Lehrstuhl für Intelligente Netze und Management Verteilter Systeme
Institut für Telekommunikationssysteme / Deutsche Telekom Laboratories
Fakultät IV – Elektrotechnik und Informatik
Technische Universität Berlin

Inter-Domain Routing Under Scrutiny: Routing Models and Alternative Routing Architectures

vorgelegt von
Dipl.-Inf. Wolfgang Mühlbauer
aus Amberg

Von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades eines
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. habil. Odej Kao, Technische Universität Berlin
Berichter:	Prof. Anja Feldmann, Ph.D., Technische Universität Berlin
Berichter:	Prof. Jennifer Rexford, Ph.D., Princeton University

Tag der wissenschaftlichen Aussprache: 24. 10. 2009

Berlin 2009
D83

Abstract

Much of the mystery behind inter-domain routing in the Internet comes from its complexity: A large number of independently administered autonomous systems (ASs), interactions between intra- and inter-domain routing protocols, manifold routing policies, diverse peering structures between ASs, etc.

The thesis at hand aims to improve the understanding of inter-domain routing in general, presents novel approaches for the construction of inter-domain routing models with predictive capabilities, and suggests an alternative routing architecture for a future Internet.

We start with a comprehensive study of how sensitive routing optimality and diversity metrics are to factors such as policies, AS size, topology, and IGP weights, etc. Our findings reveal that intra-domain factors only have marginal impact on global path properties while routing policies and AS size (in number of routers) are the dominating factors. Moreover, it is apparently hard to improve the global properties of route selection by existing means, i.e., tweaking BGP attributes, changing iBGP graphs, etc.

Based on the obtained insights, we then discuss how to construct models of inter-domain routing with predictive capabilities regarding BGP paths. We start by showing the importance of considering more than one router per AS and introduce quasi-routers to capture path diversity as seen in observed routing data. Relying on the abstraction of quasi-routers, we show that our model provides accurate predictions for unobserved routes. Regarding routing policies our work then reveals that the granularity of actual routing policies is close to per-neighbor although the widely used AS relationship model fails to provide consistency between the routes propagated in our routing model and those seen in observed data.

Given several shortcomings of Internet routing such as routing table growth, high update rates, or lack of traffic engineering, we finally discuss long-term solutions for the future Internet. Our research in this area consists of two distinctive parts. First, we present *Trellis*, a network testbed that can be used by network researchers to implement, test, and evaluate (clean-slate) solutions for a future Internet. As such, it allows to partition a physical network into multiple logical or virtual networks, where each virtual network can define its own topology, routing protocols, and forwarding tables. Second, we introduce *HAIR*, a scalable routing architecture for a future Internet. *HAIR* uses a hybrid "edge-based" approach to reconcile network- and host-based routing architectures and is based on the following ideas: (i) use of hierarchical routing, (ii) separation of locators from identifiers, and (iii) use of a hierarchical mapping system. We estimate the expected benefits of deploying HAIR in today's Internet and report our promising experiences with a proof-of-concept implementation of HAIR.

Zusammenfassung

Inter-Domain Routing, also die Verkehrslenkung zwischen Autonomen Systemen (ASen), ist seit Jahren ein lebendiges Forschungsgebiet. Hauptursache hierfür ist dessen Komplexität: Zusammenspiel zwischen Interior und Exterior Gateway Protokollen, eine Vielfalt unterschiedlicher Routing Policies, komplexe Verbindungsstrukturen zwischen ASen, usw.

Ein wesentliches Ziel der vorliegenden Dissertation besteht in einem verbesserten Allgemeinverständnis des Inter-Domain Routings. Desweiteren werden neuartige Ansätze vorgestellt, um mit Hilfe von Routing-Modellen Vorhersagen zu treffen, und es werden alternative Konzepte für eine Routing-Architektur im zukünftigen Internet erarbeitet.

Im ersten Teil der Dissertation wird zunächst umfassend untersucht, welche Metriken für die Charakterisierung der Optimalität und Vielfalt der berechneten Pfade empfindlich sind für Faktoren wie Routing Policies, AS Größe, Topologie, IGP Gewichte, etc. Die Ergebnisse verdeutlichen, dass intra-domain Faktoren die globalen Pfadeigenschaften nur am Rande beeinflussen, während Routing-Policies und die Größe von ASen (bezüglich Anzahl der Router) die dominierenden Faktoren darstellen. Überdies erscheint es schwierig, die globalen Eigenschaften der Wegefindung durch existierende Mechanismen, wie z.B. die Manipulation von BGP Attributen oder die Modifikation von IBGP Graphen, zu verbessern.

Auf Grundlage der gewonnenen Erkenntnisse werden anschließend inter-domain Routing-Modelle zur Vorhersage von BGP Pfaden erläutert. Zunächst wird begründet, dass es wichtig ist, einzelne ASe durch mehrere Router zu modellieren. Das Konzept der *Quasi-Router* wird eingeführt, um die Pfadvielfalt zu erfassen, die man in Routing-Daten beobachten kann. Mit Hilfe von Quasi-Routern lassen sich genaue Vorhersagen auch für solche Pfade treffen, die nicht beobachtet wurden. Anschließend begründet die vorliegende Arbeit, dass die Granularität von Routing Policies überwiegend "Nachbar-zu-Nachbar" ist, obwohl das weitverbreitete Modell der AS Relationships keine Übereinstimmung zwischen den in einem Modell propagierten Routen und jenen Routen, die in der Realität beobachtet werden, erzielt.

Angesichts vorhandener Mängel wie beispielsweise Wachstum der Routing-Tabellen, hohe Anzahl an Update-Nachrichten oder ungenügende Möglichkeiten des Traffic Engineerings, werden im letzten Teil langfristige Lösungen für ein "Internet der Zukunft" erörtert. Dieser Abschnitt gliedert sich in zwei Teile: Auf der einen Seite wird *Trellis* vorgestellt, eine Netzwerk-Testumgebung, die zur Implementierung, zum Test und zur Evaluation von "clean-slate" Lösungen für das Internet der Zukunft verwendet werden kann. Das System erlaubt die Partitionierung eines physikalischen Netzes in mehrere logische oder virtuelle Netze, wobei jedes virtuelle Netz seine eigene Topologie, Routing-Protokolle und Forwarding-Tabellen definieren kann. Auf der anderen Seite wird *HAIR* vorgestellt, eine skalierbare Routing-Architektur für ein zukünftiges Internet. *HAIR* ist ein "randbasierter" (edge-based) Hybrid zwischen Netz- und Host-basierten Architekturen und stützt sich auf folgende Grundideen: (i) Verwendung von hierarchischem Routing, (ii) Trennung von Locator und Identifier, und (iii) Verwendung eines hierarchischen Mapping Systems. Es erfolgt eine Abschätzung des erwarteten Nutzens von *HAIR* für das heutige Internet und eine kurze Diskussion einer Prototyp-Implementierung für HAIR.

Publications

Parts of this thesis have been pre-published:

Peer-Reviewed Conferences and Workshop Papers

Wolfgang Mühlbauer, Anja Feldmann, Olaf Maennel, Matthew Roughan and Steve Uhlig
Building an AS-Topology Model that Captures Route Diversity
In Proceedings of ACM SIGCOMM, September 2006, Pisa, Italy

Wolfgang Mühlbauer, Steve Uhlig, Bingjie Fu, Mickael Meulle, Olaf Maennel
In Search for the Appropriate Granularity to Model Routing Policies
In Proceedings of ACM SIGCOMM, August 2007, Kyoto, Japan

Sapan Bhatia, Murtaza Motiwala, Wolfgang Mühlbauer, Vytautas Valancius,
Andy Bavier, Nick Feamster, Larry Peterson, Jennifer Rexford
Trellis: A Platform for Building Flexible, Fast Virtual Networks on Commodity Hardware
In Proceedings of 3rd International Workshop on Real Overlay and Distributed Systems (ROADS),
at CoNEXT'08, December 2008, Madrid, Spain

Anja Feldmann, Luca Cittadini, Wolfgang Mühlbauer, Randy Bush, Olaf Maennel
HAIR: Hierarchical Architecture for Internet Routing
In Proceedings of Workshop on Re-Architecting the Internet (ReArch '09),
at CoNEXT'09, December 2009, Rome, Italy

Technical Reports

Anja Feldmann, Randy Bush, Luca Cittadini, Olaf Maennel and Wolfgang Mühlbauer
HAIR: Hierarchical Architecture for Internet Routing
Technische Universität Berlin, 2008-14, ISSN: 1436-9915

Sapan Bhatia, Murtaza Motiwala, Wolfgang Mühlbauer, Vytautas Valancius,
Andy Bavier, Nick Feamster, Larry Peterson, Jennifer Rexford
Hosting Virtual Networks on Commodity Hardware
Georgia Tech Computer Science, GT-CS-07-10, 2008, Atlanta, GA, USA

Contents

1 Introduction **1**
1.1 Overview . . . 3

2 Background **5**
2.1 Internet Basics . . . 5
2.1.1 Internet Structure . . . 5
2.1.2 Packet Delivery . . . 7
2.2 Routing Overview . . . 8
2.2.1 Inter-domain Routing . . . 8
2.2.2 Intra-domain Routing . . . 9
2.3 Border Gateway Protocol (BGP) . . . 10
2.4 Internet Measurement . . . 11
2.4.1 Approaches to Data Collection . . . 12
2.4.2 Data Set . . . 13
2.5 Routing Policies . . . 13
2.5.1 AS Relationships . . . 13
2.5.2 Inference of AS Relationships . . . 14
2.5.3 AS Relationship Inference – Comparison . . . 15
2.6 Simulating Route Decisions with C-BGP . . . 16

3 Impact of Routing Parameters on Path Diversity and Optimality **17**
3.1 Overview . . . 17
3.2 Factorial design . . . 18
3.2.1 Analysis of Variance (ANOVA) . . . 18
3.2.2 Metrics . . . 19
3.2.3 Factors . . . 20
3.3 Simulation Setup and Choice of Levels . . . 20
3.3.1 Simulation . . . 21
3.3.2 Topology Generation / Choice of Levels . . . 22
3.4 Sensitivity Analysis . . . 24
3.4.1 ANOVA . . . 24
3.4.2 Impact of Individual Factors . . . 26
3.5 “Optimality” of Path Selection . . . 28
3.5.1 AS-Level Path Stretch . . . 28
3.5.2 Router-Level Path Stretch . . . 29
3.5.3 Geographical Path Stretch . . . 30
3.6 Related Work . . . 31
3.7 Summary . . . 32

4 Building an AS-Topology Model that Captures Route Diversity 33
4.1 Motivation and Overview . . . 33
4.2 Domains as Simple Nodes . . . 35
4.2.1 Data Set . . . 35
4.2.2 Route Diversity in the Internet . . . 36
4.2.3 Route Diversity in Single Router Models . . . 37
4.3 Methodology . . . 38
4.3.1 Components of the AS-Routing Model . . . 39
4.3.2 Evaluating Prediction . . . 39
4.3.3 Deriving an AS-Routing Model . . . 42
4.3.4 Example: Refining an AS-Routing Model . . . 42
4.3.5 Initial Model . . . 43
4.3.6 Iterative Refinement . . . 43
4.3.7 Using the AS-Routing Model for Predictions for other Prefixes . . . 47
4.4 Results . . . 48
4.5 Related Work . . . 51
4.6 Summary . . . 52

5 Searching for an Appropriate Granularity to Model Routing Policies 53
5.1 AS-Topology Model . . . 54
5.1.1 AS-Level Connectivity . . . 54
5.1.2 Building a Quasi-Router-Level Graph . . . 55
5.2 Bounds on Policy Granularity . . . 57
5.2.1 BGP Atoms . . . 57
5.2.2 Business Relationships . . . 58
5.3 Inferring Agnostic Per-Prefix Filters . . . 58
5.3.1 Inferring Filters . . . 59
5.3.2 Freedom in Filters Location . . . 62
5.3.3 Popularity of Filters . . . 64
5.4 Next-Hop Atoms . . . 65
5.5 Are AS Business Relationships the Right Choice? . . . 68
5.5.1 AS Relationships - No transit . . . 69
5.5.2 AS Relationship - Preferences . . . 69
5.5.3 Considering *No Transit*, Ignoring Preference . . . 70
5.6 Related Work . . . 71
5.7 Summary . . . 71

6 Trellis: A Platform for Building Flexible, Fast Virtual Networks on Commodity Hardware 73
6.1 Overview . . . 73
6.2 Trellis Requirements and Design . . . 74
6.3 Trellis Implementation . . . 76
6.3.1 Host Virtualization . . . 76
6.3.2 Link Virtualization . . . 77
6.3.3 Bridging . . . 78
6.4 Performance Evaluation . . . 79

6.4.1 Experimental Setup . . . 80
6.4.2 Forwarding Performance . . . 80
6.5 Summary . . . 83

7 HAIR: Hierarchical Architecture for Internet Routing 85
7.1 Overview . . . 85
7.1.1 Design Guidelines . . . 85
7.1.2 *HAIR*: Architecture . . . 86
7.2 Design Space and Related Work . . . 88
7.2.1 Problem Space . . . 88
7.2.2 Potential Solutions – Design Choices . . . 89
7.2.3 Current Activities . . . 92
7.3 Architecture . . . 92
7.3.1 Overview . . . 92
7.3.2 Packet Delivery . . . 95
7.3.3 Mapping System . . . 96
7.3.4 Dynamics . . . 97
7.3.5 Migration Path . . . 98
7.4 Evaluation . . . 99
7.4.1 Requirement Evaluation . . . 99
7.4.2 Estimation of Benefits . . . 101
7.4.3 Proof-of-Concept Implementation . . . 104
7.5 Summary . . . 105

8 Conclusions 107
8.1 Summary . . . 107
8.2 Directions for Future Work . . . 109
8.2.1 Inter-domain Routing Models . . . 109
8.2.2 Routing Architectures for the Future Internet . . . 109

List of Figures 113

List of Tables 115

List of Algorithms 117

Bibliography 119

Index 127

1 Introduction

"Chacun en a sa part et tous l'ont tout entier!"
"Each has his own share and all have a whole!"
(from poem "Les feuilles d'automne", Victor Hugo)

Although used in a different context, this citation of the French poet Victor Hugo summarizes the unprecedented success story of the Internet: Everyone can get access to the "network of networks" by connecting an end device such as a computer, laptop, or cell phone via wire or wireless means to other devices. In doing so, she/he becomes a part of the Internet, can reach all other devices in the Internet, is generally reachable for other nodes, and can even offer services to other users. In short, every user contributes to the Internet and everyone is making use of services provided by the Internet.

Evidently, the Internet has become an integral part of our everyday lives. Various statistics, e.g., [INT], suggest penetration rates of around 70% for Internet usage in European countries in 2008. Since its expansion into popular use in the 90s, the Internet has found its way into more and more sectors of public life and therefore has become a critical infrastructure. It is replacing TV, radio broadcasting, or newspapers as primary media devices. Aligned with this is a creeping takeover of the control about which information to distribute to which group of people. Companies rely on the Internet infrastructure to do business, e.g., by selling their goods in online shops or by ordering products from suppliers. The implications of the increasing importance of the Internet are not limited to the commercial side. Findings that the daily use of the Internet has implications for literacy, cognition, and learning [Ric] or the increasing popularity of online social networks clearly point out the sociological impact of today's online culture.

Most people take it for granted that the Internet offers its services without disruptions. Dave Clark, an American Internet pioneer, summarizes the steadfast faith in technology as follows:

"We reject kings, presidents, and voting. We believe in rough consensus and running code." (Dave Clark)

The technical arrangements which guarantee the reliability of the Internet and its services range from security mechanisms that provide integrity, confidentiality, and availability to protocols that ensure that packets can be delivered from host *A* to host *B* even in the case of network failures (e.g., fiber cuts or equipment outages). With respect to the latter task we generally distinguish between two important subtasks: *Routing* and *Forwarding*. If an Internet user types in the URL `http://www.net.t-labs.tu-berlin.de` into her/his web browser, then this name is first resolved to an IP address via DNS and the request is forwarded based on this address over multiple intermediate hops towards the destination. It is the responsibility of routing algorithms and protocols to determine the route or path from the sender to the receiver. Dynamic routing protocols can be seen as distributed algorithms. They ensure that all participating nodes, i.e., routers, in a packet-switched network know for every other destination the next hop to which they have to send

traffic such that the packets get finally delivered to the correct destination. Provided that there is topological redundancy, i.e., existence of alternate routes or paths, dynamic routing protocols can react to failures of links and nodes by potentially offering alternative paths. The remainder of this thesis focuses on routing.

Years after the initial development of the current routing protocols, we still do not fully understand which parameters have an impact on the routes chosen in today's Internet and how they affect the "optimality" with respect to certain metrics. Moreover, it is hard to predict the paths between a sender and receiver [MFM+06] or to verify the correctness of routing inside networks [FB05]. Network operators are struggling to optimize their routing, and the effectiveness of those efforts is limited.

The situation is largely due to the complexity of routing. The Internet is composed of a large number of independently administered Autonomous Systems (ASes), coupled by the Border Gateway Protocol (BGP) [RL06] into a single global spanning entity. The inter-domain routing protocol BGP does provide more than simple reachability. By offering mechanisms to implement routing policies, network operators can enforce their economic goals, e.g., preferring paths with high performance or avoiding routes that are economically expensive. These routing policies are decided *locally* by each AS but act *globally* across the entire system [GSW02]. Another reason for the complexity of Internet routing is the existence of different routing protocols. While BGP supports flexible routing policies to implement business objectives, routing protocols inside ASs (intra-domain) are designed to optimize network performance (e.g., link utilization, delay). The path followed by an IP packet, therefore, depends on the interplay between inter-domain (BGP) [RL06] and intra-domain routing protocols such as OSPF [Moy98] or IS-IS [Cal90]. The peering structure between ASs, i.e., the way how neighboring ASs exchange their reachability information, further increases the complexity of Internet routing. It is a rule rather than the exception that peering agreements require multiple physical links at multiple different locations in different regions.

The overall objective of this thesis is to better understand the complexity of today's inter-domain routing system. We start by (1) analyzing how sensitive routing optimality and diversity are to various factors such as policies or intra-domain topology. Based on the obtained insights, (2) we construct models of inter-domain routing with predictive capabilities regarding BGP paths. Unfortunately, it is unclear whether shortcomings of today's routing system such as super-linear routing table growth or high update churn can be resolved by incremental and/or band-aid solutions. In the light of recent interest in designing a new and better Internet using "clean-slate" approaches [Fel07], (3) we propose a platform for hosting virtual networks on commodity hardware that can be used for the evaluation of new protocols or architectures. Finally, we introduce a scalable routing architecture for the future Internet. Accordingly, this thesis presents the following results:

1. **Impact of routing parameters on path diversity and optimality**
   We perform a comprehensive study of how sensitive routing optimality and diversity metrics are to factors such as policies, AS size, topology, and IGP weights, etc. Surprisingly, we find that intra-domain factors only have marginal impact on global path properties – for example no setting of BGP parameters decreases the geographic path stretch significantly. In contrast, routing policies and AS size (in number of routers) are the dominating factors. Our findings reveal that it is hard to improve the global properties of route selection by existing means, i.e., tweaking BGP attributes, changing iBGP graphs, etc. This should come as no surprise, as inter-domain routing has been mainly designed to realize business objectives via routing policies, and not to propagate optimal paths.

2. **Modelling inter-domain routing**
   Models of inter-domain route selection are needed for quite a number of networking tasks, e.g., making decisions about peering relationships, choice of upstream providers, inter-domain traffic engineering. Until now, no model of the Internet has succeeded in producing predictions of acceptable accuracy. We demonstrate that the key to scalable and meaningful routing models is to incorporate intra- and inter-domain information at the minimum level-of-detail that is needed to explain the observed routing in the Internet. A solution to this is outlined by our findings on the granularity of routing policies as well as by the concept of quasi-routers that can serve as archetypes for reproducing the required intra-domain detail in future inter-domain models.

3. **Clean-slate approaches: Network testbeds and design of a scalable routing architecture**
   Researchers currently discuss major overhauls of the Internet's routing architecture given problems such as super-linear routing table growth, high update churn, insufficient support for multi-homing and traffic engineering etc. For this purpose, the research community is in need of controlled and realistic environments for evaluating new protocols, architectures, and services. We propose Trellis, a platform for hosting virtual networks on shared commodity hardware. *Trellis* allows each virtual network to define its own topology, control protocols, and forwarding tables, while amortizing costs by sharing the physical infrastructure. Moreover, we introduce *HAIR*, a scalable routing architecture for the future Internet. Our primary concern is to address the scalability problems, in particular to prevent updates from being globally visible, and to provide scalable and simple solutions for traffic engineering, mobility, multipath, etc. *HAIR* is based on the following ideas: (i) use of hierarchical routing, (ii) separation of locators from identifiers, and (iii) use of a hierarchical mapping system. It uses a hybrid "edge-based" approach to reconcile network- and host-based routing architectures and to enable migration.

Our contributions span a variety of research topics in the area of inter-domain routing. First, the results of our sensitivity study reveal the crucial factors for building scalable and meaningful models of Internet routing. Both the sensitivity analysis itself and the proposed simulation framework, are useful for evaluating and comparing new routing protocols and architectures [SCE+05, YCB07]. Second, our concepts and findings on how to capture path diversity [MFM+06] and how to model routing policies [MUF+07] are milestones on the way to an inter-domain routing model with predictive capabilities. The work carried out in this area provides an important step towards a model that allows prediction of AS paths under "what-if" scenarios, e.g., how to identify potential peering partners or upstream providers. Third, we also suggest solutions for problems that the Internet routing system is facing today. While *Trellis* provides means to evaluate new architectures on cheap commodity hardware, *HAIR* offers a long-term solution for the Internet of the future. Key in the design of *HAIR* is that we first comprehensively discuss and understand the design tradeoffs before we combine existing and novel ideas into an overall routing architecture.

1.1 Overview

The remainder of this document is structured in the following way:

Chapter 2 provides background information relevant for the remainder of this thesis: It gives a

basic introduction to the Internet and routing. In particular, we explain how routing information can be recorded in the Internet and how to simulate route decisions with C-BGP.

Chapter 3 presents a comprehensive study of how sensitive routing optimality and diversity metrics are to factors such as BGP routing policies, AS size, topologies, and IGP weights etc.

Chapter 4 describes how to capture route diversity within ASs and how to produce accurate predictions for unobserved AS paths. This is a crucial step towards developing structural models of the Internet that enable real applications.

Chapter 5 is mainly concerned with routing policies and searches for the appropriate granularity at which policies should be modeled such that the paths of a model are consistent with observable AS paths.

Chapter 6 presents our platform for hosting virtual networks on shared commodity hardware. It describes the design and implementation and evaluates the packet-forwarding rates relative to other virtualization technologies and native kernel forwarding performance.

Chapter 7 introduces *HAIR*, a scalable routing architecture for the future Internet. We analyze to what extent routing can be simplified if *HAIR* was deployed in today's Internet. Moreover, we demonstrate the feasibility of our approach by describing our experiences with a proof-of-concept implementation.

Chapter 8 provides an outlook and possible directions for future work.

2 Background

In Section 2.1 we give a short overview of the Internet, its structure, and its addressing scheme. Then, in Section 2.2, we explain how routing works and briefly review routing protocols in general, before we provide, in Section 2.3, more details on the Border Gateway Protocol (BGP). Section 2.4 is about how to collect BGP routing information from the Internet and about the data source that we use in later chapters of this thesis. Finally, we discuss routing policies with a focus on AS relationships and their inference (Section 2.5) and C-BGP, a simulator for computing BGP route choices (Section 2.6).

2.1 Internet Basics

2.1.1 Internet Structure

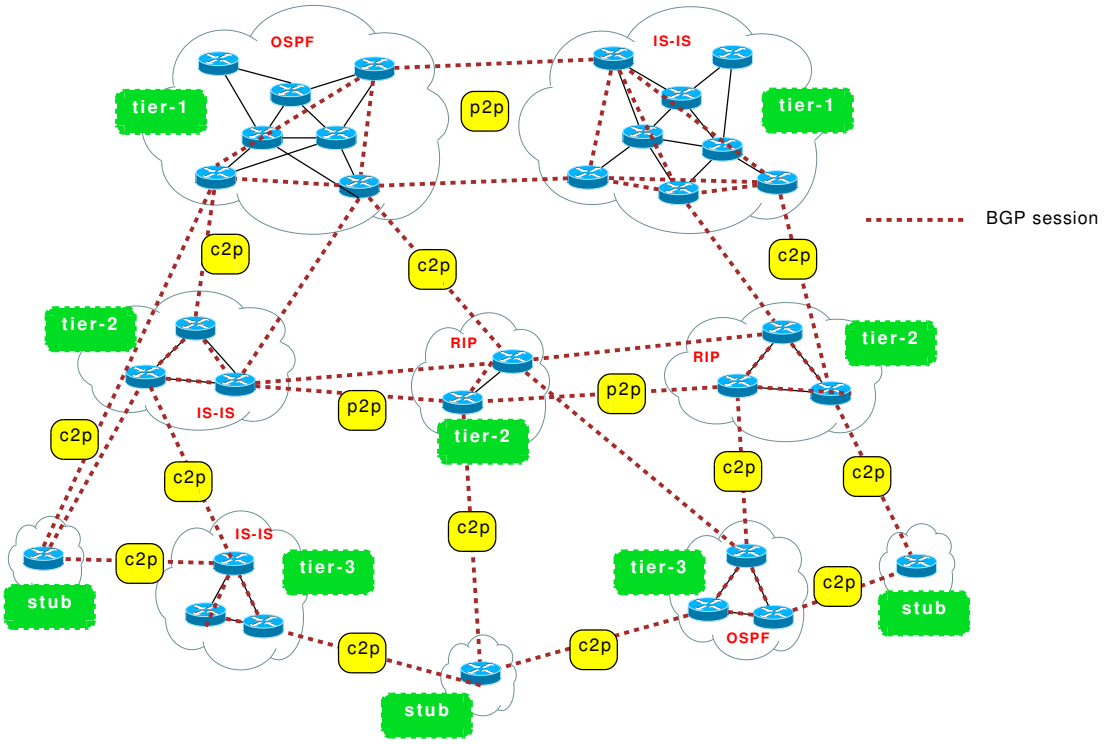
The Internet is composed of a large number of independently administered Internet domains, called Autonomous Systems (ASs). At the time of writing there are more than $30,000$ ASs, including large and small transit providers, enterprise and university networks, content and access providers, etc. *Peering*[1], i.e., the connection of two ASs via a network link, ensures global reachability. Due to the loose coupling of individual domains the Internet is frequently described as a "network of networks". Figure 2.1 illustrates the basic structure of the Internet.

In general, individual ASs are operated by single organizations or administrative authorities with their own economic objectives. With respect to the business relationship [WG03] between two peering ASs there are two main models. In the *customer-provider* (c2p) case, AS A is a customer of another AS B and A pays B money for obtaining transit through $B's$ network. To charge AS A, AS B can for example measure the amount of traffic that A sends over its *upstream* links to B. The second case is *peer-to-peer* (p2p) where neighboring ASs share the deployment and maintenance cost for the technical infrastructure that is needed to exchange traffic between the two ASs. Usually, two ASs are only willing to adopt a peer-to-peer scheme if the amount of traffic they exchange in both directions is balanced. Apart from customer-provider and peer-to-peer there exist also other less frequent business relationships. One example are *siblings* where neighboring ASs have a mutual transit agreement. Often the two ASs are merging ISPs or they adopt this scheme to obtain Internet connection backup.

Economic constraints have shaped the Internet and have led to its implicit hierarchical or "tiered" structure [SARK02], which reflects business relationships [WG03, BPP03, SARK02, MQWZ05]. A handful of so-called *tier-1* carriers, e.g., AT&T, Sprint, or Verizon, form the core of the Internet. In general, they maintain a world-wide network and have established a full mesh of p2p peerings with each other. While tier-1 ASs do not buy transit from any other AS, *tier-2* or *tier-3* ASs have by definition at least one upstream provider. Moreover, their network range may be more restricted.

[1] "Peering" is used in two contexts. It can denote the connection of two ASs and a routing policy where the costs for a connecting link between two ASs are shared.

Figure 2.1 A small Internet: tiered structure, routing policies, intra- and inter-domain routing, etc.



Finally, the majority of the observable ASs, currently more than 80% of the ASs, are *stubs*, which get their connectivity from transit or tier-1 providers and do not provide transit to any other AS. A large fraction of the stub ASs are content providers, universities, or enterprise networks.

The peering between two Internet domains can either be done via *private* or *public peering*. Since it is cheaper, networks other than tier-1 tend to rely on the latter and connect with each other at public *Internet Exchange Points* (IXP). The IXPs, e.g., DE-CIX [Dec] in Frankfurt, provide technical infrastructure, frequently a layer-2 switching fabric or cloud, which ASs can use to establish physical connectivity with all ASs, participating at the IXP. However, it is up to the individual participants whether to actually exchange traffic or not. Unlike IXPs, private peering is implemented with a dedicated layer-2 connection between the routers of the peering ASs. The costs are generally shared between the two ASs.

Since ASs are reluctant to share information about their networks, obtaining an accurate picture of the Internet topology is simply impossible. Maps of the Internet derived from collected BGP data (see Section 2.4) can only provide an incomplete view. Researchers trying to find the topology study the Internet from three different perspectives or at three levels of granularity: AS-level, PoP-level, and router-level:

- The *AS-level graph* represents the high-level structure among the different networks that compose the Internet. Each business agreement between two ASs corresponds to an AS-level edge and implies some physical connectivity between the ASs. Such a graph with ASs as nodes and an edge if there is a peering agreement between two ASs is obviously a coarse-grained model of reality. After all, it is known that peering agreements usually enforce multiple physical

links at multiple different locations in different regions. This has the advantage of minimizing the time a packet stays within an AS as long as hot-potato routing is used. Multiple physical links at different locations are also often used to increase reliability, e.g., in the form of customer backup links. Nonetheless, AS-level graphs are widely used by researchers since a view on the actual Internet AS-level map can be obtained easily if one has access to BGP routing tables. However, only a small number of observation points, i.e., BGP routing tables, are publicly available. More information on how to collect BGP routing information is presented in Section 2.4.

- The next finer level is the *PoP-level*. A Point of Presence (PoP) corresponds to a physical location where a network domain houses a collection of routers. Typically, a PoP consists of a few backbone routers in a densely connected mesh and some access routers that are connected to routers in neighboring ASs and to at least two backbone routers.

- The finest granularity is a *router-level graph*. Here, every node represents a router and every link indicates that the interfaces of two routers can reach each other on layer 2. Unfortunately, inferring router-level maps is challenging since it requires to use active measurement techniques (see Section 2.4) from a large set of vantage points and to map interface IP addresses, returned for example by `traceroute`, to actual routers. Any techniques [SMW02, MSWA02, SBS08] for inferring router-level maps provide an inherently limited view of the Internet due to the limited number of vantage points that are publicly accessible.

2.1.2 Packet Delivery

Internet users take it for granted that whenever they use the Web, e.g., by typing a URL into a web browser window, packets are automatically delivered to the appropriate destination. However, ensuring correct packet delivery is a challenging task. It requires to *identify* the endpoints of a communication session and to *locate* the destination node. The Internet Protocol (IP) [Pro81] assigns both of these tasks to the IP address.

Currently, two flavors of IP coexist, namely IP version 4 (IPv4) and IP version 6 (IPv6). Depending on whether IPv4 or IPv6 is used, IP addresses are 32 bit and 128 bit long, respectively. IP is often referred to as the waist of the hourglass since all hosts that form the Internet can understand this protocol. Roughly speaking, we can say that IP provides functionality similar to that of an envelope for snail mail. The envelope (IP header) is put around the actual letter (user data, payload) and contains amongst other the address of the originator and the receiver (IP address). Accordingly, the IP address ambiguously *identifies* the recipient of a packet. For example, the web server of our group has the IPv4 address `130.149.220.251` which should not be used by any other host in the Internet.

In addition, IP addresses serve as *locators*. While postal addresses are made up of multiple subparts such as name, street, or ZIP code and therefore indicate where a recipient is located, it is not so obvious for IP addresses how the locator information is encoded. The strategy here is to assign IP addresses in a way such that hosts or routers in the same network obtain addresses from the same IP address or IP *prefix* range. To this end, IP prefixes can then be seen as bit vectors: Two hosts are within the same network if the first x bits of their IP addresses are identical. Hence, the value of x reflects the length of the prefix. A common notation for prefixes is to append the prefix length x as $/x$ to the IP address, e.g., `130.149.220.0/34`.

In a packet-switched network each node has to know for every possible destination the next hop to which it has to send traffic such that packets are finally delivered to the correct destination. Evidently, maintaining such state for billions of end hosts is impossible. However, relying on prefixes, IP addresses can be organized hierarchically. A *route* essentially consists of two parts: (i) an IP prefix that indicates which packets are described by this route and (ii) a network interface/neighboring router to which such packets are to be forwarded. Adopting such a scheme, any router of the Internet only maintains state for complete networks in its *routing table* and not for individual hosts.

The organization of IP addresses requires some central coordination, which ensures that different networks use non-overlapping IP prefixes. To this end, IP addresses are allocated by the Internet Assigned Numbers Authority (IANA) from pools of unused address space and delegated to the appropriate Regional Internet Registries (RIR), e.g., AfriNIC, ARIN, APNIC, LACNIC, RIPE NCC, or National Internet Registries (NIR). Internet Service Provider (ISPs) can then request IP address blocks from there [HKC+00]. Similarly, the registries assign Autonomous System Numbers (ASNs) to ASs. These numbers uniquely identify an administrative domain and are used within the Border Gateway Protocol (BGP), see Section 2.3, for exchanging reachability information between two peering ASs.

2.2 Routing Overview

Routing is the task to establish state at routers or hosts in order to enable the delivery of packets to their destination networks or hosts. In principle, it is possible to manually configure *static routes* that contain the next hop for each destination in the network. Since such an approach is error-prone and does not redirect traffic to an alternate route in case of network failures, larger networks generally rely on *dynamic routing*: Each router in the network runs a routing protocol and exchanges reachability information with directly connected neighbors. In doing so, routers are supposed to converge to a state where they have a stable view of the network and know for every destination a next hop.

A plethora of routing protocols has been developed and is currently used, including IS-IS [Cal90], OSPF [Moy98], and BGP [RL06]. There are routers, e.g., at the border of ASs, that run multiple routing protocols at the same time, keeping a separate *routing table* for each. Potentially conflicting information is resolved and the result is inserted into the *forwarding table* or *general routing table*. Overall, the results of route computation depend on routing between ASs (*inter-domain*) and routing inside individual domains (*intra-domain*). While intra-domain routing protocols are mainly designed to optimize network performance, e.g., link utilization, inter-domain protocols need to provide means for routing policies so that business objectives can be realized. In the following, we briefly review routing protocols in both categories.

2.2.1 Inter-domain Routing

The Border Gateway Protocol (BGP) [RL06] is the de facto standard inter-domain or exterior gateway routing protocol. Whenever two ASs peer with each other, they establish at least one BGP session between two of their border routers to exchange reachability information. For more implementation details on BGP we refer to Section 2.3.

In contrast to intra-domain routing protocols, BGP is a *policy-based* routing protocol. We point out that BGP itself does not define policies. Rather it only provides mechanisms that can be used to

implement routing policies. The need for routing policies stems from the economic structure of the Internet, see Section 2.1.1: Some ASs earn money by selling connectivity to their *customers* while at the same time they may be customers of other *provider* networks. Given such economic conditions ASs sometimes only propagate a limited set of routes to their neighbors, and not the complete set of routes, they themselves have learned from their neighbors. Evidently, this can cause traffic to follow sub-optimal routes, i.e., routes not shortest in terms of hops. Moreover, ASs may prefer certain routes over others if sending traffic over a certain peering is cheaper. To achieve this, BGP provides mechanisms to tweak and manipulate the process of route selection and route propagation to neighboring ASs [WG03]. Section 2.5 is dedicated to such routing policies.

2.2.2 Intra-domain Routing

ASs are not atomic entities but are generally composed of multiple routers [SMW02, SBS08]. ISPs employ an intra-domain routing protocol or *interior gateway protocol* (IGP) to select paths through an AS [MSWA02]. Unlike BGP, intra-domain routing protocols are designed to make optimal use of network resources such as routers and links. Frequently, it is possible to configure weights for individual network links, which reflect for example its capacity, delay, etc. Then, the IGP prefers those routes where the sum of link weights is small. Hence, the internal structure of each AS together with its choice of IGP weights impacts the inter-domain decision. However, intra-domain routing protocols generally do not have built-in support for flexible routing policies as is the case of BGP.

Overall, interior gateway protocols can be classified as either *distance vector* or *link state*. In the former case, physically adjacent neighbors exchange only the following information: Which prefixes are reachable and which costs are associated with that prefix. Provided that there are no topology changes or link/node failures, distance vector protocols converge to a stable state. Since the implementation of a distance vector scheme is relatively straightforward, distance vector routing protocols are easy to deploy. However, problems such as count-to-infinity may induce long convergence times after link failures and can degrade the performance and scalability. The Routing Information Protocol (RIP) [Hed88] is the most well-known example of distance vector protocols.

In contrast to distance vector protocols, link state routing protocols assume that every router has a global view of the complete network topology, including all links and routers. Then, it can compute the shortest paths to each destination by solving the single-source shortest path problem using Dijkstra's [Dij59] algorithm[2]. Obtaining a complete view of the network at every router can be achieved by flooding topology information through the network. A widely used implementation of this is the *Intermediate System to Intermediate System* (*IS-IS*) [Cal90] routing protocol. It requires each router to have a global view of the topology.The *Open Shortest Path First* (OSPF) [Moy98] has the reputation of being more complex, but in contrast to IS-IS it allows to partition a network into different areas. Then, it is sufficient to acquire complete information only about the area where a router is located.

Finally, we point out that it is often necessary to use BGP inside ASs. Some routers of a network have external BGP sessions (eBGP) with peering ASs. Reachability information obtained at BGP-speaking routers needs to be redistributed to other BGP routers located in the same AS. This is done via internal BGP (iBGP) sessions, which is discussed in Section 2.3.

[2]If non-negative weights are configured for links, it can compute the shortest paths with respect to the assigned link weights.

2.3 Border Gateway Protocol (BGP)

BGP, the de facto standard inter-domain routing protocol, unites elements of the distance vector and of the link state approach. Sometimes it is referred to as a *path vector* routing protocol. Although BGP routers do not have a global view of the topology, route advertisements from neighboring routers contain an AS path towards the destination, namely the sequence of ASs that packets need to traverse to reach the destination. Insofar, BGP routers know more about the topology than in a pure distance vector scheme.

BGP sessions between two neighboring routers are established on TCP port 179. Through its BGP sessions each router receives and propagates BGP routes for destination prefixes. At the beginning of a session all routes are exchanged while afterwards routers only send *incremental updates*. In BGP routes have associated *attributes*, which can be used to rank and therefore to determine the best route in cases where multiple paths exist to a destination. Essentially, these route attributes are the main knob to manipulate route propagation and selection via policies. The BGP protocol defines four types of messages:

OPEN is sent at the beginning of a BGP session

KEEPALIVE is sent either to confirm an OPEN message or to avoid that the connection breaks during periods of inactivity, i.e., when no updates are sent for a longer time period.

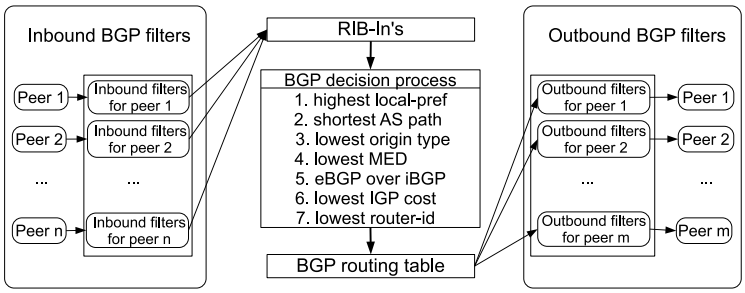
NOTIFICATION is used when an error condition is detected and causes the BGP sessions to be closed immediately.

UPDATES are sent to exchange routing information. They can advertize feasible routes (*Network Layer Reachability Information* field) to neighbors or withdraw unfeasible routes (*Withdrawn Routes* field) that have been announced previously. A BGP packet contains routing information for multiple prefixes.

A BGP router processes and generates route advertisements as shown in Figure 2.2. Administrators specify input filters on a per BGP peer basis, which are used to discard unacceptable incoming BGP advertisements. Once a route advertisement is accepted by the input filter, it is placed together with the routes originated at this router in the incoming Routing Information Base (RIB-In) of the peer, possibly after some of the route attributes have been modified according to the local routing policies. Next, the BGP decision process is used to select the *best route* for each prefix from among the available routes. This route is then placed into the BGP routing table, which we refer to as the *RIB-Out*. Finally, administrators may specify output filters for each peer, which are used to decide which best routes to propagate to a BGP neighbor. By applying input or output filters, network operators can implement routing policies and business objectives.

The *BGP decision process* consists of a sequence of elimination steps. Its final goal is to select a single best route for any given prefix. For this purpose, the BGP decision process considers the BGP routes attributes. One of the first attributes is *local-preference* (in short, *local-pref*). As local-pref is a non-transitive attribute, it can be used to locally rank routes. The next BGP attribute examined by the BGP decision process is the *AS path*. An *AS path* contains the sequence of ASs that a route crossed to reach the current AS. Routes with shorter AS paths are preferred. Next in the evaluation process is the *multi-exit-discriminator* (in short, *MED*). This attribute is used to rank routes received from the same neighbor AS but it can also be used across neighbors. Then the decision process

Figure 2.2 Layout of a BGP router, BGP decision process.



ranks routes according to the IGP cost of the intra-domain path towards the *next-hop*, preferring routes with smaller IGP cost. This rule implements hot-potato routing [TGVS04]. Finally, if there is still more than a single route left, the router breaks ties, instead of selecting the route to the neighbor that has the lowest router-id (typically one of its IP addresses).

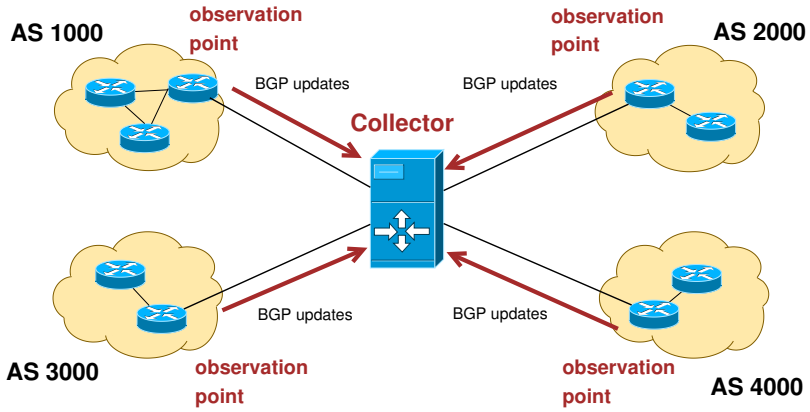
BGP comes in two flavors. External BGP (eBGP) sessions are established over inter-domain links, i.e., links between two different ASs (BGP peers), while internal BGP (iBGP) sessions are established between the routers within an AS. External BGP sessions are not set up between all routers of an AS but only on the so-called border routers. To redistribute routes learned from eBGP peers to all BGP routers within an AS, iBGP sessions are established between all BGP routers on top of the router-level topology. There are multiple ways to realize such iBGP structure, ranging from an iBGP full mesh to route-reflectors [BCC06] or confederations [TMS01]. The latter two are more scalable but reduce the visibility of routes within the AS [SMA03] compared to a full mesh.

Being in use for more than a decade now, BGP has gained the reputation of a notoriously complex protocol with high router CPU requirements. During the last decade BGP has been extended with mechanisms to reduce update rates. For example, the *Minimum Route Advertisement Interval* (MRAI) [RL06] is used to restrict the rate at which updates are propagated to neighboring routers: Before sending an update to a certain peer, a BGP router waits some time (e.g., 30s), collects more incoming updates, and finally only announces the best path. Another mechanism is *Route Flap Dampening* [VCG98], which suppresses flapping routes, i.e., routes that are continuously withdrawn and re-advertized, for a certain time period. We point out that BGP does not guarantee that routing converges to a stable state [GW99, GSW02, GSW99, GW02]. This is due to the fact that individual ASs have freedom in how and what routing policy to instantiate. Yet, business contracts between peering agreements generally prevent network operators from applying evil routing policies that potentially endanger the overall routing stability of the Internet. Even so, it usually takes quite some time until a newly announced prefix is reachable from everywhere in the Internet.

2.4 Internet Measurement

Now we describe how to obtain topology and routing data about the actual Internet. In Section 2.4.1 we give an overview of existing approaches for collecting such data and how to infer the topology of the Internet. Afterwards, Section 2.4.2 presents a data set that we frequently use throughout the remainder of this thesis.

Figure 2.3 BGP route collection.



2.4.1 Approaches to Data Collection

Obtaining an accurate and complete map of the Internet is impossible since network operators are generally reluctant to share information about their networks and their peerings with other networks. In the past, efforts have been made to infer Internet topologies on the AS-level (e.g., [SFFF03, OPW+08, Cai]), PoP-level [SMW02, SS05], or router-level [SBS08, SMW02, MSWA02]. Common to all these approaches is that they rely on a set of vantage or observation points, which record topology or routing data. Hence, any inferred topology only captures what can be observed from a limited number of observation points. Since the focus of this thesis is on inter-domain routing, we are not primarily interested in collecting router-level maps of the Internet but rather need BGP routing data.

There are many different techniques for obtaining inter-domain routing data, most notably *active* and *passive* measurement approaches. Active measurement generally relies on `traceroutes` from a set of vantage points to a set of destination prefixes in order to determine the router-level path that packets follow to reach a destination. The completeness of the obtained router-level maps is limited by the number of traceroutes, by the vantage points, by the diversity of paths they use to reach destinations, and by firewalls rules or routers that may filter or rate-limit ICMP messages. Moreover, it is challenging to map interface IP addresses into actual routers [SBS08, SMW02] if one wants to obtain an AS-level map from the router-level view that traceroute provides. The Skitter [BC01b] and the DIMES [SS05] project are examples that rely on active measurements.

One of the most common techniques for a passive measurement approach is to rely on a dedicated workstation (*collector*) running a software router that peers with "productive" BGP routers inside ASs, see Figure 2.3. We refer to each peering session from which we can gather BGP data as an *observation point*, and the AS to which we peer as the *observation AS*. Such an approach is passive since the collector does not announce any BGP updates but only collects route advertizements received from its observation points. The BGP routing table that is computed by the collector based on the received BGP updates provides a view of the Internet topology: BGP routing table entries contain for each destination prefix an AS path, i.e., the sequence of ASs that needs to be traversed to reach the destination. Generally, we can assume that there is a peering between two ASs if these

ASs are next to each other on an AS path. Again, the AS-level map of the Internet that we obtain with such an approach is going to be incomplete, as there exist observation points for only a limited number of ASs. Furthermore, some observation points apply routing policies before exporting their routes to a route collector and thus only present a partial view on their routing table to the collector. Throughout this thesis, we will mainly rely on passive measurements and use routing information from public data archives such as RIPE [RIP] and RouteViews [ROU].

2.4.2 Data Set

We now give some detail about a data set that we use frequently for our analyses. It contains BGP data from more than 1,300 BGP observation points, including those provided by RIPE NCC [RIP], RouteViews [ROU], GEANT [ID], and Abilene [Abi]. The observation points are connected to more than 700 ASs, and in 30% of these ASs we have feeds from multiple different locations.

As we are not interested in the dynamics of BGP, we use a static view of the routes at a particular point in time. The table dumps provided by the route monitors are each taken at slightly different times. We use the information provided in these dumps regarding when a route was learned to extract those routes that were valid table entries on Sun, Nov., 13, 2005, at 7:30am UTC, and that were stable in the sense that they have not changed for at least one hour.

Our data set contains routes with 4,730,222 different AS paths[3] between 3,271,351 different AS pairs. We derive an AS-level topology from the AS paths. If two ASs are next to each other on a path, we assume that they have an agreement to exchange data and are therefore neighbors in the AS-topology graph. We are able to identify 58,903 such edges.

Note, our data does not cover the complete AS topology [ZLMZ05] since not all AS relationships are observable in our data. There are relatively more observation points in the level-1 and level-2 ASs than in the other ASs [OZZ07]. Therefore, it is likely that AS-relationships involving level-2 providers are missing. Yet, their impact with regards to routing can be expected to be less significant [ZZM+07, BMRUar].

2.5 Routing Policies

Mechanisms offered by BGP to implement routing policies allow network operators to enforce their economic goals, e.g., preferring paths with high performance or avoiding routes that are economically expensive. *AS relationships* are the most wide-spread of all policy models that have been suggested in the past. We describe this policy model (Section 2.5.1), give an overview of techniques to infer AS relationships (Section 2.5.2). Finally, we present a study that compares different inference approaches (Section 2.5.3).

2.5.1 AS Relationships

The AS relationship policy model assumes that there is a unique business contract negotiated between any two ASs that are physically connected via one or multiple peering links. The business relationship between two peering ASs is implemented via routing policies that define which paths are selected for traffic forwarding and which paths are exported to neighbors [WG03]. Although AS

[3] We removed AS path prepending to prevent distraction from the task of route propagation.

relationships are the de facto standard model for routing policies, one should be aware that the reality of routing policies [WG03] and peering relationships can be far more complex [CR05, CGJ+04].

While routing policies can be rather specialized [CR05], AS relationships classify any directed link of the AS topology into one of the following types: peer-to-peer (p2p), customer-provider (c2p) or sibling (sib). While in a c2p relationship the customer pays the provider to obtain transit through the provider's network, p2p assumes that two peering ASs share the deployment and maintenance cost for the connecting link. Siblings are peering ASs that have a mutual transit agreement, i.e., merging ISPs.

The different types of AS relationships lead to constraints on path selection and propagation: *preference* of certain routes and the *valley-free property*. First, network administrators may favor longer AS paths over shorter ones due to economic reasons. In general, routes learned from customers will be preferred over routes announced over peering links and peering routes will be favored over provider routes. Second, a common assumption is that the valley-free property [WG03] holds. In particular multi-homed stub ASs want to avoid being used as a transit. For this reason, routes learned from provider and peering neighbors are not propagated to other provider or peering ASs.

2.5.2 Inference of AS Relationships

In the literature, numerous techniques have been proposed for inferring AS relationships given a set of observed AS-level paths. To obtain such AS paths, one can for example rely on passive measurement approaches, see Section 2.4.1. The goal is to assign an AS relationship (c2p, p2p or sib) to each individual link of the AS topology.

The so-called MaxTOR problem [SARK02] tries to find an assignment that maximizes the number of valley-free paths: AS-level paths that have been given as input to the algorithm should for example not contain two consecutive edges that are inferred as provider-customer and customer-provider. MaxTOR is a NP-complete problem that was tackled in the past by many heuristics. Moreover, as reported in [BEH+07], if a solution to MaxTOR with n AS relationships exists (optimal or not), then there are at least 3^n different solutions leading to the same number of valley-free paths. This finding suggests that one has to take inferred AS relationships with a pinch of salt.

In the following, we briefly summarize some of the algorithms that have been proposed to solve the MaxTOR problem:

- **gao[Gao00]:** This algorithm assumes that a provider typically has a larger size than a customer and that the size of an AS is proportional to its degree in the AS graph. Based on this assumption it identifies the "top provider" of an AS path and then tries to classify other ASs by considering the valley-free property. When valleys occur in AS paths, some conflicting relationships are supposed to support a mutual-transit relationship (label "sib").

- **sark[SARK02]:** This algorithm uses topology leaf-pruning as seen from each observation point to infer per-vantage-point AS rankings. Then a relationship for each link is inferred. When ranking of ASs is not decisive enough, some links are labeled with the "Unknown" relation.

- **csp[MM06]:** This approach takes advantage of a Constraint Satisfaction framework. A Max2CSP problem is derived from MaxTOR where each relation is a variable and each subpath of length 2 (AS triples) introduces a constraint between two relations. A tabu-search

Table 2.1 AS relationship inference: evaluation of various algorithms.

	relations			**valley-free**		**match**	
	p2p	*c2p*	*sib/* UNK	triples	paths	*CAIDA*	*Tier-1 match*
sark heuristic							
1	25688	32703	520	81.5	27.3	54.8	84.2
2	13786	34006	370	84.6	29.9	66.3	82.7
3	15630	31188	352	85.5	32.8	61.3	85.7
gao heuristic							
1	12971	44252	1688	88.3	100.0	92.6	65.4
2	5200	41453	1509	90.6	100.0	93.5	66.9
3	5361	40333	1476	90.5	100.0	94.2	69.2
caida heuristic							
1	3367	38128	229	70.5	96.1	100.0	80.0
2	3367	38128	229	73.0	96.3	100.0	80.0
3	3367	38128	229	85.4	97.4	100.0	80.0
csp heuristic							
1	18326	40585	0	99.9	99.3	95.0	94.7
2	9219	38943	0	99.9	99.3	95.5	94.0
3	8050	39120	0	99.9	99.3	94.5	94.7

algorithm runs on a restricted space of feasible solutions (unlikely relations and customer cycles are forbidden).

- **caida[DKF+07]:** Another recent algorithm claims to find more realistic solutions with a partial validation of the results. The objective function is modified to incorporate information on the degree of ASs. This mathematical program is solved by using Semi-Definite Programming and by a post-processing heuristic that tries to maximize the number of peering links.

2.5.3 AS Relationship Inference – Comparison

We now compare the four algorithms *gao*, *sark*, *csp* and *caida*, see Section 2.5.2. As input for the algorithms, we use the data set described in Section 2.4.2. Recall, that the resulting topology contains more than 4 million distinct AS paths, resulting in more than $20,000$ vertices and almost $60,000$ AS-level edges. We evaluate the number of valley-free paths and the number of valley-free AS triples for each of the four inference algorithms[4]. For each solution we report the number of inferred p2p and c2p links as well as the number of *Unknown* or sib links in Table 2.1. To obtain an understanding of the correctness of the solutions we validate the inference with two indicators: *Caida-match* and *Tier-1-match*. *Caida-match* is the percentage of relationships that are inferred as being of the same type by both the considered heuristic on our data and the downloaded *caida* solution. *Tier-1-match* is the percentage of relationships that are of the same type by both the considered heuristic on our data and those we know to be the real relationships for a tier-1 in November 2005.

According to Table 2.1 we find that the *sark* algorithm produces a small number of valley-free paths. However, the solutions match well the relationships from the tier-1 (about 80%). The *gao*

[4]Note that we were not able to run the *caida* algorithm on our data and therefore downloaded an existing solution from CAIDA for November 7th, 2005 (solution only based on traditional RIPE and Route-Views data).

heuristic has 100% of valley-free paths. This can be explained by the fact that the *gao* approach unrealistically identifies a large number of siblings and sets relationships to *mutual-transit*. In doing so, it cancels valleys next to these type of conflicting links. Note that the *gao* solutions match well the relationships of tier-1s (about 65%). The *caida* solution produces a large number of valley-free paths (more than 96% of our paths are valley-free) and a good match with the relationships of the tier-1 (80%). Finally, the *csp* heuristic generates solutions with the largest number of valley-free paths (up to 99%) and the best match with the relationships of our tier-1 (about 94%).

2.6 Simulating Route Decisions with C-BGP

Both for our study of the impact of routing parameters on path diversity and optimality in Chapter 3 and for the modeling work on inter-domain route selection presented in Chapter 4 and Chapter 5, we need a tool that allows us to compute the selected BGP routes for all routers in a given topology and policy configuration. For this purpose, we rely on C-BGP [QU05], an open-source simulator developed by Bruno Quoitin from Université Catholique de Louvain, Belgium.

C-BGP is designed to study the propagation of routing information along a topology model that consists of multiple ASs. It allows multiple routers within an AS, the setup of BGP sessions between any pair of routers, supports flat and hierarchical iBGP as well as eBGP, and computes shortest paths inside ASs using Dijkstra's algorithm. To propagate routing information, C-BGP models the propagation of BGP messages and reproduces the route selection performed by each router [HP01] based on routing policies.

Since C-BGP only computes the steady-state choice of BGP routers after the exchange of the BGP messages has converged and not the whole state machine of the BGP routing protocol, it is possible to perform large-scale simulations with thousands of routers, complex routing policies, intra-domain and inter-domain topologies in a reasonable amount of time. However, C-BGP does not model TCP connections, BGP timers such as the MRAI, or packet exchanges between simulated routers. Instead, all generated BGP messages are pushed into a global FIFO queue, which guarantees that BGP messages are received in sequence. For more details on C-BGP and its usage we refer the reader to [QU05] or to the C-BGP web page at `http://cbgp.info.ucl.ac.be`.

3 Impact of Routing Parameters on Path Diversity and Optimality

Routing in the Internet is inherently complex. It is controlled by diverse policies, decided *locally* by each AS, but acting *globally* across the entire system [GSW02]. Furthermore, it depends on protocols for routing between and within individual ASs, on the router-level topology inside Internet domains, and on the peering structure between ASs.

Years after the initial development of the current routing protocols we still lack an understanding of the impact of various parameters on the routes chosen in today's Internet. Network operators are struggling to optimize their routing, and the effectiveness of those efforts ranges from pure marketing to simply unknown.

In this chapter, we attempt an initial step towards a better understanding of inter-domain routing. We study inter-domain route selection and propagation from a global perspective: How sensitive are routing optimality and diversity metrics to factors such as policies, number of routers per AS, IGP weights, location of peerings, iBGP connectivity, etc?

3.1 Overview

To comprehensively explore all factor settings and their consequences on the routes computed by BGP, we rely on *full factorial design*. One major contribution of this chapter is to quantify sensitivity using *analysis of variance* (ANOVA) [Jai91].

Given the inherent limitations of observable routing data we rely on simulations. Within simulations we can control all parameters and can compute any desired metric on the simulated routes since we created the topology and therefore know the routing tables of each individual router. Simulations allow us to compare paths chosen by individual routers with paths that are globally optimal in terms of AS-level, router-level hops, or geographical distance. To the best of our knowledge there is no simulation work which is comparable to ours in terms of comprehensiveness, level-of-detail in modeling, and the size of the used topologies.

The benefits of our sensitivity analysis are twofold: First, they give insight into the relevant factors, i.e., the ones that need to be accurately reproduced when modeling routing in the Internet – is it really true that state-of-the art routing models neglect aspects that do have an influence on route computation? Second, our comprehensive sensitivity study can provide hints on how to improve the path selection of BGP.

Surprisingly, we find that the impact of intra-domain parameters, including IGP weights or iBGP connectivity, on *global* route propagation is low compared to other parameters. Note that this does not imply that individual routes chosen by BGP are insensitive to hot-potato routing [TSGR04, TGVS04]. Consistent with the design of BGP we find that routing policies and the size of ASs in terms of the number of routers are the dominating factors. Hence, future routing models should focus on these two aspects.

Moreover, we find that the majority of routes incur reasonable stretch in terms of AS-level/router-level hops or geographic distance. Policies slightly increase path length. Overall we did not manage to optimize geographical distance within the space of current configuration alternatives in our simulations. However, BGP and in particular BGP policies, cannot be blamed for the sub-optimality of the current paths. BGP is a policy-routing protocol for which optimality has not been a design constraint. We will face this limitation of BGP if end-to-end path quality is to become more important in the future.

The structure of this chapter is as follows. We provide our experimental design, Section 3.2, before we present implementation details, Section 3.3. Section 3.4 presents our sensitivity analysis, followed by the study of (sub)optimality of selected routes in Section 3.5. Finally, we discuss related work, see Section 3.6 and conclude in Section 3.7.

3.2 Factorial design

A factorial design [Jai91] experiment allows to study the effect of each *parameter* (factor) on the evaluation *metrics* (response variables) as well as the effects of interactions between factors on response variables. For each factor there is a set of discrete possible values or *levels*. A *full factorial experiment* explores all combinations of these levels across all factors. Analysis of Variance (ANOVA) provides statistic means to describe the impact of individual factors on the metrics. We now provide a short introduction to ANOVA and describe the metrics and factors for our sensitivity study.

3.2.1 Analysis of Variance (ANOVA)

ANOVA is an analysis of the variation present in an experiment. It helps to determine which factors strongly contribute to the metric variations observed across the different level settings. The total *sum of squares* SS_{total} measures the total absolute variability across *all* the obtained metric values x_i in a sample (where $\bar{x}$ denotes the average metric value):

$$SS_{total} = \sum (x_i - \bar{x}) = \frac{\sum x_i^2 - (\sum x_i)^2}{n}.$$

The basic idea behind ANOVA is to partition this variability of the whole data set into two components: Variability across different levels of a factor $SS_{factors}$ and variability for different repetitions of the same level configuration $SS_{repetitions}$. This can be expressed as follows:

$$SS_{total} = SS_{factors} + SS_{repetitions}$$

To estimate the impact of a certain factor on the response variable, we can compute the percentage of variability that a certain factor explains by dividing $SS_{factors}$ by SS_{total} (used in Section 3.4). Note that ANOVA is not restricted to experiments with two factors. For more details on the computation of the sum of squares, the partitioning of the variation, the regression models etc., we refer to Jain's book [Jai91].

ANOVA provides means to determine whether the fraction of variance that is explained by a certain factor is *statistically significant*. For this purpose, we will compute in Section 3.4 the F-ratios: the variation due to a factor divided by the variation due to experimental error or noise. The null hypothesis is that this ratio equals 1.0 and is rejected if the F-ratio is significantly large enough,

i.e., the likelihood that it is equal to 1.0 is smaller than a certain percentage (e.g., 5%). Again, we refer the reader to Jain's book [Jai91].

3.2.2 Metrics

We concentrate on metrics that quantify the results of route computation, optimality, and diversity, rather than the dynamic properties of BGP such as convergence time. The metrics fall into two categories: *optimality* to capture the length of the selected routes or *diversity* to capture the number of alternative paths.

AS path length (*ASLength*) reflects the number of AS-level hops of an AS path that is selected by a router.

Router-level path length (*RouterLength*): Rather than AS-level hops, this metric considers the number of router-level hops of a selected route.

Geographic path length (*GeoLength*): Our topology models include geographic coordinates. Therefore, we can compute the geographical distance for each path by summing up the distances of each link on the router-level path.

Number of selected paths (*NumSelPaths*) counts the total number of distinct AS paths that are selected as best routes[1] by the routers of an AS for a single prefix.

Number of learned paths (*NumLearnPaths*): The number of distinct AS paths learned by an AS. Rather than looking at the selected best paths, we count *all* AS paths learned by a router for any given prefix.

Ratio of learned paths by number of neighboring ASs (*LearnNeighRatio*): To measure the average route diversity learned from a neighboring AS, we divide *NumLearnPaths* by the number of AS-level neighbors of that AS.

Disjointness (*Disjointness*): For every AS, we consider all distinct AS paths, learned by its routers and compute their edge disjointness. For this we compute for each pair of AS paths p_1 and p_2:

$$m = 1 - \frac{\text{\# AS edges } p_1 \text{ and } p_2 \text{ have in common}}{\text{\# AS edges in longer path}}$$

We refer to the average of m over all these pairs of AS paths as *Disjointness*. A value of 0 implies that all learned AS paths are identical while a value close to 1 indicates a high edge disjointness. If an AS learns a single AS path, we set the metric to 1. This metric is similar to the *Novelty* metric used by Motiwala et al. [MEFV08] to characterize AS path disjointness.

[1] Each BGP router selects only one best route for each prefix.

3.2.3 Factors

We base our simulations on topologies that consist of multiple ASs, each with their own router-level topology. These router-level topologies include a location within a point of presence (PoP) for each router throughout the world. Hereby, we keep the design choices available to network operators [LAWD04] in mind.

Routing policies (*Pol*) are important for the operation of today's Internet and one of our factors. We identify two levels: no routing policy and routing policies according to AS relationships. This allows to study the degree of AS path inflation due to the use of routing policies [TGSE01, SMA03] and the impact of policies on AS-level route diversity.

AS router size (*ASsize*) can vary between ASs. Tier-1 ASs typically have more routers in their backbone part than small transit ASs. Nevertheless, previous work [MFM+06] has found that only a few routers are necessary to account for route diversity as seen from BGP data. Our level values range from 1 to 5 routers or depend on the tier of the AS.

Multiple peering links between ASs (*Peer*) are common in the Internet and increase the diversity of routes. [MFM+06] has shown that this factor is necessary to reproduce path diversity observed in the Internet. Our levels for this parameter range from 1 to 5 depending on the AS size.

IGP weights (*IGP*) often reflect the delay, the geographic distance, the link capacity, or are chosen to minimize network congestion [FT00]. Moreover, route choices are known to be sensitive to them [TGVS04]. Our levels for IGP weights range from random weights to uniform weights and even geographic distance.

iBGP topology (*Intra*) can limit the visibility of routes inside an AS. We consider iBGP full meshes as well as route-reflector hierarchies that follow the PoP structure as suggested by configuration guidelines [ZB03, BCC06].

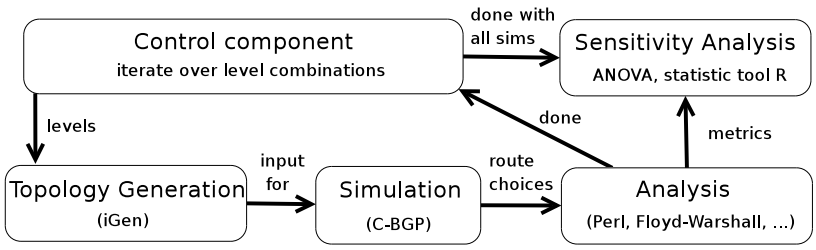
Border routers (*Border*) are not chosen arbitrarily in practice. Rather, each AS often has a limited number of them in specific locations. Moreover, peerings or customer-provider links are frequently established between close-by routers, typically routers at the same location, in order to minimize distances and costs. We use two levels: one which tries to minimize this cost and one which selects border routers randomly.

Note that AS-level connectivity is not a factor in our design for two reasons. First, it is non-trivial to combine other connectivity models such as randomly generated AS graphs with a notion of routing policies. Second, it is widely agreed that the Internet has a tiered structure [SARK02], which is unlikely to change in the foreseeable future. Therefore, we use an AS-level map obtained from [Cai] as basis for our Internet-scale simulations.

3.3 Simulation Setup and Choice of Levels

Before we execute our factorial design experiments, we dive into more details about the simulation setup and how we generate the topologies for the experiment.

Figure 3.1 Sensitivity analysis: components of the simulation framework.



3.3.1 Simulation

Our objective is to understand how sensitive the outcome of the routing process is to the different choices of levels for the topology factors. For this, we do not have to consider the BGP dynamics. Accordingly, we use the C-BGP simulator [QU05] in order to compute the paths that routers know once the BGP routing has converged [GW99].

With C-BGP it is possible to run Internet-scale simulations with more than 30,000 ASs and with complex inter-domain structures and inter-domain peerings. However, this takes a lot of time, even if we restrict number of prefixes to 100 and originate one prefix per AS for a selected set of ASs including tier-1, tier-2, and stub ASs. We found that such complex simulations may take up to 5 days to complete and may require more than 13GB memory[2]. For this reason, we decide to run such Internet-scale simulations based on AS topologies derived from CAIDA [Cai] data of April 2009 only for a limited set of parameter settings, amongst others for *defaultSim* and slight variations of *defaultSim*, see Section 3.3.2.

Our full factorial design results in 216 different combinations of the levels of our 6 factors. Together with the 10 iterations that we run per simulation this would require more than 2,000 C-BGP simulations. Obviously, we have to limit the size of the topology in terms of number of ASs for our sensitivity study[3] in Section 3.4. With regards to the size of the topologies, they have to be large enough to accommodate a tiered structure and allow for complex interconnections between ASs. Yet, they need to be small enough to allow efficient simulation of *all* routing tables and *all* prefixes. We find that a single simulation with 150 ASs, 150 originated prefixes and some 1,000 routers uses roughly 300 MB of memory and finishes within few minutes.

Despite the reduced topology size the computation of the metrics is very time-consuming, lasting roughly 30 minutes per simulation due to the computation of all shortest paths with the Floyd-Warshall algorithm[4] in weighted and non-weighted graphs. By parallelizing the processing, it is possible to finish one set of simulations and to compute the metrics within a few days, which is a reasonable time. For our Internet-scale simulations the computation of the metrics and Dijkstra's algorithm for the 100 destination prefixes take up to several hours for a single simulation.

Accordingly, our full factorial design is executed by choosing the number of ASs, in our case 150, and the number of simulation runs, in our case 10, for each choice of the factor levels. Then we generate the topology according to the factor levels, see Section 3.2.3, execute the simulation, and compute the metrics. After all simulations have been finished and analyzed, we perform the actual

[2] On AMD Opteron 865 multi-core machine with 32GB.
[3] For later discussions on "optimality" we rely on the Internet-scale simulations.
[4] Here we use an Intel Xeon Quad-Core platform with 2.4GHz processor and 8GB of memory.

sensitivity analysis. The process is summarized in Figure 3.1.

3.3.2 Topology Generation / Choice of Levels

Our guideline is a comprehensive design of simulations. Our full factorial design includes simulations configured based on configurational practices used in today's Internet (see Section 3.3.2). But we also choose levels to cover extreme cases. For example, as part of our sensitivity analysis we run simulations where no policies are configured at all. In doing so, we are able to explore the complete space of current configuration alternatives offered by BGP.

The generation of topologies is not done in one step. Rather we follow a top-down approach with three subtasks, see also Section 2: Generating an AS-level topology, the intra-domain topology, and the inter-domain interconnectivity.

AS-Level Topology

For the sensitivity study in Section 3.4.1 we rely on a "tiered" topology with 150 ASs — 5 tier-1s, 20 tier-2s, and 125 stub ASs. While this does under-represent the number of stub ASs, our goal was to include a reasonable complex set of tier-1 as well as tier-2 ASs and to roughly keep the proportion of transit to stub ASs as observed in the Internet [Cai]. Spot checks against the Internet-scale simulations show that using down-scaled topologies does not change the results drastically.

All tier-1 ASs are fully interconnected while tier-2 ASs are randomly connected to a subset of tier-1 ASs and also peer with a subset of each other. Stub ASs are either single- or dual-homed and connect to either tier-2 or tier-1 ASs. Note that the generation of the AS-level topology is non-deterministic. For example, the probability for a stub AS to be dual-homed is 50% and we randomly determine the number of tier-1 ASs to which a tier-2 AS attaches. In order to account for the non-determinism, we decide to run 10 iterations for each factor combination of our factorial design.

If the factor *Pol* is active, we configure policies according to the tiered architecture. For example, edges between two tier-2 ASs are peer-to-peer (p2p) links, while some edges between a tier-1 and a tier-2 are provider-to-customer (p2c) links and others are p2p links. Routing policies are realized via BGP filters, communities, and local-preference values and implement the no-export and no-valley property of AS relationships, see Section 2.5.1.

Intra-Domain Topology

Since intra-domain details of actual ASs are unknown and cannot be inferred, we rely on the functionality offered by IGen [Quo05], a structural topology generator, which produces plausible topologies using network design heuristics [LAWD04].

The level of the factor *ASsize* determines the number of routers for each AS. We consider 4 levels, 3 with a constant number of routers inside all ASs, i.e., 1, 2, 5. The fourth level varies the number of routers according to the position of the AS in the AS hierarchy, i.e., 30 routers for tier-1's, 15 for tier-2's and 1 per stub AS, roughly proportional with the observed structure of the Internet [SMW02, SBS08].

To be able to consider the geographic distance for the factor *IGP*, each router is placed at a specific geographic location (longitude, latitude). While large ASs are assumed to have routers all over the

Table 3.1 Parameter choice for *defaultSim*.

Pol	AS relationships applied
ASsize	Tier1: 30 routers, Tier2: 15, Stub: 1
Intra	Route reflectors inside tier-1 and tier-2
IGP	IGP weights according to geographic distance
Peer	# links per AS edge depends on size of ASs
Border	Connect to geographically-close routers

world, stub ASs are only present in a geographically limited area. Accordingly, the routers of tier-1 ASs and some tier-2 ASs are distributed world-wide, while other tier-2s and stub ASs are mainly restricted to a single continent.

As next step we setup the physical links by first constructing PoPs and then interconnecting them. Each PoP corresponds to a cluster of geographically-close nodes identified with the help of the K-Medoid[5] algorithm [KR05]. Within each PoP we distinguish between backbone nodes for connecting to other PoPs and access nodes. The backbone nodes within the different PoPs are interconnected via a clique, which corresponds to a MPLS fully meshed backbone. The access nodes are connected to the backbone nodes within the PoP using at least two edges. We need this complex intra-AS topology only if the AS is composed of a significant number of nodes and the level of the factor *Intra* is "route reflector". If the level of *Intra* is "route reflector", we configure a set of iBGP sessions that follow the physical topology. Otherwise we choose a full mesh of iBGP sessions.

The three levels of the factor *IGP* are: "uniform" which assigns uniform costs of 1 to all destinations; "random" which assigns random weights of 1 to 100 to all destinations; and "geographic" which assigns weights that correspond to the geographical distance between the source and the destination.

Inter-Domain Connectivity

As next step we need to determine the router-level connectivity for AS-level edges. The factor *Peer* determines how many router-level peering links are to be used for each AS-level edge. We again choose 4 levels, 3 with constant numbers of links per edge, i.e., 1, 2, 5. The fourth level varies the number of links according to the number of routers in the incident ASs. The larger the number of routers, the larger the number of peering links.

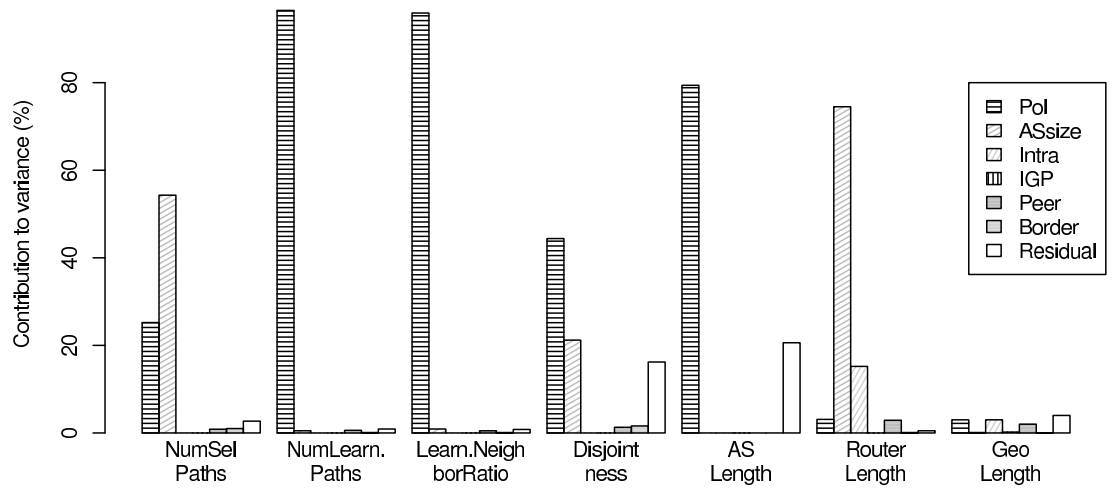
The factor *Border* determines on which routers the peerings are placed: either randomly or according to geographic distance. The base assumption of the latter case is that two domains prefer to connect at places that are geographically close to each other. For this purpose, we search among the $N_i \times N_j$ routers of AS i and j and establish the link between the geographically closest ones.

Default Simulation

In spite of recent work on topology discovery [SMW02, SBS08] and policy inference [BPP03, WG03, MUF+07], the actual routing policies and the router-level topology of the Internet are largely unknown. Yet, in order to obtain a simulation setup that is as realistic as possible, we consider

[5] For tier-1 (tier-2) ASs with 30 (15) routers we use 6 (3) clusters.

Figure 3.2 Contribution of individual factors to total variation (sum of squares).



configuration practices that correspond to common design guidelines in today's Internet and try to consider insights obtained from measurements of ISP topologies [SMW02, SBS08].

For the remainder of this chapter we will frequently refer to *defaultSim*, a choice of factor levels that is as close as possible to actual configurations in the Internet. Accordingly, such a setting uses AS business relationships, has route reflectors inside large ASs, and considers geographic locality when determining inter-domain connections and assigning IGP weights. The tier-1 and tier-2 ASs are reproduced with multiple routers, belonging to different route reflector clusters. The settings of *defaultSim* are summarized in Table 3.1. In addition to the small simulation setups with 150 ASs, we also run Internet-scale simulations for *defaultSim*, see Section 3.4.2 and Section 3.5.

3.4 Sensitivity Analysis

After identifying the metrics and factors relevant to the question of routing optimality and path diversity, we now examine the results of the sensitivity analysis.

3.4.1 ANOVA

We perform simulations for all possible combinations of factors and levels as outlined in Section 3.2. We then rely on the ANOVA technique, see Section 3.2.1, to identify those factors that strongly contribute to the variations of the metrics observed across the different choices of levels.

In our study we run 10 iterations per factor combination. Although the values of levels are constant, the non-deterministic setup of our topologies (see Section 3.3.2) leads to slight differences in the results from one simulation run to another. The variability in the results for repetitions of a given simulation setup is captured in the *residual* values.

To explore whether differences in the metrics are caused by the configuration factors or by statistical noise (due to running multiple simulation instances for each factor combination), we partition

Table 3.2 ANOVA – percentage of variation explained by individual factors ("%") and statistical significance (F-Value, "F") for all factors and metrics.

	Pol		*ASsize*		*Intra*		*IGP*		*Peer*		*Border*		Res.
	%	F	%	F	%	F	%	F	%	F	%	F	%
NumSelPaths	25.2	10^4	54.3	10^4	0.0	0	0.0	0	0.9	252	1.0	919	2.7
NumLearnPaths	96.5	10^5	0.5	420	0.0	0	0.0	1	0.6	529	0.1	142	0.9
LearnNeighRatio	95.9	10^5	0.9	885	0.0	0	0.0	0	0.5	446	0.1	611	0.8
Disjointness	44.4	6434	21.2	1025	0.0	1	0.0	1	1.3	65	1.6	228	16.2
ASLength	79.4	9023	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	20.6
RouterLength	3.1	10^4	74.5	10^5	15.2	10^4	0.0	5	2.9	4181	0.1	397	0.5
GeoLength	3.0	1786	0.1	2450	3.00	1867	0.2	54	2.0	329	0.05	10^4	4.0

the variance observed for each metric into its components using *sum of squares*. Moreover, we check for *statistical significance* relying on F-tests, see Section 3.2.1.

The influence of the 6 factors on each of the 8 metrics is shown in Figure 3.2 while Table 3.2 presents the corresponding numbers and F-values. The percentage values represent the fraction that a certain factor contributes to the total variance of the metric. Hence, a high percentage suggests that a given metric is very sensitive to that factor.

We point out that the values in each row of Table 3.2 do not always add up to 100%. For example, the fraction of variance for *NumSelPaths* (first row of Table 3.2) that can be explained by our six factors and the residuals is only 84.1%. The remaining 15.9% are due to *interactions*. Such interactions occur when it is not possible to distinguish the impact of two or more factors. In the above example the value of the metric *NumSelPaths* is sensitive to the combination of two factors: routing policies (*Pol*) and the size of ASs (*ASsize*). However, we find that the impact of such "intertwined" factors is low when compared to the impact of each individual factor.

We also observe that the residual values for the majority of our metrics are rather small (less than 4%), see Table 3.2. This suggests that the error or statistical noise of our setup is low compared to the variation across different factor configurations. This supports our choice of the factors. The factors indeed impact route computations more than the statistical variations. Nevertheless, for *Disjointness* and *ASLength*, we observe higher residual values with 16.2% and 20.6%, respectively. This is not that surprising as both metrics strongly depend on the AS-level graph, which is chosen non-deterministically (see Section 3.3.2).

From Figure 3.2 we immediately notice that the majority of our metrics are dominated by two factors: routing policies and the number of routers inside each AS. *NumLearnPaths* are mostly affected by policies (96.5%). For *NumSelPaths*, 25.2% of the total variation can be attributed to policies and 54.3% to the numbers of routers per AS. This is not too surprising given that policies determine which paths among those learned are preferred and the AS size limits the number of distinct AS paths that can be selected as best.

The two metrics that quantify the length of the selected routes in terms of AS-level and router-level hops, display a different behavior. *RouterLength* is dominated by *ASsize* which explains 74.5% of the total variation. The AS-level length of the selected routes is only sensitive to whether policies are used or not. We note that the length of the router-level paths is impacted by the choice of the intra-domain topology, i.e., if the topology has a full mesh of iBGP sessions or a route reflector hierarchy (15.2%). Finally, for *GeoLength* we cannot identify any factor that explains a large

Table 3.3 Impact of policies, intra-domain configuration and inter-domain connectivity on route selection.

	defaultSim	no policies	full mesh	random IGP weights	1 peering link	2 peering links	random border router
NumLearnPaths	2.58	6.84	2.59	2.59	2.56	2.62	2.62
Disjointness	0.81	0.83	0.81	0.81	0.81	0.81	0.81
RouterLength	11.67	7.80	8.72	11.80	12.34	10.65	11.43

fraction of the observed variation.

In summary, we find that the two most dominant factors for our metrics are routing policies and the number of routers per AS. While some of the other factors are statistically significant, their overall contribution to the variance is nevertheless small. In particular, intra-domain factors such as IGP weights or iBGP connectivity only have weak impact on global path properties.

3.4.2 Impact of Individual Factors

While the ANOVA technique helps us to understand *which* factors are responsible for the observed variations, they do not tell us *how* different parameter settings impact a certain metric. For example, we have not yet quantified how policies limit the number of paths that are learned by the routers of an AS. Therefore, we now explicitly study the impact of routing policies, intra-domain configuration and inter-domain connectivity and we investigate potential differences between tier-1, tier-2, and stub domains. The approach, we adopt, is to rely on *defaultSim* as a reference, vary one factor at a time, and then check for differences. Contrary to the preceding section, we now run Internet-scale simulations based on data from [Cai], see also Section 3.3.1. Table 3.3 presents the mean values for three of our metrics – *NumLearnPaths*, *Disjointness* and *RouterLength*– across all ASs and all routes, respectively.

Impact of Policies

If we repeat the same simulation but do not enforce AS relationships (“no policies”), the number of distinct AS paths learned by the routers of an AS increases from 2.58 to 6.84.

On the one hand, this can be seen as confirmation of the wide-spread belief that policies significantly restrict route diversity. On the other hand, the value of the *Disjointness* metric only slightly increases after removing policies from the simulation. We conclude that BGP does not necessarily propagate alternative, disjoint paths for routing. This can be seen as motivation for proposals such as path splicing [MEFV08].

For the metrics *RouterLength* and *ASLength* (not shown in Table 3.3), we observe that ignoring routing policies generally leads to shorter routes. Without policies the average distance in AS-level hops (router-level hops) is 5.67 (7.80) per selected route versus 4.30 (11.67) without routing policies. Although policies lead to longer selected routes, the consequent stretch is not dramatic, confirming previous results [TGSE01, SMA03, Hus].

Impact of Intra-Domain Configuration

The ANOVA results indicate that intra-domain design choices, e.g., iBGP full mesh versus route reflector hierarchy (*Intra*) or the strategy for assigning IGP weights (*IGP*) have a comparatively low impact on the results of route propagation and computation. Nevertheless, we find that 70% of all route decisions in *defaultSim* are reached based on other steps of the BGP decision process than local-preference and shortest AS path, confirming previous findings [TSGR04, BMU07]. To investigate, we now compare *defaultSim* against a simulation where we replace the route reflector hierarchies inside large domains with a full mesh (factor *Intra*), and another simulation where we assign IGP weights randomly rather than applying geographic-aware weights (factor *IGP*), see Table 3.3.

Varying only the settings of the two factors *Intra* and *IGP* in *defaultSim* does only induce minor changes in the values of our metrics. The only exception is the metric *RouterLength*. Here we observe that the average number of hops per route decreases from 11.67 to 8.72 when replacing the route reflector hierarchy with a full mesh.

Given these results, we conclude that it is not mandatory for studying inter-domain route propagation to emphasize intra-domain aspects such as assignment of IGP weights or the topological structure of a domain. This relative unimportance of intra-domain aspects is surprising given their importance for the BGP decision process.

Impact of Inter-Domain Connectivity

Finally, we quantify the impact of the two factors *Peer* and *Border* on our metrics. In addition to *defaultSim*, where the number of peering links between two ASs depends on the size of the neighboring ASs, we run a simulation where we realize each AS-level edge with 1 or 5 peerings. We find only minor differences for the metrics *NumLearnPaths*, *Disjointness* and *RouterLength*, see Table 3.3. In terms of *RouterLength*, the average number of router hops increases from 11.67 in *defaultSim* to 12.34 with only one peering link per AS-level edge but decreases to 10.65 for five peering links: If there are many peering links per AS-level edge, each router can choose between many egress routers. Consequently, the likelihood that a packet can be forwarded over a direct link to the neighboring AS increases.

Note, however that irrespective of how the IGP weight assignment is done, geographic-aware for *defaultSim* or via random assignment (factor *Border*), the average number of router-level hops only slightly changes.

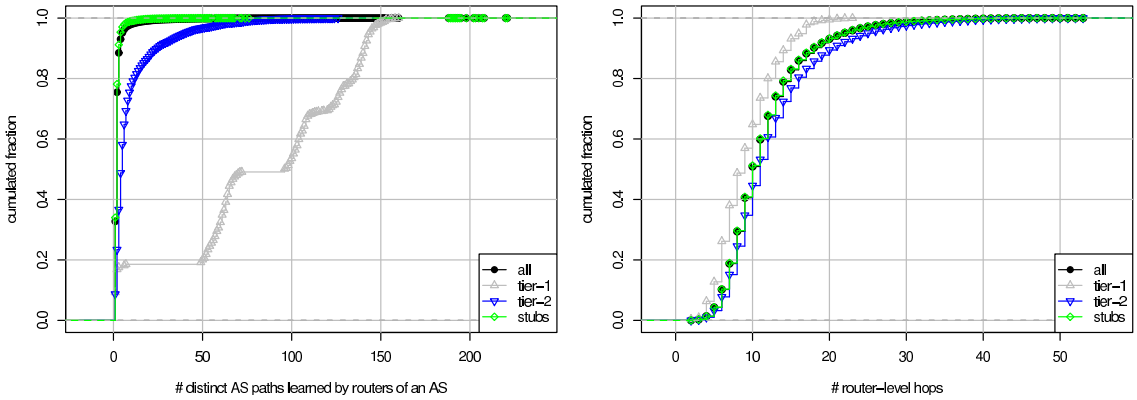
In summary, path diversity is almost always insensitive to the realization of the AS-level edges (factor *Peer* and *Border*): Neither the selection of border routers nor the number of peering links per AS-level edge have a dominating impact.

Tier-1 vs. Tier-2 vs. Stub ASs

We briefly explore if there are any significant differences between tier-1, tier-2, and stub domains. Once more, we rely on the Internet-scale simulation of *defaultSim* as basis and compute the results of our metrics separately for each type of AS.

Figure 3.3a) shows the cumulative distributions of the number of distinct paths learned by the routers of tier-1, tier-2, and stub domains (*NumLearnPaths*). We observe that path diversity is highest for tier-1 ASs (mean: 81.8 paths), followed by tier-2s (mean: 9.8) and stubs (mean: 2.2

Figure 3.3 Sensitivity analysis: differences between tier-1, tier-2 and stub ASs for *defaultSim*.



(a) Number of learned paths per AS (*NumLearnPaths*) (b) Number of router-level hops (*RouterLength*)

paths). This is mainly due to the AS-level connectivity of today's Internet. While stub ASs[6] are only connected to a small number of upstream ASs, tier-1 and tier-2 domains have peerings with many neighbors.

Figure 3.3b) shows for tier-1, tier-2 and stub domains the distribution of path lengths in terms of router-level hops. In general, tier-1 ASs can use shorter paths towards destinations. While tier-1 domains have paths with an average of 9.3 router hops, stub (tier-2) domains have a mean path length of 11.6 (12.8) hops. Overall, the differences observed among the types of ASs are consistent with what we would intuitively expect.

3.5 "Optimality" of Path Selection

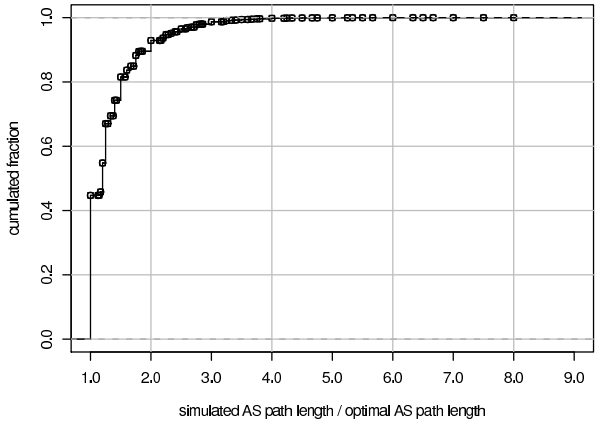
After the general sensitivity analysis we now ask: How "optimal" are the computed routes? While the term "optimality" can be interpreted in many different ways, we rely on the metrics of Section 3.2.2. We explore the length of the selected routes in terms of AS-level (*ASLength*) and router-level hops (*RouterLength*) as well as geographical distance (*GeoLength*). These path properties contribute to the delay that a packet sees when routed along a path.

The key idea for judging the "optimality" of routing is to compare the routes computed in the simulations with globally optimal routes. The latter can be determined by applying Dijkstra's algorithm: We compute routes with a minimum number of AS-level hops, router-level hops, or minimal geographical distance. Since we use simulations, we can in principle judge "optimality" for *any* pair of source/destination router.

While Section 3.5.1 analyzes the AS-level path stretch of the simulated routes vs. the "optimal" routes, Section 3.5.2 studies the router-level path stretch. Finally, Section 3.5.3 investigates the geographical distance that is covered if packets are forwarded along a route. For the remainder of this section we will use an Internet-scale simulation based on *defaultSim* as a basis.

[6] a considerable fraction of them is still only single-homed.

Figure 3.4 AS-level path stretch for *defaultSim*.



3.5.1 AS-Level Path Stretch

The AS-level path stretch is determined by comparing the number of AS hops of the selected AS path with the minimal path length in terms of AS hops that connects the same pair of routers. This is done for more than 3 million pairs of source and destination ASs. While we take the length of the selected AS path directly from the simulation results, we use Dijkstra's algorithm to obtain the shortest paths from more than 30,000 sources ASs to the 100 destination prefixes.

Figure 3.4 shows the distribution of the stretch factor, i.e., number of simulated AS hops divided by the optimal AS path length. While the stretch ratio averages 1.3, we observe for almost half of the source/destination pairs a simulated path with same length as the optimal path. There exists only one case with a stretch factor of 8. Overall, the observed AS-level stretch is reasonable. We point out that the observed AS-level path stretch is exclusively due to the use of AS relationship policies, which imply "preference" rules and the "no-valley" property [Gao00].

3.5.2 Router-Level Path Stretch

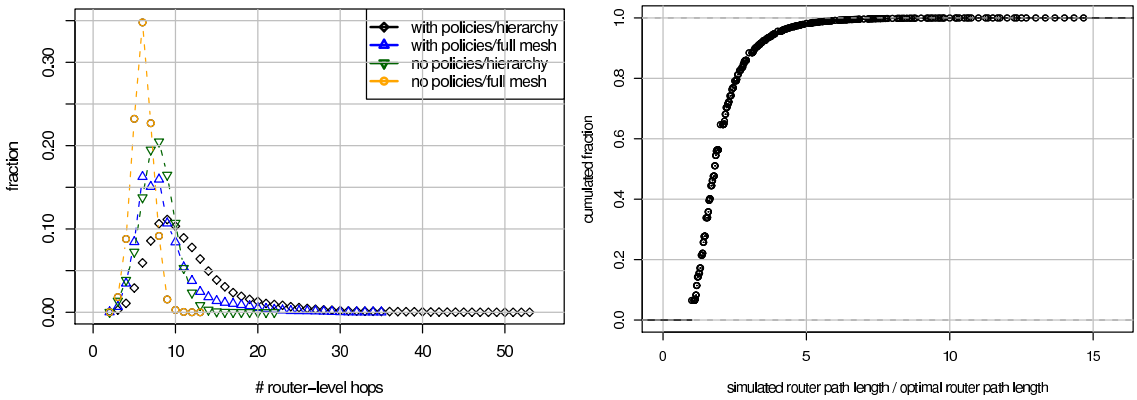
Contrary to AS-level paths, router-level paths may incur stretch for two reasons: Routing policies and ASs with multiple routers/intra-domain configuration. Both aspects are investigated in this section. The general approach for analyzing the router-level path stretch incurred by the selected routes is similar to Section 3.5.1. Again, we rely on *defaultSim* and our AS-level topology from CAIDA [Cai].

Figure 3.5a) shows the distribution of router-level path lengths for 4 different setups. We compare *defaultSim* against a setup (i) where we ignore routing policies, (ii) where we replace the route reflector hierarchy with a full mesh of iBGP sessions, and (iii) where we have both no policies and a full mesh instead of a route reflector hierarchy. Apart from that the *defaultSim* remains unchanged.

The general observation is that the longest router-level paths occur for the setup with policies using a route reflector hierarchy, followed by "with policies/full mesh" and "no policies/hierarchy". If we replace the intra-domain structure by a full mesh and if we remove policies, the length of the routes selected in our simulation decreases from 11.7 to 7.8.

The router-level stretch factor, i.e., number of router-level hops in simulated path divided by the

Figure 3.5 Optimality of router-level paths in terms of router-level hops.



(a) Length of router-level paths

(b) Router-level stretch of *defaultSim*

optimal path length is shown in Figure 3.5b). Overall, the majority of the routes in our simulations experience moderate stretch with an average stretch of 2.1. There are only very few extreme cases with a stretch ratio of more than 2.4 (3rd quantile).

These observations clearly demonstrate that both policies as well as having ASs with multiple routers impact the length of the paths of the best routes. With regards to the intra-domain configuration, we observe that routes incur stretch if ASs consist of multiple routers. In such a case, packets may have to visit more hops in order to traverse a transit AS domain. Routing policies can cause BGP to prefer longer AS paths over a shorter one if local-preference values are applied. Hence, we observe a correlation between the number of AS-level and router-level hops of a path in our simulation. Moreover, we find for more than 5% of the routes in *defaultSim* that local-preference values are used as final criterion to select a best route among the set of learned ones. While this looks like a small percentage, it is actually significant as such routing decisions are then potentially propagated to many neighbors. This underlines the impact of policies on the "optimality" of router-level paths.

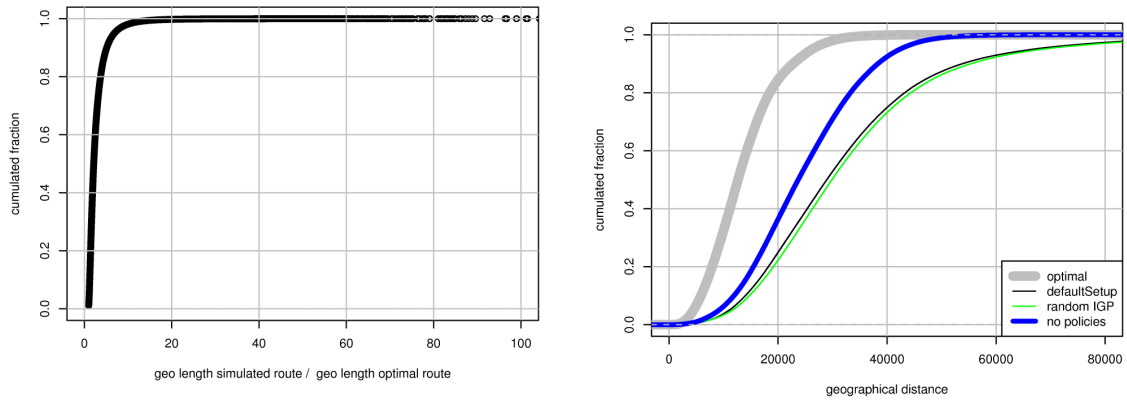
Again, we study the BGP decision process and find that criteria such as "hot-potato routing" are frequently used as decision criteria – in some ASs for more than 60% of the decisions. This underlines the importance of intra-domain routing structures on inter-domain routing and explains why the router-level path stretch is sensitive to the choice of the use of a full mesh or a route reflector hierarchy. Nevertheless, BGP path choice as a whole is mostly insensitive to intra-domain factors such as IGP weights (see Section 3.4.1).

3.5.3 Geographical Path Stretch

In general, geographic properties have only indirect impact on BGP routing decisions. For example, IGP weights may be chosen to prefer routes with geographically short distances. Given that the geographic distance and the delay that packets experience on the path are correlated, we now study the degree to which routes are "optimal" in terms of the geographical distances of the links on the forwarding path.

Again, we rely on *defaultSim* and run Internet-scale simulations based on this choice of levels. Since we assigned routers to geographic locations in our topologies, we can compute the geographic distance of each route by adding up the distances of the individual links it traverses to reach its

Figure 3.6 Optimality of routes in terms of geographical length.



(a) Distribution of geographical stretch

(b) Distribution of geographical lengths

destination. To determine the geographically shortest paths from more than 30,000 source ASs to our 100 prefixes, we apply Dijkstra's algorithm on a weighted router-level graph, where weights correspond to the distances between adjacent routers.

Figure 3.6a) shows the cumulative distribution of the geographic stretch ratio, i.e., the geographic length of simulated path divided by length of the geographically shortest path, for the more than 3 million pairs of source and destination ASs. The results suggest that most routers incur a stretch ratio that is comparable to that of the router-level stretch: The average geographic stretch is 2.9 (average router-level stretch: 2.1) while for 25% of the source/destination pairs we observe a stretch ratio of more than 3.3 (router-level: 2.4). Since BGP route decisions are based on the BGP-level topology and on routing policies that do reflect business goals rather than geographical distances, we can even observe paths with a stretch ratio of 100.

We now check how routing policies as well as different strategies for IGP weight assignment impact the geographical stretch of the selected routes. To find out, we modify the strategy for assigning IGP weights or remove routing policies but otherwise we keep the simulation setup identical. Figure 3.6b) shows the cumulative distribution of the geographical length of the selected routes. The four curves display the results for *defaultSim*, which assigns IGP weights to reflect geographical distances, for modified simulations where random IGP weights are applied or no policies are configured, and for the "optimal" lengths.

At first glance, we infer from Figure 3.6b) that the curves for all our simulations are considerably below the curve for the "optimal" lengths. This implies that neither removing policies nor changing the strategy for IGP weight assignment significantly shortens the geographical lengths of the selected paths. The simulation run closest to the "optimal" curve is the configuration where we ignore routing policies. Comparing different IGP assignments, the setup of *defaultSim*, where IGP weights reflect the geographical distance, performs only slightly better than random IGP weights.

Overall, it seems hard to globally optimize route propagation and selection with respect to the geographic length of the selected routes. In our opinion, the reason is that BGP has rather been designed to support flexible routing policies for implementing business objectives. If the geographical length of paths is to be globally optimized, routing policies of individual ASs need to be consistent and need to optimize the same objective: Geographically short routes and *not* routes that agree with

existing business agreements.

3.6 Related Work

While there are technical [KK77] and economic reasons (e.g., business agreements) to expect that Internet routing is "non-optimal", a large-scale simulation-based study and a quantitative analysis of sensitivity, as performed in this chapter, has not been the subject of research so far.

Closest to our work, Savage et al. [SCH+99] investigate the degree to which end-to-end performance of paths is determined by the routing system and to understand which mechanisms are responsible. For their analysis they rely on traceroute measurements and use round-trip time, loss rates, and bandwidth to judge the optimality of paths. As measurement-based approaches are inherently limited regarding observability, we rely on a completely controlled simulation environment and are more flexible regarding the computation of our metrics.

Besides, previous work has studied the impact of policies on the properties of the selected routes. For example, by comparing the shortest path with the shortest policy-compliant path, Tangmunarunkit et al. [TGSE01] find that a considerable number of Internet routes are inflated due to routing policies. Spring et al. [SMA03] take a step forward by characterizing the root causes of path inflation. Based on a trace-driven study they speculate that intra-domain traffic has minimal impact on path inflation while peering policies and inter-domain routing lead to significant inflation. This is in accordance with the findings of our chapter. However, we also study the impact of other factors such as intra-domain configuration.

Several approaches to the generation of Internet router-level topologies suitable for simulation have been proposed in the literature. The first and most natural approach is to rely on existing network topologies. This approach is limited due to the difficulty of obtaining the topology of operational networks. Then, there have been proposals for inferring network topologies at the router-level from measurements [SMW02, SBS08]. Another approach consists in generating synthetic topologies, sharing selected properties with the Internet. Available generators such as BRITE [MLMB01] and GT-ITM [KCNSZ03] produce topologies that respect graph properties seen in the Internet. Finally, Alderson et al. [ADGW03] present a novel approach for designing and generating Internet topologies based on the economical and technical forces that drive the growth of the Internet.

3.7 Summary

In this chapter, we perform a comprehensive study of how sensitive routing optimality and diversity metrics are to factors such as policies, AS size, topology, and IGP weights, etc. We find that intra-domain factors only have marginal impact on global path properties – for example no setting of BGP parameters decreases the geographic path stretch significantly. In contrast, routing policies and AS size in number of routers are the dominating factors.

Our results reveal that it is hard to improve the global properties of route selection by purely tweaking BGP attributes or changing iBGP graphs, etc. This is consistent with inter-domain routing design, that mainly supports the flexible implementation of routing policies, e.g., for enforcing business objectives, but not the propagation of optimal paths. Improving the global properties of Internet paths will require more than tuning BGP.

Our work is an important step towards understanding which and how parameters impact the optimality of inter-domain routing. This is crucial for a wide variety of tasks, e.g., for building scalable

and meaningful models of routing, for designing or improving routing protocols etc. Both our sensitivity analysis and our simulation framework will prove useful for the evaluation and comparison of routing protocols and architectures.

4 Building an AS-Topology Model that Captures Route Diversity

In the preceding chapter we have taken a vital step forward towards the overall goal of this thesis: Getting a grip on the complexity of inter-domain routing in the Internet. One key insight from our sensitivity analysis is that the global properties of route selection, e.g., diversity and optimality, are predominantly determined by routing policies and by the number of routers of which an AS is composed.

Whenever it is hard to understand or predict in reality, researchers resort to models that correctly reproduce those aspects that are relevant within the scope of the model's application. There is wide consensus among researchers that state-of-the art models for the Internet are not sophisticated enough to predict the behavior of the Internet under specific questions. For example: What will happen if a certain peering link is removed or if we change policies? How can a network operator identify potential ASs for peering and predict the benefits of a new peering?

In this and the following chapter we explore how to build models of the Internet so as to be able to have useful predictive capabilities regarding BGP paths. We are aiming at effective models that find the right balance between what is important and what is not: Aspects that are relevant within the scope of the application need to be reproduced with a high level-of-detail while others can be safely ignored to keep the model scalable. One lesson from our sensitivity analysis in Chapter 3 is that routing policies and the size of ASs in terms of routers are the main factors that impact global route choices. Hence, we adopt a twofold approach for constructing models with predictive capabilities: While Chapter 4 mainly investigates how to incorporate the minimum amount of intra-domain information, the following Chapter 5 searches for the right granularity to reproduce routing policies.

4.1 Motivation and Overview

In the past, high-level features of the inter-domain topology have been used to make generic inferences about its behavior, e.g., power-law distributions [FFF99] have been used to imply important "centralized" nodes (see [LAWD04] for a discussion of this issue). These types of generic inference are useful in terms of scientific understanding of the Internet as it evolves but do not allow one to answer specific questions about the current Internet.

We seek to be able to answer specific what-if questions, e.g., what if a certain peering link was removed, or what-if we change policies thus? In principle, knowledge of the Internet's inter-domain topology can be used to answer such questions, and the capability would provide great utility for providers. This is particularly true given that the focus for large providers has moved from simply providing connectivity, to maintaining contractual or business relationships that may require resilience despite changing traffic demands or link failures, in addition to supporting customers who demand more control over their traffic flows [RS04, BQ03].

Despite the requirements, current practice is quite limited. Often, the only available approach is "tweak and pray" [FBR03, FWR04, FR07]; that is, providers make changes with limited ability to predict the results, and then observe to see if the desired effect occurred. We propose to build an AS-routing model which enables us to predict unobserved Internet paths with good accuracy.

It is known [MQWZ05], that for the extracted model to be useful in prediction, it must be substantially better than those tested so far. Until now, models of the network structure have been predominantly inter-domain level models that do not worry about the details of the ASs [MRWK03, MJR+04, MQWZ05]. However, ASs are not simple nodes in a graph — they are comprised of routers. The internal structure of an AS *does* matter. It influences inter-domain routing, for instance via hot-potato routing [TSGR04, TDRR05]. Furthermore, there are multiple connections between ASs, typically from different routers, and this adds to the diversity of known routes [TMSV03]. Even where policy is uniform across an AS, internal features of the AS may result in different route choices for each router — this is a feature of BGP that allows behaviors such as hot-potato routing. Such diversity is commonly observed in public routing databases such as RouteViews [ROU]. An AS which is a single node must always choose a single best path to pass to its neighbors, and therefore cannot represent this type of diversity.

In addition, inter-domain routing is controlled by diverse *policies*, decided locally by each AS, but acting globally across the entire system [GSW02]. Hence, the topology of the inter-domain graph is not, in itself, sufficient to make predictions about Internet routing. In addition, policies need to be considered. Many policy relationships may be described as "customer-provider" or "peer-peer", and in these simple cases policies are enacted using simple, well-known filtering rules, see Section 2.5.1. Several papers have discussed inference of these simple policy rules [Gao00, WG03, BPP03], but unfortunately, not all policies fit these simple rules: for instance, in some cases of multiple links between two ASs, the policies may vary even between links. Our approach to all of these issues is to remain agnostic about what practices occur or do not occur in the current Internet. We make minimal assumptions about inter-domain routing and let the data speak for itself.

Of course, it is impossible to infer all of the internal details of an AS's policies. We are not seeking to reverse engineer the Internet. Our model does not necessarily correspond to the policies actually used by the ASs. Rather the results are analogous to the IGP (Interior Gateway Protocol) link weights inferred by Rocketfuel [MSWA02, SMA03], which do not correspond to those of the real networks investigated, but are nevertheless useful in understanding intra-domain topologies. We introduce policies into our AS-routing model with the goal of making predictions about the behavior of unobserved paths.

Likewise, we do not seek to reproduce a Rocketfuel-like detailed intra- and inter-domain connectivity map [SMA03], as a significant part of this information is not used in determining routes. Rather, we shall build topological models incorporating intra- and inter-domain information at the minimum level of detail needed to explain the observed routing in the Internet. The resulting simplicity allows us to derive insight into the relationship between routing policies, path diversity, and the actual choice of the paths propagated across the Internet without having to model the complexity of the routing inside an AS [QU05]. It gives us the ability to determine precisely where internal details matter, and how much.

Our approach is based on the idea of building a topology and policy model that is consistent with the observed routing in the Internet. To this end, we exploit BGP observations, relying on the data set introduced in Section 2.4.2. We separate our data into two datasets: a training and a validation dataset. The training set is used to build a topology and policies consistent with observed routing. We do so using a set of simulation-based iterative refinement heuristics (described in Section 4.3)

that introduce a minimal set of topology and policy changes required to match observed routing. We accommodate path diversity by creating multiple quasi-routers within each AS. A quasi-router represents a group of routers all making the same choice about best route, and so the "quasi-router topology" does not represent the physical router topology of a network but rather the logical partitioning of its policy rules. Importantly, we try to minimize the assumptions we make about "likely" policies, e.g., we do not assume that relationships fall into neat categories. We find that we can build an AS-routing model that matches the training set *exactly*. However, remember this is not the real topology, and the fact that it can match the training set exactly is not sufficient to show that the results are of practical use. We test the usefulness of the model by making a set of predictions about routes and validating them with the data excluded from training. We find that we can match the predictions down to the final BGP tie break in more than 80% of the test cases, see Section 4.4.

The work in this chapter is not (principally) concerned with modeling Internet routing dynamics. The dynamics are clearly important but considerable effort has already gone into such modeling e.g., [GW99, LABJ00, LMJ99, TGSE01, FMM+04, GSW02]. In our first prototype for predicting Internet behavior we model the equilibrium behavior of this system for the (vastly) predominant case that a stable routing solution exists. It is these equilibrium behaviors that are of most interest for the questions posed earlier. Attempting to model and incorporate dynamic information into our predictions, is a worthwhile goal (for example see [AFBB02]) but beyond the scope of this thesis.

To summarize our contributions: We present a methodology for deriving an AS-routing model that can reproduce all observed AS paths and predict unobserved routes with reasonable accuracy. Furthermore, we show the importance of considering more than one router per AS and accommodating a wide range of policies. Another major distinction of our work is that we use simulations to refine our model based on a large set of BGP data from diverse vantage points but evaluate the results using a separate set of BGP vantage points.

4.2 Domains as Simple Nodes

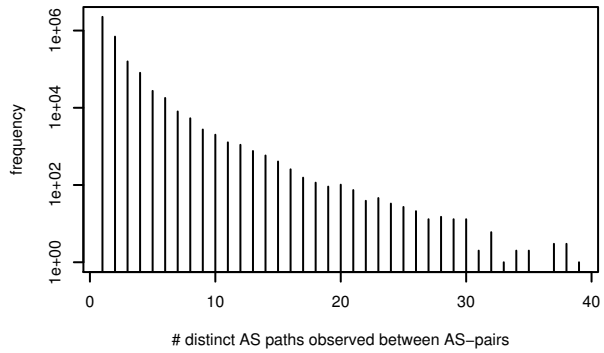
In this section we use measured routing data to illustrate the need to go beyond treating ASs as simple nodes in a graph. We first analyze the degree of route diversity present in the current Internet and then examine the limitations of single-node AS models for predicting path choices throughout the Internet accurately. The data shows that one must have a way to capture some internal details of routing at least for a subset of ASs.

4.2.1 Data Set

We rely on the data set presented in Section 2.4.2. Initially, we identify level-1 providers by starting with a small list of providers that are known to be tier-1. An AS is added to the list of level-1 providers if the resulting AS-subgraph between level-1 providers is complete, that is, we derive the AS-subgraph to be the largest clique of ASs including our seed ASs. This means that the AS-graph contains edges for all level-1 AS-pairs. This results in the following 10 ASs being referred to as level-1 providers (174, 209, 701, 1239, 2914, 3356, 3549, 3561, 5511, 7018). Note, this list is not complete. However, all found ASs are well-known tier-1 provider.

There are 7,994 ASs that are neighbors of a `level-1` provider in the BGP graph. We refer to these as `level-2`. All other 13,174 ASs are grouped together into the class `other`. Of the 21,178 ASs 3,486 provide transit for some prefixes in the sense that they appear at least once in the middle of

Figure 4.1 Histogram of # of distinct AS paths between pairs of originating and observation ASs.



an AS path. Among those ASs that do not provide transit, called stub-ASs, we distinguish between those that are observed to have a single upstream provider (are single-homed) and those that have multiple providers (are multi-homed). We find that there are $6,611$ single-homed and $11,077$ multi-homed ASs. Single-homed ASs that do not provide transit only add limited information about the AS-topology as long as any path information gathered from prefixes originated at such stub-ASs is transfered to a prefix originated at its AS neighbor. Removing single-homed stub-ASs and AS paths with loops from the AS-topology results in a graph with $14,563$ nodes and $52,288$ edges.

Note, our data does not cover the complete AS topology [ZLMZ05] since not all AS relationships are observable in our data. There are relatively more observation points in the level-1 and level-2 ASs than in the other ASs. Therefore, it is likely that AS-relationships involving level-2 providers are missing. Yet, their impact with regards to routing can be expected to be less significant.

4.2.2 Route Diversity in the Internet

To investigate the significance of route diversity in the Internet we examine how many different routes can be seen for each originating and observation AS pair (over all prefixes advertised by the origin). Figure 4.1 plots a histogram of the number of distinct AS paths using a logarithmic y-axis. Note, that for more than 30% of the AS-pairs we see more than one AS path. Indeed, there are more than $5,000$ pairs with more than 10 different paths.

Each AS may originate multiple prefixes and an AS path may be used by many prefixes. Indeed, we find that there are very popular AS paths used by more than $1,000$ different prefixes while the number of AS paths that are only used by a single prefix is less than 50%. When plotting the histogram of how many prefixes are propagated along an AS path on a log-log plot, one can see a linear relationship (plot not shown). In terms of route diversity, we observe that most prefixes are only propagated through a single AS path. Yet, there are quite a number of prefixes whose propagation samples the full path diversity between two ASs.

Obviously, one router per AS is not sufficient to capture the full diversity imposed by intra-domain routing. A single router can only propagate the route it chooses as best. With multiple routers each router within the AS can select its own best route and propagate it.

To motivate the need for modeling ASs with several routers, let us consider a concrete example from our data for the prefix `202.94.48.0/20` at AS 5511 shown on Figure 4.2.

Figure 4.2 Example of path diversity observed in the Internet.

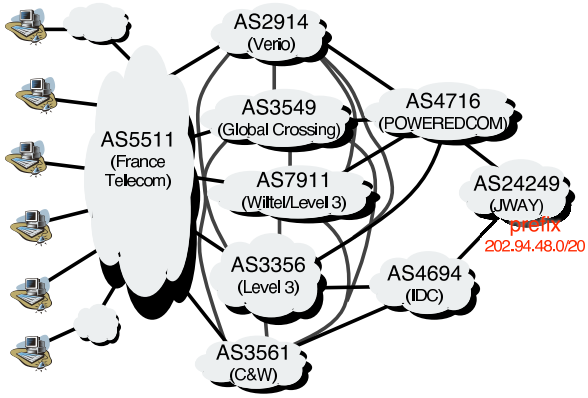


Table 4.1 Maximum route diversity received for all ASs.

Percentile	25	50	75	90	95	98	99	100
max # of unique AS path	1	2	4	5	7	10	10	23

AS 24249, which originates this prefix, is multi-homed to two ASs: AS 4694 and AS 4716. From these two providers the route is propagated to five level-1 providers: AS 2914, AS 3356, AS 3549, AS 3561, and AS 7911. Since AS 3356 propagates multiple AS paths to AS 3356, it needs to be modeled by at least two different routers. Which route is propagated can depend on the specific setup within the AS. Yet, path diversity within the ASs is only partially responsible for the route diversity. Another reason is the large interconnectivity in the core of the Internet; in this case 5 out of 8 AS paths. Still AS 3356 needs eight routers to propagate all paths further downstream.

To judge how much of the path diversity is due to multiple routes per ASs rather than multiple routes from different ASs, we determine the distribution of the maximum number of distinct unique paths each AS receives towards any destination prefix. This value is a lower bound on how many routers are needed inside an AS to propagate all these paths to downstream ASs. Table 4.1 shows the larger quantiles of this distribution. We observe that more than 50% of the ASs receive two unique AS paths for at least one destination prefix, 10% more than 5, and 2% more than 10, respectively. This highlights the importance of not loosing such path diversity.

4.2.3 Route Diversity in Single Router Models

In the past, large-scale models of routing in the Internet frequently assume that each AS consists of a single router, e.g., [MQWZ05]. To judge how appropriate this is for answering practical questions, we now examine how accurately it can predict AS path choices throughout the Internet.

We use the BGP simulator C-BGP [QU05] to compute AS-level paths on the AS-level graph after eliminating the stub-ASs. We originate one prefix per AS, resulting in 14,563 prefixes. Originating multiple prefixes per AS does not provide more information since at this point we do not consider per-prefix specific policies. To evaluate the quality of the model, we compare the predicted and

Table 4.2 Agreement between predicted and observed AS paths (single router per AS).

Criteria	Shortest Path	Customer/ Peering Policies
AS-Paths which agree	23.5%	12.5%
AS-Paths which disagree due to	76.4%	87.5%
AS path not available	49.4%	54.5%
shorter AS path exist	4.7%	5.7%
lowest neighbor ID	22.2%	27.3%

observed AS paths. Table 4.2 summarizes the results. Not surprisingly, we have agreement for only 23.5% of the AS paths. The main problem is again that for slightly less than 50% of the prefix/observation point combinations the observing AS does not even learn the "correct" AS path. For the remaining 50.6% only 4.7% of the incorrect decisions occur due to the shortest-path step of the BGP-decision process (Figure 2.2). If a router learns the "correct" route, it seems to be able to choose the "correct" one in roughly 50% of the cases.

Today's Internet does not use shortest-AS path routing as we assumed above. Most BGP peerings come with routing policies of which the most common ones can be classified as customer-provider and/or peering relationships. Relying on the BGP data, we use a simple heuristic for inferring customer-provider relationship utilizing the valley-free assumption [Gao00, SARK02, BPP03]. We start by declaring all links between the level-1 ASs as peering and then iteratively infer customer-provider relationships. We verify our classification by using data from several ASs whose peering policy we have access to. This results in $34,087$ customer-provider peers, $7,290$ peering relationships, and 640 siblings. All other edges cannot be classified. We then realized appropriate policies based on the local-pref BGP attribute and route filters[1] in the simulator and rerun the simulations. The results are fairly discouraging with only 12.5% agreement on the AS paths. The main problem is that for a lot of the prefix/observation point combinations the observing AS does not learn the "correct" path. Overall, this indicates a low accuracy for AS path prediction if an AS-routing model is solely based on AS-relationship inference.

Unfortunately, an agreement of less than $1/4$ for the selected best AS paths and just above $1/2$ for the available AS path, while not too bad, is not sufficient to answer, e.g., what-if questions such as how the routing in the Internet would change if a peering is added or de-peering of some provider occurs. Accordingly, we tackle the task of deriving more accurate models. In order to account for route diversity and to predict unobserved Internet paths, we allow for *routing policies* as well as for *multiple routers* inside ASs.

4.3 Methodology

The goal of this section is to propose a methodology for building an AS-routing topology model that captures the outcome of the routing policies and the internal structure of all ASs from observed BGP

[1] We treat siblings in the same manner as peerings relationships and set the same local-preference for unknown AS edges as for peerings.

data in order to answer practical questions about routing. The example question we use to highlight the capabilities of our model concerns predicting Internet path choices for previously unobserved AS paths.

We consciously choose an approach which allows for multiple routers, so called quasi-routers, within an AS and that is agnostic about inferred relationships such as customer-provider and/or peering relationships. After all, the real world knows many variants of such relationships [Nor]. We take the approach of modeling what we actually observe. In this manner we can avoid many potential pitfalls that arise from incomplete assumptions or trying to press BGP into some fixed schema.

In the following we first introduce the components of our AS-routing model and then show how one can evaluate its predictive capabilities. Next, we introduce our principle approach and then give an example of how to use it for deriving an AS-routing model from gathered BGP data. Finally, we discuss how to use the model for predicting previously unconsidered AS paths and how to improve it for previously unconsidered prefixes.

4.3.1 Components of the AS-Routing Model

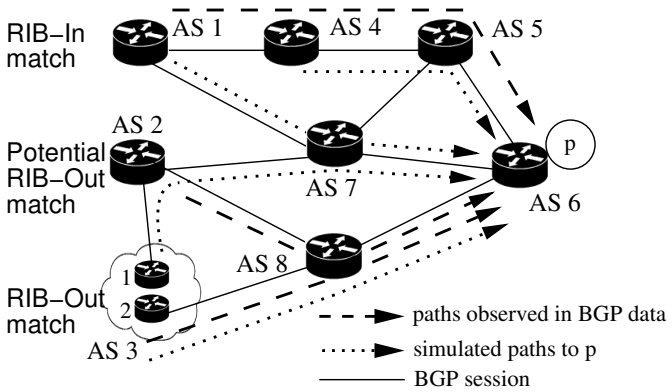
The AS-routing model should be capable of predicting AS-level paths, as used in the Internet, and so it needs to have a notion of inter-domain connectivity. Since it should capture the impact of intra-domain routing, it needs to account for the diversity and connectivity within each AS. Furthermore, as BGP is used to implement policies, we must accommodate this in our model.

Based on these criteria and the fact that we do not yet consider BGP dynamics, we propose to use a class of topology models that can also be used as input to the C-BGP simulator [QU05], see Section 2.6. Since C-BGP only computes the steady-state choice of the BGP routers after the exchange of the BGP messages has converged and not the whole state machine of the BGP routing protocol, it is thus possible to perform large-scale simulations for single prefixes on topologies with more than $16,500$ routers split among $14,500$ ASs in 2 – 45 minutes with 200 MB – 2 GB memory consumption depending on the complexity of the routing policies. C-BGP's capability of simulating large-scale propagation of BGP routes not only allows us to test how accurately the model can answer our example question, it also enables us to refine an AS-routing model incrementally.

While deriving the model we make the simplification that we only originate one prefix per AS. This allows us to address questions regarding path diversity while keeping the model manageable. For similar reasons we again exclude stub ASs but keep their AS path to ensure that we do not loose any path information.

We capture the inter-domain connectivity via an AS-topology graph as extracted from the BGP data. In order to represent the intra-domain routing diversity, we allow each AS to consist of multiple quasi-routers . A *quasi-router* represents a group of routers within an AS all making the same choice about best route, and so the "quasi-router topology" does not represent the physical router topology of a network but rather the logical partitioning of its policy rules. Each edge (AS 1, AS 2) of the AS-topology is realized by establishing a BGP session between one or more quasi-routers from AS 1 to one or more quasi-routers from AS 2. Propagation of routes can be restricted by applying route filters and/or by introducing other routing policies.

Figure 4.3 Example for metrics to measure agreement between observed and simulated routes.



Observation point	RIB-In	RIB-Out	
		observed	simulated
AS 1	1-4-5-6, 1-7-6	1-4-5-6	1-7-6
AS 2	2-7-6, 2-8-6	2-8-6	2-7-6
AS 3 (router 1)	3-2-7-6	3-8-6	3-2-7-6
AS 3 (router 2)	3-8-6	3-8-6	3-8-6

4.3.2 Evaluating Prediction

C-BGP enables us to predict, using an AS-routing model as input, the AS path along which the routing information for any prefix, originated at any node, is propagated to any other node.

For a fair evaluation we need one dataset to derive the AS-routing model, called `training` , and another separate one, called `validation`, to evaluate the quality of the AS-routing model. We divide the available BGP data randomly into two subsets by assigning observation points to either subset. This places *all* paths, observed at an observation point, into one of the two subsets. The `training` set is then used to derive the AS-routing model while the `validation` set is used for evaluation purposes.

An alternative way of slicing the data is to split the set of AS paths according to the originating ASs into two subsets. One can then compare how well an AS-routing model derived from a subset of the prefixes predicts the AS paths for another set of prefixes. Furthermore, one can combine both approaches and partition the obtained `training` or/and `validation` subsets according to the originating AS.

The evaluation proceeds by executing a C-BGP simulation for each prefix and then comparing the predicted AS path according to the AS-routing model with the actual observed AS path in the Internet. In this manner we can evaluate the predictive capabilities of the model. Since routing decisions are determined independently for each prefix, we run a separate simulation for each prefix.

After the simulation runs one has access to the routing information base (RIB) of all quasi-routers. Therefore, we can now compare for each AS the AS path that is recorded in the BGP data to the AS paths chosen in the simulation. Some mismatches have to be expected. We measure the degree of mismatch by determining if a route with the AS path is received by a quasi-router within an AS (RIB-In) if it is selected by a quasi-router (RIB-Out) or if it could have been selected but was not due to an “unlucky” decision in the last step of the BGP decision process, the tie-breaker (potential RIB-Out). More precisely we use the following metrics:

RIB-In match: The observed route at an observation point is contained in the simulated RIB-In for at least one quasi-router in the observed AS. Note, this does not say that the simulated and observed RIB-Ins are the same. as the observation point only sees the best routes advertised by the monitored AS. The metric provides an upper bound on the prediction accuracy — we can only expect a RIB-Out match if we have a RIB-In match. A RIB-In match is a necessary but not sufficient condition for a RIB-Out match.

Potential RIB-Out match: A RIB-In match where in the process of choosing a best route the observed route is eliminated in the last tie-breaking step of the BGP decision process in the simulation ("Lowest Neighbor IP address").

RIB-Out match: At least one quasi-router in the AS has selected the route with the observed AS path as its best route and propagates it to its neighbors.

Furthermore, we count for how many prefixes we find RIB-Out matches for at least 50%, 90%, or 100% of their respective unique AS paths.

To visualize the various possibilities, Figure 4.3 shows a toy example with 8 ASs, three observation points (at AS 1, AS 2, and AS 3), and one prefix p originated at AS 6. The dashed arrows[2] indicate the traffic flows along the observed AS paths while the dotted arrows indicate the paths chosen by the simulation. Consider first AS 1 — its RIB-In contains the learned routes `1-7-6` and `1-4-5-6` to reach AS 6. The path `1-7-6` is chosen instead of `1-4-5-6`, which has been observed in BGP data. This represents a RIB-In match but no RIB-Out match. Since the observed AS path is longer than the simulated path, the used policies are clearly wrong. Next, consider AS 2. Once again, we see that there is a RIB-In match (neighbor AS 8 propagates the "correct" suffix path to AS 2). But there is no RIB-Out match. In this case, the best path is chosen "wrongly" in the final BGP tie-break. We call this a potential RIB-Out match because the choice is made based on the tie-breaker. This mismatch is due to an unlucky decision in the simulation, rather than using incorrect policies. In real routing IGP weights, etc., are also used to break these ties. Finally, AS 3 has a RIB-Out match: simulation and observation agree for router 2 of AS 3.

4.3.3 Deriving an AS-Routing Model

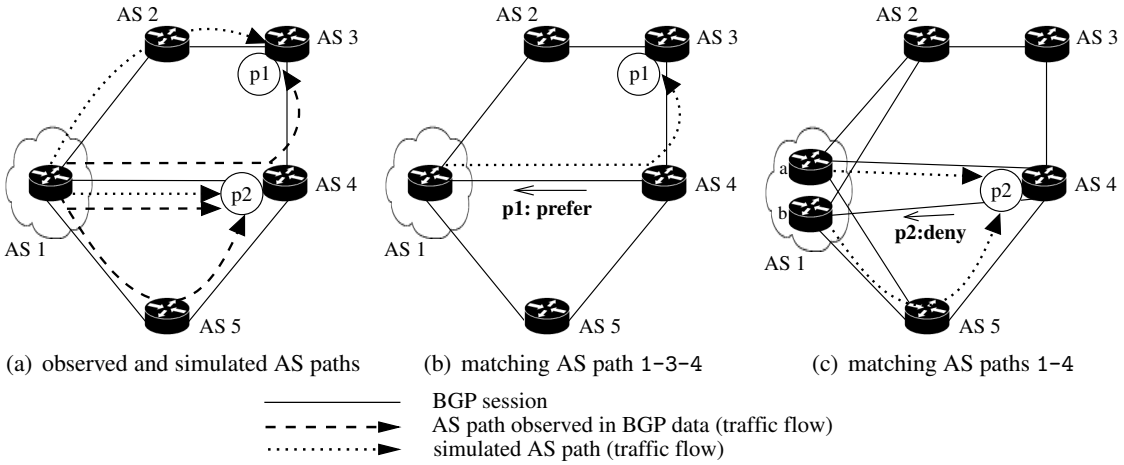
In this section, we introduce the details of our iterative approach for constructing an AS-routing model based on a `training` set of BGP data from multiple vantage points in the Internet.

We start from the simplest AS-model possible. It consists of one quasi-router per AS and contains one edge between any two connected ASs of the AS-level graph. Accordingly, this model only includes information that is easy to derive from the input data set. Then we determine for the `training` set, where the AS paths predicted by the current AS-routing model differ from those observed in the Internet (those in the `training` set). This can be due to two reasons: First, the model prefers the shortest AS path in the absence of more complex policies. Second, the quasi-routers inside an AS do not suffice to capture the required route diversity.

To reduce the discrepancies between the observed AS paths and those predicted by the model, we *alter* the model iteratively by either adding routing policies or quasi-routers. Adding quasi-routers enables us to propagate more than one best route to the next AS, a necessity as the data

[2] In all figures routes are directed according to the flow of traffic.

Figure 4.4 Example for model refinement: applying changes at AS 1 for prefixes $p1$, $p2$.



(a) observed and simulated AS paths (b) matching AS path `1-3-4` (c) matching AS paths `1-4`

analysis shows (see Section 4.2.2).[3] By adding policy rules we ensure that the appropriate AS path is selected and can be propagated, even though it may not be the shortest one.

We do not aim at inferring the actual policies used by the ASs. Rather, it is our goal to derive an AS-routing model where the simulated AS paths correspond to the observed AS paths for the `training` set. By doing so, we hope to, and indeed do, improve the predictive capabilities of the AS-routing model over the models discussed in Section 4.2.3. We are in this way capable of removing the limitations of the "one router per AS" model of the Internet.

In effect, each iteration of the heuristic, see Algorithm 1, consists of comparing the AS paths predicted by the model to those in the `training` data. Based on the results changes to the model (new quasi-routers or changes to the policies) are determined and the path propagation is re-simulated for all prefixes that are effected by the changes. This cycle is repeated until the desired level of agreement for the `training` set is achieved. In the following, we present more details about the initial model and how the iterative refinement proceeds.

4.3.4 Example: Refining an AS-Routing Model

Since any simple AS-routing model with just one quasi-router per AS is unlikely to match reality, we now illustrate with an example how to use routing policies and topology diversification to improve the model. Suppose there are five ASs, interconnected as shown in Figure 4.4 (a), with two prefixes $p1$ originated at AS 3 and $p2$ at AS 4, and one observation point at AS 1, which observes a route with AS path `1-4-3` for $p1$ and routes with paths `1-4` and `1-5-4` for $p2$. These AS paths are visualized via dashed lines. The AS paths currently chosen after a simulation run are paths `1-2-3` for $p1$ and `1-4` for $p2$ (dotted lines).

Starting with prefix $p1$, the heuristic detects that in the simulations the path `1-2-3` is chosen instead of the path `1-4-3` at AS 1. This mismatch is due to the fact that in our setup the quasi-router of AS 2 has a lower IP address than the quasi-router at AS 4. To correct this "wrong" tie-break

[3] Keep in mind that a quasi-router does not have to correspond to an actual router. It is just an entity responsible for routes.

decision, our heuristic sets up a policy at the quasi-router in AS 1 to prefer routes learned from AS 4 for prefix $p1$. We re-simulate, and now the path `1-4-3` is selected instead of the path `1-2-3` (see Figure 4.4(b)).

Next, consider the two AS paths observed for prefix $p2$ at AS 1. A route with the shorter AS path `1-4` is already selected by the quasi-router in AS 1; therefore no changes are required. Yet, in order to account for the AS path `1-5-4`, a second quasi-router inside AS 1 is needed. Therefore, a new quasi-router b is created as an identical copy of the existing quasi-router a with the same neighbors as quasi-router a (see Figure 4.4(c)). Thus, quasi-router b will have a RIB-In match for a route with AS path `1-5-4` but does not select it as best route (the AS path `1-4` is shorter). In order to correct this at router b of AS 1, two policy rules are used. A filter at AS 4 prevents routes for prefix $p2$ from being propagated to quasi-router b of AS 1 and a ranking policy is set to prefer routes for $p2$ announced by AS 4. This ensures that quasi-router b of AS 1 can select the route with AS path `1-5-4` as its best route.

4.3.5 Initial Model

To derive the initial model we use *all* available BGP feeds, `training` as well as `validation`, to derive an AS-graph from the AS path information. Such an AS graph is likely to be incomplete as it is probable that there are other peerings that are not used by any of the AS paths recorded at our vantage points. It is possible to further improve the coverage of the AS-graph by adding additional observation points or information from the routing policy database or traceroute data. Yet, as these additional data sources come with some uncertainties [SF04], we only focus on data from our observation points.

Initially, all ASs consist of a single quasi-router, and peerings are established according to the edges of the AS graph. Next, we assign IP address to each quasi-router. This choice is important as the IP address is used as the final tie-breaker in the BGP decision process. (In case of a tie a quasi-router prefers the AS paths announced by the quasi-router with the lower IP address.) Therefore, this choice can directly influence the quality of the prediction process. We choose to use IP addresses such that the high order 16 bits are set to the AS number and the low order bits are a unique ID for each quasi-router within the AS.

4.3.6 Iterative Refinement

The goal of the iterative refinement process, see Algorithm 1, is to modify the AS-routing model until one achieves the desired level of agreement between the predicted AS paths and the observed AS paths. Accordingly, we now introduce our *refinement heuristic*, which, by adding quasi-routers and BGP policy rules, reduces the discrepancies between the simulated and observed AS paths for a set of prefixes.

We have two main reasons for using an iterative process instead of trying to correct the discrepancies in a single step. First, route propagation itself is an iterative process. For the AS path of `1-2-3-4` from the origin (AS 4) to be observable at the observation point (AS 1), AS 3 first has to select an appropriate route and propagate it to AS 2. Then AS 2 has to select this route as its best one and propagate it to the observation point. To reproduce this step-by-step process in the AS-routing model, we move from the origin of the route towards the observation points and change the policies or the topology at the AS where the path chosen in the model differs from the one observed in the `training` set. The change ensures that the desired route is propagated one AS further

Algorithm 1 Routing models with predictive capabilities: methodology for model refinement

```
ITERATIVE REFINEMENT:
run simulations for prefixes p (initial model)
while not RIB-out match for all observed paths do
    apply heuristic (see below), compute changes
    restart simulations
end while

HEURISTIC (1 ITERATION):
for all prefixes p do
    for all ASs a do
        O = (suffixes of) AS paths for p observed at AS a
        for all o ∈ O do
            if RIB-Out match then
                no change
                mark the quasi-router as used
            else if RIB-In match then
                duplicate a quasi-router if necessary
                add policies: filtering, ranking (MED)
                mark the quasi-router as used
            end if
        end for
    end for
end for
```

towards the observation point in the next iteration. This is reasonable since this is a local decision and one does not have to determine how the changes influence the overall route propagation beyond the local changes. This task is delegated to C-BGP. Accordingly, our second motivation for the iterative approach is that we do not have to re-implement the full routing logic of C-BGP to determine the necessary changes to the AS-routing model. Note that it is not necessary to proceed AS-hop by AS-hop. Rather in each iteration one determines the AS which is closest to the originating AS with a discrepancy between the observed AS path and the selected best route and fixes this discrepancy at this AS.

In the following we first introduce our principle approach; then explain how the policies are adjusted; and finally how they may have to be corrected.

Refinement heuristic – principle approach:
The heuristic proceeds prefix-wise starting with the results of all C-BGP simulations runs for all prefixes of the respective `training` set based on the initial or previous AS-routing model. For each prefix p with AS path P of the `training` set and each AS a on the path it checks the following conditions and if necessary takes appropriate actions:

RIB-Out match: Condition: The observed path up to this AS (the suffix up to a) is selected as best route by at least one quasi-router inside the AS.

Action: We choose among this set of quasi-routers the one with the lowest quasi-router ID and mark/reserve this quasi-router as being responsible for this AS path and not available

for matching another observed AS path for the same prefix.

RIB-In match but no RIB-Out match: Condition: There is at least one quasi-router that learns the observed AS path up to this AS. But none of the quasi-routers has selected it as best route and none of these quasi-routers are already reserved for other routes for this prefix.

Action: We choose among this set of quasi-routers the one with the lowest router ID and mark/reserve it as being responsible for this AS path. Then, we adjust this prefix' BGP policy at this quasi-router by either adding filters or setting MED values as described below.

Condition: Same as above but all quasi-routers are already reserved for other routes for this prefix.

Action: In this case we choose to "duplicate" one quasi-router with a RIB-In match. The new quasi-router has the same neighbors and policies as the copied one to ensure that it also has a RIB-In match for the prefix p. Then the BGP policy for this prefix is adjusted as in the previous case.

No RIB-In match: Condition: No quasi-router at the current AS has learned a route with the observed AS path.

Action: No action as a route with an appropriate AS path first has to be propagated to this AS.

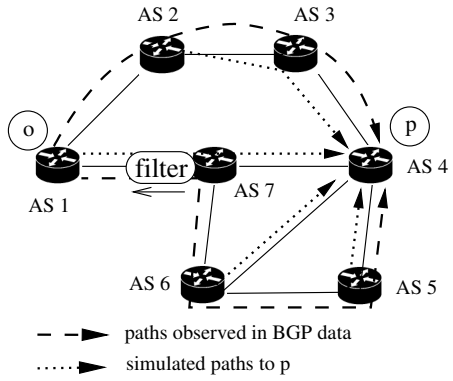
Refinement heuristic – policy adjustment:

Two ideas are central to our refinement process: First, new quasi-routers are added to account for path diversity. Yet, contrary to the routers in the Internet we do not establish iBGP sessions between the quasi-routers within an AS. Experiments with such an approach have shown that it is extremely difficult to control route selection, in particular to install different routes at neighboring iBGP routers. Therefore, we choose to use quasi-routers instead of routers. Each new quasi-router receives the appropriate routes by duplicating the BGP sessions to the neighboring ASs but remains isolated from other quasi-routers inside the AS. In effect, we short-circuit the intra-AS route propagation process. As a result each AS can consist of multiple separate quasi-routers, which do not exchange their reachability information.

Second, we use policy rules on a per-prefix basis to filter and rank routes at each selected quasi-router such that the route with the desired AS path can be selected as the best route. Suppose that a quasi-router learns a route with the correct (suffix) path for a certain prefix, yet it does not select it as its best route (RIB-In match but no RIB-Out match). This can happen at any one of the steps in the BGP decision process, see Figure 2.2. At the same time this multi-step decision process provides us with many different ways in which we can change the decision: by either adding a policy at the current quasi-router or through a filter at the announcing neighbor which ensures that a route is no longer available at the current quasi-router. At this point our goal is not to infer the specific routing policy used by the AS. Rather we want to account for all possible weird routing policies.

The first step in the decision process is based on the BGP attribute *local-pref*. It has been shown in [GSW99] that the preference of routes with longer AS paths over those with shorter ones can lead to divergence. Attempts to use *local-pref* for building our routing model resulted in divergence problems which are very hard to debug. Therefore, we choose to not rely on this attribute. Rather we use BGP filters to ensure that routes with shorter AS paths than the route we are looking for are not propagated to the current quasi-router. This is achieved by setting a filter policy for this prefix at

Figure 4.5 *Refinement heuristic*: necessity of filter deletion.



the announcing neighbor. To avoid further reduction of route diversity, we do not filter those routes that have the same AS path length as the one we are looking for. Instead, we take advantage of the next step in the BGP decision process that relies on the *MED* attribute. If two routes have the same local-pref and the same AS path length, the one with the lower MED value is selected. We assign a lower MED value to routes announced by the AS from which the observed AS path is learned. We require that MED values are always compared during the BGP decision process, even for routes learned from different neighbor ASs. Since quasi-routers inside an AS are not connected in our model, no iBGP divergence can arise [GW02]. Simply changing the ID of the router does not work as this will affect all routes.

It should be noted that our choice of BGP policies - filtering and MED values - is arbitrary and in general does not correspond to the policies actually used in the Internet. In the Internet, *local-pref* is often used to implement business relationships and for traffic engineering. Yet, prioritizing AS paths via MED is also not uncommon, as MED allows the realization of cold-potato routing [FMR04]. However, as noted above we are not concerned about reverse engineering real policies: rather we aim at understanding the impact of routing policy on route diversity.

Refinement heuristic – filter deletion:
If one could process all AS paths in a single step, it would be easy to determine when an AS needs multiple quasi-routers to propagate AS paths of different length. Using our iterative process this is not possible. It can happen that a filter is set while processing the "shorter AS path", which stops the "longer AS path" from being propagated. This filter has thus to be removed in a later iteration.

In Figure 4.5 observation point AS 1 observes two routes with AS paths `1-2-3-4` and `1-7-6-5-4` for prefix p, originated by AS 4. Neither of the two AS paths is selected as best route when simulating the initial model. The quasi-router at AS 1 chooses a route with AS path `1-7-4` to reach prefix p (dotted arrow from AS 1 to AS 4). However, the heuristic detects a RIB-In match for `1-2-3-4` at AS 1 during the first iteration. To prevent the shorter AS path `1-7-4` from being propagated to AS 1, a filter at the egress of AS 7 to AS 1 is set. Restarting the simulations results in a RIB-Out match for AS path `1-2-3-4`.

With regards to the second AS path `1-7-6-5-4`, the quasi-router at AS 7 does not select the correct suffix as best path until a later iteration. However, when it does select it as best route, it cannot propagate it to its neighbor AS 1 due to the egress filter set during the first iteration. As

a consequence, AS 1 does not learn the observed AS path `1-7-6-5-4`. When we do not find a RIB-Out or RIB-In match for a suffix of an observed AS path, we check for a RIB-Out match at all announcing neighbor ASs. Provided that there is a RIB-Out match at this AS, we remove any filter rule that prevents the propagation of the observed AS path towards the observation point.

The removal of the filter in Figure 4.5 leads to the creation of a new quasi-router at AS 1 for a route with AS path `1-7-6-5-4`. After the next iteration the route with this path is selected as best route by AS 1 and the above problem is circumvented and progress is ensured and no cycles will occur. Perfect RIB-Out matches are achieved after a total number of iterations that is a multiple of the maximum AS path length.

4.3.7 Using the AS-Routing Model for Predictions for other Prefixes

At this point we can use the AS-routing model derived from a `training` set as input to the C-BGP simulator and predict likely AS path choices for the prefixes of the `training` set to previously not considered observation points.

But as the policies are determined on a per-prefix basis, it is unclear so far how to take advantage of the AS-routing model for predicting AS paths for prefixes that are *not part* of the `training` set but for which we have AS path information for some observation points. One approach is to use multiple iterations of the *refinement heuristic* with the drawback of ignoring the routing policy information accumulated in the AS-routing model derived from the `training` set.

To overcome this limitation we introduce the *reuse policy heuristic*. The key assumption behind this heuristic is that most ASs specify their policy rules on a per-peer basis — reflecting the economic relationship between peering ASs — and not on a per-prefix basis. Accordingly, independent of the success of this heuristic, we can improve our understanding of the correlation between routes for different prefixes, i.e., whether different prefixes are treated equally or differently by the policies within an AS. In the following, we explain *from where* and *which* policy rules are reused.

In order to determine from where policy rules are transferred we again proceed prefix by prefix. For each of the new prefixes and observation points we have an observed AS path o. For each such AS path the *reuse policy heuristic* tries to find an AS path a that satisfies the following conditions:

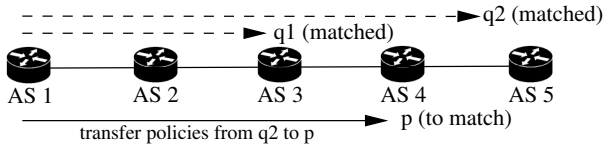
1. The AS path a is part of the `training` set, i.e., the input to the *refinement heuristic* and the AS-routing model shows a RIB-Out match for a.

2. Both AS paths end at the same observation points, i.e., the first ASs of both AS paths are identical. Furthermore we require that both AS paths share at least the first two edges. The underlying assumption is that the policies applied for the "new" prefix are the same as for the "old" prefix.

3. There is no other AS path x that satisfies the first two conditions and that is longer than a.

The example shown in Figure 4.6 illustrates this process for a simple topology that consists of five ASs. AS 1 is again our observation point. Prefix p is originated by AS 4, $q1$ by AS 3, and $q2$ by AS 5. We assume that the AS paths for prefixes $q1$ (`1-2-3`) and $q2$ (`1-2-3-4-5`) result in RIB-Out matches after using the *refinement heuristic* to derive an AS-routing model. The goal is to find a sensible policy for prefix p with AS path `1-2-3-4`. Since both AS paths (`1-2-3` and `1-2-3-4-5`) satisfy the first two conditions, the longer path is selected. In the absence of this AS path the shorter one would have been chosen.

Figure 4.6 *Reuse policy heuristic*: transfer of policies across prefixes.



Policies that allow the propagation of AS path `1-2-3-4-5` are likely to ensure the propagation of the similar path `1-2-3-4`, too. The underlying assumption is that policies in the Internet are generally specified for complete BGP sessions (neighbor-basis) and not on a per-prefix basis.

In the example of Figure 4.6 we transfer policies from the sub-path `1-2-3-4` of `1-2-3-4-5` to the current one for p. If there is a policy (MED, filter) for prefix $q2$ along `1-2-3-4`, it is converted into a policy rule for prefix p. In contrast to the *refinement heuristic*, no new quasi-routers are added.

4.4 Results

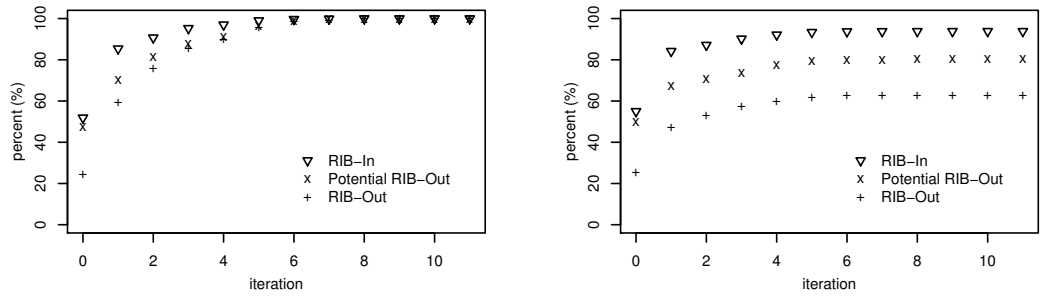
In this section we evaluate the *refinement* and *reuse policy heuristics* by using them to derive an AS-routing model for various sets of `training` data, and evaluate their effectiveness using separate `validation` data.

Data:
Of the $1,300$ BGP observation points, see Section 4.2.1, we randomly assign $2/3$ to the `training` set and the remainder ones to the `validation` set. We sub-select the AS path information from $1,000$ ASs and their corresponding paths from both the `training` and the `validation` sets to derive our base AS-routing model. In order to ensure a reasonable coverage of the AS-graph, we include all level-1 ASs as well as randomly selected ASs of the groups level-2 and other. We refer to this set of prefixes and their AS paths as `psetA` . To evaluate the effectiveness of the *reuse policy heuristic*, we select two other disjoint sets of prefixes and their AS paths in a similar manner. These sets, referred to as `psetB` and `psetC`, again consist of $1,000$ randomly chosen prefixes.

Training:
The inference of the AS-routing model uses an iterative process that incrementally refines the model with the goal of achieving an exact match between the AS paths predicted by the model and the `training` set. Figure 4.7(a) shows the progress of the heuristic with each iteration as measured in terms of RIB-In matches, potential RIB-Out matches, and RIB-Out matches. The length of the longest AS path is 10, and 11 iterations happen to suffice to achieve our goal of perfect RIB-Out matches. Notice that the early progress of the heuristic is excellent. Just one iteration more than doubles the percentage of RIB-Out matches from 24.5% to 59.3%, and increases the potential RIB-Out matches and RIB-In matches to more than 70% and 85% respectively. Given that the average length of the AS paths is about 4.3 it is not surprising that we achieve RIB-Out matches for all but 5% of the AS paths after five iterations

Further inspection of the data reveals that matching the AS paths for some prefixes and some observation points requires more policy adjustments than for others. After the fifth iteration we start to see a significant number, 238 out of the $1,000$ prefixes with RIB-Out matches for all observation

Figure 4.7 *Refinement heuristic*: results



(a) Progress for `training` (`psetA`)

(b) Progress for `validation` (`psetA`)

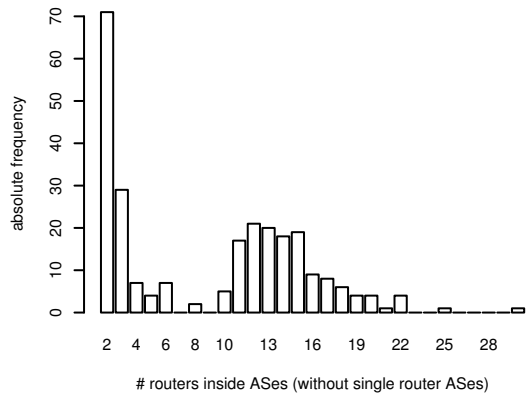
points. This number increases with the next iterations via 481 and 683 to 969 after the eighth iteration. This means that at this point we only have a very small percentage of unmatched AS paths. Note that if we do require RIB-Out matches for 90% of the AS paths for each prefix, already more than 40% of the prefixes satisfy this condition after two iterations. For the other two subsets of prefixes, `psetB` and `psetC`, even faster improvements are observable

Validation:
Given an AS-routing model we can now evaluate its predictive capabilities for our example question for a different set of observation points. We find based on the subset of `validation` for `psetA` that we improve our prediction capabilities from 25.5% (without routing policies) to 63% for RIB-Out matches, and if we ignore the final tie-breaking rule of the decision process, from 50% to more than 80% (see Figure 4.7(b)). For RIB-In matches we see an improvement from 55% to 93%. Let us point out that the major improvements happen during the first six iterations.

To judge the qualitative improvement of our results vs. those reported by Mao et al. [MQWZ05], we point out that our results hold across more than 300 observation points rather than 3 ASs and are significantly better. In terms of RIB-Out matches, which correspond to *exact matches*, we have 63% vs. their 35%, 10%, and 3%; in terms of RIB-In matches, which correspond to *matches*, we have 93% vs. their 82%, 64%, and 16%. To better compare the results we focus on the same three ASs (7018, 2152, 8121) as Mao et al., moving them from the `training` to the `validation` set and rerunning the *refinement heuristic* for 200 prefixes of `psetA`. We find perfect RIB-In matches for AS 7018, almost perfect matches for AS 2152, and 44.2% for AS 8121.

Characteristics of the inferred AS-routing model:
The main reason for the effectiveness of our *refinement heuristic* is that an AS can, if necessary, consist of multiple quasi-routers. This raises the question how many of these quasi-routers are needed. Figure 4.8 shows a histogram of the number of quasi-routers per AS for `psetA` (for those ASs with more than one quasi-router). For almost all ASs (14,305) one quasi-router suffices. For 71 we need two. Yet, there are 138 ASs that need more than 9 quasi-routers. Note, not all of these quasi-routers are needed for all prefixes. Not surprisingly, we find that our level-1 ASs are among the ASs with many quasi-routers. After all, level-1 ASs offer peerings at quite a number of peering locations, peer with and provide service to many other ASs, and have a sizable backbone network. Most of the others are level-2 ASs, but there are also some in the "other" group. The average number

Figure 4.8 *Refinement heuristic*: number of quasi-routers per AS (ignoring 14,305 ASs with one quasi-router)



of quasi-routers for level-1 ASs is 17.1 while for all other ASs it is 1.03.

Effectiveness of the *reuse policy heuristic*:
Given an AS-routing model we can now evaluate its predictive capabilities for our example question for a different set of prefixes. We find, based on the subset of `training` for `psetB` (`psetC`), that we improve our prediction capabilities by almost a factor of two from 24.6% (23.7%) (without routing policies) to 46.6% (45.5%) for RIB-Out matches. If we ignore the final tie-breaking rule of the decision process, the number of Potential RIB-Out matches increases from 48.3% (46.7%) to 59.0% (58.3%). This implies that the assumption that policies are well captured by the AS-routing model and that the *reuse policy heuristic* can take advantage of this capability. In addition it shows that some of the policies are applicable on a per-peer basis. Yet, as we do not get perfect matches one should restrain from over-generalizations.

The *reuse policy heuristic* gives an AS-routing model for `psetB` and `psetC` which utilizes the results of `psetA`. If one wants to derive an AS-routing model for `psetB`/`psetC`, one can either start from the `psetA` AS-routing model or start from scratch using the *refinement heuristic*. With regards to potential RIB-Out matches we have for `psetB` after the *reuse policy heuristic* 46.6% vs. 24.6% for the initial model, after the first iteration 65.8% vs. 55.3%, and after the second iteration 83.2% vs. 81.0%. We note that the additional information from `psetA` helps both in terms of progress during the early iterations as well as with regards to the predictive capabilities after the initial iterations. Yet, of course the AS-routing model that utilizes the *reuse policy heuristic* is based on a much larger knowledge base that has needed significant computation time to derive.

Revisiting the effectiveness of the *refinement heuristic*:
Even though the heuristics are very effective in terms of predicting AS paths we also need to investigate how and why they may fail. First, we point out that it is quite possible for an AS path to be contained both in the `training` as well as the `validation` set. This can occur since we have multiple observation points in some ASs. We happen to have at least one observation point in both sets for 168 ASs. As a result, we find that 22.2% of all AS paths for `psetA` are in both. For the subset of these AS paths in the `validation` set we have RIB-Out matches by design, in this case for 49% of the paths in the `validation` set of `psetA`.

Table 4.3 Effectiveness of *refinement heuristic* by uncovered AS-edges.

Path class # uncovered	# paths	RIB-In	Pot. RIB-Out	RIB-Out	Not found
0 AS-edge	9,486	90.0%	59.1%	43.3%	10.0%
1 AS-edge	43,823	86.3%	74.2%	25.9%	13.7%
2 AS-edges	5,624	34.2%	26.3%	6.6%	65.8%
3 AS-edges	45	0.0%	0.0%	0.0%	100.0%

If the AS path is not contained in both the `training` and the `validation` sets, then, even though we hope to have configured the appropriate policies, the heuristics may have failed to propagate the route along the observed AS path. A route may not be propagated in the AS-routing model either due to another policy decision or due to a missing policy. Given that we do not have training data from all ASs we have to predict policy choices to some degree, especially if the data used to derive the AS-routing model (`training` set) does not include an AS path for all AS-edges. We refer to such AS-edges as uncovered.

Table 4.3 shows the effectiveness of the *refinement heuristic* by uncovered edges, considering only AS paths contained in the `validation` set but not in the `training` set of `psetA`. We find that the majority of these AS paths contain at most one uncovered AS-edge. No AS path includes more than three uncovered AS-edges. Of these uncovered AS-edges almost all, 93.4% to be precise, occur next to the observation point.

As the number of uncovered edges on an AS path increases, the likelihood of achieving a RIB-In match for these paths decreases. For AS paths with 0 uncovered edges we have 90.0% RIB-In matches. For paths with 1 uncovered edge we get 86.3% RIB-In matches, which further decreases to 34.2% and 0% for 2 and 3 uncovered AS-edges, respectively. Similar observations hold for potential RIB-Out and RIB-Out matches. As shown in Table 4.3, the percentages of RIB-In matches do not decrease as fast as the ones for RIB-Out matches when there are uncovered AS-edges on the path. This is not surprising as RIB-In matches are not as sensible to the specific policy choices. These results agree with the intuition behind Mao's et al. [MQWZ05] approach. If one can determine the first hop then the uncertainty about the remaining AS path is reduced.

In order to predict how good our results can be for arbitrarily chosen observation points, we randomly select a set of $6,000$ ASs and compute for all possible originating ASs how many uncovered AS-edges one may have to predict. We find that for 0.7% there is no uncovered edge. For 54.8%, 36.7%, 6.6% there are one, two, and three, respectively. Only 1.2% require more than three. These numbers are a bit more pessimistic than for our data sets. Yet, as shown above we can expect reasonable results, when just one or two AS-edges are uncovered.

4.5 Related Work

Improving our understanding of network topology has been a topic of huge interest over the last few years [HRI+08]. A lot of attention has been given to the dynamics of the BGP protocol, e.g., to understand why convergence time of BGP can be rather long [LABJ00, GW99, LMJ97]. Oscillations in BGP [BOR+02] can occur; see [MC05] for a review of their possible causes. Apart from the aspects related to the time required for BGP to converge, divergence anomalies as defined in [MC05]

are permanent failures of BGP to converge towards a stable path. Divergence anomalies stem from two typical causes: Conflicting eBGP policies [GSW02] and iBGP oscillations [BOR+02, FR09]. The first can be explained within a model that only contains one router per AS, the second cannot.

Further work has investigated the interactions of routing on traffic within an AS. Based on data gathered from the Sprint network [ACBD04] it was shown that the impact of external BGP events on its traffic matrix is limited. Still hot-potato disruptions [TSGR04, TDRR05] can have a significant impact on transit ASs. This highlights that routing dynamics can be complex, even when just looking at a single AS.

AS-level topology inference [Gao00, SARK02, BPP03] provides another dimension to the complexity of routing in the Internet. Routing policies are typically partitioned into a few classes which capture the most common practices in use today [CR05]. Unfortunately it is also known that the reality of routing policies [WG03] and peering relationships is far more complex than those few typical classes [CR05, CGJ+04]. Current approaches for AS-level topology inference rely on a top-down approach. They first define a set of policies and then try to match those policies with their observations of the system. Yet, policies as used by ISPs have to realize high-level goals [CR05]. Assuming any kind of consistency of such policies across ASs is questionable, especially as in practice policies are often configured on a per-router or per-peering basis [CR05]. This means that observed BGP routes do not even have to be consistent with the high-level policies of the AS.

Traceroute is one of the most widely used tools for discovering end-to-end paths and can be used in combination with other tools to derive AS-level paths [MRWK03, MJR+04]. These tools require network access. To overcome this problem Mao et al. [MQWZ05] proposed a first methodology for inferring AS-level paths based on presumed BGP routing policies. They find that the accuracy of the estimation depends on the precision of the AS relationship inference and the ability to incorporate additional information regarding the first hop.

4.6 Summary

In this chapter, we have made four fundamental contributions. Firstly, we show that AS topologies using a single router are limited – they cannot capture all kinds of route diversity in the Internet today. This is in contrast to the majority of prior work on modeling the large scale structure of the Internet. Secondly, we show how to use simulation-based heuristics to iteratively build an AS model that is consistent with *all* observed paths. Thirdly, we show that the model can be used to predict previously unobserved AS paths. The accuracy is in practice limited to around 63% due to policy decisions on parts of the network that we have not previously observed in our data, and tie-break decisions that are inherently hard to predict. Ignoring tie-breaks we reach an accuracy of more than 80%. Finally, our results provide some insights about the structure of the Internet.

5 Searching for an Appropriate Granularity to Model Routing Policies

We adhere to the objective of constructing routing models of the Internet with predictive capabilities. In the preceding Chapter 4 we described how to build AS-topology models that capture route diversity and that enable us to predict previously unobserved paths with high accuracy. For this purpose, we introduced the concept of quasi-routers and a simplified notion of routing policies. Evidently, we did not infer the actual policies used by ASs in the Internet, since we were mainly interested in deriving a model where simulated paths correspond to the set of observed input paths.

Contrary to Chapter 4, we now turn to a more precise analysis of actual routing policies. For any modeling, a crucial task is to find the right balance between what details are important and what details can be safely ignored. Before we can think about actual policy implementations to use in our model, we need to understand at which granularity routing policies have to be captured. This chapter takes the step of searching for the appropriate granularity at which routing policies should be modeled.

Routing policies are typically partitioned into a few classes that capture the most common practices in use today [CR05]. Unfortunately, it is known that the reality of routing policies [WG03] and peering relationships is far more complex than those few classes [CR05, CGJ+04]. After all, policies encompass the business and engineering decisions made by individual ASs, both commercial agreements (business relationships) and technical aspects (router configuration, inter-domain routing behavior, etc.).

In this chapter we aim to capture the appropriate level-of-detail about policies to be used in a model of the Internet. Our ultimate objective, which is not achieved yet, is to build a model of the Internet with a sufficiently detailed view of the AS-level connectivity and its policies so as to be able to have useful predictive capabilities regarding BGP paths. However, we need to understand at which granularity routing policies have to be reproduced before we can think about actual policy implementations to use in our model.

The de-facto standard model for routing policies are AS relationships, that assume that the business type of an AS-level peering is either customer-provider (c2p), peer-to-peer (p2p) or sibling (sib), see Section 2.5.1. Techniques have been proposed in the past that allow to a certain degree the inference of AS relationships [Gao00, SARK02, BPP03].

The general guideline for our search of the appropriate way to model policies in the Internet is to be agnostic: We do not rely on unnecessary assumptions, but let the data speak for itself. Rather, we rely purely on what we observe in BGP data and attempt to learn as much as possible about the "correct" level-of-detail needed to model actual routing policies. Our main concern is not to shrug off existing approaches but to pinpoint their advantages and disadvantages and how they are related to one another. To this end, we introduce "agnostic" policies since it is impossible to infer all of the details of an AS's policies without access to router configurations. However, in contrast to Chapter 4 we compare inferred policies with business relationships. The gained insights are important for our

study of the right granularity to model routing policies.

Our work reveals two dimensions to policies: (i) which routes are allowed to propagate across inter-domain links (route filtering); and (ii) which routes among the most preferred ones are actually chosen (route choice) and thus observed by BGP monitors. In terms of the first dimension we show that the granularity of business relationships is largely consistent with observed paths. AS relationships provide the right abstraction to prevent unnecessary paths from propagating in a model of the Internet. For the second dimension, however, the classes of neighbors defined by business relationships are not precise enough. When only business relationships are used as policies in a model of the Internet, there are still many possible candidate paths among which the best path can be chosen. Business relationships are not sufficient to determine among all possible valid paths, which one should be chosen as best by the model to be consistent with observed BGP data.

To crystallize the choice of paths an AS makes, we introduce a new concept: *next-hop atoms*. Next-hop atoms capture the different sets of neighboring ASs that each AS chooses for its best routes. We show that a large fraction of next-hop atoms correspond to per-neighbor path choices. A non-negligible fraction of path choices, however, correspond to hot-potato routing and tie-breaking within the BGP decision process, very detailed aspects of Internet routing.

The remainder of this chapter is structured as follows. Section 5.1 presents the AS-topology model, we use for this chapter. Section 5.2 analyzes the known bounds for policies studied in the literature. To search for the right granularity to model policies, we infer in Section 5.3 agnostic per-prefix filters. In Section 5.4 we introduce an abstraction to describe the granularity of observed path choice and thus of routing policies before Section 5.5 tries to find out whether AS business relationships suffer from systematic errors that cause the propagation of the "wrong" paths. The related work is described in Section 5.6. Finally, Section 5.7 concludes and discusses further work.

5.1 AS-Topology Model

To study the granularity of policies, we need a topology model of the Internet. For this purpose, we rely on the data of Section 2.4.2 and re-use the concept of quasi-routers as defined in Chapter 4. Section 5.1.1 reviews some properties of the AS connectivity observed in our data before we explain in Section 5.1.2 how the AS graph of our model is built from the observed paths in our data set of Section 2.4.2. Note that in any data analysis results of this chapter we do not consider stub ASs, i.e., ASs that appear as the last AS hop on any AS path in our data (pure originating AS)[1].

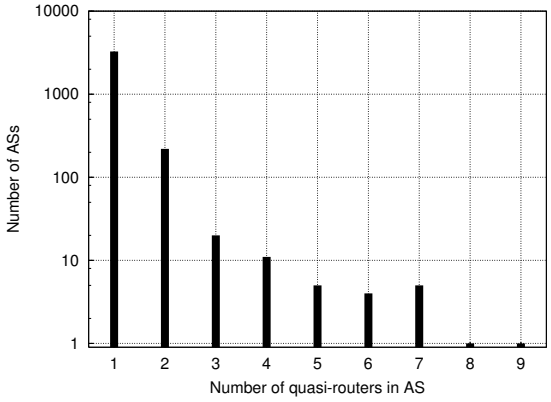
5.1.1 AS-Level Connectivity

As already shown in [MQWZ05, MFM+06], for an AS topology model to capture route diversity, ASs cannot be considered atomic entities. In order to represent intra-domain routing diversity, we allow each AS to consist of multiple quasi-routers, see Chapter 4. Based on our observed AS paths, we compute a lower bound on the number of required quasi-routers: If for a given AS we find x distinct AS paths or AS path suffixes towards the same origin AS, we reproduce the AS with x quasi-routers. Figure 5.1 provides this minimum number of quasi-routers per AS.

Among the 3,535 remaining ASs, 267 require more than a single quasi-router. Only 2 ASs need as many as 8 and 9 quasi-routers to account for their observed routing diversity. Typically, well-known tier-1 ASs require several quasi-routers. This is consistent with [TMSV03] which showed,

[1] Although being transit domains, some ASs may only have one AS neighbor after removing stub ASs.

Figure 5.1 Number of quasi-routers per AS that are required according to observed data.



based on active measurements, that tier-1 ASs have high path diversity. On the other hand, a low number of quasi-routers per AS is due both to the sampling of the available paths of the observed BGP paths, as well as the loss of BGP routing diversity inside ASs [UT06].

Diversity of the AS paths is strongly related to the AS-level connectivity. Figure 5.2, in which we consider the same 3,535 ASs as in Figure 5.1, shows a scatterplot of the relationship between the number of required quasi-routers and the number of neighboring ASs. We observe that ASs that do not have many neighbors also tend to have a small number of quasi-routers . Highly connected ASs on the other hand may have many quasi-routers although this is not necessarily always true. Some ASs have hundreds of neighbors, yet a single quasi-router is enough to account for their routing diversity.

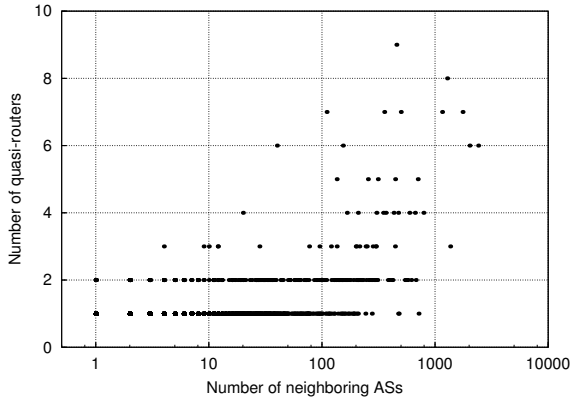
As previously stated, there are two reasons why an AS requires several quasi-routers: (i) the AS receives and selects as best multiple paths towards a given prefix from a given neighbor; and (ii) the AS receives and selects as best different paths towards a prefix but from different neighbors. From Figure 5.2 we can observe that highly connected ASs have a far larger number of neighbors than quasi-routers. ASs thus select a very small subset of best paths compared to the number of paths they may receive from their neighbors for any prefix. Note that the first reason why an AS might need several quasi-routers does not seem to be common. For only 623 pairs of neighboring ASs do we observe in the data that an AS chooses from one of its neighbors more than one path towards at least one prefix. Furthermore, in only 19 cases do we observe an AS receiving more than 2 distinct paths from a given neighbor towards at least one prefix.

5.1.2 Building a Quasi-Router-Level Graph

For our study of the granularity of routing policies we need a topology model of the Internet. We capture the inter-domain connectivity via quasi-router AS-topology graph as extracted from the BGP data (see Section 2.4.2).

There are two border cases regarding the connectivity of quasi-routers between neighboring ASs A and B: Either we insert a link between *all* pairs of routers a, b, where a is in AS A and b is in AS B (*maximum connectivity*) or we use *minimal connectivity* between the quasi-routers of two neighboring ASs. In principle, the former option, aligned with our simulation-based iterative refinement

Figure 5.2 Relationship between number of neighboring ASs and number of quasi-routers.



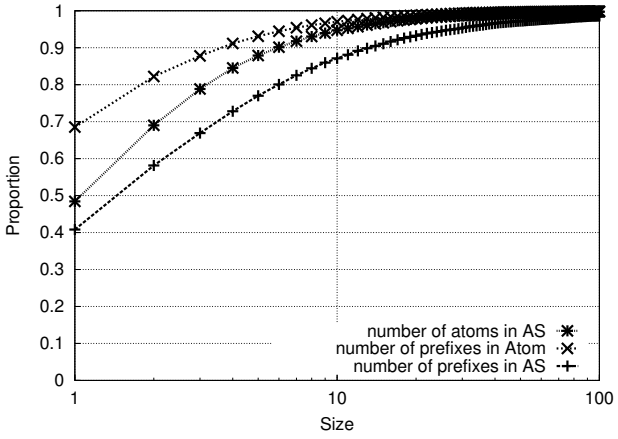
heuristic of Chapter 4 enables us to match observed paths within a routing model and to a certain degree even allows to predict AS paths. Yet, such an approach potentially propagates more paths than necessary.

Contrary to Chapter 4, we decide to rely on minimal connectivity between the quasi-routers of two neighboring ASs. Hence, the topology is built when assigning to quasi-routers AS path suffixes observed in the data. A suffix s of an AS path P is any substring Q such that $P = Qs$. The AS topology we create has as few quasi-routers per AS as possible, and an edge exists between two quasi-routers if some suffix has to be propagated between the two quasi-routers.

Our assignment works on a per-prefix basis. First, we set all quasi-routers as free to be assigned paths towards the considered prefix and set all suffixes towards this prefix as to be assigned. Then, as long as there are suffixes that are not assigned, we try to assign them by starting with those suffixes closest to the originating AS(s) of the prefix. When assigning suffixes, we first re-use existing connectivity between quasi-routers. If no link between the first two ASs on the suffix can be re-used for this prefix, we then create a new link between a free quasi-router in the first AS on the suffix and the next AS. Note that the creation of the topology (links between quasi-routers) follows directly from the path assignment.

The number of necessary quasi-routers in an AS is not the only parameter that matters for allowing an AS topology model to reproduce the paths observed in BGP data. If an AS requires the same number of inter-domain links as it has neighboring ASs, it means that even though this AS might have many neighbors, only a single neighbor at a time is used as next hop AS in the best routes for any prefix. If an AS in our model has substantially more inter-domain links than neighboring ASs on the other hand, it means that the considered AS uses several neighboring ASs for its best routes towards some prefixes. $3,150$ among the $3,535$ transit ASs of our data require a single inter-domain link with any of their neighboring ASs. Only 386 ASs require more than one inter-domain link per neighbor, and 41 ASs more than 2 inter-domain links. As seen from BGP data, only a very small fraction of the ASs choose their best paths from several neighbors at the same time towards any of their prefixes.

Figure 5.3 Atoms structure for dataset of Section 4.2.1.



5.2 Bounds on Policy Granularity

To find an appropriate way to model policies in the Internet, it is important to start with realistic bounds that define the finest and coarsest granularities at which policies are applied in the Internet. There are two ends to this spectrum. The finest granularity is the one of BGP atoms [ABSBB02, BC01a], which are sets of prefixes originated by a given AS that receive equivalent treatment by routers in the Internet. BGP atoms are as fine as the set of policies that the observed BGP paths encounter, which can be as fine as on a per-prefix basis. The coarsest granularity does not depend on the originated prefixes, but only on the neighbors from which routes are received. It is the granularity of business relationships, see Section 2.5.1. ASs may configure policies as coarse as per-neighboring AS, hence treating all prefixes, received from a given neighbor, in the same way.

5.2.1 BGP Atoms

BGP atoms [ABSBB02, BC01a] are defined as groups of prefixes (originated by a given AS) that receive *equivalent treatment* by a set of BGP routers. As BGP monitors see only a sample of the outcome of routing policies through observed AS paths, not the policies themselves, a BGP atom is defined as a set of prefixes that share the same set of AS paths, as seen from a set of BGP routers [BC01a]. Two prefixes belong to the same BGP atom if their `AS-PATH` is the same, as seen by all observation points. The finest granularity of a BGP atom is a single prefix, whereas the coarsest is all the prefixes originated by an AS (or a set of origin ASs in the case of MOAS prefixes [ZPW$^+$01]).

Figure 5.3 presents the atoms structure of our dataset (see Section 4.2.1). The graph displays three cumulative curves: the number of prefixes per origin AS, the number of prefixes in each atom, and the number of atoms in each AS. We observe that the distribution of the number of prefixes per origin AS is virtually the same as in [ABSBB02], which relies only on RIPE NCC data. More than 40% of the origin ASs advertise a single prefix: at least 40% of the atoms hence consist of a single prefix. However, the curve giving the number of prefixes per atom shows that 30% of the atoms consist of more than a single prefix. For us this is an indication that policies in the Internet are applied at a granularity coarser than per-prefix.

We believe that we observe finer atoms in the data compared to [ABSBB02] because our data

from Section 2.4.2 provides a more extensive coverage of the actual BGP paths. With an increasing number of paths, we also observe the effect of more policies, leading to smaller atoms due to more diverse path choices. The curve in Figure 5.3, showing the number of atoms in each AS, confirms that data used in this study sees ASs that have more atoms than that observed using RIPE. In our data, about 30% of the ASs contain two or more atoms, whereas [ABSBB02] observed only slightly more than 25% of ASs with two atoms or more.

The problem of BGP atoms is that their computation depends on the used data source: Two prefixes are put in the same atom if their `AS-PATH` is the same, as seen by *all* observation points. Hence, atoms are biased by the sampling of the Internet AS-level topology. Furthermore, BGP atoms actually do not describe routing policies but rather the outcome of BGP propagation and selection. Therefore, BGP atoms for example do not capture situations where a large fraction of the policies in the Internet are defined as coarse as per-neighboring AS while only a small subset of ASs configure policies on a per-prefix level. Concluding from a high number of BGP atoms for a given origin AS that the majority of routing policies is fine-grained, therefore, could be wrong.

5.2.2 Business Relationships

The most popular model for routing policies are *AS business relationships*, [Gao00, SARK02, BPP03] which rely on a coarse-grained notion of routing policies. They assume that there is a unique business contract negotiated between any two ASs, see Section 2.5.1, and define BGP policy and filter rules on a per-neighbor basis. For example, BGP route manipulations and filters are used to ensure that customer routes are preferred over provider routes and that an AS does not propagate routes received from a provider or peer-peer neighbor to any other provider or peer-peer neighbor.

However, business relationships may be inadequate if path choices are made on a finer granularity. After all, the BGP decision process selects best paths on a per-prefix and not on a per-neighbor basis. Therefore, we will in the following adopt an "agnostic" approach to find an appropriate granularity to model routing policies.

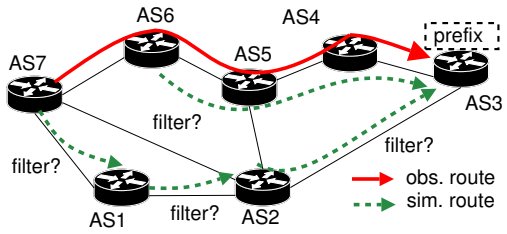
5.3 Inferring Agnostic Per-Prefix Filters

Since we believe that neither BGP atoms nor business relationships give an ultimate answer to the problem of which granularity should be used for modeling routing policies, we now start our search from scratch and rely on the finest granularity possible: per-prefix filtering. We identify fine-grained policies by analyzing what we see in our data of Section 2.4.2 and by comparing it to the routes selected in our topology model, see Section 5.1, without implemented policies. The motivation behind this approach is to compare the obtained per-prefix filters with coarse-grained policies as imposed by inferred business relationships, see Section 5.5.

As it is impossible to extract information about the implementation of policies only based on observed BGP data, we restrict ourselves to per-prefix filtering: if there is disagreement between some observed path and the corresponding route selected in our model, a set of filter rules is identified to prevent the propagation of "wrong" paths. Section 5.3.1 explains in detail how sets of filter policies are computed.

Section 5.3.2 tries to estimate the amount of freedom we have in terms of equivalent policies when trying to achieve consistency between best routes in our model and observed AS paths. In Section 5.3.3 we make an important step in our search for the appropriate granularity of policies.

Figure 5.4 Inferring candidates for filtering policies - Example.



Given a large set of per-prefix filtering rules, which has been computed exclusively based on observed data, we try to find out if there is possible aggregation across prefixes. More precisely, we check whether there are locations on the AS connectivity graph that seem to benefit more frequently from a filter than others. As we detect some very popular locations for filtering in Section 5.3.3, we conclude that the implementation of actual routing policies is somewhere in-between per-prefix and per-neighbor policies.

5.3.1 Inferring Filters

We now describe how to identify sets of per-prefix policies in order to obtain agreement between the routes selected in our model and those observed in the data. The guideline in this approach is to rely only on what we see in the data. We account for this basic principle as follows: First, the physical connectivity of our AS topology of Section 5.1 is sufficient to make the propagation of all observed AS paths possible if policies are to be installed properly. Second, policies are introduced on a per-prefix basis, the finest granularity for which policies can be configured. Third, we want to make as weak assumptions as possible about where to place a policy. If an observed AS path is not selected in the topology model of Section 5.1, we have a large choice about where and what policy to introduce. Different policy types and different AS-level peerings may have the same effect in terms of path propagation for the observed AS path. Therefore, we try to identify multiple "candidate" policies first and in a later step we will use heuristics to pinpoint likely policies.

The example in Figure 5.4 illustrates the many possible locations for policies if the only goal is to allow for the propagation of an observed AS path. We observe at AS 7 an AS path `7-6-5-4-3` originated by AS 3. However, reproducing the BGP route selection in this topology without any policies will show that AS 7 selects the shorter path `7-2-3` to reach the prefix. In this case, a preference policy at AS 7 or filtering at least one link both of the paths `7-2-3` and `7-1-2-3` will have the same effect in terms of the propagation of the observed AS path `7-6-5-4-3`. Note that it is even possible to apply an arbitrary subset of all "candidate" policies that will have the same effect.

In order to reproduce BGP route selection in our AS-level topology, we use C-BGP, see Section 2.6. As a consequence we know for every router in our model which routes it learns to reach a prefix and also which route is selected as best. According to the assignment of observed AS path suffixes to quasi-routers in our AS-topology graph (see Section 5.1), many of the routers in our model are supposed to select a specific path to reach a certain prefix. However, without properly configured policies the paths chosen by our model might be different than those observed in the data. A *mismatch* is referred to a situation where a quasi-router chooses an AS path which is inconsistent

with the path assignment of Section 5.1. In our approach, each mismatch gives a hint about where policies are required. We now distinguish between two different cases of mismatch.

The first case of mismatch can occur when a router does not select the path consistent with the assignment of Section 5.1, due to the existence of some shorter AS paths. In this case, we will introduce per-prefix BGP *filters* on the link from the announcing neighbors to prevent the shorter paths from being propagated to the router. In Figure 5.4, both AS 1 and AS 2 will propagate routes to AS 7 which are shorter than the observed AS path `7-6-5- 4-3`. In the following, we denote a filtering rule in our model between AS X and AS Y, where Y does not propagate a prefix towards X, by $X \nleftarrow Y$. Thus, configuring the filter rules $7 \nleftarrow 1$ on link `7-1` and $7 \nleftarrow 2$ on link `7-2` can be used to obtain the observed path at AS 7.

The second case will occur if a router does not select the "correct" AS path due to a wrong "tie-breaking" decision in our model. Provided some router receives multiple routes with equal AS path lengths, the BGP decision process will have to break ties, e.g., by preferring the route learned from the neighbor with the lowest IP address. We ignore those situations since no policy is identified. Indeed, we cannot be sure whether a policy is actually needed to get the correct propagation. We do not want the uncertainty of the BGP decision process and its implementation to impact our study of the granularity of policies. Reconsidering the example in Figure 5.4, we see that AS 5 may not select the observed suffix `5-4-3` due to a "wrong" tie-breaking decision: the C-BGP simulation will prefer the path `5-2-3` if the router of AS 2 has a lower IP address than the router of AS 4.

Let us now define three notions that will be used to explain the detection of filtering policies:

Candidate filter: A per-prefix filter rule which helps to allow the selection of an observed path as best route in our model. In general, several candidate filters (e.g., a filtering combination) will be needed. Additionally, shorter paths do not necessarily have to be filtered at the location of the mismatch. To obtain the observed path at AS 7 in Figure 5.4, filtering on the link `2-3` has the same effect as having filters on both `7-1` and `7-2`. Altogether, we identify four candidate filters in our example: $7 \nleftarrow 2$, $7 \nleftarrow 1$, $1 \nleftarrow 2$, $2 \nleftarrow 3$.

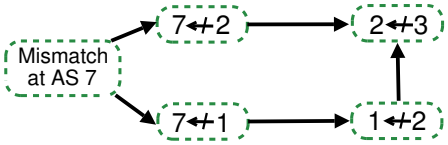
Filtering combination: A set of filter rules for a mismatch which satisfies two conditions: (i) Applying *all* filters in this set clears the mismatch, i.e., there will be agreement between the observed suffix path assigned to a quasi-router, and the route currently selected in our model; and (ii) the set of policies in this set is *minimal*, i.e., if *any* policy from a filtering combination is removed the mismatch will not disappear. In the example in Figure 5.4, there are three filtering combinations:

(1) $7 \nleftarrow 2$ and $7 \nleftarrow 1$
(2) $7 \nleftarrow 2$ and $1 \nleftarrow 2$
(3) $2 \nleftarrow 3$.

However, the set of filter rules $7 \nleftarrow 2$, $7 \nleftarrow 1$ and $1 \nleftarrow 2$ is not considered as a filtering combination, as either $7 \nleftarrow 1$ or $1 \nleftarrow 2$ can be removed while the router of AS 7 still chooses the assigned suffix `7-6-5-4-3`.

Dependency graph: A data structure used to store the identified filter candidates and their dependencies for a certain prefix. Nodes in this graph represent candidate filters or mismatches whereas directed edges between nodes reflect dependencies. The direction of the edges is determined by our algorithm. Basically, the algorithm recursively walks back from the "mismatched AS" to the originating AS, detecting filters along the way. Dependency edges are

Figure 5.5 Dependency graph for mismatch in Figure 5.4.



always directed towards filters which are closer to the originating AS. The idea now is that a "filter node" is not needed provided that (i) *all* its children nodes or (ii) *all* its parent nodes in the dependency graph are used or (iii) if there are no parent and children respectively. Figure 5.5 shows the dependency graph for the mismatch at AS 7 in Figure 5.4. There are five nodes, with one representing the mismatch at AS 7 and the remaining nodes the four candidate filters. Assume that filter 2↚3 is used. In this case, 7 ↚ 2 as well as 7 ↚ 1 are redundant, with all their children nodes (filter 2 ↚ 3) already used.

The benefits of this data structure are two-fold. First, compared to keeping all filtering combinations, the dependency graph scales as its size is bounded by the number of links and ASs in our topology. Second, it prevents losing information about possible dependencies between detected filters rules.

Our algorithm to compute a set of candidate filters for a given prefix is summarized in Algorithm 2. It takes as input the observed routes to a specific prefix, an AS topology including the assignment of observed AS paths to quasi-routers (see Section 5.1.2), and the routes selected in our model when simulating BGP route propagation with C-BGP. For each mismatch, a set of candidate filters is identified and inserted into the dependency graph. The result is a dependency graph for the prefix, with all candidate filters being associated to at least one mismatch.

As shown in Algorithm 2, the algorithm proceeds by consecutively looking at all routes observed for the prefix. For each route, we walk along the AS path from the originating AS to the observation AS and check at each hop for an inconsistency, i.e., a disagreement between the observed suffix path and the simulated route at the assigned quasi-router. If there is a mismatch, the function *findCandidates* is called recursively to identify filter candidates.

The recursion serves the purpose of considering filters that are not directly located at the mismatch but closer to the originating AS. The basic idea is that the function *findCandidates* has as a parameter the current AS hop h and recursively calls itself on neighboring ASs from which it learns routes that are too short in terms of AS path length.

To know which routes need to be filtered, we use another parameter l, the maximum path length which an AS is allowed to propagate. Provided that AS h of *findCandidates* selects in the simulation a route with a strictly shorter AS path than l, a filter between h and AS c – the AS from which this recursion has been called – will be added to the list of candidate filters. At the same time, we insert a dependency edge between the new filter and the candidate filter d_n found at AS c.

In general, recursion terminates when we arrive at an originating AS or when the current AS does not select a route shorter than the maximum allowed path length l. There are many other situations where recursion is stopped. For example, we allow the specification of a threshold for the maximum recursion depth. Additionally, no recursion is required if we arrive at an already visited AS hop. In our topology of Figure 5.4 the filters 7↚1 and 7↚2 are detected while looking at neighbor AS 1

Algorithm 2 Detecting mismatches and computing candidate filtering policies.

```
1: for all observed paths p to the given prefix do
2:    start at originating AS and walk to observation AS
3:    for all hops h of path p do
4:       o = suffix of p from AS h to observation AS
5:       s = simulated path at the router assigned for suffix o
6:       o_length = length of suffix o
7:       if s not equal o then
8:          add "mismatch" m to dependency graph
9:          findCandidates(h, o_length, m, 1)
10:      end if
11:   end for
12: end for
13: sub findCandidates(hop h, length l, from_policy f_from, depth r):
14: if r > threshold or h == originating AS then {not all termination criteria shown}
15:    terminate
16: end if
17: for all physical neighbors n of h do {some neighbors can be skipped, not shown}
18:    n_length = length of path announced from n
19:    if n_length < l then
20:       add "FILTER " f_new to dependency graph
21:       findCandidates(n, l − 1, f_new, r + 1)
22:       add link from f_from to f_new in dependency graph
23:    end if
24: end for
```

and AS 2 at recursion depth 1. While at AS 1, there will be a recursive call for AS 2 with recursion depth 2. However, AS 2 has already been visited and thus the candidate filters have been already computed. Recursion can thus be stopped safely without losing information.

5.3.2 Freedom in Filters Location

We now apply the algorithm in Figure 2 to compute *candidate combinations* on the AS-topology of Section 5.1. The goal is to give an estimate of the choice we have in terms of filter candidates when trying to achieve consistency between best routes in our model and observed AS paths. For this, we randomly select an extensive number of prefixes, called `psample` (see Table 5.1).

`psample` contains more than 2 million AS paths to 50,000 prefixes. For each prefix we have a mean of 160 distinct AS paths, with an average of 3.6 prefixes sharing a common AS path. While running our algorithm, we detected in total more than 10 million mismatches, i.e., AS hops that do not select the "correct " suffix of an observed route. Even for a single prefix the number of detected mismatches is considerable, with 3,328 on average.

To study the impact of recursion on the number of filter candidates found, we run our algorithm with three different thresholds for the maximum recursion depth. The results are summarized in Table 5.2. Allowing filters only on links incident to the AS hop with the mismatch (recursion depth 1) results in an average of 32.9 filter candidates per mismatch. This number is surprisingly high but

Table 5.1 Statistics on observed paths towards sample prefix set "psample".

# prefixes	50,000
# originating ASs	10,575
# distinct AS paths	2,267,296
# prefixes per AS path	
- mean	3.6
- standard deviation	11.8
# distinct AS paths to a prefix	
- mean	161
- standard deviation	42
# mismatches per prefix	
- mean	3,328
- standard deviation	5,191

Table 5.2 Number of candidate filters per mismatch for psample.

	recursion depth 1	recursion depth 2	recursion depth 3 [1]
mean	32.9	1,103	2,952
standard deviation	116	4,518	12174
min	1.0	1.0	1.0
max	1,847	49,040	80,050

[1] only for a subset of 2,000 prefixes

can be explained by some ASs having a large number of neighbors from which routes have to be filtered. With a recursion depth of 2 (3) this increases to more than $1,000$ ($3,000$) candidate filters on average.

To measure the freedom we have in combining those candidate filters, we use the notion of *filtering combinations* defined in Section 5.3.1. We slightly modify the recursive function *findCandidates* of Algorithm 2 to return the number of possible filtering combinations. Recall that each filtering combination ensures that no path is selected at the current AS hop that is strictly shorter than the maximum allowed path length. In general, there are multiple "bad" neighbors from which we have to filter out shorter paths. The number of possible filtering combinations is the number of non-empty subsets of lines from the filtering combinations that contain the "bad" neighbor.

Obviously, we only obtain a single filtering combination when recursion is terminated at depth 1. However, with a maximum recursion depth of 2 the average number of filtering combinations per mismatch is already in the order of 10^{500}, increasing to $10^{13,000}$ for a maximum recursion depth of 3. Note that these numbers are only rough estimates. Still, they illustrate the freedom we have in filter locations. There is even more choice if we do not restrict ourselves to non-redundant *filtering combinations* and were to allow other policies, e.g., local-preference.

Figure 5.6 Popularity of candidate filtering policies for `psample`.

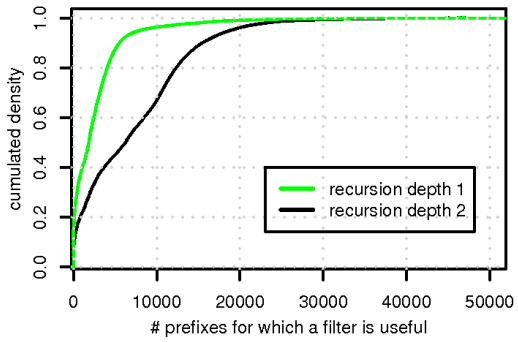


Table 5.3 Popularity of candidate filtering policies for recursion depth 1 and 2 in `psample`.

Percentile	25%	50%	75%	90%	95%	100%
#prefixes (depth 1)	236	1,888	3,604	5,548	8,004	46,921
#prefixes (depth 2)	1,480	6,237	11,389	15,523	18,896	47,032

5.3.3 Popularity of Filters

In the previous section, we computed an extensive number of candidate filters. Applying those per-prefix filters is supposed to ensure correct propagation of observed paths. The main idea now is to check whether there are filter locations that are more popular than others. We call a filter on an AS-level link *popular* if the link is identified as a possible filtering location for many prefixes by our algorithm of Section 5.3.1. A large number of such popular filters suggests that per-prefix policies are too fine and should be aggregated into coarser policy entities.

To detect popular filters, we run the algorithm of Section 5.3.1 on the observed routes of `psample` (see Table 5.3). Using a maximum recursion depth of 1 and 2 in our algorithm reveals the impact of the recursion depth on the popularity of the identified filters. For each directed AS-level link we count the number of prefixes for which a filter candidate is identified as "useful" on that link. The distribution of filters popularity for both recursion depths is plotted in Figure 5.6.

Figure 5.6 shows that some filters are more popular than others. While for a recursion depth of 1, less than 5% of the detected candidate filters are useful for at least 10,000 prefixes (out of 50,000), this is more than 30% for recursion depth 2. A similar trend is observed for larger recursion depths. The reason for this may be that large recursion depths add a lot of noise, i.e., they identify candidate filters at locations that are unlikely to be related to the mismatches we try to fix.

Table 5.3 provides further details about the popularity of filters. There are some locations for filtering which seem to be very popular. With a maximum recursion depth of 1, 5% of the identified filter candidates are "useful" for more than 8,000 prefixes.

At the same time we see filter candidates that are identified for only a very small number of prefixes. 25% of the detected filter candidates affect less than 236 prefixes (out of a total of 50,000) if a recursion depth of 1 is used. For a recursion depth of 2 this number increases to 1,480. Selecting

the 2,290 most popular filtering locations for recursion depth 1 (*pfilters*), we check how many of the filter candidates for recursion depth 2 are not redundant with them, i.e., are not be needed to achieve agreement between observations and the routes in our model if the filters in *pfilters* were configured. For this purpose, we take the computed dependency graph of recursion depth 2 and initially mark each filter in *pfilters* as "covered". Then, other filters in the dependency graph can be recursively marked as "covered" if either all children policies or all parent policies are already marked. By doing so, we see that the average ratio of covered filters is 75%. This number is surprisingly high given that there are many prefixes with more than 60,000 filters being detected for a maximum recursion depth of 2.

The main lesson of this section is that a non-negligible part of our filter candidates can be aggregated into coarser policy entities if the only goal is consistency between propagation in our model and the observed data. Higher recursion depths are not very helpful. They add more noise thereby making it more difficult to identify popular locations for filtering.

5.4 Next-Hop Atoms

An inter-domain routing model with predictive capabilities needs to incorporate routing policies that are consistent with actual route propagation and that are scalable. On the one hand, relying on per-neighbor based business relationships is scalable as little configuration is required in the model. Per-prefix filtering, on the other hand, allows for models highly consistent with observed path choices, but the information, that is needed to infer per-prefix filters, (i.e., observed routing data) is incomplete. The answer to the question of what is the right granularity to implement routing policies in an Internet-wide model is still partly open.

Given the insights from the preceding sections, we know that the appropriate granularity to model routing policies is in between per-prefix and per-neighbor. To narrow this down, we introduce an abstraction to characterize the granularity of routing policies: *next-hop atoms*.

A *next-hop atom NH* of an AS X is a subset of X's neighbors that X chooses as next-hops for its best routes towards a given set of BGP atoms[2]. All BGP atoms for which we see that an AS uses the same set of neighbors for its best routes belong to the same next-hop atom. The reason to define next-hop atoms in terms of BGP atoms is that BGP atoms define the finest granularity at which sets of prefixes share the same path choices. One might choose to use prefixes instead of BGP atoms.

Figure 5.7 illustrates the concept of next-hop atoms. It shows the path choices made by AS X towards five different BGP atoms. AS X is composed of two quasi-routers, QR_X1 and QR_X2. It has three neighboring ASs: A, B, and C, each composed of a single quasi-router. The best path, AS X chooses towards BGP atom 1, has as next-hop AS A. To reach atoms 2 and 3, X uses as its next hop AS B, whereas the best paths towards both atom 4 and 5 go through AS B and C. In this example AS X requires two quasi-routers because it has to choose two different best paths towards atoms 4 and 5.

In the case of the example in Figure 5.7, AS X has three different next-hop atoms: NH_1 contains next-hop A towards BGP atom 1, NH_2 contains next-hop B towards BGP atom 2 and 3 (since AS X chooses its best routes towards BGP atom 2 and 3 via AS B), and NH_3 contains next-hops B and C towards BGP atom 4 and 5 (because AS X chooses its best routes towards BGP atom 4 and 5 via AS B and AS C). Among all possible combinations of next-hop ASs, only a subset will actually be

[2]The definition of next-hop atoms can be trivially extended to next-hop routers if more detailed information about ASs is available.

Figure 5.7 Next-hop atoms - example.

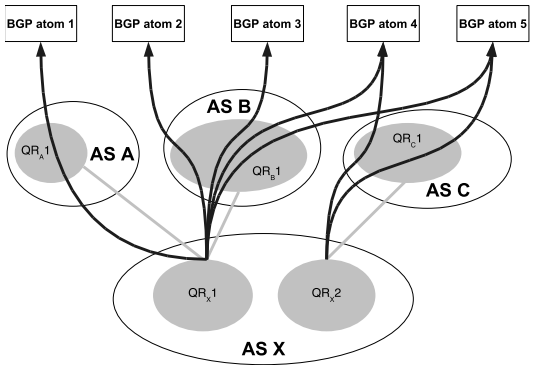
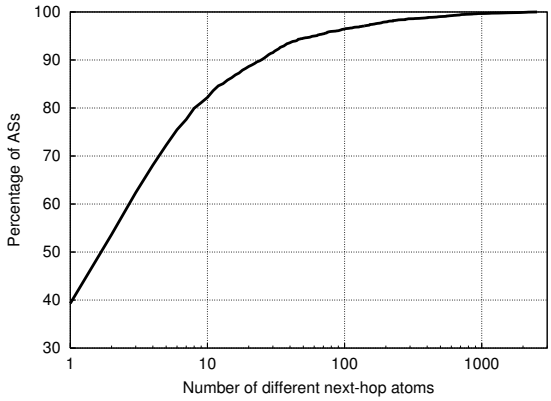


Figure 5.8 Number of next-hop atoms per AS.



used to send traffic towards BGP atoms. In our example, we only need three distinct combinations of neighboring ASs towards the five considered BGP atoms.

Note that next-hop atoms do not reveal why some AS prefers some paths to others. Next-hop atoms only describe the choice ASs make, not the reasons for their choice. However, for our study or policy granularity they are important as they capture the coarsest granularity (across prefixes) at which an AS chooses its best paths in distinct ways (among its neighbors).

The simplest way an AS can select its best paths is by always using the same set of neighbors for all prefixes. Such an AS has the same next-hop atom towards all prefixes. Single-homed ASs are in this situation as they have a single neighbor from which to choose their paths. Large transit providers on the other hand are expected to have a large number of different next-hop atoms due to their larger number of neighbors.

Figure 5.8 shows the distribution of the number of next-hop atoms per AS over the $3,535$ transit ASs, considered in Section 4.2.1. We observe that about 40% of the $3,535$ ASs have a single next-hop atom. Modeling routing policies for those ASs is trivial: they select for all prefixes the same set of neighbors. For the remaining 60% of the transit ASs, there can be between a few next-hop atoms up to hundreds.

Figure 5.9 shows how many neighboring ASs the next-hop atoms contain. Among all next-

Figure 5.9 Neighboring ASs in next-hop atoms.

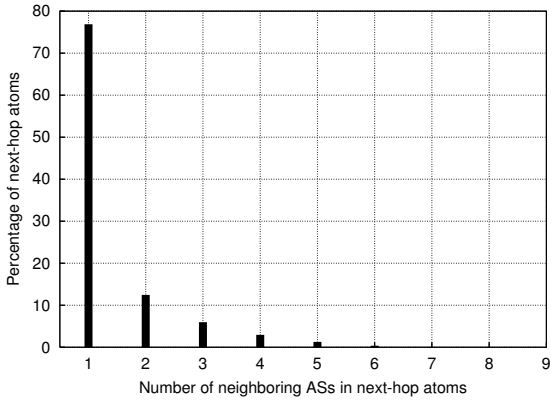
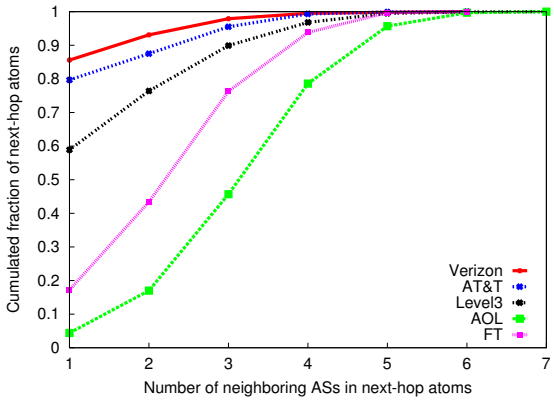


Figure 5.10 Number of neighboring ASs in next-hop atoms for large ASs.



hop atoms from our 3,535 transit ASs, more than 75% contain a single neighboring AS. Only for those next-hop atoms can we configure per-neighbor policies. For the remaining next-hop atoms preferring a single over all others does not work. In that case we can only speculate about the actual BGP policies that lead to the route propagation as we observe it in our data: local-preference, MED, IGP cost or other tie-breaking steps of the BGP decision process. One cannot hope to model such detailed information about path choices by routers, especially by relying only on BGP data from a limited set of vantage points.

Further complexity arises from differences in the next-hop atom structures of various ASs. Figure 5.10 provides the number of neighboring ASs in the next-hop atoms of 5 large ASs we selected: Verizon (AS701), AT&T (AS7018), LEVEL3 (AS3356), AOL (AS1668), and OPENTRANSIT (AS5511). We observe huge differences in the fraction of next-hop atoms that are made of a single neighbor (per-neighbor path choices). Verizon has more than 85% of its next-hop atoms consisting of a single neighbor. AOL on the other hand, has less than 5% of its next-hop atoms consisting of a single neighbor. AOL's next-hop atom granularity reflects its business as content provider. AOL is more likely to leverage its path diversity so as to optimize geographical path selection.

Table 5.4 AS relationships: agreement between observed and simulated routes.

	business relations	bus. rel. (no pref)
AS-Paths which agree	14.6%	31.8%
AS-Paths which disagree	85.6%	68.2%
due to route		
not available	61.0%	44.6%
learned but not selected	24.6%	23.6%

5.5 Are AS Business Relationships the Right Choice?

At this stage it is paramount to remember our ultimate objective: Finding an appropriate granularity to model routing policies. Both studying next-hop atoms (see Section 5.4) for our data set and the existence of popular per-prefix filters (see Section 5.3.3) suggest that the appropriate granularity of policies is generally close to per-neighbor. Yet, the question remains of which types of policies should be used in an inter-domain routing model. Next-hop atoms are only an abstraction to characterize the granularity of path choices and thus of routing policies. But there is still the need for a policy model that is scalable (i.e., no per-prefix policies) and at the same time more realistic than any agnostic routing policies that have been discussed so far.

AS business relationship appear to be the perfect choice for two reasons. First, the classification of AS-level edges into customer-provider (c2p), peer-peer (p2p) and siblings is a widely used policy model which considers *actual* business agreements between Internet domains and is not purely agnostic. Second, AS business relationships are enforced by BGP routing policies on a per-neighbor basis, something which appears reasonable from our results. In the following, we study AS relationships and try to understand whether there are fundamental errors to this policy model or not.

One motivation behind our search for the appropriate granularity of routing policies is to construct a model that is both scalable and that results in high agreement between routes, observed in the Internet, and those chosen in the model. Hence, we now explore the degree to which AS business relationships achieve this consistency. To find out, we take our topology from Section 5.1 and infer AS business relationships using the CSP algorithm (see Section 2.5.2). We use C-BGP (see Section 2.6) to compute the set of selected and learned routes at every router of our AS-level topology.

The column "business relations" of Table 5.4 shows the results when we compare the computed routes against those that we observe in our data (Section 2.4.2). For each observed path, we check if there is at least one quasi-router that selects the observed AS path as best route in the simulation. Only 14.6% of the paths agree between the simulation and the observations. For 61.0% of the paths, the corresponding path is not even propagated to the AS that should observe that path in the simulations. Only 24.6% of the paths are learned by the right AS but not selected as best path by any quasi-router of that AS.

We find these results disappointing. Applying business relationships does not seem to solve any inconsistencies between the paths propagated in our model and the routes actually observed in the Internet. Therefore, we now check for systematic errors that lead to the propagation of "wrong" paths in our model.

Enforcing business relationships has two consequences for route propagation and selection: *preference* of certain routes and *no-transit* for some routes. First, routes learned from customers will

Table 5.5 Comparing AS relationships with popular candidate filters.

total # of edges (directed)	117,822
total # of triples	30,351,164
total # of valleys	5,383,862
total # of (popular) filters	2,290
# filtered triples	991,268
# filtered valleys	602,619
ratio: filtered valleys to filtered triples	60.7%
# filters in at least one valley	2,283

be preferred over routes announced over peering links or provider links. Second, multi-homed stub ASs want to avoid being used as transit. For this reason, provider and peer routes are not propagated to other provider or peering ASs.

5.5.1 AS Relationships - No transit

We first shed light on the impact of the *no-transit* principle on route propagation. In Section 5.3.3 we identified filter candidates and found that there are some popular locations for filtering. The idea is now to compare such filter candidates with business relationships and to find out whether some of the popular filters in our per-prefix approach possibly implement *no-transit* policies in the AS-relationship "world". If so, this can be seen as a reason to consider business relationships as some form of routing policy.

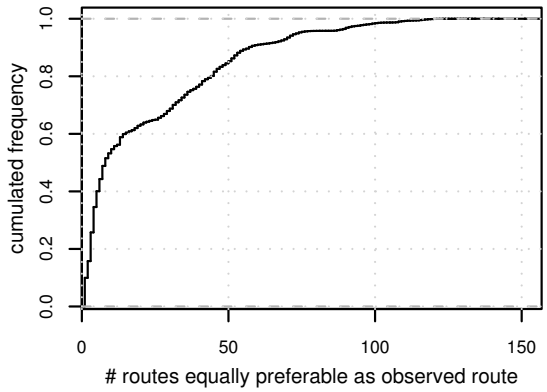
We compare business relationships with our candidate filters as follows: as a first step, popular locations for filtering are identified. According to Table 5.3, 5% of the filters found are useful for more than 8,000 prefixes. We select those filters and obtain a total of 2,290 popular filters (see Table 5.5). Based on our AS-level topology, we then compute all ASs triples. Altogether, there are more than 30 million such triples in our topology. The next step is to identify triples which violate the so-called *valley-free* property. In our terminology, a *valley* is a triple `A-B-C` along which no route should be propagated if the *no-transit* rule is correctly enforced. Let us assume that AS C and AS A are both providers of AS B according to the inferred business relationships. In this case, AS B will not announce any route learned from one of its providers to the other provider. According to Table 5.5 we find more than 5 million valleys.

Now we check how popular filters and valleys are related to each other. For this purpose, we collect all triples `A-B-C` such that any popular filter appears as either `A-B` or `B-C`. This results in 991,268 filtered triples. Surprisingly, almost all popular filters (2,283) are applied on AS-level links which are part of valleys. Popular filters hence frequently correspond to a non-transit policy, a situation where according to business relationships no path should be propagated. Henceforth, we conclude that the popular filters we identified suggest that the valley-free property used to infer business relationships is indeed correct.

5.5.2 AS Relationship - Preferences

The question remains of why using inferred business relationships exhibits this high level of disagreement when comparing the routes selected in our model with those observed in the data. Now,

Figure 5.11 AS relationships: freedom in path choice for observed paths.



we study the effectiveness of business relationships in terms of *preference*, i.e., preferring the "correct" observed path.

For this purpose, we again take the AS-topology of Section 5.1 and run a simulation based on business relationships inferred with the CSP algorithm (see Section 2.5.2). The goal is to find out how much choice each router has to select a best path. In spite of business relationships, a router may still have the choice between a set of equally preferred routes. Therefore, we determine for each observed path whether the observing AS learns it from a provider, peer, or customer AS according to the inferred AS relationships. Then, we look at the corresponding AS and quasi-router in our simulation and count the number of learned paths which are of the same "type" as the observed path, i.e., also a customer, provider or peering path. Figure 5.11 shows the distribution of this number of alternative path over all observed paths.

For only approximately 10% of the observed paths, there is a single path from which to select the best one. However, for more than 10% of all observed paths, we obtain more than 50 paths that are equally as good as the observed path. Given these results, we believe that business relationships do not reveal sufficient information about the actual *preference* policies used in the Internet.

5.5.3 Considering No Transit, Ignoring Preference

All in all, the valley-free property is apparently effective while AS business relationships only provide incomplete information about the actual preference of paths. We are now curious to see how much consistency in terms of path propagation can be achieved if we run a simulation that only considers the valley-free property but completely neglects preference, i.e., ignores whether routes are learned from customers or providers. The second column of Table 5.4 shows the results for such a simulation.

Surprisingly, we observe that the agreement between paths in our model and those actually observed is significantly higher compared to the case where we apply default business relationships. The question whether there is a systematic error in the way AS business relationships model preferences requires further investigation. Yet, we believe that future routing models should rely on the policy model of business relationships provided that open issues such as preferences of routes have been better understood.

5.6 Related Work

Inference of business relationships between ASs [Gao00, SARK02, BPP03, HK07] has been the most widely studied dimension of routing policies. Routing policies are typically partitioned into a few classes that capture the most common practices in use today [CR05]. Unfortunately, it is also known that the reality of routing policies [WG03] and peering relationships is far more complex than those few typical classes [CR05, CGJ+04]. The current approaches for business relationships inference rely on a top-down approach. They first define a set of policies and then try to match those policies with their observations of the system. Yet, policies as used by ISPs have to realize high-level goals [CR05]. Assuming any kind of consistency of such policies across ASs is questionable, especially as in practice, policies are often configured on a per-router, per-peering, or per-prefix basis [CR05]. Observed BGP routes do not have to make those high-level policies visible.

Our work is similar to [MQWZ05] in allowing the propagation of multiple paths across ASs. The authors in [MQWZ05] aimed at predicting AS paths between any pair of ASs without direct access to the concerned end-points and relied on a new inference of business relationships, as well as other information to predict the AS paths used between any pair of ASs.

5.7 Summary

In this chapter we searched for an appropriate granularity for modeling policies in the Internet. Additionally, we studied how and where to configure policies in this model in such a way that the routes in the model be consistent with paths observed by BGP from multiple vantage points.

By comparing business relationships with per-prefix filters, we investigated the role and limitations of business relationships as a model for policies. We observed there is a large freedom in the location of filters in the model if the goal is to obtain path choices consistent with observed BGP data. We also observed that the popular locations where filtering is necessary in our model correspond to the *valleys* where no path should be propagated according to business relationships inference. To capture the way individual ASs choose their best paths, we introduced a new abstraction: next-hop atoms. Next-hop atoms capture the different sets of neighboring ASs an AS uses for its best routes. We showed that a large fraction of next-hop atoms correspond to per-neighbor path choices. A non-negligible fraction of path choices however do not correspond to simple per-neighbor preferences, but hot-potato routing and tie-breaking within the BGP decision process, which are very detailed aspects of Internet routing.

To summarize, our work reveals that the the granularity of actual routing policies is close to per-neighbor. Although the per-neighbor based AS business relationships fail to provide consistency between the routes propagated in our routing model and those seen in observed data, they provide the right abstraction to prevent unnecessary paths from propagating in a model of the Internet. However, more research is needed to find out whether there are systematic error in the way business relationships model preferences.

Our concepts and findings in Chapter 4 and Chapter 5 are milestones on the way to more powerful inter-domain routing models. They show that the fundamental ingredients to an inter-domain routing model which is capable of answering what-if questions are a quasi-router-level topology combined with some kind of business relationship model.

6 Trellis: A Platform for Building Flexible, Fast Virtual Networks on Commodity Hardware

So far, this thesis has been mainly concerned with the status quo of inter-domain routing in the Internet: What are the factors that predominantly impact global route propagation and selection? How can we construct models of inter-domain routing that are scalable and sufficiently detailed to predict paths chosen in the Internet?

Regardless of whether we want to implement and evaluate new routing, forwarding, or addressing schemes for the future Internet [Fel07], or whether we simply want to improve parts of today's Internet, there is the need for a network testbed that supports experimentation. To alleviate the operation of such experimentation facilities, an increasing number of researchers follow the paradigm of network virtualization: A physical network is partitioned into multiple logical networks that behave like a physical network. Hence, multiple users can work simultaneously without interference. In addition, network virtualization is sometimes regarded as a long-term solution for the future Internet, since multiple architectures can be supported at the same time [FGR07].

In this chapter, we describe our solution *Trellis*, a platform for hosting virtual networks on shared commodity hardware. *Trellis* allows each virtual network to define its own topology, control protocols, and forwarding tables while amortizing costs by sharing the physical infrastructure.

6.1 Overview

Although existing infrastructures like PL-VINI [BFH+06] can run multiple networking experiments in parallel, forwarding packets in user space significantly limits scalability. In addition to a realistic, controlled experimental setting, network researchers need a testbed that provides the following properties:

- *Speed.* The platform should forward packets at high rates. For example, if the platform forwards packets in software, the packet forwarding rates should approach that of "native" kernel packet forwarding rates.
- *Flexibility.* The platform should allow experimenters to modify routing protocols, congestion control parameters, forwarding tables and algorithms, and, if possible, the format of the packets themselves.
- *Isolation.* The platform should allow multiple experiments to run simultaneously over a single physical infrastructure without interfering with each other's namespaces or resource allocations.

This chapter presents the design, implementation, and evaluation of *Trellis*, a platform that aims to find a "sweet spot" for achieving these three design goals given that it is difficult to achieve all

three simultaneously. The main question we address in our evaluation is the extent to which we can provide experimenters both flexibility and speed, without compromising forwarding performance. Many existing "building blocks" can provide functionality for implementing the two key components of a virtual network (*i.e.,*, virtual nodes and virtual links). Our main challenge is *synthesizing* existing mechanisms for implementing virtual nodes and virtual links in a manner that achieves the design goals above.

Currently, there are many initiatives in the area of network virtualization. For example, OpenFlow [OPE] introduces virtualization concepts and allows to run experimental protocols in production networks. However, isolating resources of different virtual networks is not the primary focus. Moreover, recent work does not only aim at providing experimental infrastructure for the design of a future Internet [Fel07], but also studies potential architectures and the required players that are needed to offer virtual network-based services to everyone, e.g., [SWP+09]. Our approach *Trellis* is similar to the enhanced Emulab testbed features for virtualizing nodes. Therefore, we are collaborating with the Emulab developers to integrate Trellis with the Emulab testbed. Emulab has focused mostly on resource allocation [HRS+08]; in contrast, we focus on the mechanisms for implementing the virtual network components.

To implement *virtual nodes*, two options are virtual machines (*e.g.,*, Xen) and "container-based" operating systems (*e.g.,*, VServer, OpenVZ). Container-based operating systems provide isolation of file system and the network stack without having to run an additional (potentially heavyweight) instance of a virtual machine for each experiment. Trellis's container-based OS approach provides experimenters *flexibility* by allowing them to customize some aspects of the IP network stack (*e.g.,*, congestion control) by giving each virtual network its own network namespace. In the current implementation, processing "custom" non-IP packets requires sending packets to user space, as in PL-VINI [BFH+06]. In this chapter, we evaluate the *speed* of such an approach.

Tunneling is a natural mechanism for implementing *virtual links*; unfortunately, existing tunneling mechanisms do not provide the appearance of a direct layer-two link, which some experiments might need. To solve this problem, we implement an *Ethernet GRE* (EGRE) tunneling mechanism that gives a virtual interface the appearance of a direct Ethernet link to other virtual nodes in the topology, even if that virtual link is built on top of an IP path.

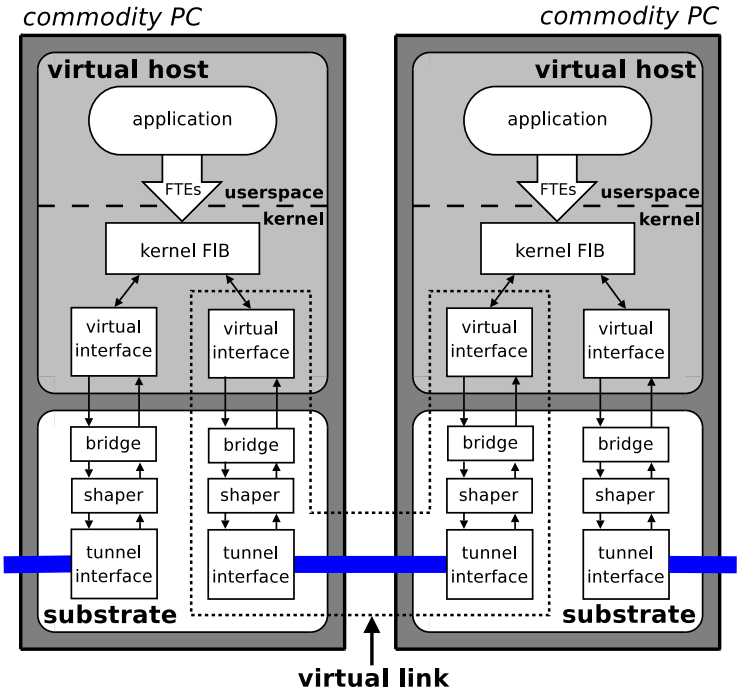
Finally, Trellis must connect virtual nodes to virtual links; existing mechanisms, such as the bridge in the Linux kernel, allows virtual interfaces within each virtual node to be connected to the appropriate tunnels. To improve forwarding performance, we propose an optimization called *shortbridge*, which improves forwarding performance over the standard Linux bridge by avoiding unnecessary look-ups on MAC addresses and copying of frame headers.

The rest of this chapter is organized as follows. Sections 6.2 and 6.3 describe the Trellis design and implementation, respectively. Section 6.4 compares Trellis's forwarding performance relative to other approaches (*e.g.,*, virtual machines), as well as to in-kernel packet forwarding performance. Section 6.5 concludes and describes our ongoing work.

6.2 Trellis Requirements and Design

A *virtual network* comprises two components: **virtual hosts**, which run software and forward packets; and **virtual links**, which transport packets between virtual hosts. We describe the requirements for Trellis, as well as its high-level design. We then describe mechanisms for creating virtual hosts and links.

Figure 6.1 Overview of *Trellis* design.



We identify four high-level design requirements for Trellis. First, it must *connect virtual hosts with virtual links* to construct a virtual network. Second, it must run on *commodity hardware* (*i.e.,*, server-class PCs) in order to keep deployment, expansion, and upgrade costs low. Third, it must run a *general-purpose operating system* inside the virtual hosts that can support existing routing software (e.g., XORP [HHK02] and Quagga [Qua]) as well as provide a convenient and familiar platform for developing new services. Finally, *Trellis* should support *packet forwarding inside the kernel* of the general-purpose OS to reduce system overhead and support higher packet-forwarding rates. An application running in user space inside a virtual host can interact with devices representing the end-points of virtual links, and can write forwarding table entries (FTEs) to an in-kernel forwarding table (forwarding information base, or FIB) to control how the kernel forwards packets between the virtual links.

Figure 6.1 illustrates a virtual network hosted on Trellis. The function of the virtual network is spread across three layers: user space inside the virtual host; in the kernel inside the virtual host; and outside the virtual host in a substrate layer that is shared by all virtual networks residing on a single host. The elements inside a virtual host can be accessed and controlled by an application running on that virtual host. Elements in the substrate cannot be directly manipulated but are configured by the *Trellis* management software on behalf of an individual virtual network. Multiple virtual hosts can run on the same physical hardware (not shown in the figure). Physical network interfaces are also not shown because they are hidden behind the tunnel abstraction. We note several salient features of this design:

- *Per-virtual host virtual interfaces and tunnels.* Each virtual host is a node in a larger virtual network topology; thus, *Trellis* must be able to define interfaces and associated tunnels specific to that virtual network.

- *In-kernel, per-virtual-host forwarding tables.* Each virtual host must be able to define how traffic is forwarded by writing its own forwarding-table entries. A virtual host's forwarding table must be independent of other forwarding tables, and processes running on one virtual host must not be able to affect or control forwarding table entries on a different virtual host.
- *Separating virtual interfaces from tunnel interfaces.* Separating the virtual interface from the tunnel endpoint enables the creation of point-to-multipoint links (*i.e.,*, the emulation of a broadcast medium). In addition, this separation allows the substrate to enforce a rate limit on each virtual link, to ensure resource isolation between the virtual networks.

The challenge in building *Trellis* was to identify and combine individual virtual host and virtual link technologies that satisfied our design requirements, or implement new components in cases where existing ones did not meet the design requirements. The next section describes these design choices in the context of the *Trellis* implementation.

6.3 Trellis Implementation

Trellis synthesizes host and link virtualization technologies into a single, coherent system. In this section, we explain the implementation decisions we made when building *Trellis* to achieve our goals of speed, flexibility, and, where applicable, isolation.

6.3.1 Host Virtualization

Flexibility *Virtual hosts* must allow experimenters to implement both custom control-plane and data-plane functions, without compromising speed (*i.e.,*, forwarding performance). Most types of host virtualization support control-plane customization; a thornier issue is custom data plane operations, such as forwarding non-IP packets, which requires modifications to the network stack in the operating system. In full virtualization, this customization requires modifications to the guest OS. Container-based virtualization does not provide this flexibility because all virtual hosts share the same data structures in the kernel but providing in-kernel data-plane customizability might ultimately be possible by partitioning kernel memory and data structures analogously to how similar systems have done this in hardware (e.g.,[T+07]).

In addition to providing fast forwarding and flexibility, *Trellis* should *scale*: it should support a large number of networks running simultaneously. Previous work, as well as our experiments in Section 6.4, show that container-based virtualization scales better than other alternatives: specifically, given a fixed amount of physical resources, it can support more concurrent virtual hosts than full virtualization. This better scalability makes sense because in container-based virtualization only a subset of the operating system resources and functions are virtualized.

Decision 1. *Create virtual hosts using Container-based Virtualization (not full virtualization).*

We combined two container-based approaches, Linux VServer [Linc] and NetNS [Net], to serve as the virtual hosting environment of Trellis. Since the PlanetLab OS is also based on VServer, this allows us to leverage PlanetLab's management software to run a Trellis-based platform. NetNS virtualizes the entire Linux network stack, rather than simply providing each container with its own forwarding table. This enables Trellis to support experiments that want to configure, for example, TCP congestion-control parameters or IP packet manipulations; in addition, NetNS has recently

Table 6.1 Container-based virtualization vs. full virtualization.

Criteria		Full Virtualization	COS
Speed	Packet forwarding	No	Yes
	Disk-bound operations	No	Yes
	CPU-bound operations	Yes	Yes
Isolation	Rate limiting	Yes	Yes
	Jitter/loss/latency control	Unknown	Yes
	Link scheduling	No	No
Flexibility	Custom data plane	Guest OS change	No
	Custom control plane	Yes	Yes

been added to mainline Linux, making the use of NetNS especially appealing. Another possible choice for container-based host virtualization is OpenVZ, which has essentially the same functionality as our combination of VServer and NetNS; we evaluate both our approach and OpenVZ in Section 6.4.

6.3.2 Link Virtualization

Virtual links must be flexible: they must allow multiple virtual hosts on the same network to use overlapping address space, and they must provide support for transporting non-IP packets. We tackled these problems by implementing a new tunneling module for Linux, Ethernet-over-GRE (EGRE). *Trellis* uses GRE [FLH+00] for tunneling because it has a small, fixed encapsulation overhead and also uses a four-byte key to demultiplex packets to the right tunnel interface.

Decision 2. *Implement virtual links by sending Ethernet frames over GRE tunnels (EGRE).*

EGRE tunnels allow each virtual network to use overlapping IP address space since hosts can multiplex packets based on an Ethernet frame's destination MAC address. This also allows *Trellis* to forward non-IP packets, which allows virtual networks to use alternate addressing schemes, in turn providing support for existing routing protocols that do not run over IP (*e.g.,*, IS-IS sometimes runs directly using layer 2 addresses). Forwarding non-IP packets would require running custom algorithms in user space, as in PL-VINI [BFH+06], or complex modifications to the kernel.

Speed Virtual links must be fast. First, the overhead of transporting a packet across a virtual link must be minimal when compared to that of transporting a packet across a "native" network link. Therefore, encapsulation and multiplexing operations must be efficient. Trellis's EGRE-based tunneling approach is much faster than approaches that perform a lookup on the source, destination address pair. Other user-space tunneling technologies like `vtun` [VTU] impose considerable performance penalty compared to tunnels implemented as kernel modules.

Isolation Trellis's virtual links must be isolated from links in other virtual networks (*i.e.,*, traffic on one virtual network cannot interfere with that on another), and they must be flexible (*i.e.,*, users must be able to specify many policies). To satisfy these goals, *Trellis* terminates virtual links in the root context, rather than in the virtual host contexts.

Decision 3. *Terminate tunnels in the "root context", outside of virtual host containers.*

Terminating the tunnel in the root context, rather than inside the container, allows the infrastructure administrator to impose authoritative bandwidth restrictions on users. Applications running on a virtual host have full control over the environment in a container, including access to network bandwidth. To enforce isolation, *Trellis* must enforce capacity and scheduling policies *outside the container*. *Trellis* terminates tunnels in the root context; an intermediate queueing device between the tunnel interface and a virtual host's virtual interface resides in the root context and shapes traffic using `tc`, the Linux traffic control module [Linb]. The virtual device inside the virtual host's context is bridged with the tunnel endpoint. This arrangement allows them to apply traffic shaping policies and packet-filtering rules, and, ultimately to implement packet scheduling algorithms that provide service guarantees for each virtual interface. Users though can still apply their own traffic shaping policies on the virtual network interfaces inside their respective containers for their traffic.

Terminating the tunnel endpoints outside the network container also provides flexibility for configuring topologies. Specifically, this choice allows users to create point-to-multipoint topologies, as discussed in more detail in Section 6.3.3. It also allows containers to be connected directly when they are on the same host, instead of being forced to use EGRE tunnels.

6.3.3 Bridging

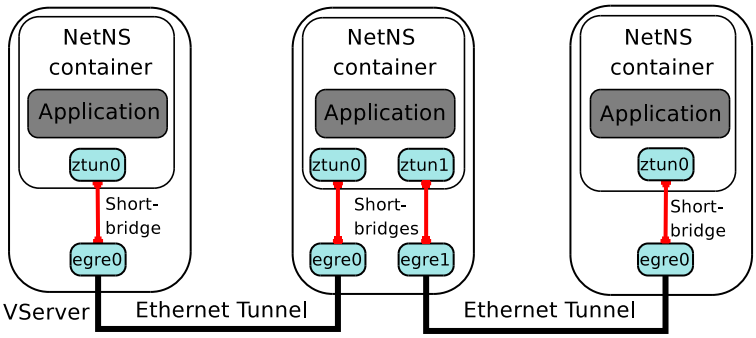
Terminating tunnels in the root context rather than in the host container creates the need to transport Ethernet frames between the tunnel interface (in the root context) and the virtual interface (on a virtual host). We explore two options for bridging EGRE tunnels to virtual interfaces: (1) the standard Linux *bridge* module [Lina]; and (2) *shortbridge*, a custom, high-performance device that we implemented specifically for bridging a single virtual interface directly to its corresponding tunnel interface. Each option offers different benefits: the bridge module offers additional *flexibility* in defining the network topology, while the shortbridge offers better *speed* (*i.e.,*, higher packet-forwarding rates). We use the standard Linux bridge for point-to-multipoint links; and *shortbridge* to maximize performance for interfaces that are connected to point-to-point links.

Decision 4. *For point-to-multipoint virtual links, connect tunnel interfaces with virtual interfaces using a bridge.*

Flexibility Some networks require bus-like, transparent multipoint topologies, where a set of interfaces can have the appearance of being on the same local area network or broadcast medium. In these cases, *Trellis* connects an EGRE tunnel to its corresponding virtual interface using (1) `etun`, a pair of devices that transports packets from a host container to the root context; and (2) the Linux bridge module, which emulates the behavior of a standard Layer 2 bridge in software and connects interfaces together inside the root context. One `etun` device is located inside a user container (`etun0`) and the other, `etun1` is located in the root context; this configuration is necessary because the bridge lies outside of the container, yet it must have an abstraction of an interface to connect to for the corresponding device inside the container. The Linux bridge module connects the end of the virtual interface that resides in the root context to the appropriate tunnel endpoint.

Unfortunately, as our experiments in Section 6.4 show, using the bridge module slows packet forwarding due to additional operations: copying the frame header, learning the MAC addresses, and performing the MAC address table lookup itself (*i.e.,*, to determine which outgoing interface corresponds to the destination Ethernet address). When network links are point-to-point, this lookup

Figure 6.2 High speed forwarding using shortbridges.



is unnecessary and can be short-circuited; this insight is the basis for the "shortbridge" optimization described below.

Decision 5. *For point-to-point virtual links, connect tunnel interfaces with virtual interfaces using a "shortbridge".*

Speed Forwarding packets between the virtual network interface and the tunnel interface must be fast, which implies that the bridge should determine as quickly as possible which outgoing interface should carry the traffic. A potential bottleneck for transporting traffic is thus the lookup at the bridge (*i.e.,*, mapping the destination MAC address of the Ethernet frame to an outgoing port).

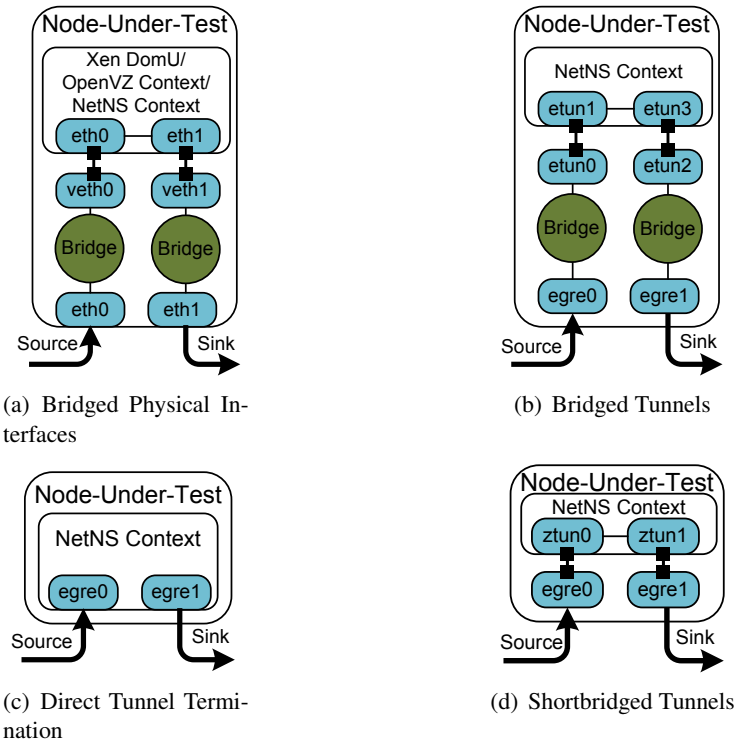
For point-to-point links, we have implemented an optimized version of the bridge module called *shortbridge*. We have also implemented a new device, `ztun` which, unlike the `etun` device, is a *single* virtual interface inside the container that the shortbridge can connect directly to the tunnel interface without requiring a corresponding interface in the root context. The `ztun` interface is instantiated as a single interface inside a host container and connects directly to the shortbridge. Figure 6.2 shows a configuration using the shortbridge device; a single shortbridge device connects one virtual interface (*i.e.,*, `ztun` device) to one tunnel interface (*i.e.,*, `egre` device).

Shortbridge achieves a performance speedup by avoiding a bridge table lookup: traffic can simply be forwarded from the single `egre` device to the single `ztun` device, and vice versa. The `ztun` device always connects to a tunnel endpoint; thus, shortbridge maintains a pre-defined device-naming scheme which allows each `ztun`/`etun` pair to have a static mapping, avoiding potentially slow lookups. Additionally, shortbridge avoids an extra header copy operation by reusing the packet data structure for the two devices that are connected to the shortbridge.

6.4 Performance Evaluation

Ultimately, we aim to evaluate whether *Trellis* satisfies our design goals of *speed* and *isolation*. To this end, we focus on speed and specifically on Trellis's packet-forwarding performance compared to other environments, including Xen, OpenVZ, and forwarding in user space. Our experiments show that *Trellis* can provide packet-forwarding performance that is about $2/3$ of kernel-level packet forwarding rates, which is nearly a tenfold improvement over previous systems for building virtual networks [BFH+06].

Figure 6.3 Experiment setup. Each setup has a source, a sink and a node-under-test.



(a) Bridged Physical Interfaces

(b) Bridged Tunnels

(c) Direct Tunnel Termination

(d) Shortbridged Tunnels

6.4.1 Experimental Setup

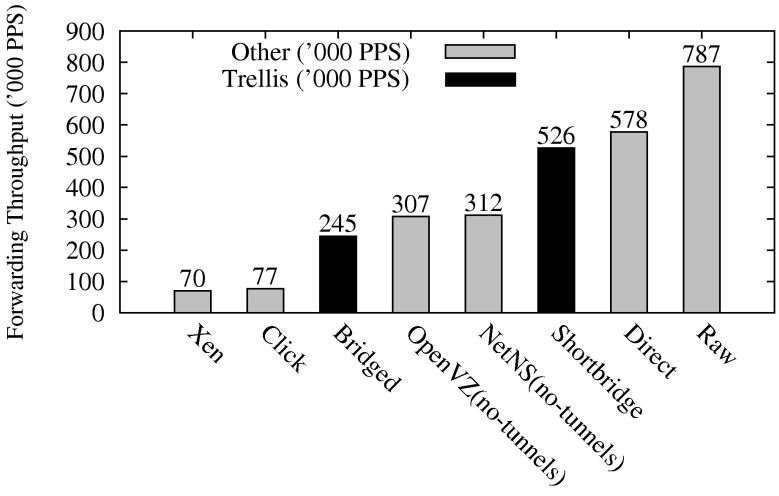
Test Nodes We evaluated the performance of *Trellis* and other approaches using the Emulab [WLS+02] facility. The Emulab nodes are connected through a switched network with stable 1 Gbps rates and negligible delays. The Emulab nodes were Dell Poweredge 2850 servers with 3.0 GHz 64-bit Intel Xeon processor with 1MB L2 cache, 800 MHz FSB, 2GB 400MHz DDR2 RAM and two Gigabit Ethernet interfaces. We used a customized 2.6.20 Linux kernel patched with Linux VServer and NetNS support and our custom kernel patches to provide support for EGRE and shortbridge.

Traffic Generation Tools such as *iperf* or *netperf* are not sufficient for our needs because these tools generate packets from user space which can hardly exceed more than 80,000 packets per second (pps). Instead, we generated traffic using *pktgen* [PKT], a kernel module that generates packets at a very high rate. We gradually varied load from high to low and noted the peak throughput.

6.4.2 Forwarding Performance

We evaluate the forwarding performance for various virtualization technologies. We performed packet-forwarding experiments for all of the environments shown in Figure 6.3 (including Xen, OpenVZ, and NetNS in the case of Figure 6.3(d) and compared each of these to the baseline forwarding performance of the native Linux kernel.

Figure 6.4 Trellis: peak forwarding performance (in pps) with 64-byte packets.



Comparison of virtualization approaches

User-Space Click To evaluate the baseline performance of forwarding packets in user space, we forwarded traffic through a Click [MKJK99] user-space process, as in the original PL-VINI environment [BFH+06], as shown in the Figure 6.3(b). We used a simple, lightweight Click `Socket()` element to forward UDP packets. Figure 6.4 shows that the peak packet-forwarding rate for 64-byte packets was approximately 80,000 pps. PL-VINI sustained even worse performance because it used a large set of Click elements with complex interactions between them.

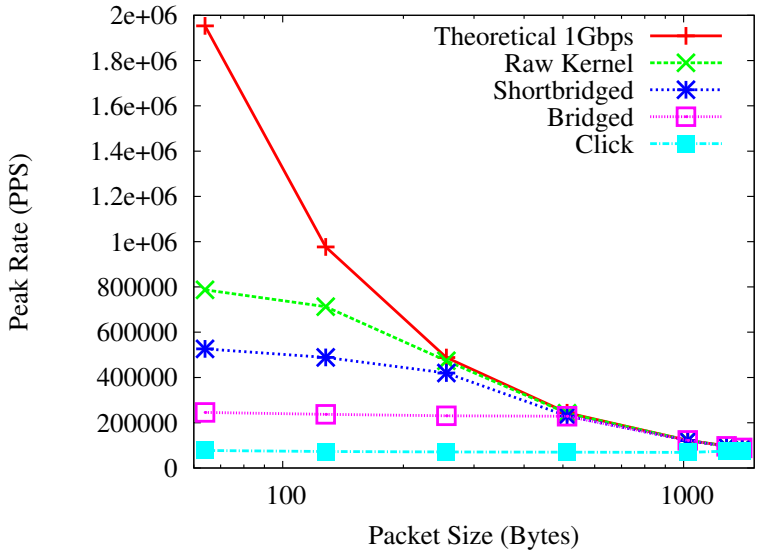
Full Virtualization: Xen We measured the forwarding performance of Xen 3.0.2. We bridged the virtual interfaces in DomU (the user domain) to the physical interfaces in the privileged domain, Dom0, using the Linux bridge module as shown in Figure 6.3(c). Unfortunately, Xen was unstable under packet rates of more than 70,000 packets per second, which is consistent with other studies [MCZ06, PZW+07].[1] Recent activity in the Xen community suggests that newer versions might have a more stable network stack [MCZ06].

Container-Based Virtualization: OpenVZ and Trellis We evaluated OpenVZ to compare Trellis's performance with another container-based virtualization system. OpenVZ does not provide EGRE or shortbridge features; thus, we connected the nodes directly, without tunnels and used a regular bridge module to connect the physical interfaces to the virtual interfaces. Figure 6.3(c) shows our configuration for the OpenVZ setup and for a *Trellis* setup with no EGRE tunnels and a regular bridge module (*i.e.,*, NetNS+VServer); this setup is analogous to our setup for the forwarding experiment with Xen.

Figure 6.4 shows that the performance of OpenVZ is comparable to that of *Trellis* when plain Ethernet interfaces and bridging are used; with this configuration both systems achieve peak packet-

[1] After about 15 seconds of such load, the DomU virtual interfaces stopped responding. Increasing the traffic load further, to more than 500,000 pps, caused the hypervisor to crash. We repeated the experiment with the same setup and similar hardware on our own nodes and found similar behavior.

Figure 6.5 Trellis: peak forwarding rate (in pps) for different packet sizes.



forwarding rates of approximately 300,000 pps. This result is not surprising because both OpenVZ and *Trellis* have similar implementations for the network stack containers. This result suggests that *Trellis* could be implemented with OpenVZ, as opposed to VServers+NetNS, and achieve similar forwarding rates.

Optimizing container-based virtualization

We evaluate the effects of various design decisions within the context of container-based virtualization: In addition to the five environments above, we evaluated various optimizations and implementation alternatives within the context of Trellis. Specifically, we examined the effects of (1) where the tunnel terminates and (2) using bridge vs. shortbridge on both packet-forwarding performance and isolation.

Overhead of terminating tunnels outside of container Directly terminating EGRE tunnels inside the container context *inside the container context*. This approach provides the infrastructure administrator little control over the network resources that the container uses (*i.e.,*, it is not possible to schedule or rate-limit traffic on the virtual links), but it offers better performance by saving a bridge table lookup. To quantify the overhead of terminating tunnels outside of containers, we perform a packet forwarding experiment with the configuration shown in Figure 6.3(e).

Figure 6.4 shows that directly terminating the tunnels within the container (Figure 6.3(e)) achieves a packet-forwarding rate of 580,000 pps (73% of native forwarding performance). This performance gap directly reflects the overhead of network-stack containers and EGRE tunneling.

Bridge vs. Shortbridge To evaluate the performance improvement of the shortbridge configuration over the standard Linux bridge module, we evaluate packet-forwarding performance with two setups. First, Figure 6.3(d) shows the setup of bridged experiment for Trellis. A similar setup is

used for evaluating forwarding performance in Xen and OpenVZ where a bridge is used. In Xen and OpenVZ the bridge joins the virtual environment with the physical interfaces on the node, but in *Trellis* the bridge connects the virtual environment to EGRE tunnels. Second, we replaced the Linux bridge module with our custom high-performance forwarding module *shortbridge* to connect virtual devices with their corresponding physical devices, as shown in Figure 6.3(f). We perform this experiment to determine the performance improvement over the regular bridging setup.

The shortbridge configuration achieves a forwarding rate of 525,000 pps (about 67% of native forwarding performance). The performance gain over the bridge configuration results from avoiding both copying the Ethernet frame an extra time, as well as performing bridge table lookup for each Ethernet frame. The bridged setup can forward packets at around 250,000 pps.

Effects of packet size on forwarding rate

Figure 6.5 shows how the packet-forwarding rate varies with packet size, for the bridge and shortbridge configurations, with respect to the theoretical capacity of the link and the raw kernel forwarding performance. For larger packets, the rate is limited by the 1 Gbps link. Trellis's packet-forwarding performance with shortbridge approaches the performance of native forwarding for 256-byte packets; for 512-byte and larger packets, both the bridge and shortbridge configurations saturate the outgoing 1 Gbps link.

6.5 Summary

This chapter has presented Trellis, a platform that allows each virtual network to define its own topology, routing protocols, and forwarding tables, thus lowering the barrier for enterprises and service providers to define custom networks that are tailored to specific applications or users. *Trellis* integrates host and network stack virtualization with tunneling technologies and our own components, EGRE tunnels and shortbridge, to create a coherent framework for building fast, flexible virtual networks.

Hosts in virtual networks require the appearance of dedicated network interfaces: the behavior of each virtual link (*e.g.,*, packet loss rate, latency, jitter) must *not* depend on the traffic patterns or load on other virtual networks that are sharing the physical infrastructure, which implies that the isolation provided by the virtual host must perform two functions: rate limiting and scheduling. At the moment, both full virtualization and container-based virtualization support traffic shaping in the root context. In principle, it is also possible for the root context to *schedule* the traffic on each virtual link to ensure that no virtual host sees inordinate delays in sending or receiving traffic, though no such scheduling mechanisms yet exist for either full virtualization or container-based virtualization. Incorporating a scheduling mechanism for a fair allocation of resource across containers is an area for future work.

7 HAIR: Hierarchical Architecture for Internet Routing

Many researchers agree that super-linear routing table growth, high update churn, insufficient support for multi-homing and traffic engineering etc. are significant deficiencies of today's Internet. Currently, it is widely questioned whether these shortcomings can be resolved by conventional incremental proposals that provide "band-aids" to the current Internet [Fel07]. Suggesting a major overhaul of today's Internet has become a popular research topic. Network testbeds as described in the preceding chapters are required to enable experimentation with new ideas, protocols, and architectures.

This chapter presents a proposal for designing a new routing architecture for the future Internet. We introduce *HAIR*, a scalable routing architecture and long-term solution for today's routing problems. Our primary concern is to address the scalability problems, in particular to prevent updates from being globally visible, and to provide scalable and simple solutions for traffic engineering, mobility, multipath, etc.

7.1 Overview

A plethora of proposals have been made to remedy the problems of the current routing architecture. For example, the shortage of host IP addresses is addressed by incremental approaches, e.g., CIDR, as well as re-engineering proposals, e.g., IPv6. Yet, none of these solutions eliminate the need for globally routable IP address space. Whereas this is not the case for IPv6, NAT, another incremental solution, has other disadvantages, e.g., breaking the end-to-end principle.

Giving a full overview of all proposals is beyond the scope. Rather we focus on those tackling the routing problem. One striking observation is that, despite the multitude of proposals, there is a wide consensus among scientists to separate the *identifier* from the *locator* functionality, e.g., [JMY+08, FFO+09, NB09, Vog08b]. Currently, both are mangled together within one address, the IP address. It identifies end-points in the transport layer and also specifies the position of an interface within the network. There are two main approaches to tackle this separation: host-based (e.g., [NB09]) or network-based (e.g., [FFO+09]). One main advantage of an host-based approach is that it can be self-deployed without cooperation of the network operators. The main advantage of network-based approaches is that they are capable of transparently supporting legacy hosts.

7.1.1 Design Guidelines

In spite of the multitude of suggested solutions and many ongoing discussions, e.g., [RRG], there is still no consensus on a new "better" routing architecture. It is our belief that this is partially due to the observation that most proposals do not comply with at least one of the following requirements for a new routing architecture:

Comprehensive solution: Any future-proof routing architecture has to address *all* the problems of today's routing system, including resolving the scalability limitations and the insufficient support for traffic engineering and mobility.

Easy deployment: A feasible migration path is key. It has to support native hosts, i.e., devices that already use the new architecture, as well as legacy hosts, i.e., devices that cannot be upgraded, at the same time.

Hierarchical scheme for mapping and routing system: We have to leverage the structure or hierarchy that is inherently present in the Internet [DD08, SFFF03], e.g., to overcome the scalability problems. Among the various benefits of using hierarchical approaches is the limited visibility of updates and their inherent scalability. Moreover, it is said that all known scalable addressing schemes exploit some kind of hierarchy [Day08].

In the context of a routing scheme, where locator and identifier are separated, we have to adopt hierarchical schemes for both the routing *and* the mapping system. In this way the updates to the routing system (due to routing changes) as well as the updates to the mapping system (due to mobility or traffic engineering) are no longer globally visible.

Reconcile network- and host-based approaches: Past evolution in the Internet has shown that it is easier to introduce innovations at the "edge" rather than in the "core". For example, new edge mechanisms such as NAT, firewalls, etc. are in common use while mechanisms that require support by the core are still waiting for wide spread use, e.g., IPv6.

In the context of locator/identifier approaches network-based [FFO+09] as well as host-based[NB09] solutions are discussed. We note that these alternatives have different objectives and at least in part complement one another. We therefore argue for a hybrid *edge-based* solution. Such an approach can transfer significant control to the edge hosts or networks while keeping some elements of the network-based approaches to address scalability limits and migration issues.

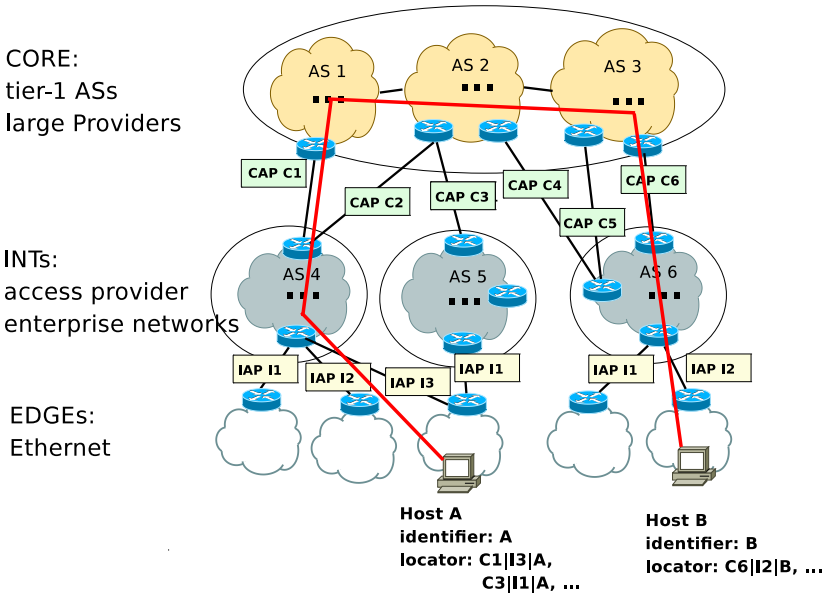
7.1.2 HAIR: Architecture

In this chapter, we introduce *HAIR* [FBC+08], a scalable routing architecture for the future Internet. *HAIR* uses a *hierarchical* approach motivated by graph theoretical insights. Graph theory [Gav01] suggests that hierarchical routing is very effective in terms of scalability as long as the number of "separators" between different hierarchical entities can be kept small. Fortunately, this is the case for the Internet. Most Internet researchers and engineers agree [CGH03] that the Internet is composed of a stable "core" formed by the transit providers and a more dynamic "edge".

Therefore, we propose to differentiate between *CORE*, *INTERMEDIATE*, and *EDGE* networks. Intuitively, the *CORE* may contain all large transit providers, e.g., tier-1 ASs. An *INTERMEDIATE* may either correspond directly to a smaller network, e.g., enterprise networks, or multiple smaller networks, e.g., access networks. Note, we use *INT* as short for *INTERMEDIATE*. Each of these in turn consists of many *EDGEs*. The hosts themselves are then located in *EDGEs*, e.g., Ethernet domains. The above describes a three-layer hierarchical schema consisting of one *CORE*, multiple *INTs*, and many *EDGEs*, see Figure 7.1. Yet, nothing limits us to three levels.

Our motivation for suggesting the above partition is information hiding. Any specific part of the hierarchy only needs a small subset of all topological information. In addition, routing updates that

Figure 7.1 *HAIR*– Basic principle.



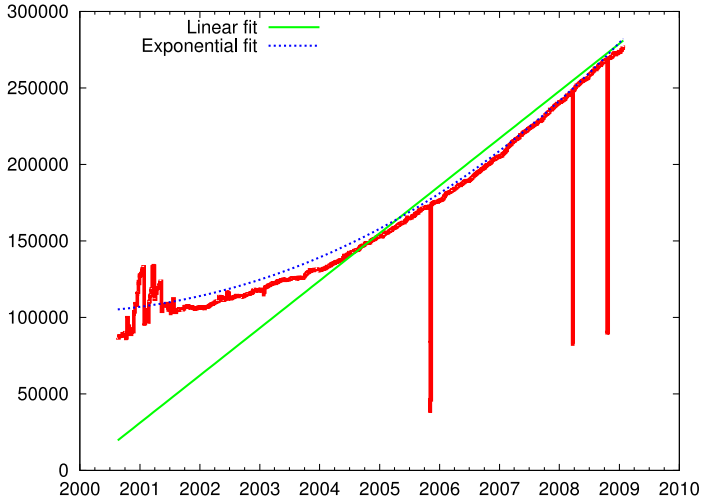
only affect the *CORE*, a *INT*, or a *EDGE* can stay local. Hence, routing updates in remote parts of the network do not have to be globally visible and do not need to be propagated.

To allow for (end-host) mobility and to avoid issues such as provider lock-in, we decouple locators from identifiers. Before sending a packet, a host asks a mapping service for the corresponding locator(s) of a given identifier. For scalability reasons, we again suggest to realize the mapping service as a distributed system where mappings are stored locally, i.e., by the authorities that own the mappings. A locator encodes a loose source route[1] by specifying one possible exit point from the *CORE* area to the *INT*; one possible exit point from the *INT* to the *EDGE*; and finally the identifier of the destination host.[2] In effect the locator is a list of separators between the hierarchy levels. In our case a separator is nothing else than a network *attachment point*. For example, the locator of host A (see Figure 7.1) consists of the separators between the different hierarchy levels, namely $CAP1|IAP3|A$. One crucial design decision is that every host can have multiple locators. This enables fast failure recovery and multi-path routing. We propose an *edge-based* approach. Accordingly, it is the responsibility of the end hosts to add the locators to the packet headers. Therefore, the gateway devices at the attachment points (*CAP* and *IAP*) can be kept simple. Their main task remains to forward packets according to information in the packet header, i.e., the locator. To support legacy hosts, the responsibility of end hosts can be delegated to a network intermediate, e.g., a NAT gateway or firewall, which is one piece of our hybrid solution. In addition, it is the responsibility of the networks themselves to maintain appropriate mapping entries. Hence, they do not only maintain their ability to control their traffic flows, rather they now have enhanced abilities for traffic engineering at their disposal.

Others have proposed some of the ideas used in this architecture. The contribution of *HAIR* are (a) to combine existing and novel ideas into an overall architecture and (b) an extensive discussion

[1] This is does not correspond to the loose source route option of IP.
[2] As mentioned *HAIR* supports any fixed number of hierarchy levels

Figure 7.2 Evolution of BGP forwarding table size observed for AS 3333 at RIPE *rrc00*.



of the design tradeoffs. We emphasize that our *edge-based* hybrid approach borrows concepts from both network-based as well as host-based proposals. By adopting a hierarchical scheme for both the mapping *and* the routing system, we avoid the trap of moving the full burden from the routing to the mapping system.

Evaluating the global impact of a new architecture proposal is an intrinsically difficult task. Yet, we perform an analysis that estimates the extent to which routers of the current Internet could benefit if *HAIR* were deployed in today's Internet. Also we discuss a potential migration path and present our experience with a proof-of-concept implementation of the architectural components of *HAIR*.

The rest of this chapter is structured as follows. In Section 7.2 we give an overview of related work and discuss the design space of future routing systems. Next, in Section 7.3, we present our architecture *HAIR*, before we describe its evaluation in Section 7.4. Finally, we summarize in Section 7.5.

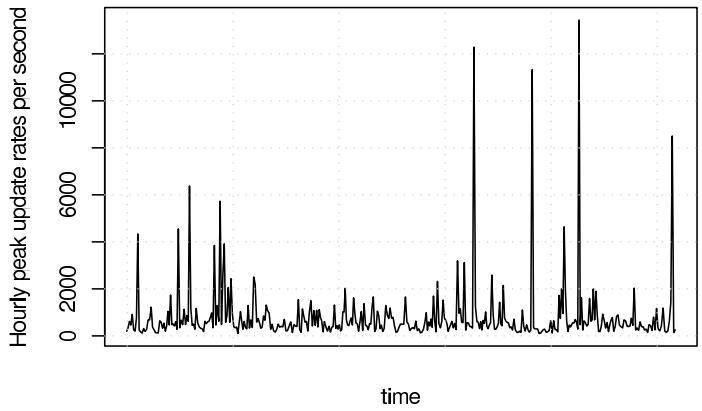
7.2 Design Space and Related Work

To better understand the fundamental design choices that need to be considered for a future-proof routing system, we need to learn from the strengths and weaknesses of previous proposals. We now review the problems that today's Internet is facing. Then, we characterize the design space for a routing architecture by reviewing related work in this area. Finally, we give a short overview of current activities in this area.

7.2.1 Problem Space

The most immediate problem faced by today's routing system is *scalability*. Routing and forwarding tables in the default-free zone (DFZ) of the Internet already contain more than 300,000 IP prefixes. Indeed, this number continues to grow super-linearly, e.g., due to traffic engineering or multi-homing, see Figure 7.2. So far, the technical progress in routers and memory technologies has accommodated this growth. But network operators are becoming more and more concerned about

Figure 7.3 Hourly peak update rates per second in November 2008, observed for AS 3549 at RIPE *rrc00*.



the ever increasing update rates, see Figure 7.3. Currently, Internet routers experience peak update rates of more than $10,000$ updates per second. This can impose such a high load on routers that it can take minutes or even hours for the routing to stabilize after events such as link failures. Many people anticipate this to become even worse with the imminent deployment of IPv6, which is likely to exacerbate such problems, e.g., calculating a routing table for IPv6 addresses is expected to take even more time [Day08].

Scalability is not the only reason why a re-design of the routing architecture is needed [Fel07]. Currently, the Internet does not provide proper means for *traffic engineering*. To control where traffic enters the network, some network providers have therefore chosen to disaggregate their assigned IP prefix ranges. They announce more-specific prefixes to upstream providers, which increases the number of entries in DFZ routing tables. In addition, more and more ASs are *multi-homed*. Such ASs also require efficient mechanisms for outbound as well as inbound traffic engineering. Current means unfortunately inflate the DFZ routing tables sizes. Furthermore, multi-homed stub ASs frequently request provider-independent (PI) addresses to avoid *provider lock-in* and *renumbering*. This impedes IP address aggregation by upstream providers. More generally, there is a *shortage of available IPv4 addresses*, e.g., [Hus] and, in spite of a huge number of proposals, the *mobility* problem in the Internet has not yet been resolved.

7.2.2 Potential Solutions – Design Choices

In this section, we discuss the design alternatives for novel routing architectures and illustrate, based on proposed solutions, the advantages and disadvantages of certain design choices. Given the abundance of proposals, it is not possible to discuss all of them in this section. Rather we pick representatives to cover the design space.

A large subset of the proposed solutions focus on a subset of the problems (see Section 7.2.1) that the Internet is currently facing. This includes simple solutions such as exclusively assigning provider aggregatable (PA) addresses or complex ones such as *Mobile IP*. Exclusively assigning provider addresses causes provider lock-in and may induce frequent renumbering of addresses. *Mobile IP* [Per02] enables end-host mobility but is not designed to reduce the size of routing tables in the core of the network. Other examples include the *Host Identity Protocol* (HIP) [MN06]. HIP

Table 7.1 Some design choices for locator/identifier (LOC/ID) separation.

Design choice	Option 1	Example	Option 2	Example
Separate or same namespaces for LOCs and IDs?	separate	LISP, i3	same	SHIM6, ROFL
Flat or hierarchical namespace for LOCs and IDs?	flat	ROFL	hierarchical	today's Internet
Host-based or network-based solution?	host-based	SHIM6	network-based	LISP
Core-edge separation?	yes	APT, LISP	no	today's Internet, ROFL
Mapping/forwarding in network-based solutions	map&encap	LISP	address translation	SixOne

decouples two namespaces – namely the names of a host's network interface and the names of a location – which are currently mangled together within the IP address. Thus, it enables easy renumbering of the inter-networking layer. However, it does not specify any mechanism to reduce the size of routing tables. Unlike HIP [MN06], the *Hybrid Link-State Path-Vector* (HLP) [SCE+05] protocol focuses on reducing update churn and routing table size in the core. But it neglects for example mobility.

For the remainder of this section we restrict our discussion to proposals in the realm of *locator/identifier separation* or Loc/ID split. These explicitly decouple the identifier from the locator functionality and thus solve some essential problems in the current Internet. For example such solutions inherently support mobility[3] but they do not necessarily resolve all scalability issues in today's Internet.

One decision that needs to be made is whether to use *single or separate namespaces* for identifiers and locators. The former option is chosen by more "radical" proposals such as ROFL [CCK+06] or VRR [CCN+06] that use flat labels as identifiers and locators or by *shim6* [NB09]. While relying on a single namespace seems appealing for reasons of simplicity, it violates the guideline of modular design: entities with different functionalities and different characteristics (contrary to locators the identifiers are stable) should be separable. Such an approach is taken, e.g., by *Locator/Identifier Separation Protocol* (LISP) [FFO+09], *Internet indirection Infrastructure*(i3) [SAZ+02] and ISLAY [Kas02].

Both identifiers and locators can be organized in either a *hierarchical or a flat namespace*. Today's Internet has two global namespaces (DNS and IP addresses). Both of these are tied to a pre-existing structure (administrative domains and network topology) and can therefore be considered as hierarchical. Unfortunately, this particular choice of hierarchies hampers mobility and causes renumbering of addresses. To solve this, one can rely on flat identifiers, see [CCK+06, CCN+06, MN06], at the cost of losing the ability to aggregate identifiers. Ahlgren et al. [AAER06] adopt a hybrid approach which uses a hierarchy of heterogeneous networks and can still support mobility and communication across them. However, they do not consider migration and do not transfer the hierarchical structure of their network to the required mapping system.

The same tradeoff applies for locators. But apparently it is possible to offer some compromises as well. For example Eriksson et al. [EO07] suggest to dynamically build hierarchical locators on demand, such that they are always aligned with the network topology. Unfortunately, the locator

[3] When separating locators and identifiers, only the identifier will be used as connection identifier in a TCP session.

construction comes with some overhead.

Frequently, one distinguishes between *host-based* [NB09, MN06] and *network-based* [FFO+09, JMM+07, Her] solutions. This classification refers to the components that have to be modified to implement the locator/identifier split. With *shim6* [NB09] end hosts possess a set of locators and choose one of them as identifier. Hence, state is maintained at the end hosts. In contrast, network-based proposals such as LISP [FFO+09] use a stable, non-routable IP address as identifier. Identifiers are mapped onto globally routable addresses by a mapping service that the (edge) network provides. It is sometimes feasible to combine network-based approaches with complementary host-based ones. For example, *LISP* [FFO+09] can be used to address the lack of globally routable IP addresses while end hosts may use *shim6* [NB09] to switch between different locators.

Network-based solutions address the shortage of global-routable IP addresses and routing table size. Host-based solutions allow for end-host mobility. But host-based solutions still require a globally routable locator for each end host. Thus, they do not directly address the problem of routing table growth. Recent discussions within the IETF's Routing Research Group [RRG] highlight that the main difference between host-based and network-based approaches is the deployment roadmap: in the limit, a network-based solution that has been pushed all the way to the hosts becomes a host-based solution.

Another design choice of routing architectures is whether to limit the scope of routing protocols or not [JMY+08]. Proposals that rely on *core/edge separation* include, among others, LISP [FFO+09], TRRP [Her], GSE [O'D97], and APT [JMM+07]. Generally, they need to map end host identifiers onto global-routable IP addresses, owned by the transit providers. Moreover, there exist solutions such as HLP [SCE+05]: It routes based on AS numbers and adopts a link-state routing protocol within each hierarchy of ASs (as specified by provider-customer relationships). But it uses a path-vector protocol between hierarchies. The alternative choice is to use a global flat routing space as in today's Internet or to deploy more "radical" solutions such as ROFL [CCK+06].

Irrespective of the specific Loc/ID split approach, network-based solutions need a *mapping service* inside the network. The indirection step to go from identifier to locator can be implemented either with tunneling, e.g., [FFO+09] or without tunneling, e.g., [Vog08b, Atk09, O'D97]. While the proposals in the former category encapsulate packets and add a header with the locator, the latter either use a NAT-style translation between identifier and locator addresses, e.g., [Vog08b, O'D97], or require addresses that consist of an identifier and a locator part [Atk09]. The obvious drawback of tunnel-based approaches is that they reduce the MTU size due to the encapsulation overhead. The drawback of address rewriting techniques is that they may require special boxes to keep state.

A mapping system design offers additional degrees of freedom. For example, it can be "one-to-one" or "many-to-one" (see, e.g., [Vog08a]). One proposal [Vog08b] maps edge addresses one-to-one onto the transit addresses from its provider. To then preserve the routing table scalability, the transit addresses have to be organized in an easy to aggregate manner. LISP [FFO+09] adopts an alternative approach. Multiple endpoint IDs (EIDs) can be mapped onto multiple routing locators (RLOCs). The assumption is that there is a limited number of RLOCs that need to be globally routable.

Moreover, the mapping service can be implemented in different ways. Many of the proposed solutions, e.g., [MN06], extend DNS and rely on DNS lookups to determine locators for a given identifiers. Others propose to use overlays [FFM09, MIB08, SAZ+02]. Overlays are frequently based on DHTs such as Chord or CAN [SMK+01, RFHK01]. Frequently, proposals strive to keep the routing system agnostic about the details of the specific mapping solution. This loosens the coupling between the two systems. For example, LISP leaves the choice of whether to use LISP-

ALT [FFM09], which is based on an overlay of BGP routers, LISP-DHT [MIB08], relying on DHT techniques, or any other solution that understands LISP queries. The drawback is a lack of mutual benefits.

Table 7.1 summarizes the main design choices we discussed in this section. Note that this list is not complete. Further design choices include the question of backward-compatibility and the type of routing algorithm to use for locators: link-state versus distance vector approaches, shortest path routing versus compact routing, etc. This discussion is beyond the scope of this thesis.

7.2.3 Current Activities

The future of the routing architecture is currently widely debated. In the following we briefly summarize ongoing discussions in this area.

There are efforts within the IETF to standardize some of the aforementioned solutions. The Routing Research Group (RRG) [RRG] provides the forum to discuss the pros and cons of the various solutions that have been suggested so far. Currently, there is consensus on a classification of potential solutions into classes such as *core-edge separation combined with map&encap* ("strategy A") or *hierarchical arrangement of locators combined with aggregation* ("strategy B") etc.. According to the RRG mailing list [RRG], most researchers appear to favor "strategy A". Yet, it has been noted that different approaches (e.g., "strategy B") may coexist with, or even evolve towards, some kind of "strategy A" solutions. Currently, there is an ongoing discussion if and why certain strategies can be excluded. The goal behind this discussion is to narrow down the solution space and to then move forward within the IETF to prototype, implement, and experiment with specific solutions.

Within the research community a plethora of research projects that adopt the clean slate approach to design a new routing architecture have been initiated, e.g., within FIND [NSF] in the US, or the "7th Framework Programme" of the EU [EU] or their local counterparts, e.g., in Korea, Japan, or Germany. They share the principle that the shortcomings of today's Internet cannot be resolved by conventional band-aid style of academic and industrial networking research. Hence the goal of this work is to initiate new proposals, reuse existing ones, or/and combine them to form an integrated architecture for a new routing system.

7.3 Architecture

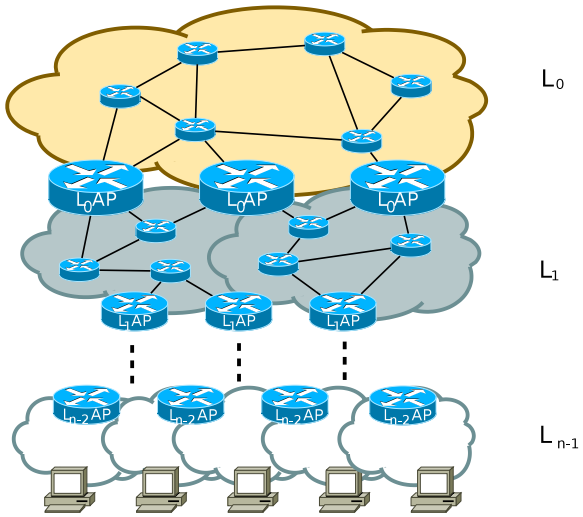
After giving an overview of *HAIR* in Section 7.1.2, we now present more details about our proposal: packet forwarding, mapping between identifiers and locators, reaction to failures, and migration from today's Internet to *HAIR*.

7.3.1 Overview

In spite of the multitude of suggested solutions and many ongoing discussions [RRG], there is still no consensus for a "new" routing architecture, as most existing proposals do not fulfill the design criteria outlined in Section 7.1.1. The approach of *HAIR* is: an *edge-based hybrid* routing architecture that *separates locators from identifiers* and adopts a *hierarchical scheme* for both the routing and the mapping system.

We will show that *HAIR* scales well as it localizes updates and therefore puts an end to unlimited table growth. Moreover, it offers inherent support for mobility, multi-homing, traffic engineering, etc.

Figure 7.4 N-layer hierarchical network structure.



Hierarchical Scheme On the one hand, all scalable addressing schemes, known so far, exploit some kind of hierarchy [Day08]. Indeed, results from graph theory [Gav01] show that hierarchical routing is very scalable as long as one can find small "separators". The key concept is information hiding: each entity only has aggregate rather than complete information about entities elsewhere in the hierarchy.

On the other hand, today's Internet is said [DD08, SFFF03] to consist of a stable "core", formed by large transit providers, and a more dynamic "edge", consisting of enterprise networks or small access providers. The typical Internet cost structure together with peering policies ensures that the number of links between an "edge" network and the "core" is limited. Most "edge" networks have a small number of upstream providers in the "core" and they typically, due to costs, do not have too many upstream links to a single provider. Accordingly, one possible Internet routing hierarchy is to group the "core" networks into level 0, the "edge" or *intermediate* networks into level 1, and the local area networks into level 2.

Whereas the above discussion suggests a *3-layer hierarchy*, see Figure 7.1, our architecture supports an arbitrary number of hierarchy levels, see Figure 7.4 for an *n-layer hierarchy*. *HAIR* consists of the following *components*:

***EDGE* (Level $n-1$):** This is the bottom layer of the hierarchy. Within this level routing is direct. An *EDGE* is an access network which attaches the hosts and the servers. A prototypical example is a (switched) layer-2 network such as an Ethernet LAN.

***INT*: (Levels 1 to $n-2$):** This is an intermediate layer in the hierarchy. A *INTERMEDIATE* network (*INT*) provides routing between attached *EDGEs* or *INTs* of the next higher level. Hence, its routing table needs entries for all routers within the *INT*, routes to the attached *EDGEs*/*INTs*, and default routes to the *INTs* of the next higher level or the *CORE*.

A *INT* can be administered by a single authority or it can consist of multiple domains that have agreed on a common routing protocol. Contrary to *EDGEs*, *INTs* only consist of routers. In the current Internet, *INTs* may correspond to access providers, enterprise networks, or content distribution networks.

***CORE*:** (Level 0) This is the top layer of the hierarchy. The *CORE* is a network with a single routing protocol that routes packets between the *INTERMEDIATE* networks of level 1. Thus the *CORE* ensures global reachability. Its routing table contains entries about all routers within the *CORE* as well as routes to the directly attached *INT*s.

Figure 7.4 illustrates the relationships of the above components. Routing domains such as *CORE*, *INT*s, or *EDGE*s are connected via *attachment points*: a "level k" attachment point (L_kAP) connects a level k routing domain to a level $k+1$ one. Interconnections within the same hierarchical layer ("peerings") are possible.

Note that the number of hierarchy levels does not have to be the same network-wide: for instance, one organization may organize its own network in two hierarchical levels, while another one may partition its network into four or five layers. Moreover, the structure in Figure 7.4 is orthogonal to the typical classification of ASs as "transit" and "stub": in fact, even a transit AS usually has a portion of its network that is solely interested in network access. On the other hand, some enterprises (e.g., Boeing), despite owning "stub" ASs, have reached such a size that they may need to provide transit for their own traffic.

One way to map today's Internet to a 3-layer hierarchy is to group all the routers in the backbone portion of autonomous systems that provide transit into the *CORE*. These ASs fulfill the criteria of a *CORE*, as they use a single routing protocol, BGP, and ensure global reachability between the stub-ASs. The *INT*s are those ASs whose core Internet business is not in the transit business, e.g., enterprise customers, content/hosting/access providers. These again meet the criteria, as they use a single routing protocol within the *INT*. The *EDGE*s are then the layer-2 networks.

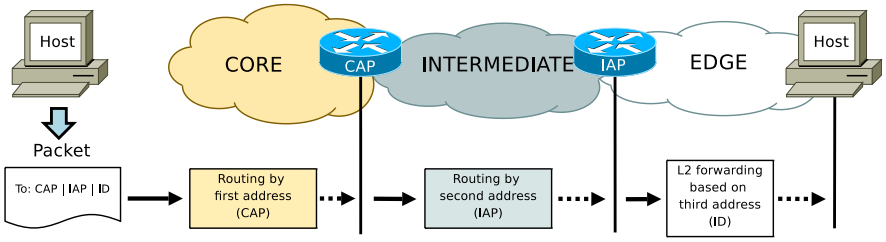
We achieve our scalability goals – small routing table size and low update churn – by following a core-edge separation strategy (see Table 7.1). At the top of the hierarchy there is a single routing domain – the *CORE*– where inter-domain traffic is exchanged. Within this part, all participating routers need to agree on a common inter-domain routing protocol. In principle, the policy-based BGP routing protocol can be used for this purpose. Intermediate networks – "edge" networks – only need their providers for upstream. Hence they have the freedom to choose which routing protocol to use inside their network. Due to the scope of routing domains the updates are also localized. Higher hierarchical levels do not have to know about topology or routing changes at lower levels. Provided that the number of L_0APs in the *CORE* is limited[4], the routing table size is drastically reduced as well.

Locator/Identifier Separation To allow for (end-host) mobility and to avoid issues such as provider lock-in, we decouple locators from identifiers. An immediate consequence of this design choice is the need for a new architectural component – a *mapping service*: Given a certain identifier it returns the current locator(s). For scalability reasons the mapping service is implemented as a distributed system. It ensures that mappings are stored locally, i.e., by the authorities that own the mappings. See Section 7.3.3 for more details on how to realize such a service.

HAIR presumes that end host identifiers are organized in a global flat namespace. This gives us the equivalent of a Provider Independent (PI) address and thus allows us to avoid provider lock-in and renumbering. In principle, *HAIR* is compatible with IDs that are self-certifying, e.g., AIP [ABF+08].

[4] L_0APs may correspond to the Points of Presence (PoPs) that are used to interconnect transit providers in today's Internet. This number should be in the magnitude of tens of thousands and is not expected to change considerably in the future (see [DD08]).

Figure 7.5 Packet forwarding in *HAIR*.



In contrast to identifiers, the namespace of locators has a local scope. They are managed by the individual *INTs*. A full locator for an end host consists of multiple parts and encodes a *loose source route*[5] towards that host: It consists of a sequence of attachment points that need to be traversed to forward a packet from the *CORE* to the host, see Section 7.3.2. To support an arbitrary number of hierarchical layers, locators can have variable length. Whenever a packet arrives at an attachment point, i.e., a hop on the loose source route, the local routing task within the *INT* (*CORE*) is only required from inside the *INT* (*CORE*) and its attached *EDGE* (*INT*).

The fact that each host can have multiple locators can be used for traffic engineering as well as multi-homing. Moreover, it enables inherent support for multi-path routing.

Edge-Based Approach We propose an *edge-based hybrid* solution. While some lightweight functionality is added to a limited set of devices in the network (i.e., to routing domain borders), resource hungry features are pushed as close to the edge as possible. It is the responsibility of the end host to query the mapping system for the locator and then add it to the packet headers. Since it may be necessary to support legacy networks, the host functionality can also be realized by either a NAT box or a firewall somewhere in the network.

7.3.2 Packet Delivery

Sending a packet from a source host to a destination host consists of three steps: 1. forwarding the packet up to the *CORE*, 2. forwarding within the *CORE* and 3. sending the packet from the *CORE* down to the destination host. The source *INT*, for part 1, can either use a default route, leverage the loose source encoded in the source locator, or combine these approaches. Routing within the *CORE*, part 2, is based on the destination *CORE* attachment point encoded in the destination locator. Since the *CORE* mainly consists of large transit providers (e.g., tier-1s), routing inside this area is likely to remain policy-based, e.g., via BGP. Finally, routing from the *CORE* towards the destination, part 3, follows the "loose source route", encoded in the destination locator: Whenever one of the *AP*s specified in the destination locator is traversed, forwarding is continued based on the next one. Hence, the destination locator determines where traffic enters a *INT* and therefore enables inbound traffic engineering. Note that the addresses of the attachment points do not have to be globally unique as the scope of these is limited by the *INT*.

In the following, see Figure 7.5, we illustrate using a 3-layer hierarchy how packets are forwarded from source A with identifier ID_A to destination B with identifier ID_B. In this case the locator consists of the *CAP* (i.e., the *CORE* AP or L_0AP) and the *IAP* (i.e., the *INT* AP or L_1AP). Host

[5] This does not correspond to the loose source route option of IP.

A is reachable via CAP_A and IAP_A and thus has locator $LOC_A = CAP_A|IAP_A$. Host B has locator $LOC_B = CAP_B|IAP_B$.

When host A wants to send a packet to B it first has to find its identifier ID_B, e.g., via DNS. Next it queries the mapping system to determine B's locator: $LOC_B = CAP_B|IAP_B$. The composition of these, $CAP_B|IAP_B|ID_B$, is the destination address of the packet. In addition, A also includes the source address $CAP_A|IAP_A|ID_A$. This avoids the locator lookup at the destination host.

Using, e.g., a default route, this packet is now forwarded to the *CORE* as A and B are not in the same *INT*. If A and B are in the same *INT*, i.e., if $CAP_A == CAP_B$, the next step is skipped. Within the *CORE* routing is based on the *CAP* portion of the destination address: CAP_B. Once the packet reaches CAP_B, it is handed over to the *INT*. Routing is now based on the *INT* attachment point: IAP_B. Finally, the packet reaches the *EDGE*. It is now IAP_B's responsibility to resolve the identifier, ID_B, to a layer-2 address and forward the packet to destination B.

We close this discussion with some observations: 1. the source *INT* controls the way its traffic leaves its *EDGEs*; 2. it is possible to incorporate direct "peerings" between *INTs*. In this case each of the *INTs* has to distribute the information within its routing domain and a packet that is destined to the other *INT* can be redirected directly to that *INT* on its way to the *CORE*.

7.3.3 Mapping System

Whenever a host wants to send a packet to a destination, it needs to resolve an identifier to one or several locators. To minimize lookups, hosts should maintain a cache[6] of such mappings in a similar manner to ARP or DNS caches.

We propose a hierarchical mapping system in which the structure of the routing architecture is mirrored. Thus, we propose to use a *"global" distributed lookup service*, e.g., a global DHT, managed by the *CORE* and a set of *Intermediate Network Mapping Service* (*IMSs*) managed by the *INTs* (or a set of *INTs*). The "global" DHT is used to get the hierarchy started. It stores for all identifiers a pointer to the *IMS* which currently holds the mapping for the identifier. The actual mappings are, therefore, stored in the *IMSs*. *IMSs* are administered by *INT* networks: the advantage of such a design choice is that the control over the paths remains with the *INTs* who manage the *IMSs* (see Section 7.4).

If a host wants to retrieve the locator for identifier ID_B in a 3-layer hierarchy, it first queries the global DHT which returns a pointer to the *IMS*, IMS_B, which stores the mapping for identifier ID_B. Then, the host asks IMS_B for the locator of ID_B. The result is the locator $LOC_B = CAP_B|IAP_B|ID_B$. Note, mapping an identifier to a locator involves two queries. Both requests can be issued by the host, or the global DHT redirects the request to the *IMS*, which has the option to send the answer directly back to the host. We point out that it is in principle possible to reduce the overhead by merging such queries with the DNS resolution steps.

While *HAIR* proposes the use of a DHT to resolve identifiers to *IMSs*, we do not put any restrictions on how to implement the *IMS* service. In principle, any distributed directory service can be used for both the global directory at Level 0 as well as the *IMSs*. However, we recommend to use one of the very scalable DHT designs at Level 0 since the number of required entries in the DHT corresponds to the number of identifiers – the number of hosts. To recruit enough servers for the DHT, registries that assign resources such as AS numbers or IP addresses may require each AS to dedicate resources to the global DHT.

[6] The exact cache design is out of scope of this thesis.

In the above example, we assume that the host knows how to contact the global DHT. This can either be resolved via a bootstrapping protocol, such as DHCP, by including them in DNS, or by ensuring that the locators of some of the DHT servers are well-known. The servers participating in the global DHT are expected to be professionally maintained and highly available, thus mitigating the frequency of failures. Participation in the global DHT requires authentication and authorization by a third party, e.g., routing registries or IANA.

Our motivation for mirroring the routing hierarchy in the mapping system is twofold: First, by controlling the *IMS* component, *INTs* are able to perform effective inbound traffic engineering. They determine the mappings used by the *IMS*. Second, we need to limit update churn. Any time a host changes *EDGEs*, the corresponding entries in the mapping system have to be updated. But if the two *EDGEs* are attached to the same *INT*, only the *INT* mappings have to be changed. No updates are required to the global DHT. Our design is based on the assumption that hosts switch more frequently between *EDGEs* attached to the *same INT* than between *EDGEs* of different *INTs*. This is plausible since nodes frequently move within an organization or to a geographically-close location which is likely to be associated with the same *INT*.

In case that a host moves to another *INT*, the update process is slightly more complex than with a non-hierarchical design. But we expect this to be less frequent. The locators of the mobile host have to be transferred from the previous *IMS* to the new *IMS* and the attachment point information needs to be updated. Then the global DHT is notified that the locators of the host have moved to a new *IMS*. At first glance this may seem time-expensive. However, updating information in a DHT involves only a minimum of additional work besides the lookup to find the node where the information has to be changed.

One might argue that *HAIR* simply shifts the burden of handling churn from the routing to the mapping system. In fact, this is a feature: *HAIR* enables us to separate a currently intertwined functionality into two systems which can now be designed for simpler tasks and be tuned separately. Moreover, the mapping service can be offered on commodity systems at relatively low cost. It does not require expensive router hardware.

7.3.4 Dynamics

For a new routing architecture to be successful it has to handle network dynamics appropriately. Accordingly, we now discuss the scenarios that require updates: 1. if a node or a link other than an attachment point is temporarily or permanently unreachable due to a network event, 2. if an attachment point is unreachable, and 3. if a node moves to another network and thus changes its locator.

Link/Router failure: Let us presume that node v or link (v,w) inside the routing domain D, either a *INT* or *CORE*, failed. If the routing protocol inside D is able to find an alternative route between all pairs of attachment points, then no information has to be communicated beyond the routing domain D. Hence, *HAIR*'s hierarchical design ensures local visibility of routing updates. In general, even routing updates inside the *CORE* are localized in scope to the *CORE* and thus only affect the transit providers participating in the *CORE*.

Failure or unreachable attachment point: One of the main drawbacks of using loose source routing in order to reach a locator is that if one of the entries is unavailable the route is no longer valid. In such a situation the mapping system and the hosts that use a locator that includes the failed

attachment point have to be updated. But note, no routing update has to be propagated beyond the affected routing domain.

Regarding updating the mapping system, we point out that the *IMS* for a domain is responsible for choosing the sequence of attachment points. Hence, the *IMS* can monitor its attachment points and, if they change, update its part of the mapping system. This approach can be augmented with a mechanism where end hosts that notice an unreachable attachment point in their locator may notify the appropriate *IMS*.

Updating the locators for ongoing sessions is slightly more complicated as end hosts have to be able to detect that an attachment point is no longer reachable. This can be done either via explicit signaling (e.g., ICMP), by a timeout, or by monitoring reply packets. Alternatively, a router close to the attachment point replaces the entry for the failed one with a valid one towards the next level *IMS* or *EDGE*, if it is still reachable. Upon detection the end host can switch to an alternative locator, either by issuing a new query or by using an alternative locator previously retrieved from the mapping system. In addition, the host should inform the mapping system. We enable our mapping system to return multiple locators for a given identifier. One advantage of this is that the two endpoints are then able to switch locators without issuing updates/queries to the mapping system [AB09].

Change of locator Locator changes occur either when a node moves to another network or when an *IMS* changes the mapping, e.g., for traffic engineering purposes. In the latter case the updates automatically propagate to those hosts that do not have an entry in their cache or that update their cached entries.

The former case differs from the attachment point failure case as the host is usually aware of the change. Hence, upon detection, the host can notify any end points of active transport sessions by sending them a packet with its identifier and its new locator. Since this information is included in every packet anyway, it is easy to piggyback the notification (e.g., in a TCP ack packet). Therefore, movements of sessions will incur a negligible delay. After all, there is no need to wait for neither the routing nor the mapping system.

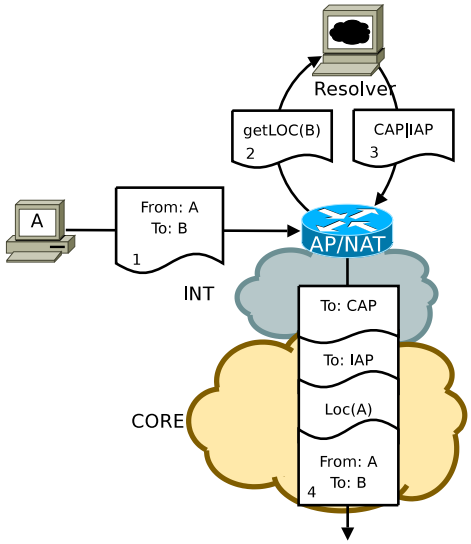
Still, the host has to inform the mapping system about the new locator if the previous locator is no longer available. Otherwise, the host could be no longer reachable by newly initiated connections. In terms of updates to the mapping systems, this implies an update to the *IMS* anytime the hosts moves within the *INT*, and updates to the *IMS* and eventually the global DHT if it moves beyond the *IMS*. Yet, given our assumption that most hosts stay within the range of a *INT*, the update frequency can be expected to be manageable by the DHT.

7.3.5 Migration Path

As presented so far the *HAIR* architecture requires changes to network devices at the attachment points and to *all* end-systems. As such a change is not going to happen overnight, we have to be able to support legacy devices, i.e., hosts that do not use *HAIR* in native mode. To enable the transition to *HAIR*, we propose to use *AP/NAT* boxes as middle-boxes, see Figure 7.6.

If an *EDGE* chooses to support legacy hosts, e.g., host *A*, it has to either include a *AP/NAT* box or to know how to redirect packets to such a box. For host *A* to send a packet to host *B*, it generates the packet using a traditional IP address for *B*. This packet is intercepted by the *AP/NAT* box, which then obtains the *HAIR* locator for *B*, adds it to the packet, e.g., by IP-in-IP encapsulation, and then

Figure 7.6 AP/NAT box.



sends it on. Upon receiving a packet destined to *A*, it strips the locator and sends it onward to *A* using traditional IP.

In effect, the *AP/NAT* box is acting as a NAT box as well as a *HAIR* AP. By pushing the *AP/NAT* box as close to the end-system as possible, we keep the system scalable and avoid the need for extensive state management – during the locator resolution step the packets have to be buffered. We decided to buffer the packets rather than sending them via the mapping system [FFM09] to keep the separation between routing and mapping system.

Regarding the transition of *INTs* from legacy to *HAIR*, we point out that it is easy to accommodate traffic from an upgraded to a legacy *INT* by using fallback mappings where identifiers and locators are the same. The reverse direction can be handled either by globally routable identifiers or translation mechanisms.

A short overview of our prototype implementation of the *AP/NAT* box is shown in Section 7.4.3 and is available via the *HAIR* Web page. To deploy this in an incremental manner, we need to deploy *CAPs* at the transit providers and one *AP/NAT* box inside each *EDGE* with legacy hosts. For the latter we propose to leverage existing devices. Today, almost all hosts either connect to the Internet via a NAT box, a firewall, or a wireless gateway. By modifying their firmware and then upgrading them, it will be possible to cover most end hosts at very reasonable costs. Moreover, our first experiences with our prototype implementation are very promising.

7.4 Evaluation

After discussing why *HAIR* meets the requirements for a future Internet routing system, we evaluate the potential benefits of deploying it. Finally, we give an overview of our proof-of-concept implementation of both native as well as legacy mode.

7.4.1 Requirement Evaluation

Scalability of the routing system: As most *EDGEs* and *INTs* can be expected to be of reasonable size, we focus on the *CORE*. Within *HAIR* the size of a *CORE*'s routing table is determined by the number of *CAPs*. As core routers are part of the *CORE* while access routers form the local *INT*, we expect that each ISP will have a limited number of *CAPs*, e.g., a handful for each Point of Presence (PoP). Indeed, many of today's operators already control traffic at the PoP level [ABF+08]. Given that the number of PoPs even inside today's top tier providers is generally limited [SMW02, SBS08], we do not expect any scalability problems.

To not restrict current routing policy conventions, we propose to stick to BGP as the routing protocol in the *CORE*. Due to the different structure of the *CORE*, and its smaller size, i.e., only *CAPs*, we expect that the well-known BGP scalability problems regarding convergence can be handled. Routing updates from dynamic, fault-prone edge networks will not even pass the *INT* boundaries. Moreover, given the high level of physical redundancy that is inherent to the *CORE* [LAF99], the need for routing updates is greatly reduced. In addition, it is no longer necessary to disaggregate address space for traffic engineering or multi-homing purposes. All in all, this should significantly reduce the number of updates visible to the *CORE*.

Finally, only the *CORE* needs to use a shared (public) address space. Therefore, we expect no shortage of routable addresses as it is currently the case with IPv4 addresses.

Scalability of the mapping system: Given that each *INT* operates its own mapping service, the number of entries per *IMS* is limited. The scalability of the global directory service and keeping costs reasonable is achieved by relying on a DHT. Moreover, the hierarchical design of our architecture ensures that updates to the mapping system are localized: Based on our assumption that changes between different *INTs* are infrequent compared to updates to the same *INT*, the update rate of the global DHT stays reasonable.

Mobility: Support for mobility is inherent in *HAIR* due to separation of locators and identifiers. Moreover, updates to the mapping system are initiated by the end-systems rather than by the network components, which ensures scalability, see Section 7.3.4.

Note, that it is even possible to set up agreements between cooperating *IMSs* to support "foreign" identifiers and thus avoid updates to the global DHT. In this case, if a host *H* moves from *INT A* to *INT B*, the *IMS* of *INT A* is notified by the *IMS* in *INT B* and updates the mapping for *H* accordingly, eliminating the need to inform the global DHT.

While seamless hand-overs during network changes are not the primary objective of our architecture, this is inherently possible as every packet carries both the locator and the identifier. Hence, a remote endpoint learns about the new locator as soon as the first "new" end-to-end data packet (the one with the new locator) arrives. This enables lightweight hand-overs with session survivability. For newly initiated connections, however, we need to wait until the mapping system has been updated.[7] *HAIR* also supports network mobility, e.g., assume a *INT* adds a new provider. This requires updating the local *IMS* to change some of the locators to include the new APs.

[7] Note, *HAIR* can be combined with existing approaches, e.g., MobileIP [Per02]

Multipath: With *HAIR* multipath routing is possible when more than one locator per identifier is available. Multipath routing can be used to improve throughput and availability of network paths [NB09].

Traffic engineering (TE): By choosing which locator(s) to assign to which destination, ISPs have a new knob for traffic engineering. Each *INT* can influence where traffic enters its network by simply changing the locators in its *IMS*. Such updates do not need time to converge as no routing protocol is involved. Hence, each *INT* can do inbound traffic engineering at a host granularity while still preventing other hosts from interfering with its network-wide TE policies. The latter is a deployment hindrance for [NB09]. On the other hand, a *INT* may delegate, e.g., for some special hosts or servers, the ability to influence how they are reachable. All in all, *HAIR* removes the need to burden the routing system in the core with hacks, such as more-specific prefixes, to enable at least a minimum form of inbound traffic engineering.

No changes to the core: *HAIR* adopts a hybrid *edge-based* solution. Most of the control is handed to the end-systems, respectively the edge networks. Yet, we give the ISPs the ability to control their traffic flow. Moreover, elements of the network-based approach, e.g., attachment points, are used to enable hierarchical routing and to address the scalability problems. Therefore, attachment points are the only routers in the core that have to be modified. One positive side effect of embedding both the locator and the identifier in the packet header is that routers never need to query the mapping system (e.g., not even when generating error messages). Consequently, buffering packets at routers is not necessary. To prevent DDoS attacks we are currently working on an address authentication mechanism.

Easy migration: The design of *HAIR* allows a smooth transition path where changes are kept to a minimum number of devices and that supports legacy systems.

Business incentives: *HAIR* provides different incentives to different players. Mobile edge networks can benefit from enhanced mobility. We expect this incentive to drive the early stages of deployment. The new way of enforcing TE policies via changing local mappings is expected to attract interest from content/hosting providers. Finally, by reducing routing table sizes and routing overhead in the *CORE* area, *HAIR* is a real alternative for ISPs.

7.4.2 Estimation of Benefits

Evaluating the impact of an architectural proposal is intrinsically difficult. Nevertheless, we now estimate the benefit to the routing system if *HAIR* were deployed in today's Internet. Thereby, we focus on the following questions: How much is the DFZ routing table size reduced? To what extent can the *INTs* isolate the *CORE* from update churn originated at the edge?

Data Sets

For our analysis we rely on two data sources: 1. a classification of the ASs [DD08] according to their type of business: transit provider, enterprise networks, or content provider, and 2. BGP updates and table dumps from Internet observation points. The latter reveals information about routing table size and update churn in today's Internet. The former allows us to distinguish "core" from "edge" ASs.

Table 7.2 AS types according to[DD08], # observation points per type.

AS type	# ASs	# obs. points
Enterprise Customers (EC)	31,704	3
Small Transit Providers (STP)	1,663	44
Large Transit Providers (LTP)	30	17
Content/Access/Hosting Providers (CAHP)	979	116
Total	34,376	180

Based on manual classification, an initial training set, and machine-learning techniques, Dhamdhere et al. [DD08] distinguish four types of ASs: Enterprise Customers (EC), Small Transit Providers (STP), Large Transit Providers (LTP), and Content/Access/Hosting Providers (CAHP). Most ASs ($31,704$) are enterprise customers or small transit providers ($1,663$), see Table 7.2.

Our second data source are the BGP collectors from RIPE [RIP]: *rrc00*, *rrc01* and *rrc03*. We obtain data from 180 observation points in more than 142 ASs of different types, see Table 7.2. Most observation points are inside CAHP ASs while only 3 observation points are inside enterprise networks. In addition to snapshots of routing tables from November 1st 2008, we analyze BGP updates for the first week in November of the years 2001 to 2008.

While both data sources are not 100% accurate, they still allow us to estimate the benefits of *HAIR*. The BGP data only includes a limited number of observation points and constrains the visibility of the actual Internet topology. Moreover, misclassifications of ASs are possible. Nevertheless, Dhamdhere et al. [DD08] report accuracies between 78% and 86% for EC, STPs and CAHPs ASs[8].

Routing Table Size

To evaluate the benefits of *HAIR*[9] on the routing table size, we have to identify the pieces of the current Internet that would form the 3-layers. As proposed in Section 7.3.1 we assume that the *CORE* includes all large and small transit providers (LTPs and STPs) while each CAHP and EC forms separate *INTs*.

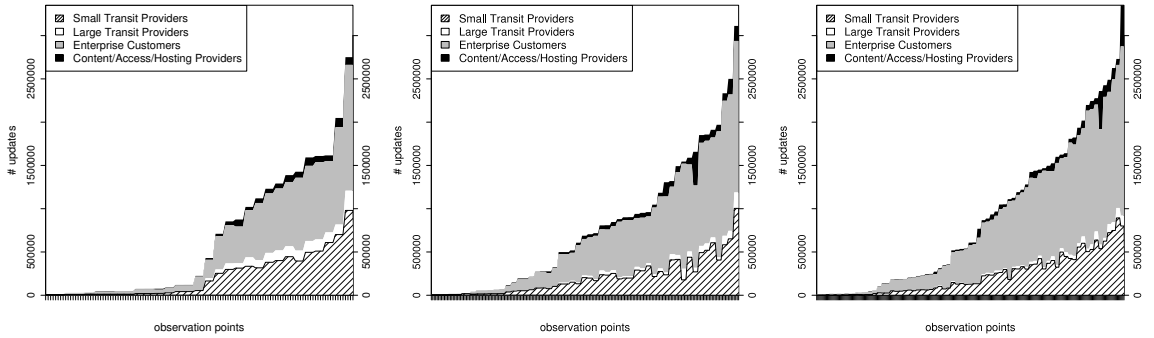
Given the $1,700$ STP and LTP ASs that are then part of the *CORE* and the assumption that each operates, on average, less than 100 Points of Presences (PoPs) [SMW02, SBS08] one may estimate that the total number of locators that need to be routable inside the *CORE* is less than $1,700 \cdot 100 \approx 170,000$. Even if some large ASs have more PoPs or more APs per PoP, others are likely to be significantly smaller. As such we now have $\approx 170,000$ locators rather than $300,000$ prefixes. This is a considerable improvement over the current status, in particular as recent work [DD08] suggests the Internet mostly grows at the Internet edge (EC and CAHP domains) while the number of transit providers is stabilizing. Moreover, the PoPs of different ISPs frequently overlap, i.e., are in the same location. By performing prefix aggregation in the *CORE*, or by adding additional levels to the hierarchy, routing table size can be shrunk even further.

Another reason why *HAIR* helps to reduce the size of routing tables is that it removes the motivation to inject more-specific prefixes in the routing protocol. Multihoming and inbound traffic engineering are examples of network practices that put pressure on routing tables by injecting more-

[8] LTPs are manually identified and thus 100% correct according to definition.

[9] For simplicity we consider only a 3-layer deployment.

Figure 7.7 Number of updates per type across multiple observation points observed over one week.



(a) 1st week of November, 2001 (b) 1st week of November, 2004 (c) 1st week of November, 2008

specific prefixes. Using a routing table snapshot of a tier-1 provider from November 2008 we find that 57% of all prefixes are subprefixes for which we find a prefix that is "less specific", i.e., for which the address range contains the subprefix. Upon deployment of *HAIR*, most of these prefixes should disappear since these tasks can be achieved more comfortably by tuning the locally administered mapping service, i.e., the *IMS*.

Update Churn

Next, we study how effectively *HAIR* can keep updates local. For this we rely on the updates collected each year in November from 2001 to 2008. For each update trace we iterate over all updates and check whether the update affects a prefix originated by a LTP, STP, EC or CAHP domain. Our assumption is that an update will not be visible in the *CORE*, formed by LTP and STP domains if it affects a prefix from the EC or CAHP domains.

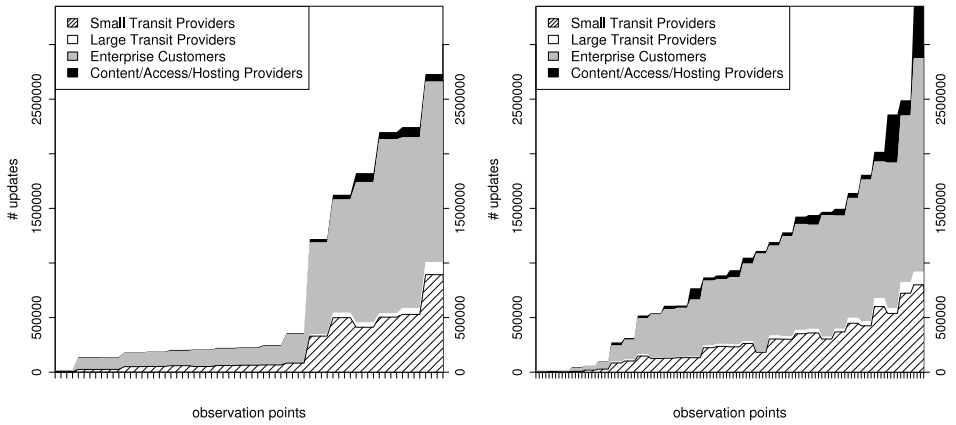
Figure 7.7 shows the number of updates observed at our observation points for each of our four types of ASs. As some observation points only propagate updates for a small subset of prefixes, we sort the observation points by the number of prefixes they receive along the x-axis. The stacked area plot partitions the number of updates, seen at each of our observation points, into the four categories LTP, STP, EC or CAHP. Hence, the y-value represents the total number of updates, seen at an observation point.

Figure 7.7c reveals that a large fraction of the updates – for most observation points more than 60% – are due to prefixes originated by enterprise networks (EC). This number is significantly lower for STP (around 30%) and LTP networks (around 5%). This indicates a considerable reduction in updates rates for routers in the *CORE* if *HAIR* were deployed in today's Internet. Looking back in time (2001 to 2008) we see that similar observations hold.

However, a closer study of the historical evolution, see Figure 7.7, shows that increasingly more updates are originated by EC domains while the number of updates from STP is more or less stable: For the majority of observation points the fraction of updates from EC domains is 60% in 2008, yet slightly less than 50% in 2001. This shift is largely a consequence of the fact that the Internet is growing much faster at the edge than in the core [DD08].

Finally, we explore if the location of an observation point affects its perception of update churn. For this purpose, we determine for each of our 180 observation points its location, in LTP, STP, EC or CAHP. Figure 7.8a and b show the corresponding plots for only considering observation points

Figure 7.8 Sensitivity to the location of observation points.



(a) collectors in LTP ASs (b) collectors in CAHP ASs

in LTP or CAHP. Both types of observation point see a considerable number of updates for prefixes originated by EC domains. Hence, not only edge routers in CAHP or EC networks can benefit from *HAIR* but also routers in large transit providers –those that currently suffer most from poor scalability, see Figure 7.8a.

7.4.3 Proof-of-Concept Implementation

It is not our goal yet to present a final implementation for *HAIR*. Rather, our proof-of-concept implementation, which is available online,[10] demonstrates the feasibility of our approach in general.

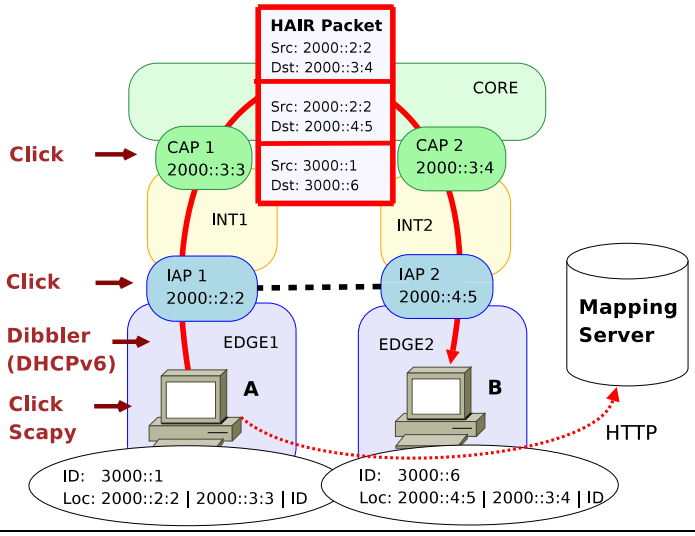
As shown by our test setup, see Figure 7.9, we implement *all forwarding elements*, namely two end hosts *A* and *B*, the *IAP* (*IAP*1 and *IAP*2), and the *CAP* (*CAP*1 and *CAP*2) devices. One major stumbling block of our implementation is *transparency to the transport protocol layer*. We decide to use IPv6 addresses for both locators and identifiers[11]. This allows us to use standard IP forwarding and existing network applications, e.g., BGP in the core. Keeping as much functionality as possible in the *user space* is a major concern for our proof-of-concept implementation. After all, this will facilitate larger setups of *HAIR* in testbeds such as PlanetLab in the future. Finally, our implementation incorporates means for easy *bootstrapping*: When an end host enters a *EDGE*, it automatically obtains its *CAPs* and *IAPs* via DHCP-like mechanisms and thus can compose its locator.

To achieve our goal of sending packets from *A* to *B*, we have implemented 3 components: End hosts, *IAP* and *CAP*. At *end hosts* all packets follow a default route to a `tun` interface and are passed to user space. If the locator for the destination identifier in the packet is unknown, an HTTP-like query is sent to a well-known mapping server and resolved mappings are cached locally. For now the mapping service only keeps mappings in a text file. As soon as the locator of the destination is known, we generate the packet, appending the locators of the *CAP* and *IAPs* to the packet header. Whenever a node joins a *EDGE*, it obtains information about its gateways, i.e., the *CAPs* and *IAPs* from a DHCPv6 server (`Dibbler`), which is running at the *IAP* gateway. Moreover, both the *IAP* and the *CORE* run instances of `Click` [MKJK99], a customizable software router, and forward the

[10] http://sites.google.com/site/hairarchsite

[11] Locator and identifier namespaces are nevertheless disjoint.

Figure 7.9 *HAIR*: Proof-of-Concept implementation



packet based on the locators of the incoming packet. We point out that our implementation also supports direct peerings such as the dashed thick link between *IAP*1 and *IAP*2.

Based on the setup of Figure 7.9, we run tests with various applications. When using `ping6` from *A* to *B*, we only see a higher latency for the first ICMP request. This can be explained by the required lookup to the mapping system. When host *A* moves during an ongoing file copy with `scp` to *EDGE*2, the session is preserved. Overall, the deployment of *HAIR* only requires changes to a small number of devices and can be done using existing pieces of software.

7.5 Summary

In this chapter, we have introduced *HAIR*, a scalable routing architecture for the future Internet. Our primary concern is scalability. In particular, we want to prevent updates from being globally visible and to provide scalable and simple solutions for traffic engineering, mobility, multipath, etc. *HAIR* is based on the following ideas: 1. use of hierarchical routing, 2. separation of locators from identifiers, and 3. use of a hierarchical mapping system. It uses a hybrid "edge-based" approach to reconcile network- and host-based routing architectures and to enable migration.

The contribution of *HAIR* is a comprehensive discussion of the design tradeoffs and to combine existing and novel ideas into an overall routing architecture. To evaluate the potential benefits we show that if *HAIR* were deployed in today's Internet the routing table sizes as well as the update load could be substantially reduced. Moreover, our experiences with a proof-of-concept implementation show that support for mobility and traffic engineering are indeed inherent in *HAIR*, and a smooth deployment roadmap that supports legacy systems is feasible. Our future focus will be on the implementation and evaluation of the mapping service. Moreover, we are working on a security model and analysis for *HAIR*.

8 Conclusions

8.1 Summary

The results presented in this thesis provide a variety of insights into Internet routing and demonstrate ways to build sophisticated models of inter-domain routing and to design scalable routing architectures for the future Internet.

One major concern of this thesis is to gain a deeper understanding of routing in general by analyzing how sensitive route optimality and diversity are to various factors such as policies or intra-domain topology, see Chapter 3. The obtained insights are crucial for constructing scalable and meaningful models of inter-domain routing that incorporate predictive capabilities, see Chapter 4 and Chapter 5. In the light of recent interest in designing a new and better Internet using "clean-slate" approaches [Fel07], we propose a platform for hosting virtual networks on commodity hardware (Chapter 6), that can be used for the evaluation of new protocols or architectures. Finally, we introduce a scalable routing architecture for the future Internet (Chapter 7).

We start in Chapter 3 with a comprehensive study of how sensitive routing optimality and diversity metrics are to factors such as policies, AS size, topology, and IGP weights etc. Surprisingly, we find that intra-domain factors only have marginal impact on global path properties – for example no setting of BGP parameters decreases the geographic path stretch significantly. In contrast, routing policies and AS size in number of routers are the dominating factors. All in all, our results reveal that it is hard to improve the global properties of route selection by purely tweaking BGP attributes or changing iBGP graphs, etc. This is consistent with inter-domain routing design, which mainly supports the flexible implementation of routing policies, e.g., for enforcing business objectives but not the propagation of optimal paths. Improving the global properties of Internet paths will require more than tuning BGP. Our sensitivity analysis is an important step towards understanding which parameters impact the optimality of inter-domain routing in what manner. This is crucial for a wide variety of tasks. With respect to clean-slate approaches for the future Internet, both the sensitivity analysis and the simulation framework that we propose will prove useful for evaluating and comparing new routing protocols and architectures.

There is a more direct benefit from our sensitivity analysis as well. We find that routing policies and AS size in number of routers are the dominating factors. This insight is key for constructing scalable and meaningful routing models that incorporate intra- and inter-domain information at the minimum level-of-detail required to explain observed routing in the Internet, even allowing for the prediction of routes. Such routing models are needed for quite a number of networking tasks ("what-if questions"), e.g., making decisions about peering relationships, choice of upstream providers or inter-domain traffic engineering.

Our concepts and findings in Chapter 4 and Chapter 5 are milestones on the way to more powerful inter-domain routing models. First, we show the importance of considering more than one router

per AS and introduce quasi-routers to capture path diversity as seen in observed routing data. Relying on the abstraction of quasi-routers, we show that models can predict previously unobserved AS paths with an accuracy of more than 80%, ignoring tie-breaks. Second, regarding routing policies our work reveals that the granularity of actual routing policies is close to per-neighbor. Although the per-neighbor based AS business relationships fail to provide consistency between the routes propagated in our routing model and those seen in observed data, they provide the right abstraction to prevent unnecessary paths from propagating in a model of the Internet. All in all, the fundamental ingredients to an inter-domain routing model are a quasi-router-level topology combined with routing policies on a per-neighbor granularity.

Suggesting a major overhaul of today's Internet has become a popular research topic given several shortcomings of the Internet such as routing table growth, high update rates, lack of traffic engineering, or security. Indeed, clean-slate approaches to Internet design and the development of new routing protocols rely on the premise that we know what features are important in the current routing system. In this respect, our sensitivity analysis also provides indispensable insights and offers a simulation framework that can be used to evaluate and compare any new protocol or architecture.

Our research in the context of a clean-slate design of the Internet consists of two distinctive parts. First, we present *Trellis*, a network testbed that can be used by network researchers to implement, test, and evaluate (clean-slate) solutions for the Internet of the future. As such, it allows to partition a physical network into multiple logical or virtual networks, where each virtual network can define its own topology, routing protocols, and forwarding tables. In principle the proposed virtualization techniques also lower the barrier for enterprises and service providers to define custom networks that are tailored to specific applications for users. The major objectives of our work are to provide complete isolation between the different logical or virtual networks. *Trellis* integrates host and network stack virtualization with tunneling technologies and our own components, EGRE tunnels and shortbridge, to create a coherent framework for building fast, flexible virtual networks. The evaluation of our system demonstrates that it is possible to build network testbeds relying on cheap commodity hardware and to obtain reasonable performance at the same time. Many of the concepts and techniques that we introduced with Trellis have flown into the design of VINI[1], a virtual network infrastructure with 37 nodes at 22 sites, distributed across the US.

Second, we introduce *HAIR*, a scalable routing architecture for the future Internet. Our primary concern is scalability. In particular, we want to prevent updates from being globally visible and to provide scalable and simple solutions for traffic engineering, mobility, multipath, etc. *HAIR* is based on the following ideas: 1. use of hierarchical routing, 2. separation of locators from identifiers, and 3. use of a hierarchical mapping system. It uses a hybrid "edge-based" approach to reconcile network- and host-based routing architectures and to enable migration. The contribution of this work is a comprehensive discussion of the design tradeoffs and to combine existing and novel ideas into an overall routing architecture. To evaluate the potential benefits we show that if *HAIR* were deployed in today's Internet, the routing table sizes as well as the update load could be substantially reduced. Moreover, our experiences with a proof-of-concept implementation show that support for mobility and traffic engineering are indeed inherent in *HAIR*, and a smooth deployment roadmap that supports legacy systems is feasible.

[1] http://www.vini-veritas.net

8.2 Directions for Future Work

The results of this thesis suggest several promising directions for future work, including *inter-domain routing models for answering "what-if questions"* as well as the *design of routing architectures for the future Internet*.

8.2.1 Inter-domain Routing Models

With respect to the former, we plan to continue our work on improving the existing routing models with the goal of increasing the accuracy to predict routes in the Internet. Our ultimate goal, which is not fully achieved yet, is to enable network administrators to answer what-if questions such as "what if a certain peering link was removed" or "what if we change policies thus"? According to the findings of this thesis, the core ingredients to such an inter-domain routing model are a quasi-router-level topology combined with policies that are defined on a per-neighbor basis.

Two further aspects are crucial for future research in this area. Frequently, inter-domain routing models will be applied from the local perspective of a specific ISP. Therefore, inter-domain routing routing models should be refined with local details about topology and routing policies. After all, a specific network administrator knows the internal topology of his own domain and the actual policies applied for peerings with neighbor ASs. Therefore, an inter-domain routing model, which accurately reproduces the local topology and policies with a high level-of-detail, but only incorporates some details about remote Internet domains, will be useful for many what-if questions.

Second, we need to evaluate the capabilities of routing models to answer what-if questions. For this purpose, we propose to identify a routing event in observed routing data (e.g., a de-peering). Then, we build an inter-domain routing model based on observed routing tables or updates from before that event. Finally, we replay the routing event in our model and check if the prediction of our routing model matches what we see in observed routing tables or updates from the time after that event.

8.2.2 Routing Architectures for the Future Internet

With respect to the design of alternative routing architectures Internet, we plan to further develop HAIR.

Amongst others, we intend to refine our *proof-of-concept implementation*, see Section 7.4.3. For example, we extend our current implementation with better support for network dynamics such as link/router failures, unreachable attachment points, or locator changes. Future test cases will include more complex *CORE* and *INT* topologies to test the interactions with routing protocols such as BGP and OSPF. Moreover, we plan to quantify the overhead, e.g., in terms of RTT, if we compare HAIR against the legacy Internet.

To avoid the trap of moving the full burden from the routing to the mapping system, we envision a careful study of the design choices and tradeoffs of different mapping approaches. Based on this, we hope to come up with a more detailed specification of a mapping system that satisfies the needs of HAIR. The core design requirements include high scalability in terms of number of entries, i.e., key-value pairs in the global DHT, control of mappings by authorities which own the mappings, and support for host and network mobility.

Finally, we are working on a security model and analysis for *HAIR*. For example, it is critical to control access to the mapping system, i.e., who is able to change, insert or update mappings.

Acknowledgments

Probably, acknowledgments are the most challenging part of any Ph.D. thesis. In any case, I will do my very best to not forget anyone in the following.

Without any doubt, I am indebted most of all to my advisor Anja Feldmann. As a member of her research group at TU München and later at TU Berlin/T-labs, I learned all the basics of research, in particular how to approach research topics and how to write scientific papers. I am extremely grateful that she pointed me to people working on similar research topics, and thereby helped me to get in touch with the research community. Whenever I was stuck in a problem, she had time for discussions and inspired me with ideas how to solve them. Without the mutual trust it would have been difficult to partly work from my remote office in Munich. Overall, it was more than a fortunate coincidence that I had met Jörg Liebeherr during a semester abroad at University of Virginia in 2003 and that he had recommended me to work with Anja.

Moreover, I would like to express my cordial thanks to Jennifer Rexford. She made it possible to do an internship at Princeton University where I worked on an interesting project together with great people, including Andy Bavier, Murtaza Motiwala, Larry Peterson, Vytautas Valancius, Nick Feamster, just to name a few. I have always been fascinated by Jennifer's readiness to help others and by her prompt replies to my e-mails. Without doubt, it is a great honor to have her in my Ph.D. committee.

My deep gratitude goes to Olaf Maennel, Steve Uhlig, and Bruno Quoitin, who have accompanied my work for the last few years. It is really a luxury to have so many "Ph.D. advisors".

I already got to know Olaf during my master at TU München where he advised a student project and my diploma thesis. His fascination for routing and BGP infected me soon, and thanks to Olaf I had the smoothest start possible into my Ph.D. His readiness to help and to discuss problems is unexcelled. With Steve Uhlig, the three of us formed the successful BGP team "Plip-Plap-Rrrums".

Equally, I owe a lot to Steve Uhlig. For me it has always been great fun to have metaphysical discussions with Steve. He was never angry about "*I do not concur*" or about "*Je fais la grève*" replies, and continuously inspired me with new ideas. There are not many researchers who succeed in combining humor with high-quality research in the same way as Steve. Merci beaucoup pour la excellente coopération!

And, of course, not to forget Bruno Quoitin from Université Catholique de Louvain-la-Neuve, Belgium. Without tools such as C-BGP or IGen, this thesis would not have been possible. I admire Bruno's enthusiasm for implementing code and his thoroughness when reviewing papers. Special thanks goes to Bruno, Olivier Bonaventure and his group who were always perfect hosts whenever I visited Louvain-la-Neuve.

Certainly, it is possible to double the length of this thesis by thanking everyone in Anja's research group at TU München and TU Berlin/T-Labs individually. *All* of my colleagues contributed to a very enjoyable working atmosphere. At this point, I would like to apologize for my live commentatorship during some Foosball matches. A special thanks goes to Petra Lorenz and Britta Liebscher who helped me with bureaucracy and made sure that I'm reimbursed for my business trips. Thanks also to Nils Kammenhuber and Vinay Aggarwal who provided me the text sources of their thesis and

thus alleviated my work a lot. And, not to forget Vinay Aggarwal and Gregor Maier who offered me to stay in their flats while being in Berlin. I will definitely not forget the interesting discussions and the good food that I had at Vinay's place. Last but not least, I would like to acknowledge Fabian Schneider and Deti Fliegl who both suffered from sharing office with me, and my students whose theses and projects helped me a lot for my own work.

For the last three years I have been enjoying the luxury of two offices, one at T-Labs/TU Berlin and one at TU München. I'm indebted to Georg Carle and Thomas Fuhrmann for providing me an office at TU München in Garching. Whenever I worked in my Garching office, I enjoyed discussions during lunch and coffee breaks. Meeting good friends at TU München, in particular the participants of the "K@ffchen", was my main motivation for going to Garching.

Without participating in Albert's Krafttraining, without hiking trips, without skiing, without having a good beer in one of Munich's "Biergärten", life would have been miserable. Thanks a lot to Achim, Bene, Deti, Katrin, Naffy, Stefan and all "Freunde des Klettersports"!

Last but not least, I thank all my friends, my sister and my parents for their support in all these years!

Vergelt's Gott!

List of Figures

2.1 A small Internet: tiered structure, routing policies, intra- and inter-domain routing, etc. . . . 6
2.2 Layout of a BGP router, BGP decision process. . . . 11
2.3 BGP route collection. . . . 12

3.1 Sensitivity analysis: components of the simulation framework. . . . 21
3.2 Contribution of individual factors to total variation (sum of squares). . . . 24
3.3 Sensitivity analysis: differences between tier-1, tier-2 and stub ASs for *defaultSim*. 28
3.4 AS-level path stretch for *defaultSim*. . . . 29
3.5 Optimality of router-level paths in terms of router-level hops. . . . 29
3.6 Optimality of routes in terms of geographical length. . . . 30

4.1 Histogram of # of distinct AS paths between pairs of originating and observation ASs. 36
4.2 Example of path diversity observed in the Internet. . . . 37
4.3 Example for metrics to measure agreement between observed and simulated routes. 40
4.4 Example for model refinement: applying changes at AS 1 for prefixes $p1$, $p2$. . . . 41
4.5 *Refinement heuristic*: necessity of filter deletion. . . . 46
4.6 *Reuse policy heuristic*: transfer of policies across prefixes. . . . 47
4.7 *Refinement heuristic*: results 49
4.8 *Refinement heuristic*: number of quasi-routers per AS (ignoring 14,305 ASs with one quasi-router) 50

5.1 Number of quasi-routers per AS that are required according to observed data. . . . 55
5.2 Relationship between number of neighboring ASs and number of quasi-routers. . . . 56
5.3 Atoms structure for dataset of Section 4.2.1. . . . 57
5.4 Inferring candidates for filtering policies - Example. . . . 59
5.5 Dependency graph for mismatch in Figure 5.4. . . . 61
5.6 Popularity of candidate filtering policies for `psample`. . . . 64
5.7 Next-hop atoms - example. . . . 66
5.8 Number of next-hop atoms per AS. . . . 66
5.9 Neighboring ASs in next-hop atoms. . . . 67
5.10 Number of neighboring ASs in next-hop atoms for large ASs. . . . 67
5.11 AS relationships: freedom in path choice for observed paths. . . . 70

6.1 Overview of *Trellis* design. . . . 75
6.2 High speed forwarding using shortbridges. . . . 79
6.3 Experiment setup. Each setup has a source, a sink and a node-under-test. . . . 80
6.4 Trellis: peak forwarding performance (in pps) with 64-byte packets. . . . 81
6.5 Trellis: peak forwarding rate (in pps) for different packet sizes. . . . 82

7.1 *HAIR*– Basic principle. 87
7.2 Evolution of BGP forwarding table size observed for AS 3333 at RIPE *rrc00*. . . . 88
7.3 Hourly peak update rates per second in November 2008, observed for AS 3549 at RIPE *rrc00*. 89
7.4 N-layer hierarchical network structure. 93
7.5 Packet forwarding in *HAIR*. 95
7.6 AP/NAT box. 99
7.7 Number of updates per type across multiple observation points observed over one week. 103
7.8 Sensitivity to the location of observation points. 104
7.9 *HAIR*: Proof-of-Concept implementation 105

List of Tables

2.1 AS relationship inference: evaluation of various algorithms. 15

3.1 Parameter choice for *defaultSim*. 23
3.2 ANOVA – percentage of variation explained by individual factors ("%") and statistical significance (F-Value, "F") for all factors and metrics. 25
3.3 Impact of policies, intra-domain configuration and inter-domain connectivity on route selection. 26

4.1 Maximum route diversity received for all ASs. 37
4.2 Agreement between predicted and observed AS paths (single router per AS). 38
4.3 Effectiveness of *refinement heuristic* by uncovered AS-edges. 51

5.1 Statistics on observed paths towards sample prefix set "`psample`". 63
5.2 Number of candidate filters per mismatch for `psample`. 63
5.3 Popularity of candidate filtering policies for recursion depth 1 and 2 in `psample`. . 64
5.4 AS relationships: agreement between observed and simulated routes. 68
5.5 Comparing AS relationships with popular candidate filters. 69

6.1 Container-based virtualization vs. full virtualization. 77

7.1 Some design choices for locator/identifier (LOC/ID) separation. 90
7.2 AS types according to[DD08], # observation points per type. 102

List of Algorithms

1 Routing models with predictive capabilities: methodology for model refinement . . 44

2 Detecting mismatches and computing candidate filtering policies. 62

Bibliography

[AAER06] B. Ahlgren, J. Arkko, L. Eggert, and J. Rajahalme. A Node Identity Internetworking Architecture. In *Proc. IEEE Global Internet*, 2006.

[AB09] J. Arkko and I. Beijnum. Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming, RFC 5534. 2009.

[ABF+08] D. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable Internet Protocol (AIP). In *Proc. ACM SIGCOMM*, 2008.

[Abi] Abilene. The Abilene Observatory: Abilene Routing Data. `http://abilene.internet2.edu/observatory/`.

[ABSBB02] Y. Afek, O. Ben-Shalom, and A. Bremler-Barr. On the Structure and Application of BGP Policy Atoms. In *Proc. of the 2nd ACM SIGCOMM Workshop on Internet measurement*, 2002.

[ACBD04] S. Agarwal, C. Chuah, S. Bhattacharyya, and C. Diot. The Impact of BGP Dynamics on Intra-Domain Traffic. In *Proc. ACM SIGMETRICS*, 2004.

[ADGW03] D. Alderson, J. Doyle, R. Govindan, and W. Willinger. Towards an Optimization-Driven Framework for Designing and Generating Realistic Internet Topologies. In *Proc. ACM SIGCOMM CCR*, 2003.

[AFBB02] D. Andersen, N. Feamster, S. Bauer, and H. Balakrishnan. Topology Inference from BGP Routing Dynamics. In *Proc. ACM IMW*, 2002.

[Atk09] R. Atkinson. ILNP Concept of Operations. Internet Draft, draft-rja-ilnp-intro-02.txt, 2009.

[BC01a] A. Broido and K. Claffy. Analysis of RouteViews BGP Data: Policy Atoms. In *Proc. of the Network-Related Data Management Workshop*, 2001.

[BC01b] A. Broido and K. Claffy. Internet Topology: Connectivity of IP Graphs. In *Proc. of SPIE International Symposium on Convergence of IT and Communication*, 2001.

[BCC06] T. Bates, R. Chandra, and E. Chen. BGP Route Reflection - An Alternative to Full Mesh IBGP. Internet Engineering Task Force, RFC4456, 2006.

[BEH+07] G. Battista, T. Erlebach, A. Hall, M. Patrignani, M. Pizzonia, and T. Schank. Computing the Types of the Relationships between Autonomous Systems. *IEEE/ACM Transactions on Networking*, 2007.

[BFH+06] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In VINI Veritas: Realistic and Controlled Network Experimentation. In *Proc. ACM SIGCOMM*, 2006.

[BMRUar] R. Bush, O. Maennel, M. Roughan, and S. Uhlig. Internet Optometry: Assessing the Broken Glasses in Internet Reachability. In *Proc. ACM IMC*, 2009 (to appear).

[BMU07] M. Buob, M. Meulle, and S. Uhlig. Checking for Optimal Egress Points in iBGP routing. In *Proc. of the 6th IEEE International Workshop on the Design of Reliable Communication Networks (DRCN)*, 2007.

[BOR+02] A. Basu, C. Ong, A. Rasala, F. Shepherd, and G. Wilfong. Route Oscillations in I-BGP with route reflection. In *Proc. ACM SIGCOMM*, 2002.

[BPP03] G. Battista, M. Patrignani, and M. Pizzonia. Computing the types of the relationships between autonomous systems. In *Proc. IEEE INFOCOM*, 2003.

[BQ03] O. Bonaventure and B. Quoitin. Common Utilizations of the BGP Community Attribute, 2003. Internet Draft, draft-bq-bgp-communities-00.txt.

[Cai] The CAIDA Web Site. `http://www.caida.org/data/`.

[Cal90] R. Callon. Use of OSI IS-IS for Routing in TCP/IP and Dual Environments. IETF RFC1195, December 1990.

[CCK+06] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica. ROFL: Routing On Flat Labels. In *Proc. ACM SIGCOMM*, 2006.

[CCN+06] M. Caesar, M. Castro, E. Nightingale, G. O'Shea, and A. Rowstron. Virtual Ring Routing: Network Routing inspired by DHTs. In *Proc. ACM SIGCOMM*, 2006.

[CGH03] D. Chang, R. Govindan, and J. Heidemann. The Temporal and Topological Characteristics of BGP Path Changes. In *Proc. ICNP*, 2003.

[CGJ+04] H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Towards Capturing Representative AS-Level Internet Topologies. *Computer Networks*, 44(6), April 2004.

[CR05] M. Caesar and J. Rexford. BGP Routing Policies in ISP Networks. *IEEE Network Magazine*, 2005.

[Day08] J. Day. *Patterns in Network Architecture: a Return to Fundamentals*, pages 176,297–316. Prentice Hall, 2008.

[DD08] A. Dhamdhere and C. Dovrolis. Ten Years in the Evolution of the Internet Ecosystem. In *Proc. ACM IMC*, 2008.

[Dec] DE-CIX – German Internet Exchange. `http://www.de-cix.net/`.

[Dij59] E. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269, 1959.

[DKF+07] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, K. claffy, and G. Riley. AS Relationships: Inference and Validation. *Proc. ACM SIGCOMM CCR*, 37(1), 2007.

[EO07] A. Eriksson and B. Ohlmann. Dynamic Internetworking Based on Dynamic Locator Construction. In *Proc. IEEE Global Internet Symposium*, 2007.

[EU] EU. Seventh Framework Programme (FP7), ICT-2007.1.1 The Network of the Future. `http://cordis.europa.eu/fp7/`.

[FB05] N. Feamster and H. Balakrishnan. Detecting BGP Configuration Faults with Static Analysis. 2005.

[FBC+08] A. Feldmann, R. Bush, L. Cittadini, O. Maennel, and W. Mühlbauer. HAIR: Hierarchical Architecture for Internet Routing. Technical Report Rote Reihe Informatik 2008/14, Technische Universität Berlin, 2008.

[FBR03] N. Feamster, J. Borkenhagen, and J. Rexford. Guidelines for Interdomain Traffic Engineering. *Proc. ACM SIGCOMM CCR*, 2003.

[Fel07] A. Feldmann. Internet Clean-Slate Design: What and Why? In *Proc. ACM SIGCOMM CCR*, 2007.

[FFF99] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On Power-Law Relationships of the Internet Topology. In *Proc. ACM SIGCOMM*, 1999.

[FFM09] D. Farinacci, V. Fuller, and D. Meyer. LISP Alternative Topology (LISP+ALT). Internet draft, draft-fuller-lisp-alt-05.txt, 2009.

[FFO+09] D. Farinacci, V. Fuller, D. Oran, D. Meyer, and S. Brim. Locator/ID Separation Protocol (LISP). Internet draft, draft-farinacci-lisp-12.txt, 2009.

[FGR07] N. Feamster, L. Gao, and J. Rexford. How to Lease the Internet in your Spare Time. *Proc. ACM SIGCOMM CCR*, 37(1):61–64, 2007.

[FLH+00] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic Routing Encapsulation (GRE), RFC 2784. 2000.

[FMM+04] A. Feldmann, O Maennel, Z. M. Mao, A. Berger, and B. Maggs. Locating Internet Routing Instabilities. In *Proc. ACM SIGCOMM*, 2004.

[FMR04] N. Feamster, Z. Mao, and J. Rexford. BorderGuard: Detecting Cold Potatoes from Peers. In *Proc. ACM IMW*, 2004.

[FR07] N. Feamster and J. Rexford. Network-Wide Prediction of BGP Routes. 2007.

[FR09] A. Flavel and M. Roughan. Stable and Flexible iBGP. In *Proc. ACM SIGCOMM*, 2009.

[FT00] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *Proc. IEEE INFOCOM*, 2000.

[FWR04] N. Feamster, J. Winick, and J. Rexford. A Model of BGP Routing for Network Engineering. In *Proc. ACM SIGMETRICS*, 2004.

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

[Gav01] C. Gavoille. Routing in Distributed Networks: Overview and Open Problems. *SIGACT News*, 32(1):36–52, 2001.

[GSW99] T. Griffin, F. Shepherd, and G. Wilfong. Policy Disputes in Path Vector Protocols. In *Proc. ICNP*, 1999.

[GSW02] T. Griffin, F. Bruce Shepherd, and G. Wilfong. The Stable Paths Problem and Interdomain Routing. *IEEE/ACM Transactions on Networking*, 2002.

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

[GW02] T. Griffin and G. Wilfong. On the Correctness of IBGP Configuration. In *Proc. ACM SIGCOMM*, 2002.

[Hed88] C. Hedrick. Routing Information Protocol. IETF RFC1058, June 1988.

[Her] W. Herrin. Tunneling Route Reduction Protocol (TRRP). `http://bill.herrin.us/network/trrp.html`.

[HHK02] M. Handley, O. Hudson, and E. Kohler. XORP: An Open Platform for Network Research. In *Proc. HotNets*, 2002.

[HK07] B. Hummel and S. Kosub. Acyclic Type-of-Relationship Problems on the Internet: An Experimental Analysis. In *Proc. ACM IMC*, 2007.

[HKC+00] K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg, and J. Postel. Internet Registry IP Allocation Guidelines, RFC 2050. 2000.

[HP01] B. Halabi and D. Mc Pherson. *Internet Routing Architectures (2nd Edition)*. Cisco Press, 2001.

[HRI+08] H. Haddadi, M. Rio, G. Iannaccone, A. Moore, and R. Mortier. Network Topologies: Inference, Modeling, And Generation. *Communications Surveys & Tutorials, IEEE*, 2008.

[HRS+08] M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb, and J. Lepreau. Large-Scale Virtualization in the Emulab Network Testbed. In *Proc. USENIX*, 2008.

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

[ID] Intel-DANTE. Intel-DANTE monitoring project. `http://www.cambridge.intel-research.net/monitoring/dante/`.

[INT] Internet World Stats. `http://www.internetworldstats.com`.

[Jai91] R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, Inc., 1991.

[JMM+07] D. Jen, M. Meisel, D. Massey, L. Wang, B. Zhang, and L. Zhang. APT: A Practical Transit Mapping Service. Internet draft, draft-jen-apt-01.txt, 2007.

[JMY+08] D. Jen, M. Meisel, H. Yan, D. Massey, L. Wang, B. Zhang, and L. Zhang. Towards a New Internet Routing Architecture: Arguments for Separating Edges from Transit Core. In *Proc. HotNets*, 2008.

[Kas02] F. Kastenholz. ISLAY: A New Routing and Addressing Architecture. Internet Draft, draft-irtf-routing-islay-00.txt, 2002.

[KCNSZ03] S. Merugu K. Calvert, J. Eagan, A. Namjoshi, J. Stasko, and E. Zegura. Extending and Enhancing GT-ITM. In *Proc. ACM SIGCOMM workshop on models, methods and tools for reproducible network research*, 2003.

[KK77] L. Kleinrock and F. Kamoun. Hierarchical Routing for Large Networks: Performance Evaluation and Optimization. *Computer Networks*, 1:155–174, 1977.

[KR05] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons, 2005.

[LABJ00] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. An Experimental Study of Internet Routing Convergence. In *Proc. ACM SIGCOMM*, 2000.

[LAF99] C. Labovitz, A. Ahuja, and F. Fahanian. Experimental Study of Internet Stability and Backbone Failures. In *Proc. International Symposium on Fault-Tolerant Computing*, 1999.

[LAWD04] L. Li, D. Alderson, W. Willinger, and J. Doyle. A First-Principles Approach to Understanding the Internet's Router-level Topology. In *Proc. ACM SIGCOMM*, 2004.

[Lina] Linux BRIDGE-STP-HOWTO. `http://www.faqs.org/docs/Linux-HOWTO/BRIDGE-STP-HOWTO.html`.

[Linb] Linux Advanced Routing and Traffic Control. `http://lartc.org/`.

[Linc] Linux VServers Project. `http://linux-vserver.org/`.

[LMJ97] C. Labovitz, G. Malan, and F. Jahanian. Internet Routing Instability. In *Proc. ACM SIGCOMM*, 1997.

[LMJ99] C. Labovitz, R. Malan, and F. Jahanian. Origins of Internet Routing Instability. In *Proc. IEEE INFOCOM*, 1999.

[MC05] R. Musunuri and J. Cobb. An Overview of Solutions to Avoid Persistent BGP Divergence. *IEEE Network Magazine*, 2005.

[MCZ06] A. Menon, A. Cox, and W. Zwaenepoel. Optimizing Network Virtualization in Xen. In *Proc. USENIX*, 2006.

[MEFV08] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala. Path Splicing. In *Proc. ACM SIGCOMM*, 2008.

[MFM+06] W. Mühlbauer, A. Feldmann, O. Maennel, M. Roughan, and S. Uhlig. Building an AS-Topology Model that Captures Route Diversity. In *Proc. ACM SIGCOMM*, 2006.

[MIB08] L. Mathy, L. Iannone, and O. Bonaventure. LISP-DHT: Towards a DHT to Map Identifiers onto Locators. Internet draft, draft-mathy-lisp-dht-00.txt, 2008.

[MJR+04] Z. Mao, D. Johnson, J. Rexford, J. Wang, and R. Katz. Scalable and Accurate Identification of AS-Level Forwarding Paths. In *Proc. IEEE INFOCOM*, 2004.

[MKJK99] R. Morris, E. Kohler, J. Jannotti, and M. Kaashoek. The Click Modular Router. *SIGOPS Oper. Syst. Rev.*, 33(5):217–231, 1999.

[MLMB01] A. Medina, A. Lakhina, I. Matta, and J. Byers. An Approach to Universal Topology Generation. In *Proc. IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2001.

[MM06] J. Hao M. Meulle, Q. Nguyen. Formulation CSP et Approches Heuristiques pour l'Inférence des Accords d'Interconnexion dans l'Internet. In *Prof. of ROADEF'06*, 2006.

[MN06] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP), RFC 4423, 2006.

[Moy98] J. Moy. OSPF Version 2. IETF RFC2328, April 1998.

[MQWZ05] Z. Mao, L. Qiu, J. Wang, and Y. Zhang. On AS-level Path Inference. In *Proc. ACM SIGMETRICS*, 2005.

[MRWK03] Z. Mao, J. Rexford, J. Wang, and R. Katz. Towards an Accurate AS-Level Traceroute Tool. In *Proc. ACM SIGCOMM*, 2003.

[MSWA02] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. Inferring Link Weights using End-to-End Measurements. In *Proc. ACM IMW*, 2002.

[MUF+07] W. Mühlbauer, S. Uhlig, B. Fu, M. Meulle, and O. Maennel. In Search for an Appropriate Granularity to Model Routing Policy. In *Proc. ACM SIGCOMM*, 2007.

[NB09] E. Nordmark and M. Bagnulo. Shim6: Level 3 Multihoming Shim Protocol for IPv6, RFC 5533. Internet draft, draft-ietf-shim6-proto-11.txt, 2009.

[Net] Linux Containers—Network Namespace. `http://lxc.sourceforge.net/network.php`.

[Nor] W.B. Norton. The Art of Peering: The Peering Playbook. 2002.

[NSF] NSF. NeTS Initiative, Future Internet Design. `http://www.nets-find.net`.

[O'D97] M. O'Dell. GSE - An Alternate Addressing Architecture for IPv6. Internet draft, draft-ietf-ipngwg-gseaddr-00.txt, 1997.

[OPE] The OpenFlow Switch Consortium. `http://www.openflowswitch.org`.

[OPW+08] R. Oliveira, D. Pei, W. Willinger, B. Zhang, and L. Zhang. In Search of the Elusive Ground Truth: The Internet's AS-level Connectivity Structure. In *Proc. ACM SIGMETRICS*, 2008.

[OZZ07] R. Oliveira, B. Zhang, and L. Zhang. Observing the Evolution of Internet AS Topology. In *Proc. ACM SIGCOMM*, 2007.

[Per02] C. Perkins. IP Mobility Support for IPv4, RFC 3344. 2002.

[PKT] pktgen: Linux Packet Generator Tool. `http://linux-net.osdl.org/index.php/Pktgen`.

[Pro81] DARPA Internet Program. Internet Protocol. IETF RFC791, September 1981.

[PZW+07] P. Padala, X. Zhu, Z. Wang, S. Singhal, and K. Shin. Performance Evaluation of Virtualization Technologies for Server Consolidation. Technical Report HPL-2007-59, HP Labs, 2007.

[QU05] B. Quoitin and S. Uhlig. Modeling the Routing of an Autonomous System with C-BGP. *IEEE Network Magazine*, 2005.

[Qua] Quagga Software Routing Suite. `http://www.quagga.net/`.

[Quo05] B. Quoitin. Topology Generation Based on Network Design Heuristics. In *Proc. CoNEXT student workshop*, 2005.

[RFHK01] S. Ratnasamy, P. Francis, M. Handley, and R. Karp. A Scalable Content-Addressable Network. In *Proc. ACM SIGCOMM*, 2001.

[Ric] M. Rich. Literacy Debate: Online, R U Really Reading? New York Times, 2008.

[RIP] RIPE's Routing Information Service. `http://data.ris.ripe.net/`.

[RL06] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). IETF RFC4271, 2006.

[ROU] University of Oregon RouteViews Project. `http://www.routeviews.org/`.

[RRG] Internet Research Task Force – Routing Research Group. `http://tools.ietf.org/group/irtf/trac/wiki/RoutingResearchGroup`.

[RS04] B. Raghavan and A. Snoeren. A System for Authenticated Policy-Compliant Routing. In *Proc. ACM SIGCOMM*, 2004.

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

[SAZ+02] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet Indirection Infrastructure. In *Proc. ACM SIGCOMM*, 2002.

[SBS08] R. Sherwood, A. Bender, and N. Spring. DisCarte: A Disjunctive Internet Cartographer. In *Proc. ACM SIGCOMM*, 2008.

[SCE+05] L. Subramanian, M. Caesar, C. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica. HLP: A Next Generation Inter-domain Routing Protocol. In *Proc. ACM SIGCOMM*, 2005.

[SCH+99] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The End-to-End Effects of Internet Path Selection. In *Proc. ACM SIGCOMM*, 1999.

[SF04] G. Siganos and M. Faloutsos. Analyzing BGP Policies: Methodology and Tool. In *Proc. IEEE INFOCOM*, 2004.

[SFFF03] G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos. Powerlaws and the AS-level Internet Topology. *IEEE/ACM Transactions on Networking*, 11(4):514–524, 2003.

[SMA03] N. Spring, R. Mahajan, and T. Anderson. Quantifying the Causes of Path Inflation. In *Proc. ACM SIGCOMM*, 2003.

[SMK+01] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. ACM SIGCOMM*, 2001.

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

[SS05] Y. Shavitt and E. Shir. DIMES: Let the Internet Measure Itself. In *Proc. ACM SIGCOMM CCR*, 2005.

[SWP+09] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldman, R. Bless, A. Greenhalgh, M. Kind, O. Maennel, and L. Mathy. Network Virtualization Architecture: Proposal and Initial Prototype. In *Proc. of the First ACM SIGCOMM Workshop on Virtualized Infastructure Systems and Architectures*, 2009.

[T+07] J. Turner et al. Supercharging PlanetLab: A High Performance, Multi-Application, Overlay Network Platform. In *Proc. ACM SIGCOMM*, 2007.

[TDRR05] R. Teixeira, N. Duffield, J. Rexford, and M. Roughan. Traffic Matrix Reloaded: Impact of Routing Changes. In *Proc. ACM PAM*, 2005.

[TGSE01] H. Tangmunarunkit, R. Govindan, S. Shenker, and D. Estrin. The Impact of Internet Policy on Internet Paths. In *Proc. IEEE INFOCOM*, 2001.

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

[TMS01] P. Traina, D. McPherson, and J. Scudder. Autonomous System Confederations for BGP, RFC 3065, 2001.

[TMSV03] R. Teixeira, K. Marzullo, S. Savage, and G. Voelker. In Search of Path Diversity in ISP Networks. In *Proc. ACM IMC*, 2003.

[TSGR04] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of Hot-Potato Routing in IP Networks. In *Proc. ACM SIGMETRICS*, 2004.

[UT06] S. Uhlig and S. Tandel. Quantifying the Impact of Route-Reflection on BGP Routes Diversity inside a Tier-1 Network. In *Proc. of IFIP Networking*, 2006.

[VCG98] C. Villamizar, R. Chandra, and R. Govindan. BGP Route Flap Damping, RFC 2439, 1998.

[Vog08a] C. Vogt. Design Taxonomy and Analysis for Address-Indirection-Based Routing Scalability Improvements. 2008.

[Vog08b] C. Vogt. Six/One Router: A Scalable and Backwards Compatible Solution for Provider-Independent Addressing. In *Proc. MobiArch*, 2008.

[VTU] VTun - Virtual Tunnels.
`http://vtun.sourceforge.net`.

[WG03] F. Wang and L. Gao. Inferring and Characterizing Internet Routing Policies. In *Proc. ACM IMC*, 2003.

[WLS+02] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. 2002.

[YCB07] X. Yang, D. Clark, and A. Berger. NIRA: a new inter-domain routing architecture. *IEEE/ACM Transactions on Networking*, 15(4):775–788, 2007.

[ZB03] R. Zhang and M. Bartell. *BGP Design and Implementation*. Cisco Press, 2003.

[ZLMZ05] B. Zhang, R. Liu, D. Massey, and L. Zhang. Collecting the Internet AS-level Topology. *Proc. ACM SIGCOMM CCR*, 2005.

[ZPW+01] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. Felix Wu, and L. Zhang. An Analysis of BGP Multiple Origin AS (MOAS) Conflicts. In *Proc. ACM IMW*, 2001.

[ZZM+07] Y. Zhang, Z. Zhang, Z. Mao, C. Hu, and B. Maggs. On the Impact of Route Monitor Selection. In *Proc. ACM IMC*, 2007.

Index

Analysis of Variance, *see* ANOVA
ANOVA, 18
AS, *see* autonomous system
AS relationships, **13**, 20, 38, 58
    inference, 14
ASLength, 19
attachment point, 94
autonomous system, 5

BGP, 8, **10**
BGP atom, 57
Border, 20
Border Gateway Protocol, *see* BGP

C-BGP, **16**, 21, 39, 59, 68
c2p, *see* customer-provider
CAHP (content/access/hosting provider), 101
candidate filter, 60
clean slate, *see* future Internet
collector, 12, 16
container-based OS, 74
CORE, 94
core-edge separation, 92
customer-provider, 5, 14

decision process of BGP, 10
dependency graph, 60
Disjointness, 19

EC (enterprise customer), 101
EDGE, 94
edge-based, 95
EGRE, 77

factor, **18**, 20
factorial design, 18
filtering combination, 60
forwarding, 1
future Internet, 92

GeoLength, 19
granularity of routing policies, 57

HAIR, 85–105
heuristic
    reuse policy heuristic, 47
    refinement heuristic, 44
host-based, 90

identifier, 7
IGen, 22
IGP, 20
INT, 94
Interior Gateway Protocol, 9
Internet Exchange Point (IXP), 6
Intra, 20
iterative refinement, 43

LearnNeighRatio, 19
level, **18**, 20
Loc/ID split, 90, 94
local-preference, **10**, 45
locator, 7
locator/identifier separation, *see* Loc/ID split
LTP (large transit provider), 101

mapping system, 96
metric, **18**, 19
mismatch, 60
Multiple Exit Discriminator (MED), **10**, 45

NetNS, 76
network virtualization, 73
network-based, 90
next-hop atoms, 65
NumLearnPaths, 19
NumSelPaths, 19

observation point, 12

p2p, *see* peer-to-peer
parameter, *see* factor
Peer, 20
peer-to-peer, 5, 14
peering, **5**, 20
per-prefix filter, 58–65
Point of Presence, **7**, 94
Pol, 20
policies, **13**
PoP, *see* Point of Presence
Potential RIB-Out match, 41
private peering, 6
`psample`, 62
`psetA`, 48

quasi-router, 3, 39, **45**, 49, 55

response variable, *see* metric
RIB-In match, **40**, 44
RIB-Out match, **41**, 44
RouterLength, 19
ASsize, 20
routing, 1, **8**

sensitivity analysis, 24–28
shortbridge, 78
sibling, 5, 14
STP (small transit provider), 101
stretch, 28
    AS-level, 28
    geographical, 30
    router-level, 29
stub ASs, **6**, 35, 54

tiers, 5
topology, 6
    AS-level, **6**, 22
    PoP-level, **7**, 94
    router-level, **7**, 22
`training`, 40
Trellis, 73–83

`validation`, 40
valley-free, **14**, 38, 69
vantage point, 12
virtual host, 76
Vserver, 76