponding number of paths. We show the median, the 10$^{th}$ percentile and the 90$^{th}$ of the number of prefixes since this is a summary for 496 pairs in the case of the RouteViews dataset and 55 pairs in the case of the RIPE dataset. We can readily observe that there was few variation among the different pairs since the percentiles are close to the median.

When looking at the BGP paths towards the destination AS, the number of distinct paths is comprised between 0 and 2. If there is no path, that means that the destination prefixes cannot be reached. This fortunately occurs for only a small subset of the RouteViews dataset. This is probably due to the filters used by some ISPs. If there is only one path, that means that the destination prefix was not reach-

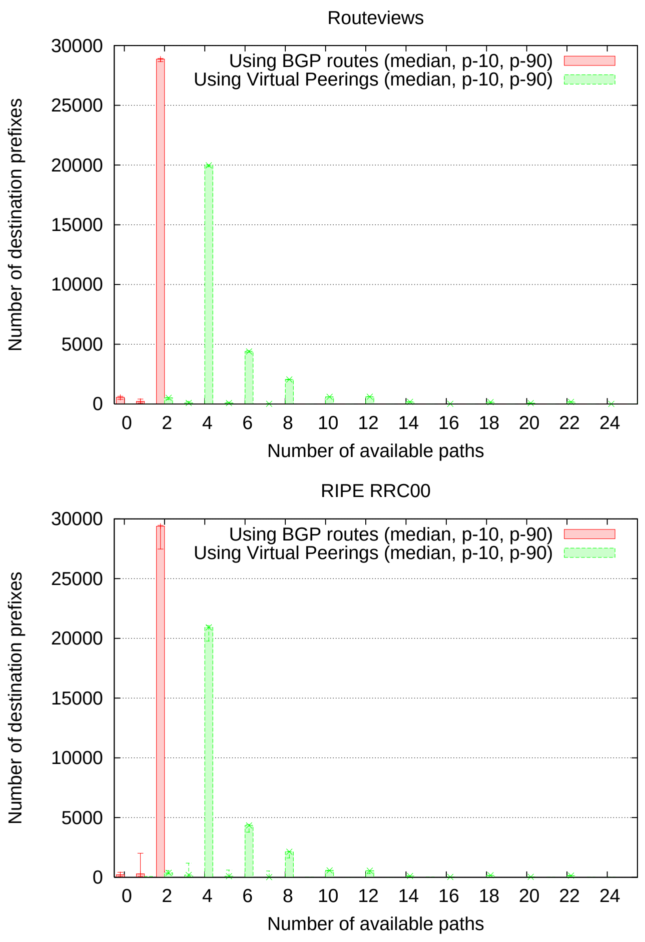


Figure 5.3: Path diversity obtained when multihoming to RouteViews or RIPE peers.

able through one of the providers. But most of the time, the destination prefixes were reachable through both providers. The number of available BGP paths cannot be more than 2 since the simulated stubs are dual-homed. Therefore, they only receive a single route for each destination prefix from each provider. Moreover, it is frequent that these paths merge at the same provider of the destination AS. The path diversity is thus low with BGP even if there are two different paths most of the time.

If we look at the routes that would be obtained by using Virtual Peerings, the path diversity increases a lot. Most destination prefixes (67.6% for RouteViews and 69.9% for RIPE) are reachable through at least 4 different paths. There is also a significant number of destination ASs that are reachable through 6 paths (14.9% for RouteViews and 14.5% for RIPE) or even more due to some destination stubs being more than dual-homed. The reason for the large majority of the destination prefixes having an even number of different paths is that the source stub is dual-homed. The simulations show that using the routes towards the providers of the destination domain brings out a lot of new paths. A rule of thumb can be derived from this observation: the number of paths leveraged by Virtual Peerings is $M \times N$, with $M$ (resp. $N$) the number of providers of the source (resp. the destination) stub.

A similar study was performed in the framework of an evaluation of IPv6 multi-homing solutions (see [dLBL03, dLQB06]). With IPv6 multi-homing, each end-system receives several IPv6 addresses, one per provider of its AS. By selecting the address that it uses to reach a destination, each host can indirectly select the interdomain path used. With IPv6, a deterministic control of the interdomain paths is possible without using virtual peerings.

## 5.4 Incoming traffic balancing

In this section, we describe how Virtual Peerings can be used to balance the load of incoming traffic on the access links of a stub domain. To achieve a good balancing of the inbound traffic, the stub AS needs to monitor the traffic received on each access link. This can be done by activating NetFlow on the border routers' interfaces and by collecting the traffic statistics in a dedicated workstation [VE04]. Then, based on the traffic statistics combined with an optimization algorithm, the stub AS identifies the source ASs that must be moved. For each source AS concerned, a Virtual Peering is established. Through the Virtual Peering, the destination requests the source AS to encapsulate its traffic in a tunnel towards a designated access link.

We show that this solution is feasible and that with a limited number of Virtual Peerings, it is possible to reach a near perfect load balance of the inbound traffic. We conduct our evaluation in two steps. First, we simulate the traffic imbalance on the stub that we want to optimize. Then, for each considered stub, we run an optimization algorithm which determines the Virtual Peerings that are required to move the sources of traffic of the stub and obtain a better balance of its inbound traffic. We describe these two steps in the succeeding sections, then we present our results.

### 5.4.1 Simulation scenario

We use a simulation scenario similar to that of Chapter 4, based on the Internet topology inferred by [SARK02] from real BGP routing tables gathered from multiple vantage points, mostly in the Internet core. The topology dates from February 10th, 2004 and contains 16,921 domains and 37,271 interdomain links. There is at most one link between two different domains and each link represents the business relationship that exists between the two domains it connects. The possible relationships are *customer-provider* and *peer-to-peer*.

We use C-BGP and model each domain as a single BGP router. We translate the business relationships between the domains into routing policies configured in each routers. These policies are composed of two parts. The first part enforces the so-called *selective export rule* [Gao00] and the second part enforces the preference that a domain has for routes learned over different relations [Gao00].

To ensure a good representativity, our simulations are performed for a large number of stubs. We consider 1000 dual-homed stubs, 1000 3-homed stubs, 295 4-homed stubs, 101 5-homed stubs and 49 6-homed stubs. This corresponds to 29% of the multi-homed stubs in today's Internet. Using C-BGP, we computed the paths used by all the domains to reach the stubs we selected. The computation is done in a distributed manner, by considering one destination prefix at a time, as explained in Section 4.5.1.

## 5.4.2 Topological imbalance and traffic imbalance

There are two main structural causes to the imbalance of the inbound traffic received by a stub domain. We call the first cause the *topological imbalance*. The topological imbalance is an imbalance in the number of inbound paths reaching the destination stub through each of its providers. This imbalance is due to the routing decisions taken by BGP in the remote domains. The importance of the imbalance depends on the connectivity of the stub domain and on the policy routing constraints. The consequence of the topological imbalance is that the sources of traffic reach the destination stub through different paths hence different providers.

We show the initial topological imbalance seen by the stub domains considered in our simulation on Fig. 5.4. We define the topological imbalance as the maximum number of paths received by a provider divided by the mean number of paths. For instance, a dual-homed stub that is reached by 75% of its inbound paths through one provider and 25% through the other provider has a topological imbalance of $\frac{75}{50} = 1.5$. On the x-axis of Fig. 5.4, we show the topological imbalance and, on the y-axis, we show the cumulative fraction of stubs that see the corresponding imbalance. The figure shows five curves. The upper one concerns 2-homed stubs. We observe that nearly 50% of the 2-homed stubs have a topological imbalance larger than 1.6, meaning that these stubs have a provider that receives more than 80% of the paths. For 3-homed stubs (the second curve), 50% of them have a topological imbalance that is larger than 2.1, meaning that these stubs have a provider that receives more than 70% of the paths. The succeeding curves show the topological imbalance for 4-homed stubs, 5-homed stubs and 6-homed stubs. For 4-homed stubs, the median topological imbalance is larger than 2.3, for 5-homed stubs, it is close to 2.7 and for 6-homed stubs, it is close to 2.8.

The topological imbalance must be combined with the *traffic imbalance*. That is, all the sources do not send the same volume of traffic to the destination stub. The distribution of the volume of traffic per source is usually highly skewed. Typically, a small number of ASs are responsible for a large fraction of the received traffic ([QUP$^+$03, UB02, FBR03]).

To model the traffic distribution in this topology, we assign traffic on all the

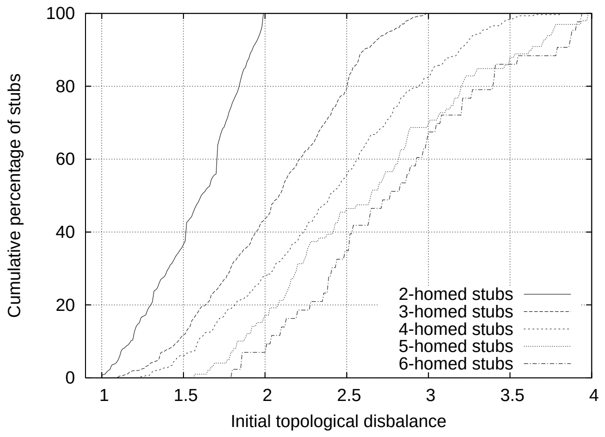


Figure 5.4: Distribution of the topological imbalance.

sources following a Weibull distribution with shape parameter $\alpha$ equal to 0.5. With this distribution, about 1000 sources are responsible for 95% of the traffic received by a stub AS. This fits very well the traffic distribution shown in [FBR03] and [UBQ03b] and the references therein.

We used those traffic distributions to weight the paths computed by C-BGP. Doing this, we obtained the distribution of the incoming traffic on each of the considered 2445 stub domains. Fig. 5.5 shows the distributions of the traffic imbalance among all those stub domains. We define the traffic imbalance as the traffic volume received by the most loaded provider divided by the mean traffic volume. On the y-axis, we show the cumulative fraction of stubs that have the corresponding imbalance. Fig. 5.5 shows the traffic imbalance for stubs that have 2 to 6 providers. We observe on the first curve that less than 25% of dual-homed stubs have their inbound traffic well balanced over their providers, that is with an imbalance smaller than 1.01. Moreover, more than 35% of dual-homed stubs have an imbalance superior to 1.2, which means that 35% of the dual-homed stubs have one provider that receives more than 60% of the traffic. Among stubs that have 3 providers, about 60% have an imbalance larger than 1.2. Discussions with ISPs reveal that such large traffic imbalances are common.

### 5.4.3 Selection of the Virtual Peerings

Searching for a better repartition of the traffic is an optimization problem which consists in allocating an access link to each source of traffic. This is a combinatorial problem. The number of possible assignments of $N$ sources on $M$ access links is $M^N$. For a dual-homed stub ($M = 2$) and on the order of 15.000 sources, this number is already huge. Several techniques can be used to solve this problem. We choose to use Evolutionary Computing techniques [ES03] implemented with

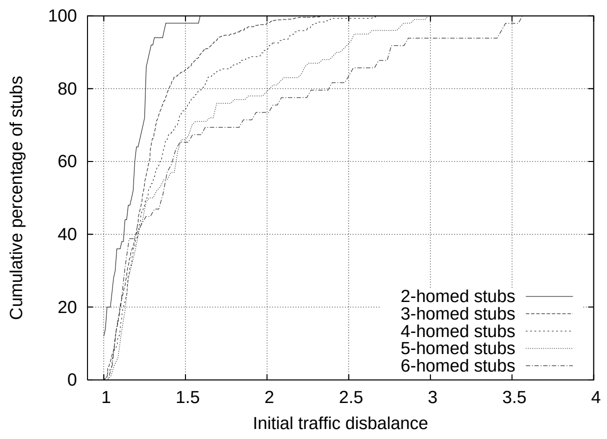


Figure 5.5: Initial traffic imbalance ($\alpha = 0.5$).

the help of the GAUL library [Adc04]. We choose an evolutionary algorithm to solve this problem because it is possible to extend it to support multiple objectives [Deb01] and this technique has already been applied to solve different interdomain traffic engineering problems [UBQ03b].

Our evolutionary algorithm (Alg. 4) relies on a population of individuals, that is, a set of potential solutions that evolves in time. In our population, an individual represents a particular assignment of the $N$ sources on the $M$ access links of the destination. An individual is thus an array $(l_0, \ldots, l_{N-1})$ of $N$ integers $0 \leq l_j < M$ where each $p_j$ is the identifier of the access link used by source $j$ to enter the network. We initialize the population with individuals that represent the initial BGP situation, that is, an individual represents the access link used by the $N$ sources. In practice, a stub network does not need to know the interdomain paths used by each source AS. It can use NetFlow on its border routers and collect the traffic statistics [VE04] to determine which source AS is received over which access link. Before starting the optimization, the algorithm slightly perturbates the initial individuals. In this way, we do not start with a population of identical individuals. The perturbation of an individual consists in replacing the access link of a randomly chosen source by a randomly chosen access link.

We fixed the population size, in an empirical manner, to twice the number of considered sources. The number of considered sources depends on the traffic volume distribution. In these results, the algorithm considers as many sources as required to cover 95% of the total traffic volume. That is, $N = 942$ sources were taken into account. During the evolution of the population, we perform mutations and crossovers. In our algorithm, a mutation consists in changing the access link of a randomly selected source (see Alg. 5). We refer the reader to [ES03] for an explanation of the crossover operation.

After each generation of the population, individuals are evaluated with a fitness

**Alg. 4** Optimization algorithm

```
Let $N$ be the number of sources
Let $M$ be the number of access links
Let $(l_j)_{0 \leq j < N}$ be the initial access link used by each source
Let $(v_j)_{0 \leq j < N}$ be the traffic volume sent by each source
 1: /* Initialize population with $2N$ individuals */
 2: $pop \leftarrow \emptyset$
 3: for $k = 0$ to $2N - 1$ do
 4:     $pop \leftarrow pop \cup mutate((l_0, ..., l_{N-1}))$
 5: end for
 6: /* Evaluate fitness of individuals */
 7: $evaluate\_fitness\_pop(pop)$
 8: /* Main loop, each generation updates the population */
 9: while $(generation < MAXGEN)$ do
10:     /* Crossover with probability 0.1 */
11:     $crossover\_pop(pop)$
12:     /* Mutation of individuals with probability 0.9 */
13:     $mutate\_pop(pop)$
14:     /* Evaluate fitness of individuals */
15:     $evaluate\_fitness\_pop(pop)$
16:     /* Terminates if a good individual is found */
17:     if ($\exists k : pop[k]$ satisfies termination criterion) then
18:         break
19:     end if
20:     /* Select best individuals based on fitness */
21:     $select(pop)$
22: end while
```

**Alg. 5** Mutate individual $(l_0, ..., l_{N-1})$

```
1: /* Choose a random access link $i$ */
2: $i = rand(M)$
3: /* Choose a random source $j$ */
4: $j = rand(N)$
5: /* The new access link used by source $j$ is $i$ */
6: $l_j \leftarrow i$
```

function. The fitness function used in our algorithm (Alg. 6) measures for an individual the deviation that it causes in term of load balancing. Formally, in order to measure the fitness of an individual $x$, the algorithm first computes the percentage of traffic $L_i$ that would be received by each access link $i$ if the configuration represented by individual $x$ was implemented with Virtual Peerings. Then, the function computes the L2 distance between the vector $(L_i)_{0 \leq i < M}$ and the equilibrium. The equilibrium represents the case where each access link receives an equal percentage of traffic $1/M$.

Finally, a selection is performed based on the fitness of individuals. The individuals that best fit the objective are kept while others are discarded.

**Alg. 6** Compute the fitness of individual $(l_0, ..., l_{N-1})$

```
1: /* Compute the load vector $(L_i)_{0 \leq i < M}$ */
2: for $i = 0$ to $M - 1$ do
3:     $L_i \leftarrow \frac{\sum_{\forall j : l_j = i} v_j}{\sum_{0 \leq j < N} v_j}$
4: end for
5: /* Compute the L2 distance from the equilibrium vector */
6: $fitness \leftarrow \sum_{0 \leq i < M} \left(L_i - \frac{1}{M}\right)^2$
```

We show in Fig. 5.6 the convergence of the evolutionary algorithm for two different domains. The first one is a 3-homed stub. The initial distribution of the traffic load is as follows: ISP1 receives 17.59% of the traffic while ISP2 receives 53.58% and ISP3 receives 28.82%. After 38 generations of the population, the algorithm has selected 30 tunnels to be established. These tunnels lead to a new distribution of the load which is as follows: ISP1 receives 33.33% of the traffic, ISP2 receives 33.80% and ISP3 receives 32.87%. This is close to the objective by less than 1% and the algorithm terminates.

The second domain is a 4-homed domain. The initial traffic load is as follows: 17.60%, 51.80%, 19.27% and 11.33%. The algorithm converges to a selection of 55 tunnels after 55 generations. The final traffic load distribution is 24.80%, 25.50%, 25.04% and 24.66%.

### 5.4.4 Results

We used this evolutionary algorithm to determine the Virtual Peerings that each of our 2445 considered stubs would have to establish to approach a perfect balance among its access links by less than 1%. This means that in the case of a dual-homed stub for instance, the number of tunnels required causes the most loaded provider to carry at most 50.9% of the traffic volume. Figure 5.7 reports the cumulative distribution of the number of Virtual Peerings established by all those stub domains to approach of the perfect balance by less than 1%. We observe that in the case of dual-homed stubs, the objective is reached with no more than 41 tunnels for 90% of the stubs. In the case of 3-homed stubs, no more than 42 tunnels are required to

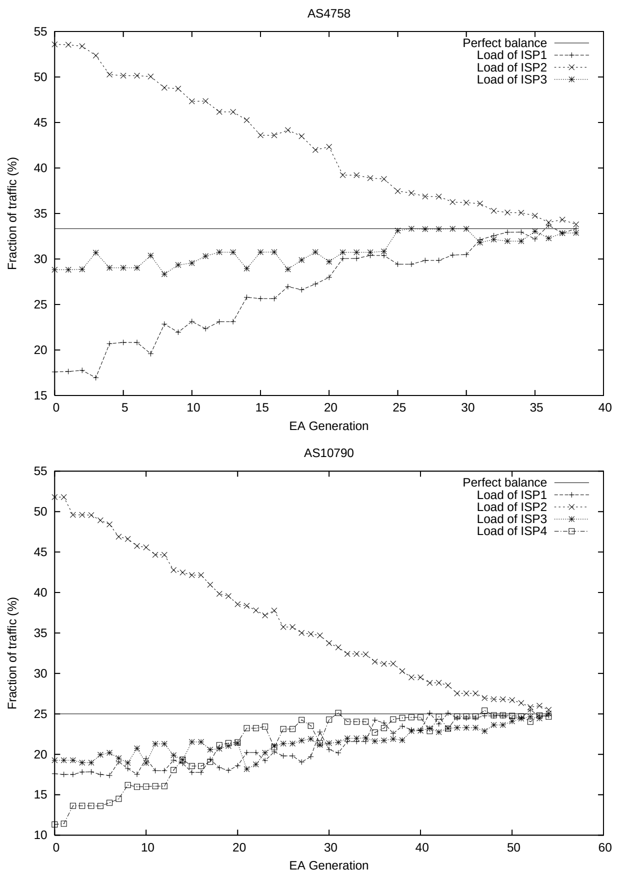


Figure 5.6: Convergence of the evolutionary algorithm for a 3- and a 4-homed stubs.

balance the traffic of 90% of the stubs. Finally, less than 50 tunnels are required to balance the traffic among the providers of 90% of the 4-homed stubs.

We studied the sensitivity of the technique to the traffic distribution. We performed the same simulation with a traffic distribution that follows a Weibull with parameter $\alpha = 0.25$. In this case, the situation is really unfavorable : 3000 source ASs produce 95% of the traffic received by a stub. The resulting traffic imbalance is shown in Fig. 5.8 and the number of tunnels is shown in Fig. 5.9. The results of this simulation reveal that the number of Virtual Peerings required to balance the inbound traffic increases, but remains quite low. With $\alpha = 0.5$, a near perfect load balancing was possible with as few as 43 tunnels for 90% of the stubs. The remaining 10% of the stubs require between 44 and 94 tunnels. With $\alpha = 0.25$, 43 tunnels allow a near perfect load balancing of 68% of the stubs. Up to 80 tunnels are required to cover 90% of the stubs. The remaining 10% of the stubs still only need not more than 148 tunnels to balance their inbound traffic.

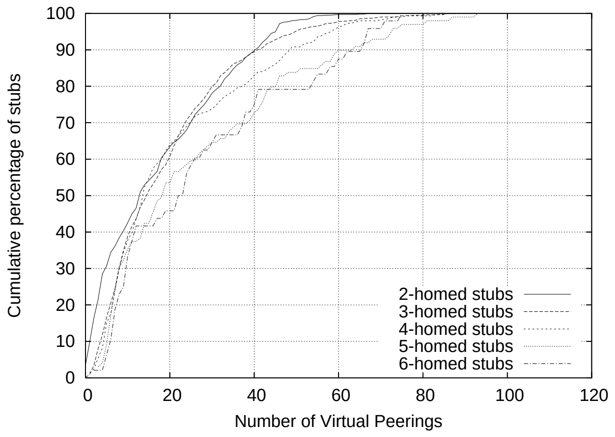


Figure 5.7: Number of Virtual Peerings to establish ($\alpha = 0.5$).

These results show that an inbound traffic engineering relying on Virtual Peerings is feasible even with an unfavorable traffic distribution. Moreover, with Multi-Objective Evolutionary Computing [Deb01] it would be possible to determine the optimal Virtual Peerings that minimize both the imbalance and the number of tunnels to establish. It would also be possible to combine load balancing with other objectives such as the latency reduction but we leave this as further work.

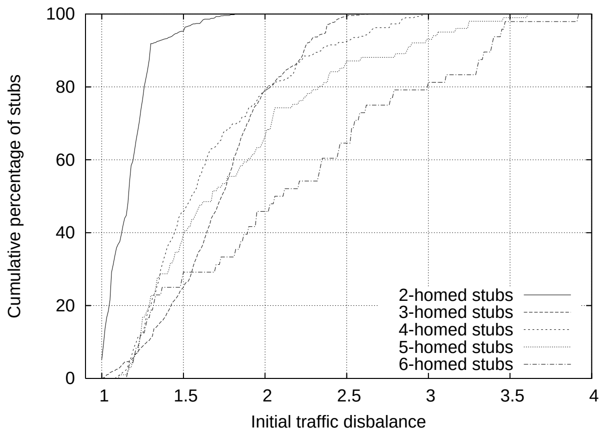


Figure 5.8: Initial traffic imbalance ($\alpha = 0.25$).

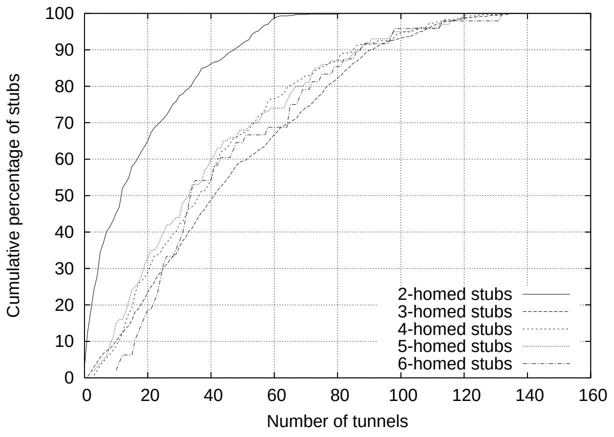


Figure 5.9: Number of Virtual Peerings to establish ($\alpha = 0.25$).

## 5.5 Latency improvement

By using appropriate Virtual Peerings, the source and destination ASs may choose to forward traffic over paths that better meet their QoS requirements. In this section we focus on the selection of paths with the lowest delay. In order to select the best path according to this metric, we need a mechanism that monitors the various paths available to join the virtual peer. We need to measure the one-way delay along each available path. Measuring the RTT is not enough since the routing can be asymmetric. Fortunately, standard solutions exist for one-way delay measurement that are discussed within the Internet IP Performance Metrics working group [AKZ99, ST04]. One-way delay measurement equipment is also available in specific hardware [FGK$^+$01] as well as in routers [Sys04]. We can therefore consider that measuring the one-way delay of available paths is possible.

In this section, we evaluate the utilization of Virtual Peerings to reduce the latency of interdomain paths. For this purpose, we use a second simulation scenario where we compare the delay along the BGP route selected by a stub domain to reach another stub domain to the delay along the routes that can be obtained with virtual peerings. To perform this simulation, we cannot rely on the same topology as in section 5.4 since it does not contain the delays along the links. Moreover, the topology used in section 5.4 contains a single router per domain.

At this time, there is no realistic Internet-scale topology available that contains delays. A possible solution could be to use a synthetic topology. For instance, BRITE [MAMB01] makes possible the generation of two-level topologies (ASs and routers) with delays. Unfortunately, BRITE does not contain a model of the interdomain business relationships. Another topology generator of interest is GT-ITM. Its authors have presented in [CDZ97] a second version of their generator that is supposed to support policies, but unfortunately, this version is not publicly available at this time. Finally, another generator called Inet [JCJ00] provides topologies with a more accurate degree distribution. However, it does not deal with business relationships.

### 5.5.1 A two-level topology with delays

Since to our knowledge, no adequate Internet topology exists, we have built an Internet topology that contains delays, IGP weights, multiple interdomain links and BGP policies. We used the AS-level topology inferred by Subramanian et al [SARK02] as a starting point. This topology contains a large fraction of the Internet domains as well as the business relationships between them.

In order to build an intradomain structure, we used a commercial database [max04] dated from June 2nd, 2004, that provides the geographical mapping (latitude and longitude) between blocks of IP addresses and locations worldwide. The database contains 1,837,457 blocks of IP addresses located in 118,489 locations. Based on this database, we were able to infer the points of presence of the domains found in the BGP routing table. To identify the domain that advertises each block, we used real BGP routing tables collected in the framework of the Route-Views project [Mey05] at University of Oregon and dated from February 10th, 2004. These routing tables contain 139,527 prefixes originated by 15,030 different domains. For each point of presence, we have placed one router in the originating domain.

Then, in order to build the internal structure for each domain, we grouped the closest points of presence of each domain in clusters using a hierarchical classification method using the euclidean distance metric. We connected all the routers of a cluster together. Then, we connected all the clusters together, using the closest routers of each cluster. We thus obtained a two level intradomain structure. Based on the coordinates of the end-points of each link, we were able to compute the distance and thus the propagation delay along the link. In addition to this, the IGP weight that we assigned to links favors hot-potato routing in the sense that shorter links are assigned a shorter cost than longer links.

For the interdomain links, we relied on the AS-level topology. For each interdomain link found between two domains in this topology, we added multiple interdomain links in our topology. We fixed the maximum number of interdomain links between two domains to $N = 5$. Then, the number of interdomain links between two domains was computed by multiplying N by the size of each domain and dividing the result by the square of the size of the largest domain in the topology. We added 1 to the result to make sure that there is at least one link. The resulting topology[3] contains 39,343 routers, 103,829 links and requires 400,148 BGP sessions.

[3]The resulting topology is available at `http://cbgp.info.ucl.ac.be/itopo` such that other researchers are able to reproduce our results. More details about the construction of the topology are also available on the web site.

### 5.5.2 Results

We then perform our simulation with C-BGP for a subset of the 8,026 multi-homed stubs contained in the topology. To reduce the simulation time, we conduct the simulation for 2,068 multi-homed stubs randomly chosen among the 8,026 multi-homed stubs. We show the results of the simulation in the succeeding figures. The delays shown in these figures can be considered as minimal bounds for the real delays, since we only take the propagation delay into account. Other factors can influence the end-to-end delay, such as limited bandwidth (transmission delay) as well as congestion (queuing delay). In addition, each hop introduces a processing delay which is not taken into account here. Nevertheless, the delays presented in Fig. 5.10 and Fig. 5.11 can be considered as being of the same order of magnitude as real delays.

First, in Fig. 5.10, we show the delay along the paths chosen by BGP. On the x-axis, we plot the delay of the paths in milliseconds (ms). On the y-axis, we plot the cumulative fraction of stub-stub paths that have the corresponding delay. We observe that about 21% of the routes chosen by BGP have a delay inferior to 10ms. About 64% of the routes chosen by BGP have a delay comprised between 10 and 50ms, 24% have a delay between 50 and 100ms and less than 1% have a delay larger than 100ms, the largest delay being 208ms. Just to get an idea of what such delays represent, the delay to go at the speed of light from one end of a diameter of the earth to the other (about 20,000km) is approximately 67ms.

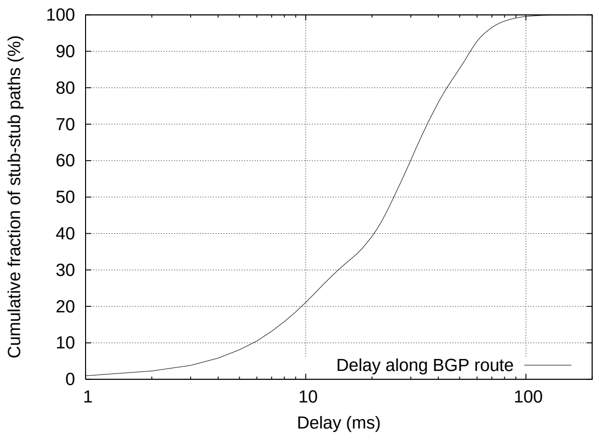


Figure 5.10: Cumulative distribution of delay along BGP routes.

In Fig. 5.11, we plot the delay along the BGP route (x-axis) against the delay along the best delay obtained with Virtual Peerings (y-axis). Each point represents a pair of delay values. The color of the point is an indication of how many pairs of stubs correspond to these delays. The diagonal line where $x$ equals $y$ represents the case were there is no improvement, since the delay along the BGP route and the best delay are equal. A first observation is that the points are never above this line. This means that the Virtual Peerings never worsen the end-to-end delay. For all points that lie strictly under the line, there is an alternative path exploitable by using Virtual Peerings and this path has a lower delay than the BGP route. We observe that a neat improvement in delay is possible. For instance, if we observe the right part of Fig. 5.11, we note that a significant number of paths chosen by BGP have a delay that is longer than 100ms. The fraction of BGP routes that have a delay larger than 100ms is about 25%. On the contrary, if we observe the upper part of the plot, we note that most of the lowest delay paths are under 100ms. The fraction of lowest delay paths that are shorter than 100ms is about 95%.

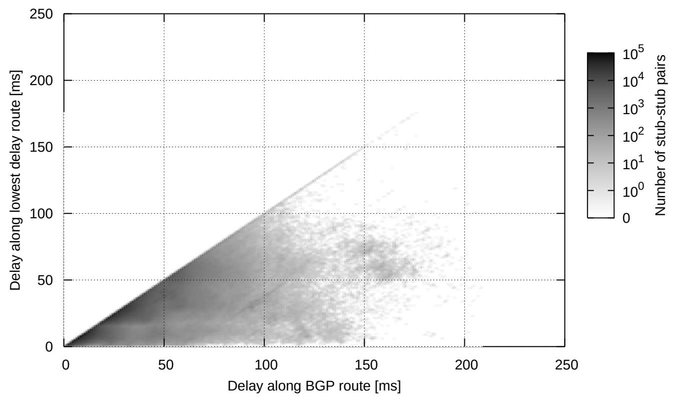


Figure 5.11: Delay along the BGP route versus delay along the lowest delay route.

The curve of Fig. 5.12 gives the distribution of absolute improvement in delay, that is the difference between the delay of the BGP route and the delay of the best alternative path that virtual peerings permit. We observe that for approximately 40% of the paths, there is no possible improvement. This means that, by chance, BGP has already selected the best route in term of delay for these paths. However, we observe that for more than 40% of the paths there is a possible improvement in terms of delay. 30% of routes can be improved by up to 5ms. 22% of the routes can be improved by 5 to 20ms and 8% of the routes can be improved by more than 20ms and up to more than 180ms.

Finally, we show in Fig. 5.13 the influence of having 2, 3, 4 or more providers for a stub network. On the x-axis of Fig. 5.13, we show the absolute improvement in delay, that is the difference between the delay of the BGP route and the best route in term of delay. On the y-axis, we show the fraction of stub-stub paths. The figure represents the same curve than the curve of Fig. 5.12, except that we split the curve depending on the number of providers of the source domain. In addition, we use a log scale in order to zoom on smaller improvements. We observe that when a stub

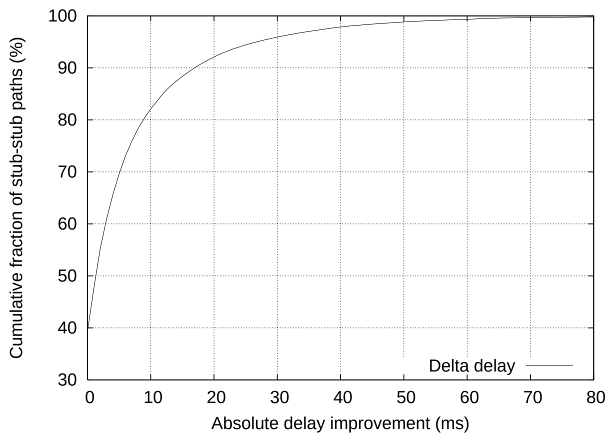


Figure 5.12: Distribution of the absolute delay improvement.

network connects to more providers, it is possible to find shorter routes in term of delay. First, having more providers implies that a larger fraction of the paths can be improved. For instance, for dual-homed stubs, there are less than 50% of paths that can be improved. For stubs that have 3 providers, more than 45% can be improved. With stubs that have 4 providers, the fraction grows to about 40% and with more 5 providers and more, up to 30% of paths can be improved. However, the marginal gain in possible delay improvement quickly decreases.

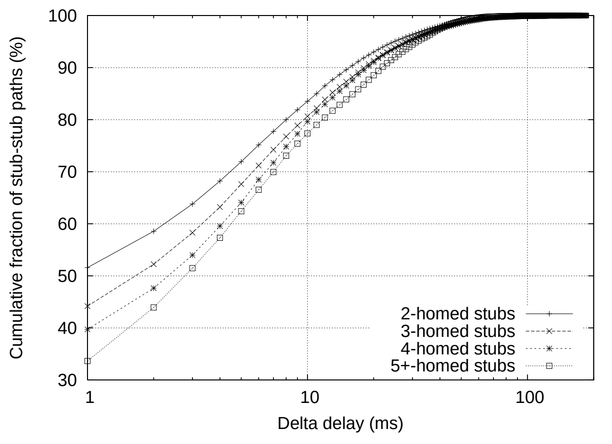


Figure 5.13: Cumulative distribution of the absolute delay improvement for k-homed stubs.

## 5.6 Conclusion

In this chapter, we have presented a cooperative approach to the engineering of interdomain traffic. This approach relies on Virtual Peerings. A Virtual Peering is a unidirectional IP tunnel between a border router chosen by the source AS and a border router chosen by the destination AS. Our solution to establish a Virtual Peering relies on three basic principles. First, there is a Virtual Peering Controller inside each AS and its IP address is attached as a BGP extended community to all BGP advertisements originated by the AS. Second, a multi-hop eBGP session is established between the VPCs of the source and destination ASs to negotiate Virtual Peerings. The source AS selects the head-end of the Virtual Peering based on its own traffic engineering objectives. Third, the destination AS selects autonomously the tail-end of the Virtual Peering. A key advantage of our approach is that **it can be incrementally deployed** inside cooperating stub ASs and does not require any change to the transit ASs. Given the size of the global Internet and the number of BGP routes, this incremental deployment is a key operational problem that must be considered.

We have shown that Virtual Peerings can be used to solve interdomain traffic engineering problems. We evaluated them on two particular instances, namely balancing the load of traffic and decreasing the latency of interdomain paths. When used to balance the load of traffic, we have shown that Virtual Peerings are a **scalable solution** since a limited number of them will be required to reach a near-perfect equilibrium. Typically, with 1000 sources responsible for 95% of the inbound traffic, on the order of 50 virtual peerings are required to balance the traffic.

In the second case, we have shown that using Virtual Peerings makes possible to forward traffic on interdomain paths with a lower delay. We compared the delay obtained on paths selected by the genuine BGP and the paths obtained by using Virtual Peerings and we showed that for more than 40% of the source/destination pairs an improvement of the delay was possible. This is in accordance with previous work by Savage et al that shown that 30-80% of end-to-end paths could benefit from a significant improvement in quality by using an alternative path [SCH+99].

# Conclusion

A key contribution of this thesis is the BGP routing solver that we introduced in Chapter 2. Its purpose is to simulate BGP routing in large networks. Our BGP routing solver contains the complete BGP decision process and versatile route filters combined with a simple message passing model. Our BGP routing solver was specificaly designed for modeling the steady-state of BGP routing. For this reason, we obtain superior performance with regard to existing packet-level network simulators. We removed the functionnalities of BGP such as the TCP connections and the BGP timers that are useless for a model of BGP in steady-state mode. Modeling these functionnalities would have limited the performance of the route computation since additional processing and resources would have been needed. We have shown that running such a model in a topology composed of several thousands of routers do not require a large cluster of computers. This is in contrast with current BGP models found in packet-level simulators such as SSFNet. We made our implementation of this BGP routing solver, C-BGP, publicly available. Then, we used C-BGP throughout the whole thesis. We first applied it to the modeling of a single AS before applying it to large Internet-scale topologies.

We have first applied C-BGP to model an ISP in Chapter 3. We started by identifying the key factors that must be encompassed by such a model. Among these factors are the network topology, the traffic, the routing protocols and the routers configuration. We have shown that obtaining this data can still be an operational issue. We illustrated the use of our routing solver through two different case studies. The first case study was an analysis of the impact of adding/removing the peers of a transit AS on its traffic. Typically, an ISP will try to decrease the delay of transit inside its network for external destinations and at the same time balance the load of traffic on its peering links. This is difficult to optimize due to the interactions between the IGP and the BGP routing protocols. We have shown that using a model of an ISP makes possible to explore various peering solutions. It would not have been possible to perform this kind of analysis without a model of the ISP that takes into account the interdomain routing information received from outside the ISP.

In a second study we investigated the impact of link and router failures on the routing and on the traffic matrix inside an ISP. This is an important problem since network events such as router hardware failures, link cuts and maintenances are frequent. C-BGP helps a network operator to identify the links and routers that could lead to large service disruptions. These links would be good candidate for the addition of parallel links or the deployment of protection techniques such as SONET-SDH and MPLS tunnels. We have shown that a pure intradomain model of an ISP such as used by the current versions of commercial network design and planning tools would miss most of the routing changes that occur under single link and router failures. This is a second motivation for taking into account the interdomain routes in a model of an ISP.

The second part of the thesis is dedicated to interdomain traffic engineering. In Chapter 4, we surveyed the current BGP-based interdomain traffic engineering techniques. We focused on the techniques that allow to control the incoming traffic. These techniques work by influencing the routing decisions taken by BGP routers in distant ASs. For this reason, a model of the Internet topology is required to evaluate them. We used C-BGP to perform large-scale simulations of BGP routing. We observed that a large fraction of the interdomain routes are selected by the tie-breaking rules of the BGP decision process. This is a bad news for the current incoming traffic engineering techniques since many routing decisions depend on local conditions in distant ASs. We further evaluated AS-Path prepending with our large-scale simulation model. We observed that the outcome of AS-Path prepending is difficult to predict. Moreover the granularity of AS-Path prepending is coarse. In practice, the usefulness of AS-Path prepending is limited to signaling backup links. A second technique that is increasingly deployed by ISPs relies on BGP Communities. Though it provides a finer granularity than AS-Path prepending, its outcome is not easier to predict. The main cause is the limited view an ISP has on the Internet topology due to the path-vector nature of BGP and the enforcement of routing policies. In addition, the BGP Communities are more difficult to use in practice since the number of possible Communities assignment is combinatorial. We concluded that the current BGP-based traffic engineering techniques are not appropriate to control the incoming traffic of an ISP in today's Internet.

In Chapter 5, we proposed an alternative solution to the interdomain traffic engineering problem which relies on cooperation. We describe the Virtual Peerings, a scalable, deterministic solution that cooperating ASs can use to better control the interdomain paths between each other. The main advantage of this approach is to be deployable in the current Internet since Virtual Peerings are transparent to the core of the Internet and they require only small modifications to the BGP routers in the cooperating ASs. We evaluated the utilization of the Virtual Peerings to solve two different interdomain traffic engineering problems.

We first focused on the traffic load balancing issue faced by multi-homed stub ISPs. Many ISPs in this situation have their incoming traffic unevenly balanced on their access links. This unbalance can cause congestion on the access links and disrupt the connectivity service. Using Virtual Peering, it is possible to contact some cooperating ASs that are responsible of an excess of traffic on an access link and ask them to forward the packets destined to the stub ISP in a tunnel headed at another ingress router. We evaluated how many tunnels would be required to perfectly balance the incoming traffic of a stub ISP in various traffic and connectivity conditions and we observed that only a limited number of them was required.

Using Virtual Peerings proves thus to be a scalable solution to the load-balancing problem. The same technique could also be used for cost-savings optimization or to enforce traffic policies such as a differentiated service through a premium provider and a best-effort provider.

The second application of Virtual Peerings that we investigated is the reduction of the latency between two ISPs. Using Virtual Peerings, it is possible to control the first and last segments of an interdomain path. We have shown that typically, the path diversity between two stub ISPs obtained with Virtual Peerings is on the order of the number of providers of one stub times the number of providers of the other stub. Among these paths, BGP is eager to select the worst one in term of latency since the BGP decision process does not include a QoS metric and the AS-Path length is seldom correlated with the latency. Based on a synthetic Internet topology, we investigated how many times the latency between a pair of stub domains could be reduced by using Virtual Peerings. We also investigated how much the latency would be improved in these cases. We observed that a significant fraction of the stub pairs (40% for our synthetic topology) could benefit from a latency improvement while such improvements could be on the order of tenths of milliseconds.

## Perspective

An important hypothesis we made in Chapter 5 is the synthetic router-level topology that we designed to evaluate the utilization of Virtual Peerings for latency improvement. Although the conclusion that a large number of AS pairs might benefit from significant latency reductions still holds, the quantitative results obtained from this evaluation could differ from those obtained in the real world Internet. There were two important motivations for designing an ad-hoc topology. First, we needed an interdomain topology with realistic delays. In our model, we assigned the delay of each link based on the link mileage [ZCBD05]. Second, we needed ASs composed of multiple routers. The internal structure of a domain is important since it constrains the paths that intradomain and interdomain routing protocols will select. Hot-potato routing and interdomain routing policies might lead to paths with an increased delay [SPK02]. Today, we do not have enough detailed measurements of the real Internet topology in order to build a validated model. No topology generator is able to produce router-level topologies of the global Internet in a satisfactory way.

The problem of obtaining an accurate picture of the Internet topology is not new as the recent publications attest [MP01, SARK02, SMW02, GR02, MRWK03, CGJ$^+$04, MH05, SS05, DRFC05, DKR05]. We do not know the exact shape of the Internet, especially at the router level and there are multiple reasons for this. First, we do not know about the internal structure of domains since their operators are often reluctant to publish the topology of their network. Attempts at inferring network topologies at the router-level from measurements such as Rocketfuel

[SMW02] sample the real network topology, sometimes missing parallel paths between routers or failing to resolve router aliases which results in links and routers that do not really exist [TMSV03]. Second, we do not know how domains are connected together. There is no map of the Internet available today. Looking at BGP routing tables from a small set of monitoring points [SARK02, DKR05] provides a gross picture of the interdomain graph. However, this approach misses a large number of peering links, mainly of the shared-cost type as shown in [CGJ+04]. These links are valuable for the interdomain routing protocol. In addition, the interdomain graph that we have today indicates that two domains are connected together, but it does not provide information on where the interconnection takes place or the link redundancy. Finally, an important characteristic of the interdomain graph are the routing policies. To the opposite of the intradomain graph, not all paths through the interdomain graph are allowed. These paths are constrained by the policies that are enforced by the domains in an independent manner [Gao00, SARK02, BPP03].

We advocate for the development of a validated model of the router-level structure of the Internet. This model should take into account the following aspects. First, the geographic location of routers [Mal02, SPK02, LBCM03] is important since it constrains the design of the network topology of each domain. Second, the geographical coverage of domains is another important factor. Typically, there are regional domains, national domains and international domains that can span multiple continents. Third, the network structure of domains, including the link capacities, the delays and the IGP weights, constrains the paths that can be used to cross a domain and drives the selection of preferred paths. Moreover, the link delays and capacities inside a domain are components of the end-to-end characteristics of interdomain paths [ZCBD05]. The iBGP configuration inside each domain is also an important aspect. A full-mesh of iBGP sessions will provide a better path diversity than a hierarchical iBGP [CPB04]. Typically, large domains will rely on a hierarchical iBGP configuration while small to moderate-size domains will use a full-mesh. Fourth, the geographical location of peering links matters. Two domains will usually establish peering links at places where they both have equipment [SPK02, RS02]. Fifth, the redundancy of interdomain links (parallel links) must be modeled. In the real world Internet, domains establish parallel links for resilience and performance reasons. Finally, the different types of business performed by domains as well as the business relationships between domains should also be modeled. Indeed, there are different types of domains such as transit domains, content providers, research networks and so on and their behaviors differ.

Besides that, the utilization of Virtual Peerings poses serious challenges from an operational viewpoint and deserves some more attention. A first further work would be to study how to make Virtual Peerings more secure. Virtual Peerings exhibit two major issues concerning security. On the side of the traffic, the utilization of a protocol such as IPSec could serve to check the identity of the participants as well as to ensure the privacy of the data. On the side of routing, the secured versions of BGP currently discussed at the IETF could be used. The main barrier for the wide utilization of both IPSec and S-BGP/soBGP is the need for a global public key infrastructure (PKI). Unfortunately, since the PKI has been initially laid out, it has appeared to be difficult to deploy and poorly scalable. In the context of Virtual Peerings, an alternative would be to rely on a more local establishment of trust through the use of a web of trust for example.

A second further work would be to study the utilization of Virtual Peerings in transit domains. In the thesis, we restricted their utilization to stub domains. In this case the impact on routing is limited to the iBGP of the cooperating stubs. Indeed, since stub domains do not provide a transit service, no BGP advertisement is leaked over eBGP sessions and the stability of the global Internet is not affected. Transit domains would also likely benefit from the utilization of Virtual Peerings and the principles we explained in the thesis still apply. However, in contrast with stub domains, a transit network would advertise the Virtual Peering routes outside its domain. In the case of regional and national transit networks these routes would only be propagated downstream to their customers. The impact on eBGP would thus be limited to these customers. In the case of large transit domains, the BGP stability of the whole Internet could be impacted. In addition to the impact on routing, using Virtual Peerings with transit networks would withdraw the predictability of the mechanism. More than the traffic originated by the transit network could be forwarded through the requested tunnel. One possible solution would be to prevent transit domains to advertise the Virtual Peering route to their eBGP neighbors. In addition, they could rely on policy-routing [Veg01] to forward traffic in a differentiated manner. Policy routing would be used to forward the traffic originated in the transit domain through the tunnel and the transit traffic would still be forwarded along the genuine BGP routes.

Finally, a possible further work would be to continue improving the scalability of the routing solver proposed in Chapter 2. C-BGP has been designed to perform the experiments presented within this thesis in a reasonable time and with a limited amount of computing resources. It made possible to perform studies of BGP behavior on topologies that are several orders of magnitude larger than in the previously published work. However, it is not an optimal BGP routing solver. Even if it outperforms current BGP routing models for the computation of the outcome of the BGP decision process in large topologies, it still requires important resources. An implementation and evaluation of the technique proposed by Hao and Kopol [HK03] in C-BGP would be valuable not only for C-BGP users but also for the entire BGP routing modeling community. The message passing model used in C-BGP also deserves some more attention. It can lead to expensive path exploration that worsens the execution time. A smarter message scheduling model would certainly prove useful.

# Appendix A

# Redundancy in BGP routing tables

In this appendix, we show additional results for the BGP routing tables redundancy measurement discussed in Section 2.5.9. The results shown here concern BGP routes collected in the Abilene network and data collected by the RouteViews project. Refer to Section 2.5.9 for more information on how to interpret the results.

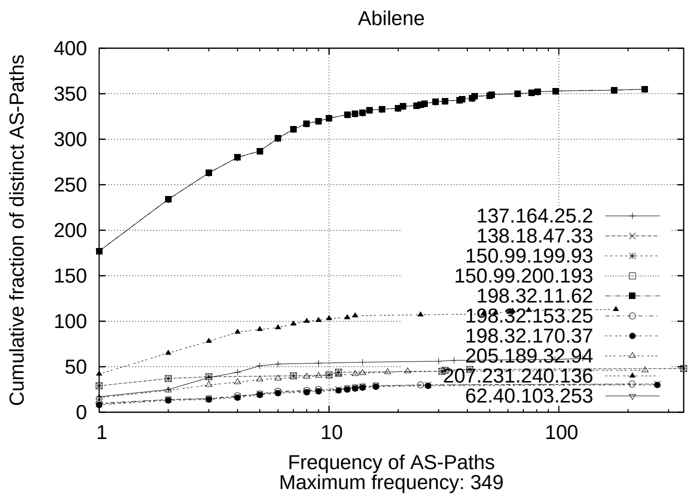


Figure A.1: Frequency of AS-Paths in Abilene routing tables.

In Fig. A.1 and Fig. A.2, we show the frequency of AS-Paths in Abilene and RouteViews RIBs respectively. We observe that nearly half the AS-Paths are present in at least 2 different routes. The maximum frequency of an AS-Path ranges from 349 in Abilene RIBs to 5906 in RouteViews RIBs. The figures also show that for a single peer, there are about 25.000 different AS-Paths in the RouteViews routing tables. For Abilene these numbers are slightly lower since Abilene only allows prefixes belonging to research and educational institutions to cross its network.

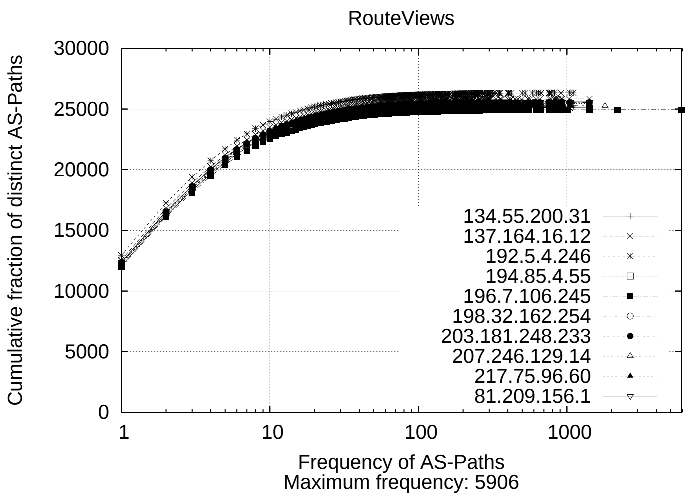


Figure A.2: Frequency of AS-Paths in RouteViews routing tables.

We show the results for the Communities redundancy in Fig. A.3 and Fig. A.4. We observe that the Communities redundancy is higher than the AS-Path redundancy. The maximum frequency of Communities ranges from 694 in Abilene RIBs to 158,158 in RouteViews RIBs.

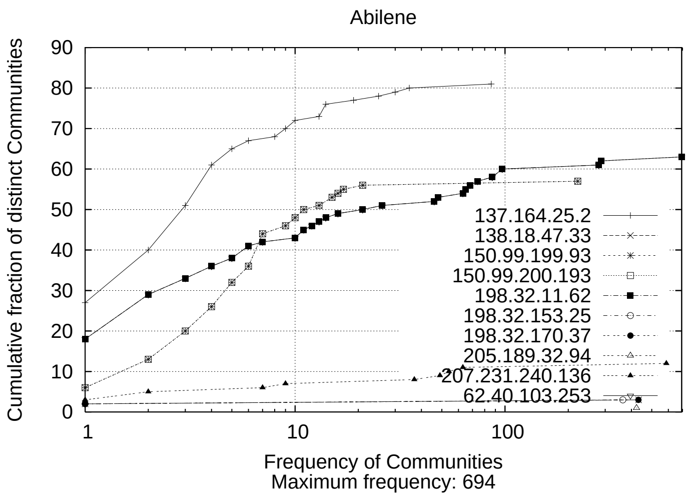


Figure A.3: Frequency of Communities in Abilene routing tables.

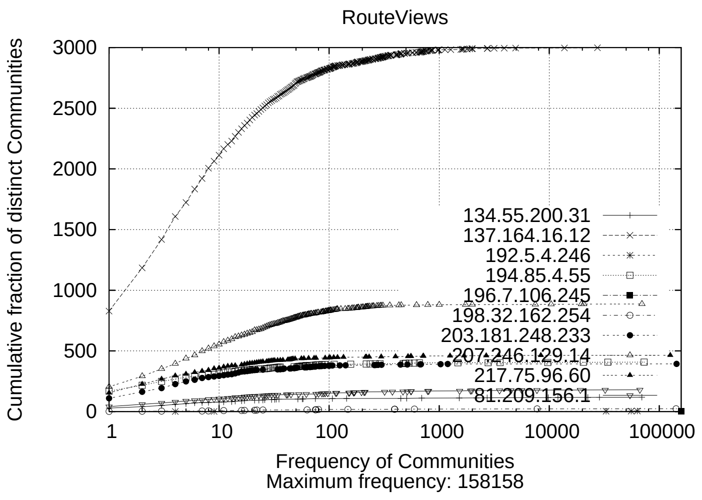


Figure A.4: Frequency of Communities in RouteViews routing tables.

# Bibliography

[AAM98] D. O. Awduche, J. Agogbua, and J. McManus. An Approach to Optimal Peering Between Autonomous Systems in the Internet. In *Proceedings of IEEE ICCN'98*, October 1998.

[ABG+01] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. RSVP-TE: Extensions to RSVP for LSP Tunnels. Internet Engineering Task Force, RFC3209, December 2001.

[ABKM01] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 131–145, 2001.

[ACBD04] S. Agarwal, C. N. Chuah, S. Bhattacharyya, and C. Diot. Impact of BGP Dynamics on Intra-Domain Traffic. *ACM SIGMETRICS Performance Evaluation Review*, 32, June 2004.

[ACE+02] D. O. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and Principles of Internet Traffic Engineering. Internet Engineering Task Force, RFC3272, May 2002.

[ACK03] S. Agarwal, C. Chuah, and R. Katz. OPCA: Robust interdomain policy routing and traffic control. In *IEEE Openarch*, New York, NY, April 2003.

[Adc04] S. Adcock. GAUL, the Genetic Algorithm Utility Library. `http://gaul.sourceforge.net`, 2004.

[AG04] S. Agarwal and T. Griffin. BGP Proxy Community Community. Internet draft, draft-agarwal-bgp-proxy-community-00, work in progress, January 2004.

[AKZ99] G. Almes, S. Kalidindi, and M. Zekauskas. A One-way Delay Metric for IPPM. Internet Engineering Task Force, RFC2679, September 1999.

[ALD+05] J. Abley, K. Lindqvist, E. Davies, B. Black, and V. Gill. IPv4 Multihoming Practices and Limitations. Internet Engineering Task Force, RFC4116, July 2005.

[All02] D. Allen. NPN: Multihoming and route optimization: Finding the best way home. *Network Magazine*, February 2002. available from http://www.networkmagazine.com/article/NMG20020 206S0004.

[ANB05] S. Agarwal, A. Nucci, and S. Bhattacharyya. Measuring the Shared Fate of IGP Engineering and Interdomain Traffic. In *Proceedings of IEEE ICNP*, November 2005.

[BA99] A. L. Barábasi and R. Albert. Emergence of Scaling in Random Networks. *Science*, pages 509–512, October 1999.

[Bar00] S. Bartholomew. The Art of Peering. *BT Technology Journal*, 18(3), July 2000.

[BBGR01] S. Bellovin, R. Bush, T. Griffin, and J. Rexford. Slowing routing table growth by filtering based on address allocation policies. preprint available from `http://www.cs.princeton.edu/~jrex/papers/filter.pdf`, June 2001.

[BCC00] T. Bates, R. Chandra, and E. Chen. BGP Route Reflection - An Alternative to Full Mesh IBGP. Internet Engineering Task Force, RFC2796, April 2000.

[BCH$^+$03] O. Bonaventure, S. De Cnodder, J. Haas, B. Quoitin, and R. White. Controlling the redistribution of bgp routes. Internet draft, draft-ietf-grow-bgp-redistribution-00.txt, work in progress, April 2003.

[BE04] U. Brandes and T. Erlebach. *Network Analysis: Methodological Foundations, LNCS3418*. Springer-Verlag, 2004.

[Ben02] J. Bengtsson. Memtime. `http://www.update.uu.se/~johanb/memtime`, 2002.

[BJ05] H. Brauer and C. Jeker. OpenBGPD. Available from `http://www.openbgpd.org`, 2005.

[BNC02] A. Broido, E. Nemeth, and K. Claffy. Internet expansion, refinement and churn. *European Transactions on Telecommunications*, January 2002.

[Bon05] O. Bonaventure. LISIS - A simple ISIS analyser. `http://totem.info.ucl.ac.be/lisis_tool/`, June 2005.

[Bor02] S. Borthick. Will Route Control Change The Internet ? *Business Communications Review*, September 2002.

[BP05] S. Bryant and P. Pate. Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture. Internet Engineering Task Force, RFC3985, March 2005.

[BPP$^+$] G. Di Battista, M. Patrignani, S. Pettini, M. Pizzonia, F. Ricci, and M. Rimondini. Netkit: The poor man system for experimenting computer networks. Available from `http://www.netkit.org/`.

[BPP03] G. Di Battista, M. Patrignani, and M. Pizzonia. Computing the Types of the Relationships between Autonomous Systems. In *Proceedings of IEEE INFOCOM*, March 2003.

[BQ03] O. Bonaventure and B. Quoitin. Common utilizations of the BGP community attribute. Internet draft, draft-bq-bgp-communities-00.txt, work in progress, June 2003.

[Bro88] R. Brown. Calendar Queues: A Fast O(1) Priority Queue Implementation for the Simulation Event Set Problem. *Communications of the ACM*, 31(10), October 1988.

[BTP$^+$03] O. Bonaventure, P. Trimintzios, G. Pavlou, B. Quoitin (Eds.), A. Azcorra, M. Bagnulo, P. Flegkas, A. Garcia-Martinez, P. Georgatsos, L. Georgiadis, C. Jacquenet, L. Swinnen, S. Tandel, and S. Uhlig. Internet Traffic Engineering. Chapter of COST263 final report, LNCS 2856, Springer-Verlag, September 2003.

[BZB$^+$97] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jarmin. Resource ReserVation Protocol (RSVP). Internet Engineering Task Force, RFC2205, September 1997.

[Cah98] R. S. Cahn. *Wide Area Network Design: Concepts and Tools for Optimisation*. Morgan Kaufmann, 1998.

[CB96] E. Chen and T. Bates. An Application of the BGP Community Attribute in Multi-home Routing. Internet Engineering Task Force, RFC1998, August 1996.

[CB05] B. Y. Choi and S. Bhattacharyya. Observations on CISCO sampled NetFlow. In *Proceedings of ACM SIGMETRICS Workshop on Large-Scale Network Inference (LSNI)*, June 2005.

[CDZ97] K. Calvert, M. Doar, and E. Zegura. Modeling Internet Topology. *IEEE Transactions on Communications*, pages 160–163, December 1997.

[CGJ$^+$04] H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Towards capturing representative AS-level Internet topologies. *Computer Networks*, 44(6):737–755, April 2004.

[CL05] R.K.C. Chang and M. Lo. Inbound Traffic Engineering for Multihomed ASs Using AS-Path Prepending. *IEEE Network Magazine*, pages 18–25, March/April 2005.

[CNO99] J.H. Cowie, D.M. Nicol, and A.T. Ogielski. Modeling the global Internet. *IEEE Computational Science and Engineering*, 1(1):42–50, January/February 1999. Available from `http://www.ssfnet.org`.

[CPB04] S. Uhlig C. Pelsser and O. Bonaventure. On the difficulty of establishing interdomain LSPs. In *Proceedings of the IEEE International Workshop on IP Operations and Management (IPOM)*, October 2004.

[CR05] M. Caesar and J. Rexford. BGP routing policies in ISP networks. *IEEE Network Magazine*, 19(6), November 2005.

[CS05] E. Chen and S. R. Sangli. Avoid BGP Best Path Transition from One External to Another. Internet draft, draft-chen-bgp-avoid-transition-04.txt, work in progress, December 2005.

[Deb01] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, June 2001.

[Dik] J. Dike. User-mode linux. Available from `http://user-mode-linux.sourceforge.net`.

[DKR05] X. Dimitropoulos, D. Krioukov, and G. Riley. Revisiting Internet AS-level Topology Discovery. In *Proceedings of the 6$^{th}$ Passive and Active Measurement Workshop (PAM)*, 2005.

[dLBL03] C. de Launois, O. Bonaventure, and M. Lobelle. The NAROS Approach for IPv6 Multi-homing with Traffic Engineering. In *Proceedings of QoFIS, LNCS 2811, Springer-Verlag*, pages 112–121, October 2003.

[dLQB06] C. de Launois, B. Quoitin, and O. Bonaventure. Leveraging network performance with IPv6 and multiple provider-dependent aggragatable prefixes. *Computer Networks*, 50:1145–1157, June 2006.

[Dom00] G. Dommety. Key and Sequence Number Extensions to GRE. Internet Engineering Task Force, RFC2890, September 2000.

[DR00] B. Davie and Y. Rekhter. *MPLS Technology and Applications*. Morgan Kauffmann, 2000.

[DR04] X. A. Dimitropoulos and G. F. Riley. Large-Scale Simulation Models of BGP. In *Proceedings of 12th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2004)*, October 2004.

[DR06] X. A. Dimitropoulos and G. F. Riley. Large-Scale Simulation Models of BGP. *Computer Networks (Elsevier)*, 50:2013–2027, August 2006.

[DRFC05] B. Donnet, P. Raoult, T. Friedman, and M. Crovella. Efficient Algorithms for Large-Scale Topology Discovery. In *Proceedings of ACM Sigmetrics*, June 2005.

[ES03] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, 2003.

[FB05] N. Feamster and H. Balakrishnan. Detecting BGP Configuration Faults with Static Analysis. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI)*, May 2005.

[FBR03] N. Feamster, J. Borkenhagen, and J. Rexford. Guidelines for interdomain traffic engineering. *ACM SIGCOMM Computer Communications Review*, October 2003.

[FdMG04] D. Fernandez, T. de Miguel, and F. Galan. Study and Emulation of IPv6 Internet-Exchange-Based Addressing Models. *IEEE Communications Magazine*, 42(1), January 2004.

[FE04] C. Filsfils and J. Evans. Deploying Diffserv in Backbone Networks for Tight SLA Control. *IEEE Internet Computing*, January-February 2004.

[FFF99] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. In *Proceedings of ACM SIGCOMM 1999*, August 1999.

[FGK$^+$01] 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 the Passive and Active Measurements Workshop (PAM2001)*, Amsterdam, April 2001.

[FGL$^+$00] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford. NetScope: Traffic Engineering for IP Networks. *IEEE Network Magazine*, March 2000.

[FLH$^+$00] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic Routing Encapsulation (GRE). Internet Engineering Task Force, RFC2784, March 2000.

[FMMF05] L. Forst, M. Mares, P. Machek, and O. Filip. BIRD Internet Routing Daemon. `http://bird.network.cz/`, 2005.

[FP01] S. Floyd and V. Paxson. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking*, 9(4), November 2001.

[FR01] A. Feldmann and J. Rexford. IP Network Configuration for Intradomain Traffic Engineering. *IEEE Network Magazine*, pages 46–57, September/October 2001.

[FRT02] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, October 2002.

[FT00] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings of IEEE INFOCOM*, pages 519–528, March 2000.

[FT03] B. Fortz and M. Thorup. Robust optimization of OSPF/IS-IS weights. In *Proceedings of INOC*, pages 225–230, October 2003.

[FWR04] N. Feamster, J. Winick, and J. Rexford. A model of BGP routing for network engineering. In *ACM SIGMETRICS*, pages 331–342, June 2004.

[Gao00] L. Gao. On Inferring Autonomous System Relationships in the Internet. *IEEE Global Internet*, November 2000.

[GCLC04] F. Guo, J. Chen, W. Li, and T. Chiueh. Experiences in Building a Multihoming Load Balancing System. In *Proceedings of IEEE INFOCOM*, March 2004.

[GDZ05] R. Gao, C. Dovrolis, and E. W. Zegura. Interdomain Ingress Traffic Engineering Through Optimized AS-Path Prepending. In *4th International IFIP-TC6 Networking Conference*, May 2005.

[GF04] F. Galan and D. Fernandez. Virtual Network User Mode Linux. Available from `http://jungla.dit.upm.es/~vnuml`, 2004.

[GH05] T. Griffin and G. Huston. BGP Wedgies, November 2005. Internet Engineering Task Force, RFC4264.

[GJT04] A. Gunnar, M. Johansson, and T. Telkamp. Traffic Matrix Estimation on a Large IP Backbone - A Comparison on Real Data. In *Proceedings of ACM IMC*, October 2004.

[GP01] T. Griffin and B. Premore. An Experimental Analysis of BGP Convergence Time. In *Proceedings of ICNP 2001*, pages 53–61. IEEE Computer Society, November 2001.

[GR00] L. Gao and J. Rexford. Stable internet routing without global coordination. In *SIGMETRICS*, 2000.

[GR02] R. Govindan and P. Radoslavov. An Analysis of The Internal Structure of Large Autonomous Systems. Technical Report Technical Report 02-777, Computer Science Department, University of Southern California, November 2002.

[Gro04] W. D. Grover. *Mesh-Based Survivable Networks*. Prentice Hall PTR, 2004.

[GW99] T. Griffin and G. Wilfong. An Analysis of BGP Convergence Properties. In *Proceedings of ACM SIGCOMM*, August 1999.

[GW00] T. Griffin and G. T. Wilfong. A Safe Path Vector Protocol. In *IEEE INFOCOM*, pages 490–499, 2000.

[GW02a] T. Griffin and G. Wilfong. Analysis of the MED Oscillation Problem in BGP. In *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP)*, pages 90–99, November 2002.

[GW02b] T. Griffin and G. Wilfong. On the Correctness of IBGP Configuration. In *Proceedings of ACM SIGCOMM*, pages 17–29, August 2002.

[HFP$^+$02] B. Huffaker, M. Fomenkov, D. Plummer, D. Moore, and K. Claffy. Distance Metrics in the Internet. In *Proceedings of IEEE International Telecommunications Symposium (ITS)*, September 2002.

[HK03] F. Hao and P. Koppol. An Internet Scale Simulation Setup for BGP. *ACM SIGCOMM Computer Communication Review*, 33(3):43–57, July 2003.

[HP00] B. Halabi and D. Mc Pherson. *Internet Routing Architectures (2nd Edition)*. Cisco Press, January 2000.

[Hus99] G. Huston. The Politics and Economics of Peering and Interconnection. In *Proceedings of the Internet Society INET Conference. Available from `http://www.isoc.org/inet99/proceedings/1e/1e_1.htm`*, June 1999.

[Hus05] G. Huston. BGP Routing Table Analysis Reports. `http://bgp.potaroo.net`, 2005.

[Hus06] G. Huston. The CIDR Report, January 2006. `http://www.cidr-report.org/`.

[Ias05] Stefano Iasi. An OSPF Model for C-BGP. Master's thesis, Politecnico di Torino, Italy, November 2005.

[ISC05] ISC. Internet Domain Survey, June 2005. `http://www.isc.org/index.pl?/ops/ds`.

[Ish96] K. Ishiguro. GNU Zebra: free routing software. Available from `http://www.zebra.org`, 1996.

[JCJ00] C. Jin, Q. Chen, and S. Jamin. Inet: Internet Topology Generator. Technical report, 2000.

[Ken05] S. Kent. IP Encapsulating Security Payload (ESP). Internet Engineering Task Force, RFC4303, December 2005.

[KKW$^+$03] H. Tahilramani Kaur, S. Kalyanaraman, A. Weiss, S. Kanwar, and A. Gandhi. BANANAS: an evolutionary framework for explicit and multipath routing in the internet. In *Proceedings of the ACM SIGCOMM, FDNA*, pages 277–288, 2003.

[KKY03] R. Katz, K. Kompella, and D. Yeung. Traffic Engineering (TE) Extensions to OSPF Version 2. Internet Engineering Task Force, RFC3630, September 2003.

[KLS00] S. Kent, C. Lynn, and K. Seo. Secure Border Gateway Protocol (S-BGP). *IEEE Journal on Selected Areas in Communications*, 18(4):582–592, April 2000.

[Knu00] D. E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, second edition, January 2000.

[KW06] D. Katz and D. Ward. Bidirectional Forwarding Detection. Internet draft, draft-ietf-bfd-base-05.txt, work in progress, June 2006.

[Lab05] Wide Area Network Design Laboratory. IP/MPLSView. `http://www.wandl.com`, 2005.

[LABJ00] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. In *Proceedings of ACM SIGCOMM*, pages 175–187, Stockholm, Sweden, September 2000.

[LBCM03] A. Lakhina, J. Byers, M. Crovella, and I. Matta. On the Geographic Location of Internet Resources. *IEEE Journal on Selected Areas in Communications*, 21(6):934–948, August 2003.

[LIN06] LINX. Pseudowire / Layer-2 Connect. `https://www.linx.net/www_public/our_services/layer2_connect/?searchterm=martini`, May 2006.

[LTG05] J. Lau, M. Townsley, and I. Goyret. Layer Two Tunneling Protocol - Version 3 (L2TPv3). Internet Engineering Task Force, RFC3931, March 2005.

[Mal02] E. J. Malecki. The Economic Geography of the Internet's Infrastructure. *Economic Geography*, 78(4), October 2002.

[MAMB01] A. Medina, A.Lakhina, I. Matta, and J. Byers. BRITE: An Approach to Universal Topology Generation. In *MASCOTS 2001*, August 2001.

[max04] MaxMind GeoIP City Database. `http://www.maxmind.com/app/city`, June 2004.

[MD91] J. Mogul and S. Deering. Path MTU Discovery. Internet Engineering Task Force, RFC1191, November 1991.

[Mey05] D. Meyer. Route Views Archive Project (University of Oregon). `http://archive.routeviews.org/`, 2005.

[MGVK02] Z. M. Mao, R. Govindan, G. Varghese, and R. Katz. Route Flap Damping Exacerbates Internet Routing Convergence. In *Proceedings of ACM SIGCOMM*, August 2002.

[MH05] D. Magoni and M. Hoerdt. Internet core topology mapping and analysis. *Computer Communications*, 28:494–506, March 2005.

[Mor68] D. R. Morrison. PATRICIA - Practical Algorithm To Retrieve Information Coded in Alphanumeric. *Journal of the ACM*, 15(4):514–534, October 1968.

[Moy98] J. Moy. *OSPF : anatomy of an Internet routing protocol*. Addison-Wesley, 1998.

[MP01] D. Magoni and J. J. Pansiot. Analysis of the Autonomous System Network Topology. *ACM SIGCOMM Computer Communication Review*, 31, July 2001.

[MPDZ] M. Mikuc, Z. Puljiz, N. Djurak, and M. Zec. IMUNES - An Integrated MUltiprotocol Network Emulator/Simulator. Available from `http://tel.fer.hr/imunes`.

[MRG] Berkeley MASH Research Group, University of California. The Network Simulator ns-2. Available from `http://www.isi.edu/nsnam/ns`.

[MRWK03] Z. M. Mao, J. Rexford, J. Wang, and R. H. Katz. Towards an AS-level traceroute tool. In *Proceedings of ACM SIGCOMM*, August 2003.

[MSM05] T. G. Mattson, B. A. Sanders, and B. L. Massingill. *Patterns for Parallel Programming*. Addison-Wesley, 2005.

[MWA02] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP misconfigurations. In *Proceedings of ACM SIGCOMM 2002*, August 2002.

[NCM02] J. Nykvist and L. Carr-Motyckova. Simulating Convergence Properties of BGP. In *Proceedings of the 11th International Conference on Computer Communications and Networks (IC3N 2002)*, pages 124–129, October 2002.

[Net04] Juniper Networks. IP services PICs : datasheet. `http://www.juniper.net/products/modules/100048.html`, 2004.

[NK99] S. Nilsson and G. Karlsson. IP-Address Lookup Using LC-Tries. *IEEE Journal on Selected Areas in Communications*, June 1999.

[NKTW05] G. Nalawade, R. Kapoor, D. Tappan, and S. Wainner. Tunnel SAFI. Internet Draft, draft-nalawade-kapoor-tunnel-safi-04, work in progress, October 2005.

[Nor02] W. B. Norton. The Art of Peering: the Peering Playbook. preprint available from wbn@equinix.com, May 2002.

[Ora90] D. Oran. OSI IS-IS Intra-domain Routing Protocol. Internet Engineering Task Force, RFC1142, February 1990.

[PAMZ04] D. Pei, M. Azuma, D. Massey, and L. Zhang. BGP-RCN: Improving BGP convergence through root cause notification. *Computer Networks (Elsevier)*, 48(2):175–194, June 2004.

[Pre] B.J. Premore. SSF Implementation of BGP-4 v1.5.0. Available from `http://www.ssfnet.org/bgp/doc/validation.html`.

[Pre01] B. J. Premore. SSF Implementations of BGP-4. available from `http://www.cs.dartmouth.edu/~beej/bgp/`, 2001.

[Pre02] B. J. Premore. SSFNet BGP User's Guide. `http://www.ssfnet.org/bgp/user-guide-ps.zip`, 2002.

[QN04] X. Qie and S. Narain. Using service grammar to diagnose BGP configuration errors. *Science of Computer Programming Journal*, 53:125–241, November 2004.

[QPBU05] B. Quoitin, C. Pelsser, O. Bonaventure, and S. Uhlig. A performance evaluation of BGP-based traffic engineering. *International Journal of Network Management (Wiley)*, 15(3), May-June 2005.

[QTUB04] B. Quoitin, S. Tandel, S. Uhlig, and O. Bonaventure. Interdomain Traffic Engineering with Redistribution Communities. *Computer Communications Journal (Elsevier)*, 27(4):355–363, March 2004.

[Qua03] Quagga. Quagga Routing Suite. Available from `http://www.quagga.net`, 2003.

[Quo03a] B. Quoitin. BGP model for J-Sim. Available from `http://www.j-sim.org/contribute.html\#Infonet\_Suite`, 2003.

[Quo03b] B. Quoitin. C-BGP, an efficient BGP simulator. `http://cbgp.info.ucl.ac.be`, September 2003.

[QUP$^+$03] B. Quoitin, S. Uhlig, C. Pelsser, L. Swinnen, and O. Bonaventure. Interdomain traffic engineering with BGP. *IEEE Communications Magazine*, May 2003.

[RIP05] RIPE. Routing Information Service. `http://www.ripe.net/projects/ris/`, 2005.

[RL95] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). Internet Engineering Task Force, RFC1771, March 1995.

[RL04] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). Internet draft, draft-ietf-idr-bgp4-26.txt, work in progress, October 2004.

[RS02] B. Raveendran and P. Smith. *Cisco ISP Essentials*. Cisco Press, 2002.

[RWXZ02] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang. BGP Routing Stability of Popular Destinations. In *Proc. of ACM SIGCOMM Internet Measurement Workshop*, November 2002.

[SAA$^+$99] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: Informed Internet Routing and Transport. *IEEE Micro*, 19(1):50–59, 1999.

[SARK02] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz. Characterizing the Internet Hierarchy from Multiple Vantage Points. In *Proceedings of INFOCOM*, June 2002.

[SCE$^+$05] L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica. HLP: a next generation inter-domain routing protocol. In *Proceedings of ACM SIGCOMM*, pages 13–24, August 2005.

[SCH$^+$99] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The End-to-end Effects of Internet Path Selection. In *Proceedings of the ACM SIGCOMM Conference*, pages 289–299, September 1999.

[Scu05] J. Scudder. BGP Monitoring Protocol. Internet draft, draft-scudder-bmp-00.txt, work in progress, August 2005.

[SF02] R. Sommer and A. Feldmann. NetFlow: Information loss or win ? In *Proceedings of ACM Internet Measurement Workshop*, November 2002.

[Sim95] W. Simpson. IP in IP Tunneling. Internet Engineering Task Force, RFC1853, October 1995.

[SL04] H. Smit and T. Li. Intermediate System to Intermediate System (IS-IS) Extensions for Traffic Engineering (TE). Internet Engineering Task Force, RFC3784, May 2004.

[SMW02] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP Topologies with Rocketfuel. In *Proceedings of ACM SIGCOMM*, August 2002.

[SPK02] L. Subramanian, V. N. Padmanabhan, and R. H. Katz. Geographic Properties of Internet Routing. In *Proceedings of USENIX Technical Conference*, June 2002.

[SS05] Y. Shavitt and E. Shir. DIMES - Letting the Internet Measure Itself. `http://www.arxiv.org/abs/cs.NI/0506099`, June 2005.

[ST04] S. Shalunov and B. Teitelbaum. One-way Active Measurement Protocol (OWAMP) Requirements. Internet Engineering Task Force, RFC3763, April 2004.

[Ste99] J. Stewart. *BGP4 : interdomain routing in the Internet*. Addison Wesley, 1999.

[STR06] S. Sangli, D. Tappan, and Y. Rekhter. BGP Extended Communities Attribute. Internet Engineering Task Force, RFC4360, February 2006.

[Sys04] CISCO Systems. CISCO IOS Service Assurance Agent. http://www.cisco.com/warp/public/cc/pd/iosw/prodlit/saang_ds.pdf, 2004.

[Sys05a] CISCO Systems. BGP Best Path Selection Algorithm. `http://www.cisco.com/warp/public/459/25.shtml`, October 2005.

[Sys05b] CISCO Systems. BGP Peer Groups. `http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080093fb7.shtml`, August 2005.

[Sys05c] CISCO Systems. CISCO Optimized Edge Routing. `http://www.cisco.com/en/US/netsol/ns471/networking_solutions_package.html`, 2005.

[Tan06] S. Tandel. BGP Converter Technical Report. `http://www.info.ucl.ac.be/~standel/bgp-converter`, February 2006.

[TAR05] R. Teixeira, S. Agarwal, and J. Rexford. BGP routing changes: Merging views from two ISPs. *ACM SIGCOMM Computer Communications Review*, 35(5), October 2005.

[TCL96] P. Traina, R. Chandrasekeran, and T. Li. BGP Communities Attribute. Internet Engineering Task Force, RFC1997, August 1996.

[Tec05a] Cariden Technologies. MATE. `http://www.cariden.com/`, 2005.

[Tec05b] OPNET Technologies. SPGuru. `http://www.opnet.com/`, 2005.

[Tel02] T. Telkamp. Traffic Characteristics and Network Planning. In *NANOG26*, October 2002.

[TGVS04] R. Teixeira, T. Griffin, G. Voelker, and A. Shaikh. Network Sensitivity to Hot Potato Disruptions. In *Proceedings. of ACM SIGCOMM*, August 2004.

[TMS01] P. Traina, D. McPherson, and J. Scudder. Autonomous System Confederations for BGP. Internet Engineering Task Force, RFC3065, February 2001.

[TMSV03] R. Teixeira, K. Marzullo, S. Savage, and G. M. Voelker. Characterizing and Measuring Path Diversity of Internet Topologies. In *Proceedings of ACM SIGMETRICS*, June 2003.

[TOT05] TOTEM. A TOolbox for Traffic Engineering Methods. `http://totem.info.ucl.ac.be`, February 2005.

[TSGR04] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of Hot-Potato Routing in IP Networks. In *Proceedings of ACM SIGMETRICS*, June 2004.

[Tya02] H. Tyan. *Design, realization and evaluation of a component-based compositional software architecture for network simulation*. PhD thesis, Ohio State University, 2002.

[UB02] S. Uhlig and O. Bonaventure. Implications of Interdomain Traffic Characteristics on Traffic Engineering. In *European Transactions on Telecommunications, special issue on traffic engineering*, 2002.

[UBQ03a] S. Uhlig, O. Bonaventure, and B. Quoitin. Interdomain Traffic Engineering with minimal BGP Configurations. In *Proc. of the* $18^{th}$ *International Teletraffic Congress, Berlin*, September 2003.

[UBQ03b] S. Uhlig, O. Bonaventure, and B. Quoitin. Interdomain Traffic Engineering with minimal BGP configurations. In *Proceedings of the 18th International Teletraffic Congress, ITC*, September 2003.

[UMB$^+$04] S. Uhlig, V. Magnin, O. Bonaventure, C. Rapier, and L. Deri. Implications of the Topological Properties of Internet Traffic on Traffic Engineering. In *Proceedings of ACM SAC'04*, March 2004.

[Var05] G. Varghese. *Network Algorithmics: An Interdisciplinary Approach to Designing Fast Networked Devices*. Morgan Kaufmann, 2005.

[VE04] G. Varghese and C. Estan. The measurement manifesto. *SIGCOMM Computer Communications Review*, 34(1):9–14, 2004.

[Veg01] S. Vegesna. *IP Quality of Service*. Cisco Press, 2001.

[Veg05] L. Vegoda. Address Space Managed by the RIPE NCC. `http://www.ripe.net/ripe/docs/ripe-ncc-managed-address-space.html`, December 2005.

[VPD04] J.-P. Vasseur, M. Pickavet, and P. Demeester. *Network Recovery: Protection and Restoration of Optical, SONET-SDH, and MPLS*. Morgan Kaufmann, 2004.

[VSA06] J.-P. Vasseur, N. Shen, and R. Aggarwal. IS-IS Extensions for Advertising Router Information. Internet draft, draft-ietf-isis-caps-06.txt, work in progress, January 2006.

[Wax88] B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, December 1988.

[Whi03] R. White. Securing BGP Through Secure Origin BGP. *The Internet Protocol Journal*, 6:15–22, June 2003.

[WJR02] J. Winick, S. Jamin, and J. Rexford. Traffic Engineering Between Neighboring Domains. July 2002.

[Yan03] X. Yang. NIRA: a New Internet Routing architecture. In *Proceedings of the ACM SIGCOMM, FDNA*, pages 301–312, 2003.

[YFMB$^+$04] M. Yannuzzi, A. Fonte, X. Masip-Bruin, E. Monteiro, S. Sanchez-Lopze, M. Curado, and J. Domingo-Pascual. A Proposal for Inter-domain QoS Routing Based on Distributed Overlay Entities and QBGP. In *Proceedings of QoFIS*. Springer-Verlag, LNCS3266, September 2004.

[YXW$^+$05] Y. R. Yang, H. Xie, H. Wang, A. Silberschatz, A. Krishnamurthy, Y. Liu, and L. E. Li. On Route Selection for Interdomain Traffic Engineering. *IEEE Network Magazine*, 19(6), November 2005.

[ZB03] R. Zhang and M. Bartell. *BGP Design and Implementation: Practical guidelines for designing and deploying a scalable BGP routing architecture*. CISCO Press, 2003.

[ZCBD05] A. Zeitoun, C.N. Chuah, S. Bhattacharyya, and C. Diot. An AS-Level Study of Internet Path Delay Characteristics. Global Internet and Next Generation Network Workshop (GINGN). Available from `http://ipmon.sprint.com/pubs_trs/pubs/supratik/gingn2004-asdelay.pdf`, November 2005.

[Zec03] M. Zec. Implementing a clonable network stack in the freebsd kernel. In *Proceedings of the 2003. USENIX Annual Technical Conference*, San Antonio, Texas, USA, June 2003.

[ZRDG03] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast Accurate Computation of Large-Scale IP Traffic Matrices from Link Loads. In *Proceedings of ACM SIGMETRICS*, June 2003.